UCLA
UCLA Electronic Theses and Dissertations

Title

Learning and Sampling in Neural Generative Models: Energy-Based Estimation, Diffusion Sampling, and Beyond

Permalink

https://escholarship.org/uc/item/63c8n4j4

ISBN

9798173790460

Author

Yu, Peiyu

Publication Date

2026-11-13

Peer reviewed|Thesis/dissertation

eScholarship.org Powered by the California Digital Library
University of California

UNIVERSITY OF CALIFORNIA

Los Angeles

Learning and Sampling in Neural Generative Models: Energy-Based Estimation, Diffusion Sampling, and Beyond

A dissertation submitted in partial satisfaction of the requirements for the degree Doctor of Philosophy in Statistics

by

Peiyu Yu

2026

© Copyright by

Peiyu Yu

2026

ABSTRACT OF THE DISSERTATION

Learning and Sampling in Neural Generative Models:
Energy-Based Estimation, Diffusion Sampling, and Beyond

by

Peiyu Yu

Doctor of Philosophy in Statistics
University of California, Los Angeles, 2026

Professor Yingnian Wu, Co-Chair
Professor Guang Cheng, Co-Chair

Deep generative models represent complex data distributions using neural networks, but expressive probability models can be difficult to learn and sample from. This dissertation investigates two coupled challenges: constructing useful estimators for neural generative distributions, and improving sampling procedures that depend on or make use of learned models.

The first part studies learning for energy-based generative models, including the relation between scaled noise-contrastive estimation and maximum likelihood, and the use of diffusion-amortized Markov chain Monte Carlo for learning latent energy-based priors. The second part studies optimization of diffusion sampling through multi-denoiser guidance and instance-dependent denoising schedules learned by policy-gradient methods with variance reduction. The final part develops structured generative modeling approaches to unsupervised foreground extraction and interpretable text representation learning.

Together, these studies highlight that probabilistic representation, statistical estimation,

and generation algorithms are interdependent choices. They present complementary methods for improving the learnability and practical usefulness of neural generative models.

The dissertation of Peiyu Yu is approved.

Qing Zhou

Aditya Grover

Yingnian Wu, Committee Co-Chair

Guang Cheng, Committee Co-Chair

University of California, Los Angeles

2026

CONTENTS

Acknowledgments **xxi**

Vita **xxiv**

1 Introduction **1**
1.1 Learning and Sampling in Neural Generative Modeling 1
1.2 Probabilistic Representations of Complex Data 2
1.2.1 Latent-variable models 2
1.2.2 Energy-based models and maximum likelihood 3
1.3 Improving Model Learning 4
1.3.1 Learning through a contrastive objective 4
1.3.2 Learning through amortized sampling 4
1.4 Improving Diffusion Sampling 5
1.4.1 Diffusion as a learned generative process 5
1.4.2 Learning guidance from multiple denoisers 5
1.4.3 Learning instance-dependent sampling schedules 6
1.5 Structured Generative Modeling and Applications 6
1.5.1 Unsupervised image decomposition 6
1.5.2 Interpretable text modeling 7
1.6 Contributions and Dissertation Organization 7

2 "Noisier" Noise Contrastive Estimation is (Almost) Maximum Likelihood **9**
2.1 Introduction 9
2.2 Background 11
2.3 "Noisier" Noise Contrastive Estimation 13
2.3.1 Noise Magnitude 13
2.3.2 Scaling Effect of Noise Magnitude 14
2.3.3 Practical Error Analysis, N^2CE Family and Regularization 16
2.3.4 An Information-Theoretic Perspective 19
2.4 Experiments 20
2.4.1 Learning Latent-space Energy-Based Model (LEBM) with "**N**oisier" **N**oise **C**ontrastive **E**stimation (N^2CE) 20
2.4.2 Reward and Critic Learning for Diffusion Distillation 22
2.4.3 Offline Black-Box Optimization 23
2.5 Conclusion 28
2.6 Acknowledgments for This Research Chapter 28
2.7 Reproducibility Details 30
2.8 Related Work 31
2.9 Detailed Derivations and Proofs 35
2.9.1 Proof of Proposition 2.3.1 35
2.9.2 Proof of Proposition 2.3.3 37

2.9.3 Proof of Theorem 2.3.2 . . . 38
2.10 Network Architecture and Implementation Details . . . 42
2.10.1 How to Learn a Latent Space Model with N^2CE . . . 42
2.10.2 Image Modeling and Anomaly Detection . . . 43
2.10.3 Reward and Critic Learning . . . 46
2.10.4 Offline Black-Box Optimization . . . 46
2.11 Additional Experiment Details & Supplementary Results . . . 51
2.11.1 Gaussian Simulation . . . 51
2.11.2 Image Modeling . . . 53
2.11.3 Anomaly Detection . . . 54
2.11.4 Reward and Critic Learning for Diffusion Distillation . . . 55
2.11.5 Offline Black-Box Optimization . . . 56
2.11.6 2D Branin Function . . . 60
2.11.7 Design-Bench . . . 60

References for Chapter 2 **69**

3 Learning Energy-Based Prior Model with Diffusion-Amortized MCMC **84**
3.1 Introduction . . . 84
3.2 Background . . . 86
3.2.1 Energy-Based Prior Model . . . 86
3.2.2 Denoising Diffusion Probabilistic Model . . . 87
3.3 Method . . . 88
3.3.1 Amortizing MCMC with DDPM . . . 89
3.3.2 Approximate MLE with DAMC . . . 91
3.3.3 Implementation . . . 92
3.4 Experiments . . . 93
3.4.1 Generation and Inference: Prior and Posterior Sampling . . . 94
3.4.2 Analysis of Latent Space . . . 98
3.4.3 Ablation Study . . . 100
3.5 Related Work . . . 102
3.6 Conclusion . . . 104
3.7 Acknowledgments for This Research Chapter . . . 105
3.8 Theoretical Discussion . . . 106
3.8.1 KL Contraction under Idealized Assumptions . . . 106
3.8.2 Discussion about Diffusion-Based Amortization . . . 107
3.8.3 Further Discussion about the Learning Algorithm . . . 109
3.9 Network Architecture and Training Details . . . 110
3.10 Pytorch-style Pseudocode . . . 114
3.11 Dataset and Experiment Settings . . . 118
3.12 Additional Qualitative Results . . . 119
3.12.1 Toy Examples . . . 119
3.12.2 Generation . . . 121
3.12.3 Reconstruction . . . 125
3.12.4 Visualization of Transitions . . . 129

3.13 Additional Quantitative Results . . . 130
3.13.1 Learning DAMC in the Latent Space of Other Deep Latent Variable Models (DLVMs) . . . 130
3.13.2 Further Ablation Studies . . . 130
3.14 Further Discussion . . . 132
3.14.1 Limitations . . . 132
3.14.2 Broader Impacts . . . 132

References for Chapter 3 **133**

4 Learning Multi-Denoiser Guidance for Diffusion Sampling **143**
4.1 Introduction . . . 143
4.2 Multi-Denoiser Guidance Formulation . . . 144
4.2.1 Pretrained models and a formal product-of-experts target . . . 144
4.2.2 Combining scores and denoiser predictions . . . 145
4.3 Learning Mixture Weights . . . 147
4.3.1 Reward-regularized distribution optimization . . . 147
4.3.2 Preference comparisons instead of explicit scalar rewards . . . 148
4.4 Preference Optimization for Diffusion Guidance . . . 149
4.4.1 From reward regularization to direct preference optimization . . . 149
4.4.2 Diffusion surrogate based on denoising losses . . . 150
4.5 Training and Sampling Procedures . . . 151
4.5.1 Learning the guidance parameters . . . 151
4.5.2 Guided generation . . . 152
4.6 Discussion and Limitations . . . 152
4.7 Chapter Summary . . . 154
4.8 Acknowledgments for This Research Chapter . . . 154

References for Chapter 4 **155**

5 Designing Instance-Level Sampling Schedules via REINFORCE with James–Stein Shrinkage **156**
5.1 Introduction . . . 157
5.2 Related Works . . . 160
5.3 Method . . . 161
5.3.1 Problem Formulation . . . 162
5.3.2 Policy Gradient with Baselines . . . 162
5.3.3 James–Stein Shrinkage Reward Baseline . . . 164
5.3.4 Single-Pass Scheduler as Dirichlet Policy . . . 167
5.4 Experiments . . . 170
5.4.1 General Text-to-Image Experiments . . . 170
5.4.2 Text Rendering and Fine-Grained Alignment . . . 174
5.5 Conclusion . . . 177
5.6 Acknowledgments for This Research Chapter . . . 179
5.7 Additional Analyses and Experimental Details . . . 180

5.7.1 Proofs . . . 180
5.7.2 Scheduler Network Architectures . . . 183
5.7.3 Experiment Settings . . . 185
5.7.4 Additional Qualitative Results . . . 188

References for Chapter 5 **197**

6 Unsupervised Foreground Extraction via Deep Region Competition **202**
6.1 Introduction . . . 202
6.2 Related Work . . . 205
6.3 Methodology . . . 207
6.3.1 Preliminaries . . . 207
6.3.2 Generative Image Modeling . . . 208
6.3.3 Deep Region Competition: from Generative Modeling to Foreground Extraction . . . 211
6.3.4 Technical Details . . . 214
6.4 Experiments . . . 215
6.4.1 Results on Foreground Extraction . . . 215
6.4.2 Ablation Study . . . 218
6.4.3 Generalizable Foreground Extraction . . . 219
6.5 Conclusion . . . 221
6.6 Acknowledgments for This Research Chapter . . . 222
6.7 Additional Analyses and Experimental Details . . . 223
6.7.1 Dataset Details . . . 223
6.7.2 Details on Models and Hyperparameters . . . 225
6.7.3 Details on Learning Objective and Regularization . . . 227
6.7.4 PyTorch-style Code . . . 231
6.7.5 Evaluation Protocols . . . 234
6.7.6 Additional Illustrations and Baseline Results . . . 235
6.7.7 Further Discussion . . . 247

References for Chapter 6 **250**

7 Latent Diffusion Energy-Based Model for Interpretable Text Modeling **261**
7.1 Introduction . . . 261
7.2 Preliminaries: Symbol-Vector Coupling Energy-Based Model . . . 264
7.3 Latent Diffusion Energy-Based Model . . . 266
7.3.1 A Symbiosis between Symbol-Vector Coupling EBMs and Diffusion Models . . . 266
7.3.2 Information Bottleneck . . . 268
7.3.3 Geometric Clustering Anchors the Modes . . . 270
7.3.4 Algorithms and Implementation . . . 271
7.4 Experiments . . . 273
7.4.1 Generative Modeling . . . 273
7.4.2 Interpretable Text Modeling . . . 276

7.4.3 Semi-supervised Classification . . . 279
7.5 Discussion and Related Work . . . 280
7.6 Conclusion . . . 282
7.7 Acknowledgments for This Research Chapter . . . 283
7.8 Additional Derivations and Discussion . . . 284
7.8.1 Derivation of Conditional Energy-Based Models . . . 284
7.8.2 Derivation of the ELBO . . . 285
7.8.3 Detailed Discussion of Symbol Coupling . . . 286
7.8.4 Derivation of the Information Bottleneck . . . 287
7.8.5 Derivation of the Learning Gradient . . . 289
7.9 Additional Experimental Details and Discussion . . . 290
7.9.1 Network Architecture and Hyperparameters . . . 290
7.9.2 Experiment Settings and Baselines . . . 292
7.9.3 Extra Details for Experiments . . . 293

References for Chapter 7 **298**

8 Conclusion and Future Directions **307**
8.1 Summary of the Dissertation . . . 307
8.2 Three Cross-Cutting Lessons . . . 308
8.2.1 Expressiveness entails inference costs . . . 308
8.2.2 Approximate computation can be optimized statistically . . . 309
8.2.3 Inference-time control is distinct from model fitting . . . 309
8.3 Limitations and Open Problems . . . 309
8.4 Future Directions . . . 310
8.4.1 Jointly optimizing guidance and schedules . . . 310
8.4.2 Bias-controlled amortized inference . . . 310
8.4.3 Statistical characterization of controlled generation . . . 310
8.4.4 Structured and controllable latent representations . . . 311
8.5 Closing Perspective . . . 311

LIST OF FIGURES

2.1 **"Noisier" NCE gradients approach the MLE gradients.** As a sanity check, we simulate the results using 2d Gaussian distributions; *true* MLE gradients can be analytically computed. In Fig. 2.1a, we can see that $M \to \infty$ leads to a trajectory-wise convergence from Noise Contrastive Estimation (NCE) to MLE. In Fig. 2.1b, as the noise magnitude M increases, "noisier" NCE gradients $\nabla_{\boldsymbol{\alpha}} \mathcal{L}_M$ approach MLE gradients $\nabla_{\boldsymbol{\alpha}} \mathcal{J}^{\mathrm{MLE}}$; gradient bias bounded at order $O(\frac{1}{M})$ under suitable moment conditions (equivalently, squared bias $O(\frac{1}{M^2})$), which is consistent with Proposition 2.3.3. Further details can be found in Section 2.11.1. 10

2.2 **Branin function level sets.** 23

2.3 **Viz. of Branin optimal samples.** (b–d) are results of our method. G-SV denotes the Gaussian prior model sampled with SVGD. MLE-LD and MLE-SV denote the model trained by Maximum Likelihood Estimation (MLE) sampled with Langevin Dynamics (LD) and Stein Variational Gradient Descent (SVGD), respectively. 23

2.4 **Results on uniformly sampled Branin w/ and w/o top-10% points.** Zoom in for details. 25

2.5 **Finite-M N^{2}CE can exhibit more stable Monte Carlo optimization than NWJ and simple reweighting.** We plot trajectories represented by L_2 norms between model and target parameters. 53

2.6 **Uncurated 1-step generation results from our distilled EDM models.** We present 1-step generation results from our distilled models on CIFAR-10 (left) and ImageNet64x64 (right) datasets. 56

2.7 **Graphical illustration of our Black-Box Optimization (BBO) framework.** We construct an energy-based latent space model $p_{\boldsymbol{\alpha}}$ for offline BBO via learning a series of ratio estimators $\{r_{\boldsymbol{\alpha}_k}\}_{k=0}^K$ with the N^{2}CE objective $\mathcal{L}_{M\to\infty}$ to optimize the ELBO *without* Markov Chain Monte Carlo (MCMC). After training, we employ stochastic samplers like LD or SVGD to perform BBO by sampling from the implicit inverse model $p_{\boldsymbol{\theta}}(\mathbf{x}|y) \propto \mathbb{E}_{p_{\boldsymbol{\theta}}(\mathbf{z}|y)}[p_{\boldsymbol{\beta},\mathbf{x}}(\mathbf{x}|\mathbf{z})]$, where $p_{\boldsymbol{\theta}}(\mathbf{z}|y) \propto p_{\boldsymbol{\beta},y}(y|\mathbf{z}) p_{\boldsymbol{\alpha}}(\mathbf{z})$ given y. Best viewed in color. 57

2.8 Histogram of normalized function values in the Hopper dataset. The distribution is highly skewed towards low function values (plot from Mashkaria et al. [59]). 61

2.9 Dataset values vs Oracle values for top 10 points. We can see from mean and standard deviation over 20 runs that oracle is noisy (plot from Mashkaria et al. [59]). 61

3.1 **Learning the DAMC sampler.** The training samples for updating the sampler to $\boldsymbol{\phi}_{k+1}$ is obtained by T-step short-run LD, initialized with the samples from the current learned sampler $\boldsymbol{\phi}_k$. The left block shows the diffusion-based sampler, the middle block shows Langevin transitions, and the upper block shows the updated sampler. 89

3.2 **Samples generated from the Diffusion-Amortized MCMC (DAMC) sampler and LEBM** trained on SVHN, CelebA, CIFAR-10 and CelebA-HQ datasets. In each sub-figure, the first four rows are generated by the DAMC sampler. The last four rows are generated by LEBM trained with the DAMC sampler. 96
3.3 **Qualitative results of StyleGAN inversion using the DAMC sampler.** In each sub-figure, the left panel contain samples from the FFHQ dataset, and the right panel contains samples from the LSUN-T dataset. 98
3.4 **Transition of Markov chains initialized from $\mathcal{N}(0, \mathbf{I}_d)$ towards $p_{\boldsymbol{\alpha}}(\boldsymbol{z})$.** We present results by running LD for 200 and 2500 steps. In each sub-figure, the top panel displays the trajectory in the data space uniformly sampled along the chain. The bottom panel shows the energy score $f_{\boldsymbol{\alpha}}(\boldsymbol{z})$ over the iterations. 99
3.5 **The 2-arm pinwheel-shaped prior distribution used in the toy example.** 119
3.6 **Evolution of the posterior distributions learned by the DAMC sampler.** In each row, we display from left to right the evolution of the learned posterior distributions from the DAMC sampler through training iterations. The last column shows the corresponding ground-truth posterior distribution obtained by running 1K-3K steps of Langevin Dynamics until convergence for posterior sampling. 119
3.7 **Samples generated from the DAMC sampler and LEBM** trained on the SVHN dataset. 121
3.8 **Samples generated from the DAMC sampler and LEBM** trained on the CelebA dataset. 122
3.9 **Samples generated from the DAMC sampler and LEBM** trained on the CIFAR-10 dataset. 123
3.10 **Samples generated from the DAMC sampler and LEBM** trained on the CelebA-HQ dataset. 124
3.11 **Reconstructed samples from the posterior DAMC sampler** trained on the SVHN dataset. 125
3.12 **Reconstructed samples from the posterior DAMC sampler** trained on the CelebA dataset. 126
3.13 **Reconstructed samples from the posterior DAMC sampler** trained on the CIFAR-10 dataset. 127
3.14 **Reconstructed samples from the posterior DAMC sampler** trained on the CelebA-HQ dataset. 128
3.15 **Transition of Markov chains initialized from $\mathcal{N}(0, \mathbf{I}_d)$ towards $p_{\boldsymbol{\alpha}}(\boldsymbol{z})$ on SVHN.** We present results by running LD for 200 and 2500 steps. In each sub-figure, the top panel displays the trajectory in the data space uniformly sampled along the chain. The bottom panel shows the energy score $f_{\boldsymbol{\alpha}}(\boldsymbol{z})$ over the iterations. 129

3.16 **Transition of Markov chains initialized from $\mathcal{N}(0, \mathbf{I}_d)$ towards $p_{\boldsymbol{\alpha}}(\boldsymbol{z})$ on CIFAR-10.** We present results by running LD for 200 and 2500 steps. In each sub-figure, the top panel displays the trajectory in the data space uniformly sampled along the chain. The bottom panel shows the energy score $f_{\boldsymbol{\alpha}}(\boldsymbol{z})$ over the iterations. 129

5.1 **Illustrations from the sampling-schedule study.** Paired images compare default (left) and learned instance-specific schedules (right), demonstrating few-step sampling, image-text alignment, text rendering, and object counting. 156

5.2 **James–Stein (JS) reward baseline.** a Simulation results from analytic policies illustrating lower measured gradient-estimator variance for the evaluated rollout counts K. The experimental setup is described in Section 5.7.3.1. b Diagram of JS baseline: combining b_{RLOO} and b_{xctx} into $b_{\mathrm{JS}} = (1 - \widehat{\alpha}_c) b_{\mathrm{RLOO}} + \widehat{\alpha}_c b_{\mathrm{xctx}}$ (Eq. (5.7)). 158

5.3 **Rescheduling improves general T2I alignment.** Head-to-head comparisons between images generated with default schedules (upper) and our learned schedules (lower) from Flux-Dev with 40 steps. Figures henceforth follow the same format. 172

5.4 **Representative training curves (Flux-Dev) for different baselines and sampling budgets L.** The horizontal axis denotes training iterations and the vertical axis the aggregate reward. The plotted JS baseline yields stronger training rewards than the compared XCTX and RLOO baselines in these settings; the effect is most pronounced at small step budgets. 173

5.5 **Rescheduling improves few-step sampling**. Comparisons between images generated with default schedules (upper) and our learned schedules (lower) from Flux-Dev, with 5 steps. 175

5.6 **Rescheduling improves text rendering**. We present comparisons between images generated with default schedules (upper) and our learned schedules (lower) from Flux-Dev. 177

5.7 **Rescheduling improves fine grained alignment**. Comparisons between images generated with default schedules (upper) and our learned schedules (lower) from Flux-Dev. 178

5.8 **Rescheduling improves few-step sampling**. Comparisons between images generated with default schedules (upper) and our learned schedules (lower) from Flux-Dev, with 5 steps. 189

5.9 **Rescheduling improves few-step sampling**. Comparisons between images generated with default schedules (upper) and our learned schedules (lower) from Flux-Dev, with 5 steps. 190

5.10 **Rescheduling improves general T2I alignment.** Head-to-head comparisons between images generated with default schedules (upper) and our learned schedules (lower) from Flux-Dev with 40 steps. 191

5.11 **Rescheduling improves general T2I alignment.** Head-to-head comparisons between images generated with default schedules (upper) and our learned schedules (lower) from Flux-Dev with 40 steps. 192

5.12 **Rescheduling improves text rendering**. We present comparisons between images generated with default schedules (upper) and our learned schedules (lower) from Flux-Dev. 193
5.13 **Rescheduling improves text rendering**. We present comparisons between images generated with default schedules (upper) and our learned schedules (lower) from Flux-Dev. 194
5.14 **Rescheduling improves fine grained alignment**. Comparisons between images generated with default schedules (upper) and our learned schedules (lower) from Flux-Dev. 195
5.15 **Rescheduling improves fine grained alignment**. Comparisons between images generated with default schedules (upper) and our learned schedules (lower) from Flux-Dev. 196

6.1 **Overview of Deep Region Competition (DRC).** (a) The model generates foreground and background regions using sampled latent variables $\mathbf{z} = \{\mathbf{z}_1, \mathbf{z}_2\}$. $p_{\beta_k},\ k = 1, 2$ represents the generator for each region. Of note, the pixel re-assignment function is absorbed in the background generator; see Section 6.3.2 for details. (b) DRC samples the latent variables $\mathbf{z}$ in an iterative manner. Let $\mathbf{x}$ denote the observed image; we use $\hat{\mathbf{x}}_t,\ t = 0, 1, ...$ to represent the image generated by $p_\beta(\mathbf{x}|\mathbf{z})$ at the t-th sampling step. DRC has a two-step workflow for learning unsupervised foreground extractors that resembles the E- and M-step in the classic Expectation-Maximization (EM) algorithm. In the E-step, it employs gradient-based MCMC sampling to infer the latent variables $\mathbf{z}$ as shown in (b). Of note, only the latent variables $\mathbf{z}$ are updated in this step. In the M-step, the sampled latent variables $\mathbf{z}$ are fed into the model for image generation as shown in (a), where the generators are updated to minimize the reconstruction error. 210
6.2 **Pixel re-assignment.** The output of β_p can be viewed as a learned re-assignment of the original background pixels that follows the mapped grid $P_{\alpha_p,\beta_p}(\mathbf{z}, C)$. Note that the re-assignment function $P_{\alpha_p,\beta_p}(\mathbf{z}, \cdot)$ might not be injective. The final background image is generated via grid sampling. 211
6.3 **Foreground extraction results for each dataset**; zoom in for better visibility. From top to bottom: (i) observed images, (ii) generated images, (iii) masked generated foregrounds, (iv) generated backgrounds, (v) ground-truth foreground masks, and (vi) inferred foreground masks. More samples and results of baselines can be found in the additional analyses below. 217
6.4 Examples of excluded images. From left to right: (i) image with a foreground object that occupied too much space, (ii) image with a low-quality mask, and (iii) image with significant foreground distractors. 223
6.5 Samples from Textured Multi-dSprites (TM-dSprites). From top to bottom: (1) observed images, (ii) background textures, and (iii) ground-truth masks. . 225
6.6 An example situation when using each individual mask as the background mask gives higher scores. Note that if we threshold the output of each individual slot and compose them, the result would be the mask shown in the last column. 235

6.7 An example situation when thresholding&combining the output of each individual slot gives higher scores. We can see from the last column that the combined mask fits the foreground objects well. 235
6.8 Additional foreground extraction results on Caltech-UCSD Birds-200-2011 (Birds) dataset. 236
6.9 Additional foreground extraction results on Stanford Dogs (Dogs) dataset. . 237
6.10 Additional foreground extraction results on Stanford Cars (Cars) dataset. . . 238
6.11 Additional foreground extraction results on CLEVR6 and TM-dSprites datasets. 239
6.12 Typical failure modes on Birds and TM-dSprites. 240
6.13 Results of GrabCut on Birds and TM-dSprites datasets. The first three columns are results from Birds dataset, and the last three are from TM-dSprites. From left to right, we display the observed image, ground-truth mask, and the foreground extraction results respectively. 241
6.14 Results of ReDO on Birds and TM-dSprites datasets. The first four columns are results from Birds dataset, and the last four are from TM-dSprites. From left to right, we display the observed image, ground-truth mask, mask from the first output channel and from the second channel respectively. 242
6.15 Results of IODINE on Birds datasets. We provide the observed image, mask from the first slot and from the second slot respectively. 243
6.16 Results of IODINE on TM-dSprites datasets. We provide the observed image and masks from four object slots respectively. 244
6.17 Results of Slot-Attention on Birds datasets. We provide the observed image, mask from the first slot and from the second slot respectively. 245
6.18 Results of Slot-Attention on TM-dSprites datasets. We provide the observed image and masks from four object slots respectively. 246
6.19 Image histograms for foreground objects and background regions from each dataset. 247
6.20 Heatmaps of ground-truth masks for each dataset. 248
6.21 Preliminary results on learning slot-based object representation. 248
6.22 Failure modes of energy-based slot representation model. 248
6.23 Prior sampling results on Birds dataset. 249

7.1 **Graphical illustration of the latent diffusion process.** We construct the forward and reverse diffusion processes in the latent space. The symbolic one-hot vector is coupled with the initial latent vector $\mathbf{z}_0$. The latent and diffused latent variables are highlighted by the red and blue plates, respectively. The cyan arrows indicate that $\mathbf{z}_0$ is connected with only $\mathbf{z}_1$. We learn a sequence of EBMs to model the reverse diffusion process $p_\alpha(\mathbf{z}_t|\mathbf{z}_{t+1})$. 262

7.2 **Evaluation on 2D synthetic data**: a mixture of 16 Gaussians (left panel) and a 10-arm pinwheel-shaped distribution (right panel). In each panel, the top, middle, and bottom rows display densities learned by Symbol-Vector Coupling Energy-Based Model (SVEBM)-Information Bottleneck (IB), our model without geometric clustering, and our full model, respectively. From left to right, each row shows the data distribution and the Kernel Density Estimations (KDEs) of: $\mathbf{x}$ generated from amortized posterior $\mathbf{z}$ samples, $\mathbf{x}$ generated from MCMC-sampled prior $\mathbf{z}$ samples, posterior $\mathbf{z}$ samples, and prior $\mathbf{z}$ samples. 273
7.3 **Visualization of color-coded data points.** We visualize data points and the corresponding inferred latent variables of two 2D synthetic datasets (*gaussian* and *pinwheel*). Data points with different labels are assigned with different colors. 274
7.4 **Visualization of $p_\alpha(\mathbf{y}|\mathbf{z}_t)$ over t.** $p_\alpha(\mathbf{y}|\mathbf{z}_t)$ is constantly around the probability of 0.5 over t. 296
7.5 Full evolution of SVEBM–IB and LDEBM variants on Gaussian and pinwheel synthetic distributions. Each panel displays representative states over training. 297

LIST OF TABLES

2.1 **FID(↓) on different datasets**. We highlight our model, the **1**st and 2nd performances; tables henceforth follow this format. Numbers from the first six rows are from Yu et al. [118]. `nz` denotes the latent dimension. `M` and `K` denote the noise magnitude and num. of stages for ratio estimation, respectively. . . 21

2.2 **AUPRC(↑) scores for unsupervised anomaly detection on MNIST**. Baseline numbers are taken from Yoon et al. [114], Yu et al. [118]. Full results with variances in found in Section 2.11.3. 21

2.3 **CIFAR-10 (DDPM backbone) and ImageNet64×64 (EDM backbone) results** shown side-by-side. First six rows are from [115]. † highlights the starting point of DxMI fine-tuning. 23

2.4 **CIFAR-10 results**. "FID-U/C (iters)" shows uncond./cond. FID and, when provided, corresponding training iterations in parentheses. Baseline numbers from [122, 123]. 24

2.5 **Conditional ImageNet 64×64 results.** 24

2.6 **Results on the top-10%-tile-removed Branin task (avg. over 5 runs).** Opt denotes the global optimum. 25

2.7 **Normalized results on design-bench with $Q = 256$ (avg. over 5 runs).** 26

2.8 **Ablation studies on design-bench (avg. over 5 runs)**, results with a budget $Q = 256$. 26

2.9 **Network structures of the generator networks** used for the SVHN, CelebA, CIFAR-10, CelebA-HQ and MNIST (from top to bottom) datasets. ConvT(n) indicates a transposed convolutional operation with n output channels. We use `ngf=64` for the SVHN dataset and `ngf=128` for the rest. LReLU indicates the Leaky-ReLU activation function. The slope in Leaky ReLU is set to be 0.2. 44

2.10 **Network structures of the encoder networks** used for the SVHN, CelebA, CIFAR-10, CelebA-HQ and MNIST (from top to bottom) datasets. Conv(n)Norm indicates a convolutional operation with n output channels followed by the Instance Normalization [104]. We use `nif=64` for all the datasets. LReLU indicates the Leaky-ReLU activation function. The slope in Leaky ReLU is set to be 0.2. 49

2.11 **Network architecture for the log-ratio estimator $\tilde{f}_{\boldsymbol{\alpha}_k}(\mathbf{z}, k)$.** N is set to 3 for image modeling and anomaly detection experiments, and 16 for the offline BBO experiments. LReLU denotes the Leaky ReLU activation function. The slope in Leaky ReLU is set to 0.2. For SVHN and CelebA datasets, we use `nz=100`. For the CIFAR-10 and CelebA-HQ datasets, we use `nz=128` and `nz=512`, respectively. We use `nz=8` for anomaly detection on the MNIST dataset; `nz=20` is the latent space dimension used for offline BBO experiments. 50

2.12 **Generator and encoder network architectures.** LReLU indicates the Leaky-ReLU activation function. $d_{\mathbf{x}}$ is the dimension of input $\mathbf{x}$ mentioned in Table 2.18. For discrete tasks, $d_{\mathbf{x}}$ specifies the length of input, and L denotes number of possible discrete tokens. 50

2.13 **MSE sweep for $n = 2$.** The finite-sample trade-off depends on the noise-sample allocation; see Proposition 2.3.3. 53
2.14 **MSE sweep for $n = 500$.** The finite-sample trade-off depends on the noise-sample allocation; see Proposition 2.3.3. 53
2.15 **AUPRC(↑) scores for unsupervised anomaly detection on MNIST**. Baseline numbers are taken from Yoon et al. [114], Yu et al. [118]. Results of our model are averaged over 10 trials. 54
2.16 **Ablative FID scores of M on CIFAR-10 under [115] setting.** 55
2.17 **Ablative uncond. FID scores of M on CIFAR-10 under [123] setting.** 55
2.18 **Design-Bench dataset statistics.** 61
2.19 **Results on (reduced size) Branin function dataset.** 64
2.20 **Results on (reduced size) D'Kitty dataset.** x% indicates the proportion of training data withheld, *i.e.*, 0% represents the full dataset. 64
2.21 **Results on design-bench. Unnormalized** results with a budget $Q = 256$. 64
2.22 **Normalized results on design-bench.** 50-th percentile results with $Q = 256$. Baseline numbers from [59]. DDOM [43] does not report 50-th percentile results; we are unable to find public model weights corresponding with the results in Table 2.7 for DDOM and therefore omit DDOM in this table. 64
2.23 **Normalized design-bench results with $Q = 128$ at 100-th percentile.** Baseline numbers from [8, 37, 68, 120]. **ChEMBL** dataset is excluded following baseline set-ups. * indicates that the baseline has slightly different set-ups; the numbers are for rough reference (see Section 2.11.7.3 for more details). . 65
2.24 **Normalized design-bench 25-th and 75-th results with $Q = 128$.** . . . 65
2.25 **Normalized design-bench 50-th percentile results with $Q = 128$.** Baseline numbers from [8, 37, 68, 120]. **ChEMBL** dataset is excluded following baseline set-ups. * indicates that the baseline has slightly different set-ups; the numbers are for rough reference (see Section 2.11.7.3 for more details). . 66
2.26 **Ablation of K, M on SPRCON. (avg. over 5 runs).** 67
2.27 **Ablation of Q on D'Kitty (avg. over 5 runs).** 67
2.28 **Random baseline results on the D'Kitty dataset.** Results from 5 runs. 68

3.1 **MSE(↓) and FID(↓) obtained from models trained on different datasets**. The FID scores are computed based on 50k generated images and training images for the first three datasets and 5k images for the CelebA-HQ dataset. The MSEs are computed based on unseen testing images. We highlight our model results in gray color. The best and second-best performances are marked in bold numbers and underlines, respectively; tables henceforth follow this format. *[41] uses a prior model with 4x parameters compared with [22] and ours. 94
3.2 **MSE(↓) and FID(↓) for GAN inversion on different datasets**. Opt. and Enc. denotes the optimization-based and encoder-based methods. 97
3.3 **AUPRC(↑) scores for unsupervised anomaly detection on MNIST**. Numbers are taken from [41]. Results of our model are averaged over the last 10 trials to account for variance. 100

3.4 **Ablation study on CIFAR-10 dataset**. VI denotes learning LEBM using variational methods. SR denotes learning LEBM with short-run LD. DAMC-G replaces the LEBM in DAMC-LEBM with a standard Gaussian distribution. NALR denotes the non-amortized Denoising Diffusion Probabilistic Model (DDPM) setting. For each set-up, we provide results using the vanilla sampling method, denoted as V, and the ones using the DAMC sampler, denoted as D. 101
3.5 **Network structures of the energy score network.** LReLU denotes the Leaky ReLU activation function. The slope in Leaky ReLU is set to 0.2. For SVHN and CelebA datasets, we use `nz=100`. For the CIFAR-10 and CelebA-HQ datasets, we use `nz=128`. We use `nz=8` for anomaly detection on the MNIST dataset, and `nz=7168` for GAN inversion. We use `ndf=512` for GAN inversion and `ndf=200` for the rest experiments. 110
3.6 **Network structures of the generator networks** used for the SVHN, CelebA, CIFAR-10, CelebA-HQ and MNIST (from top to bottom) datasets. For GAN inversion, we use the StyleGAN [10] structure as our generator network. ConvT(n) indicates a transposed convolutional operation with n output channels. We use `ngf=64` for the SVHN dataset and `ngf=128` for the rest. LReLU indicates the Leaky-ReLU activation function. The slope in Leaky ReLU is set to be 0.2. 111
3.7 **Network structure of the denoising diffusion network.** a) We use the sinusoidal embedding to embed the time index as in [8, 25]. b) We use the learned fourier feature module [75] to embed the input $\boldsymbol{z}$. c) The merged time embedding and context embedding is used to produce a pair of bias and scale terms to shift and scale the embedding of input $\boldsymbol{z}$. `nz` is the input dimension, as in Table 3.5. `nemb` is the dimension of image embedding as in Table 3.8. . 112
3.8 **Network structures of the encoder networks** used for the SVHN, CelebA, CIFAR-10, CelebA-HQ and MNIST (from top to bottom) datasets. For GAN inversion, the encoder network structure is the same as in [18]. Conv(n)Norm indicates a convolutional operation with n output channels followed by the Instance Normalization [76]. We use `nif=64` and `nemb=1024` for all the datasets. LReLU indicates the Leaky-ReLU activation function. The slope in Leaky ReLU is set to be 0.2. 113
3.9 **Further baseline results for learning DAMC on CIFAR-10 dataset.** These models are implemented using the same encoder and decoder network architectures for fair comparison 130
3.10 **Ablation Studies for the choice of Langevin steps and model capacity**. We highlight the results of our set-up reported in the main text. `T` is the posterior sampling step. `f` stands for the factor for the model capacity, *e.g.*, `f=2` means 2x the size of the original model. 131

5.1 **HPSv2 scores on HPDv2 held-out prompts.** Scores by number of inference steps (L). Rows per backbone: Default, Cross-Context RLOO (XCTX), RLOO, TPDM-style PPO (Flux only), and Ours (JS). Best per column in **bold**; second-best underlined. 171
5.2 Flux-Dev 5-step sampling on HPDv2. 175

5.3 **Text rendering at a fixed step budget ($L = 40$).** HPSv2 (↑), OCR-A (↑), OCR-P (↑), OCR-R (↑). Best per backbone in **bold**. 176
5.4 **GenEval** results by backbone and schedule. Metrics are fractions in $[0, 1]$. c.attrs. = Color Attribution. 176
5.5 **Network architecture for the Dirichlet schedule policy $\alpha_\theta(x, c)$.** The network takes an image-shaped noise tensor x, a sequence of text embeddings e_{text}, and an optional pooled text embedding $\bar{e}_{\text{text}}$ (consistent with SD-XL, SD3.5M/L and FLUX models), and outputs Dirichlet parameters $\boldsymbol{\alpha} \in \mathbb{R}_+^{L+1}$ over L schedule intervals plus one skipped interval. Default network architecture hyperparameters: `text_embed_dim` $d_{\text{text}} = 2048$, `pooled_text_embed_dim` $d_{\text{pool}} = 1280$, `image_encoder_depth` $n_{\text{conv}} = 2$, `image_encoder_width` $d_{\text{conv}} = 32$, `attention_dim` $d_{\text{attn}} = 256$, `cross_attention_heads` $n_{\text{attn}} = 4$, `number_of_blks` $N_b = 4$, `hidden_dim` $d_h = 256$, `num_mlp_layers` $n_{\text{mlp}} = 2$. 184
5.6 Synthetic variance sanity check. Average per-dimension variance of REINFORCE for an analytic Dirichlet policy. Rewards follow Eq. (5.18). Entries are RLOO → JS. JS consistently gives gain over RLOO baselines for various settings. 186
5.7 Learning rates for schedule training on HPDv2 prompts. 187
5.8 Learning rates for schedule training on text rendering and geneval-style prompts. Num. of sampling steps is 40. 188

6.1 **Foreground extraction results on training data measured in IoU and Dice.** Higher is better in all scores. *Results of W-Net and OneGAN are provided by Benny and Wolf [31]. Of note, results of these two models on Dogs and Cars datasets may **not** be directly comparable to other listed methods, as the data used for training and evaluation could be different. We include these results as a rough reference since no official implementation or pretrained model are publicly available. § indicates unfair baseline results obtained using extra ground-truth information, *i.e.*, we choose the best-matching scores from the permutation of foreground and background masks. †OneGAN is a strong **weakly supervised** baseline, which requires clean background images to provide additional supervision. We include this model as a potential upper bound of the fully unsupervised methods. 216
6.2 **Ablation study on Birds.** *We replace the LEBM with encoders to perform amortized inference for the latent variables **z** within a variational framework as in VAE [83]. †We explore the possibility of using short-run MCMC [84] instead of persistent chain sampling. 219
6.3 **Performance of DRC on training and held-out testing data.** *GrabCut is a deterministic method that does not require training. We report the results of GrabCut [5] on these splits only for reference. 220
6.4 **Performance of DRC on unseen testing categories.** *We include the testing results of models trained with data from the same categories for reference. 220
6.5 Dimension of latent variables on each dataset. 225

6.6 Architecture of the generators, LEBMs and auxiliary classifiers (see Section 6.7.3.2). UpConv3x3Norm denotes an Upsampling-Convolutional-InstanceNorm layer with a convolution kernel size of 3. Similarly, Conv4x4Norm denotes a Convolutional-InstanceNorm layer with a kernel size of 4. LReLU denotes the Leaky-ReLU activation function. The leak factor for LReLU is 0.2 in LEBMs and auxiliary classifiers, and 0.01 in generators. $*D$ represents the dimensions of the latent variables for different datasets; see Table 6.5. $\dagger K$ represents the pre-specified category number for latent variables. We use 200 for both the foreground and background LEBMs on real-world datasets, and 30 and 10 in the foreground and background LEBMs on multi-object datasets respectively. 226

7.1 **Results of language generation on PTB dataset.** We highlight our model results in gray color. The best and second-best performances are marked in bold numbers and underlines, respectively; tables henceforth follows this format. 275

7.2 **Sentence completion on JerichoWorld dataset.** The gray words in the input sentences are masked with `<unk>` token. 276

7.3 **Results of interpretable text modeling on Daily Dialog (DD).** We use mutual information (MI), BLEU, and homogeneity with actions and emotions for evaluation. 277

7.4 **Dialog evaluation results on Stanford Multi-Domain Dialog (SMD) and DD.** Models are assessed using four metrics: BLEU, average, extrema, and greedy word embedding based similarity. 277

7.5 **Samples of unsupervisedly discovered action categories and corresponding utterances on SMD.** 278

7.6 **Dialog cases generated by Latent Diffusion Energy-Based Model (LDEBM) given the context.** On SMD, we provide the same context but with different $\mathbf{y}$ values to generate each response; actions indicated by $\mathbf{y}$ are listed in parentheses. On DD, LDEBM can well capture the dialog topic; we provide the ground-truth response in each case for reference. 278

7.7 **Accuracy of sentence attribute control on Yelp.** 279

7.8 **Generated positive and negative reviews on Yelp.** 279

7.9 **Accuracy on AGNews.** We report semi-supervised classification accuracy with varied number of labeled data. 280

7.10 **Network architecture for the LDEBM prior.** N is set to 12 for all the experiments. 291

7.11 **Hyperparameters of LDEBM.** DD-CLS presents the set of hyperparameters used in unsupervised clustering on DD dataset. DD-GEN presents the set of hyperparameters used in conditional response generation on DD dataset. 292

ACKNOWLEDGMENTS

I am deeply grateful to my advisor, Professor Ying Nian Wu, for his guidance, encouragement, and support throughout my graduate studies. His perspective on probabilistic modeling and statistical learning has profoundly shaped my approach to research. I also sincerely thank Professor Guang Cheng for serving as co-chair of my dissertation committee, and Professors Qing Zhou and Aditya Grover for their time, thoughtful questions, and valuable feedback.

I am fortunate to have worked with many generous collaborators and colleagues, whose discussions and perspectives have enriched the research presented here. I thank the faculty, students, and staff in the Department of Statistics and Data Science at UCLA for the stimulating academic environment and their support. I gratefully acknowledge support from the UCLA Dissertation Year Award and the Charles E. and Sue K. Young Graduate Student Fellowship. Finally, I thank my family and friends for their enduring encouragement and understanding. I am very fortunate to have gotten to know, worked with, and learned from Hengzhi during my Ph.D. studies. It is a blessing of a lifetime to have met Lindsey during my time at UCLA.

Coauthored research included in this dissertation.

Parts of Chapters 2–7 are adapted from the following coauthored publications or collaborative research outputs. The bibliographic details and contributions of the collaborators are acknowledged below.

Chapter 2. This chapter is based on Peiyu Yu, Dinghuai Zhang, Hengzhi He, Xiaojian Ma, Sirui Xie, Ruiyao Miao, Yifan Lu, Yasi Zhang, Deqian Kong, Ruiqi Gao, Jianwen Xie, Guang Cheng, and Ying Nian Wu, "Noisier" Noise Contrastive Estimation is (Almost) Maximum Likelihood, *International Conference on Learning Representations (ICLR)*, 2026. `https://openreview.net/forum?id=qR59RrG7Om`. Peiyu Yu contributed to the conception and development of the method, experimental design and execution, and writing of the manuscript. The other coauthors contributed to research discussions and feedback on the work.

Chapter 3. This chapter is based on Peiyu Yu, Yaxuan Zhu, Sirui Xie, Xiaojian Ma, Ruiqi Gao, Song-Chun Zhu, and Ying Nian Wu, "Learning Energy-Based Prior Model with Diffusion-Amortized MCMC," *Advances in Neural Information Processing Systems*, vol. 36, pp. 42717–42747, 2023. NeurIPS proceedings record. Peiyu Yu conceived the method, designed and conducted the experiments, and wrote the manuscript. The remaining coauthors contributed through research discussions and feedback.

Chapter 4. This chapter describes collaborative research by Peiyu Yu, Morteza Mardani, and Arash Vahdat, related to their published patent application: Peiyu Yu, Morteza Mardani, and Arash Vahdat, "Enhanced Diffusion Model Guidance via Multi-Denoiser Mixing," U.S. Patent Application Publication No. US20260087199A1, March 26, 2026. `https://patents.google.com/patent/US20260087199A1/en`. All three named inventors collaborated on the underlying research. Peiyu Yu prepared the adaptation of this research for the dissertation.

Chapter 5. This chapter is based on Peiyu Yu, Suraj Kothawade, Sirui Xie, Ying Nian Wu, and Hongliang Fei, "Designing Instance-Level Sampling Schedules via REINFORCE with James–Stein Shrinkage," *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 36040–36050, 2026. CVPR proceedings record. Peiyu Yu contributed to the conception of the method, experimental design and implementation, execution of the experiments, and preparation of the manuscript. The other coauthors contributed to research discussions and feedback.

Chapter 6. This chapter is based on Peiyu Yu, Sirui Xie, Xiaojian Ma, Yixin Zhu, Ying Nian Wu, and Song-Chun Zhu, "Unsupervised Foreground Extraction via Deep Region Competition," *Advances in Neural Information Processing Systems*, vol. 34, pp. 14264–14279, 2021. NeurIPS proceedings record. Peiyu Yu conceived the method, designed and conducted the experiments, and wrote the manuscript. The other coauthors participated in research discussions and provided feedback.

Chapter 7. This chapter is based on Peiyu Yu, Sirui Xie, Xiaojian Ma, Baoxiong Jia, Bo Pang,

Ruiqi Gao, Yixin Zhu, Song-Chun Zhu, and Ying Nian Wu, "Latent Diffusion Energy-Based Model for Interpretable Text Modelling," *Proceedings of the 39th International Conference on Machine Learning (ICML)*, in *Proceedings of Machine Learning Research*, vol. 162, pp. 25702–25720, 2022. `https://proceedings.mlr.press/v162/yu22h.html`. Peiyu Yu conceived the method, designed and conducted the experiments, and wrote the manuscript. The other coauthors participated in research discussions and provided feedback.

VITA

Education

Bachelor of Engineering in Automation, Tsinghua University, 2016–2020.

Graduate study in Computer Science, University of California, Los Angeles, 2020–2021.

Selected Publications and Research Outputs

Peiyu Yu, Dinghuai Zhang, Hengzhi He, et al., "Noisier" Noise Contrastive Estimation is (Almost) Maximum Likelihood, *ICLR*, 2026.

Peiyu Yu, Suraj Kothawade, Sirui Xie, Ying Nian Wu, and Hongliang Fei, "Designing Instance-Level Sampling Schedules via REINFORCE with James–Stein Shrinkage," *CVPR*, pp. 36040–36050, 2026.

Peiyu Yu, Yaxuan Zhu, Sirui Xie, et al., "Learning Energy-Based Prior Model with Diffusion-Amortized MCMC," *NeurIPS*, vol. 36, pp. 42717–42747, 2023.

Peiyu Yu, Sirui Xie, Xiaojian Ma, et al., "Latent Diffusion Energy-Based Model for Interpretable Text Modelling," *ICML*, PMLR vol. 162, pp. 25702–25720, 2022.

Peiyu Yu, Sirui Xie, Xiaojian Ma, et al., "Unsupervised Foreground Extraction via Deep Region Competition," *NeurIPS*, vol. 34, pp. 14264–14279, 2021.

Hengzhi He, Peiyu Yu, Junpeng Ren, Ying Nian Wu, and Guang Cheng, "Watermarking Continuous Tabular Data Generation," accepted for publication in the *Journal of the American Statistical Association*, 2026.

Peiyu Yu, Morteza Mardani, and Arash Vahdat, "Enhanced Diffusion Model Guidance via Multi-Denoiser Mixing," U.S. Patent Application Publication US20260087199A1, 2026.

Chapter 1

Introduction

1.1 Learning and Sampling in Neural Generative Modeling

Generative modeling aims to learn the statistical structure of observed data and to generate new samples that exhibit similar properties. Let p_{data} denote an unknown distribution on an observation space $\mathcal{X}$. A neural generative model specifies, explicitly or implicitly, a family of distributions $\{p_\theta\}$ from which a model is learned using observations drawn from p_{data}. Once learned, the model must also support a practical procedure for generating samples, potentially subject to additional conditions or quality criteria.

Learning and sampling are distinct problems, but their interaction is central to generative modeling. In energy-based and latent-variable models, learning may require expectations under distributions from which exact sampling is difficult. Approximating these expectations introduces a tradeoff between computational cost and statistical accuracy. In diffusion models, a learned denoising network defines a family of sampling procedures whose quality and efficiency depend on choices made during inference. Thus, sampling affects not only how a trained model is used, but, in some models, how its parameters can be estimated.

This dissertation studies these connections through three related themes. The first

concerns the estimation of neural probability distributions, including contrastive objectives and the amortization of Markov chain Monte Carlo (MCMC). The second concerns the design of efficient and controllable sampling procedures for pretrained diffusion models. The third concerns structured generative representations, illustrated by unsupervised image decomposition and interpretable text modeling. Across these settings, the central question is how the choice of probabilistic representation, learning procedure, and sampling algorithm shapes the capabilities of a generative model.

1.2 Probabilistic Representations of Complex Data

1.2.1 Latent-variable models

Images and text are high-dimensional observations that often depend on underlying semantic or structural factors. Latent-variable models represent these factors through an unobserved variable z, with joint distribution

$$p_{\alpha,\beta}(x, z) = p_\alpha(z)\, p_\beta(x \mid z), \tag{1.1}$$

and marginal likelihood

$$p_{\alpha,\beta}(x) = \int p_\alpha(z) p_\beta(x \mid z)\, \mathrm{d}z. \tag{1.2}$$

Here $p_\alpha(z)$ is a prior over latent variables, while $p_\beta(x \mid z)$ describes how observations arise from them. The latent representation can capture statistical dependencies in the data and, when suitably structured, provide a basis for interpretation or control.

Although a simple Gaussian prior is convenient, it may not adequately represent a complex or multimodal latent distribution. An energy-based prior introduces a flexible correction to a reference density p_0:

$$p_\alpha(z) = \frac{1}{Z(\alpha)} \exp\{f_\alpha(z)\}\, p_0(z), \qquad Z(\alpha) = \int \exp\{f_\alpha(z)\} p_0(z)\, \mathrm{d}z, \tag{1.3}$$

where f_α is a neural function and $Z(\alpha)$ is the normalizing constant. This flexibility comes with a computational challenge: likelihood-based learning generally involves expectations under the learned prior and the latent posterior, neither of which is necessarily easy to sample from. Efficient approximation of these expectations motivates several of the methods studied in this dissertation.

1.2.2 Energy-based models and maximum likelihood

Consider an energy-based density $p_\theta(x) = \exp\{f_\theta(x)\}/Z(\theta)$. Under standard regularity conditions, its maximum-likelihood gradient is

$$\nabla_\theta \, \mathbb{E}_{p_{\text{data}}}[\log p_\theta(x)] = \mathbb{E}_{p_{\text{data}}}[\nabla_\theta f_\theta(x)] - \mathbb{E}_{p_\theta}[\nabla_\theta f_\theta(x)]. \tag{1.4}$$

The second, or negative-phase, expectation is taken under the current model. It can be approximated with MCMC, but finite-length chains may introduce bias, while longer chains increase the cost of training. The quality of the sampler therefore directly influences likelihood-based estimation.

Noise-contrastive estimation (NCE) offers an alternative by learning to distinguish observations from samples of a reference noise distribution. The resulting density-ratio objective can be optimized without explicitly computing the model's partition function. However, when the data and noise distributions differ substantially, a classifier may separate them easily without providing accurate density-ratio estimates or useful optimization gradients. This difficulty motivates the contrastive estimation method developed in Chapter 2.

1.3 Improving Model Learning

1.3.1 Learning through a contrastive objective

Chapter 2, *"Noisier" Noise Contrastive Estimation is (Almost) Maximum Likelihood*, investigates the relationship between NCE and maximum likelihood from the perspective of the effective weight assigned to the noise distribution. It shows that increasing this weight can bring the contrastive gradient closer to the maximum-likelihood gradient under appropriate conditions, without requiring a proportional increase in computational cost. The chapter develops the corresponding estimator and examines its statistical behavior and empirical performance in density modeling and related applications.

The central insight is that a suitably designed contrastive objective can approximate the optimization behavior of maximum likelihood while avoiding the need to estimate a model expectation with MCMC at every update. This addresses the learning objective rather than the sampling algorithm used to approximate it.

1.3.2 Learning through amortized sampling

Chapter 3, *Learning Energy-Based Prior Model with Diffusion-Amortized MCMC*, addresses a complementary challenge: how to obtain better samples when MCMC is part of the learning procedure. The method uses a learned diffusion sampler to amortize the effect of many MCMC transitions. Short-run MCMC refinement provides updated samples that are used to improve the amortized sampler, allowing information from successive transitions to accumulate without repeatedly running a long chain from scratch.

This approach is developed for latent energy-based priors, whose likelihood-based learning involves sampling from both the latent prior and posterior. By improving these approximations, diffusion-amortized MCMC aims to reduce the sampling bottleneck and support more reliable model estimation. Together, Chapters 2 and 3 illustrate two different routes to more tractable learning: changing the estimation objective and improving the computational procedure used

to estimate its gradients.

1.4 Improving Diffusion Sampling

1.4.1 Diffusion as a learned generative process

Diffusion models generate observations by reversing a gradual noising process. In a common Gaussian formulation,

$$x_t = \alpha_t x_0 + \sigma_t \varepsilon, \qquad \varepsilon \sim \mathcal{N}(0, I), \tag{1.5}$$

where t indexes the noise level. A denoising or score network provides predictions used to construct the reverse process. In practice, these predictions are evaluated at a finite sequence of noise levels, and the resulting samples depend on both the network and the numerical sampling procedure.

For a pretrained diffusion model, the principal challenge is often to improve generation quality or efficiency without retraining the underlying network. Choices such as how to guide denoising predictions and where to place sampling steps can substantially influence the outputs. Chapters 4 and 5 study these two complementary forms of sampling-time adaptation.

1.4.2 Learning guidance from multiple denoisers

Chapter 4, *Learning Multi-Denoiser Guidance for Diffusion Sampling*, explores how several pretrained denoisers can be combined to guide a base diffusion model. Rather than using a fixed combination of predictions, the method learns guidance weights using preference-based objectives. This provides a mechanism for adapting the sampling dynamics to desired generation criteria while retaining the underlying pretrained models.

1.4.3 Learning instance-dependent sampling schedules

Chapter 5, *Designing Instance-Level Sampling Schedules via REINFORCE with James–Stein Shrinkage*, considers the placement of denoising steps under a fixed sampling budget. It learns a distribution over schedules conditioned on information available before the denoising trajectory begins, and optimizes the schedules using reward-based policy gradients. A James–Stein shrinkage baseline is used to reduce the variance of policy-gradient estimation.

These two chapters optimize different aspects of the same sampling process. Multi-denoiser guidance changes the predictions used in the reverse dynamics, whereas schedule optimization changes when those predictions are evaluated. Both show how sampling-time decisions can improve the use of a pretrained generative model without modifying its original training objective.

1.5 Structured Generative Modeling and Applications

The preceding chapters focus on the objectives and algorithms used to learn or sample from generative models. A related question is what structure a model can represent and expose. Chapters 6 and 7 examine this question in image decomposition and text generation, where the organization of latent variables is important in addition to sample quality.

1.5.1 Unsupervised image decomposition

Chapter 6, *Unsupervised Foreground Extraction via Deep Region Competition*, develops a generative framework for separating foreground objects from their backgrounds without pixel-level annotations. It combines region-specific generative components, an energy-based prior, and an expectation-maximization-based assignment procedure. Competition between components encourages the model to identify meaningful image regions, demonstrating how probabilistic modeling assumptions can support structural interpretation.

1.5.2 Interpretable text modeling

Chapter 7, *Latent Diffusion Energy-Based Model for Interpretable Text Modeling*, studies structured latent representations for text. It combines diffusion-based learning with latent energy-based modeling and a symbol-vector coupling framework, with additional regularization to encourage interpretable latent organization. The approach addresses sampling difficulties associated with complex latent energy distributions while supporting analysis and control of text generation.

These applications emphasize that useful generative models need not be evaluated only by the realism of their samples. The choice of latent representation and the reliability of its learning and sampling procedures also determine whether meaningful structure can be recovered from data.

1.6 Contributions and Dissertation Organization

The dissertation brings together six research projects organized by their primary questions rather than by publication date. Its contributions fall into three parts:

Part I: Learning neural generative distributions. Chapter 2 develops an NCE objective whose gradient approaches that of maximum likelihood as the effective noise weight increases. Chapter 3 develops diffusion-amortized MCMC for learning latent energy-based priors, addressing the cost and bias of approximate sampling during training.

Part II: Improving diffusion sampling. Chapter 4 investigates learned guidance based on multiple pretrained denoisers. Chapter 5 develops instance-dependent sampling schedules optimized with policy gradients and variance reduction. These methods improve different components of the inference-time sampling procedure.

Part III: Structured generative modeling. Chapter 6 studies unsupervised foreground

extraction through deep region competition, and Chapter 7 develops a latent diffusion energy-based framework for interpretable text modeling. These studies illustrate how structured probabilistic representations support tasks beyond unconstrained sample generation.

Chapter 8 concludes with a synthesis of the methodological connections and remaining challenges. The common perspective of this dissertation is that probabilistic representation, statistical estimation, and computational sampling should be designed with their interactions in mind. Although the individual methods address distinct settings, together they provide complementary ways to make neural generative models more tractable, adaptable, and interpretable.

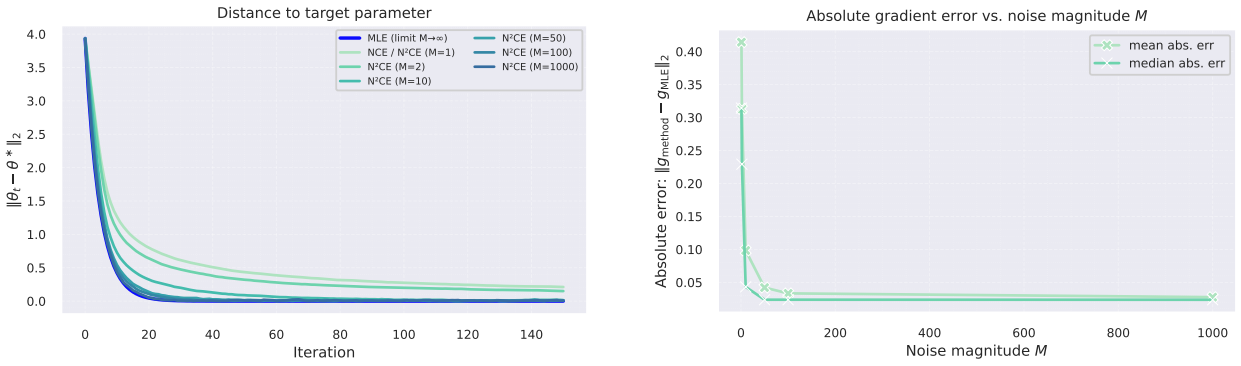
Chapter 2

"Noisier" Noise Contrastive Estimation is (Almost) Maximum Likelihood

2.1 Introduction

Generative modeling of data distributions has made impressive progress in recent years, driven by the development of a rich family of deep generative models [31, 35, 52, 55, 89, 96]. Among existing paradigms, Noise Contrastive Estimation (NCE) [24, 25], also referred to as density–ratio estimation in some literature [99], is a powerful and fundamental framework that unifies generative modeling with discriminative representation learning [9, 32, 34, 85]. NCE reduces density estimation to a classification task: learning the ratio $r(\mathbf{x}) = q_*(\mathbf{x})/q_0(\mathbf{x})$ between a target distribution q_* and a noise distribution q_0 from samples of each, thereby enabling density estimation without explicitly modeling the target density itself [24, 99].

Despite its wide adoption, most existing NCE-based objectives suffer from a fundamental drawback: when the target and noise distributions differ substantially, the neural classifier can achieve near-perfect discrimination accuracy while still providing a poor estimate of the density ratio. This issue, often referred to as the *density-chasm* [87], arises when the gap between the two distributions is large, *e.g.*, when the KL divergence between q_* and



(a) (b)

Figure 2.1: **"Noisier" NCE gradients approach the MLE gradients.** As a sanity check, we simulate the results using 2d Gaussian distributions; *true* MLE gradients can be analytically computed. In Fig. 2.1a, we can see that $M \to \infty$ leads to a trajectory-wise convergence from NCE to MLE. In Fig. 2.1b, as the noise magnitude M increases, "noisier" NCE gradients $\nabla_{\boldsymbol{\alpha}} \mathcal{L}_M$ approach MLE gradients $\nabla_{\boldsymbol{\alpha}} \mathcal{J}^{\text{MLE}}$; gradient bias bounded at order $O(\frac{1}{M})$ under suitable moment conditions (equivalently, squared bias $O(\frac{1}{M^2})$), which is consistent with Proposition 2.3.3. Further details can be found in Sec. 2.11.1.

q_0 exceeds tens of nats. Such situations are common in modern high-dimensional or highly multimodal datasets. Although NCE estimators are asymptotically consistent, meaning they recover the true density ratio in the infinite-sample limit, the convergence rate is provably slow: even an exponential increase in sample size yields only a linear decrease in estimation error [60, 81], and the issue persists even with infinite data [53].

In this chapter, we revisit NCE from a less explored perspective: the magnitude of the noise distribution used for ratio estimation. We show that, under mild conditions, introducing a virtually scaled noise magnitude allows the gradient of the NCE objective to align with that of Maximum Likelihood Estimation (MLE). This observation has two key implications. First, it establishes a principled *gradient-level* connection between NCE and MLE, where the NCE gradient serves as a controlled approximation to the MLE gradient. This perspective highlights that NCE can be understood as approximating the MLE in terms of the optimization trajectory rather than merely matching asymptotic error rates established by [25]. Second, it offers a new lens on the density-chasm problem, revealing that a "noisier" NCE naturally mitigates convergence issues. Building on this insight, we propose N^{2}CE, a simple modification of vanilla NCE whose computational cost depends on the allocation

of negative samples. As we show in Sec. 2.4, this framework proves broadly applicable, demonstrating strong empirical performance in various downstream tasks (see Fig. 2.1 for an illustrative example).

Our main **contributions** are as follows: i) *Theoretically*, we show that, under mild conditions, a "noisier" NCE objective yields gradients that align with those of maximum likelihood, thus bridging the gap between NCE and MLE at the optimization trajectory level. Further, our analysis reveals that a virtually scaled noise magnitude naturally alleviates the convergence issues that arise when the target and noise distributions differ substantially. ii) *Practically*, we propose N^2CE, a drop-in modification to vanilla NCE with a simple weighted classification objective. N^2CE achieves strong empirical performance across diverse tasks including image modeling, anomaly detection, and offline black-box optimization. It produces 10-step and even 1-step samplers that match or surpass state-of-the-art methods on the CIFAR-10 and ImageNet64×64 datasets, with up to half as many reported training iterations in the compared distillation setting.

2.2 Background

Noise Contrastive Estimation (NCE) and Nguyen-Wainwright-Jordan (NWJ) Noise Contrastive Estimation (NCE) [24, 25] offers a discriminative route to training unnormalized statistical models, sidestepping the need for direct likelihood maximization. Let q_* denote the target distribution and q_0 a known noise distribution. Rather than directly modeling $p_{\boldsymbol{\alpha}}$ to approximate q_*, NCE estimates the ratio $r_{\boldsymbol{\alpha}}(\mathbf{x}) = \frac{p_{\boldsymbol{\alpha}}(\mathbf{x})}{q_0(\mathbf{x})}$ by training a classifier to distinguish between samples from q_* and q_0. When the model is correctly specified, the optimal ratio satisfies $r_{\boldsymbol{\alpha}*} \approx q_*/q_0$, which in turn implies $p_{\boldsymbol{\alpha}*} \approx q_*$. In practice, $r_{\boldsymbol{\alpha}}$ is often parameterized as the exponential of a scalar-output neural network $\tilde{f}_{\boldsymbol{\alpha}}$. This parameterization naturally aligns with the energy-based view of $p_{\boldsymbol{\alpha}}$ [11, 50, 111], where $\tilde{f}_{\boldsymbol{\alpha}}$ incorporates the

partition function. A common choice of classification loss is the logistic loss:

$$\mathcal{L}(\boldsymbol{\alpha}) = \mathbb{E}_{\mathbf{x}\sim q_*}\left[\log \frac{r_{\boldsymbol{\alpha}}(\mathbf{x})}{1 + r_{\boldsymbol{\alpha}}(\mathbf{x})}\right] + \mathbb{E}_{\mathbf{x}\sim q_0}\left[\log \frac{1}{1 + r_{\boldsymbol{\alpha}}(\mathbf{x})}\right]. \tag{2.1}$$

Numerous extensions of NCE have been developed, including generalized loss functions [53, 61, 79, 81, 99] and multi-stage ratio estimation strategies [87, 110], which greatly enrich the paradigm. In this work, we revisit NCE from the perspective of the noise distribution's magnitude, showing that virtually scaling the noise magnitude offers a simple, effective remedy to long-standing convergence issues with an implementation-dependent sampling cost.

Another fundamental approach, Nguyen-Wainwright-Jordan (NWJ), addresses the ratio–estimation problem through a variational characterization of f-divergences [70]. This variational view reveals that likelihood–ratio estimation can be performed by maximizing a convex risk functional, yielding M-estimators that are statistically consistent and minimax-optimal under suitable regularity assumptions. Algorithmically, NWJ induces a tractable objective of the form

$$\mathcal{L}^{\mathrm{NWJ}}(\boldsymbol{\alpha}) = \mathbb{E}_{\mathbf{x}\sim q_*}[T_{\boldsymbol{\alpha}}(\mathbf{x})] - \mathbb{E}_{\mathbf{x}\sim q_0}[\exp T_{\boldsymbol{\alpha}}(\mathbf{x})]\,, \tag{2.2}$$

where $T_{\boldsymbol{\alpha}} = \log r_{\boldsymbol{\alpha}}$. Provably, the optimum critic obtained by maximizing Eq. (2.2) retrieves the optimal ratio estimator. In contrast to NCE, which originates from a binary classification surrogate, NWJ is derived directly from convex duality and makes no explicit reference to a discriminative viewpoint. Intriguingly, however, as we demonstrate in Sec. 2.3.4, the noise-scaled NCE objectives studied in this work establish an implicit connection between these ostensibly distinct paradigms. This perspective positions our approach as a unifying family of likelihood-ratio estimators.

Maximum Likelihood Estimation (MLE) Maximum Likelihood Estimation (MLE) remains the generative route: it fits probabilistic models by maximizing $\mathbb{E}_{q_*}[\log p_{\boldsymbol{\alpha}}(\mathbf{x})]$ over

data samples. A common parameterization of $p_{\boldsymbol{\alpha}}$ [11, 50, 111] is

$$p_{\boldsymbol{\alpha}}(\mathbf{x}) := \frac{1}{Z_{\boldsymbol{\alpha}}} \exp\left(f_{\boldsymbol{\alpha}}(\mathbf{x})\right) q_0(\mathbf{x}), \tag{2.3}$$

where $f_{\boldsymbol{\alpha}}$ denotes the unnormalized log-density, $Z_{\boldsymbol{\alpha}}$ is the partition function, and $q_0(\mathbf{x})$ is the base noise distribution. The model in Eq. (2.3) can be interpreted as an energy-based correction or exponential tilting of q_0 [111]. The corresponding gradient takes the general form

$$\nabla_{\boldsymbol{\alpha}} \mathcal{J}^{\mathrm{MLE}}(\boldsymbol{\alpha}) = \mathbb{E}_{q_*}[\nabla_{\boldsymbol{\alpha}} f_{\boldsymbol{\alpha}}(\mathbf{x})] - \mathbb{E}_{p_{\boldsymbol{\alpha}}}[\nabla_{\boldsymbol{\alpha}} f_{\boldsymbol{\alpha}}(\mathbf{x})]. \tag{2.4}$$

The first expectation over the data distribution is straightforward to approximate, but the second requires sampling from $p_{\boldsymbol{\alpha}}$. This is often intractable when the model involves an unknown partition function. In such cases, MCMC methods such as Langevin dynamics [107] are commonly employed. While theoretically valid, MCMC sampling can converge slowly in high-dimensional or multimodal settings, making MLE difficult to apply in practice [72]. This sampling bottleneck is a key motivation for alternatives such as NCE, which avoids direct sampling from $p_{\boldsymbol{\alpha}}$ by reframing density estimation as a classification problem — an idea we revisit from a less-explored perspective in the next section.

2.3 "Noisier" Noise Contrastive Estimation

2.3.1 Noise Magnitude

We now make precise what we mean by the *noise magnitude* in N^2CE, meaning the effective noise-to-data weighting rather than the variance of q_0. Recall the standard objective in Eq. (2.1), where samples from the base distribution q_0 serve as negatives. More generally, we may scale the contribution of q_0 by a positive factor $M > 1$, which we refer to as the *noise*

magnitude. This leads to the following "noisier" NCE objective [25]:

$$\mathcal{L}_M(\boldsymbol{\alpha}) = \mathbb{E}_{q_*(\mathbf{x})}\left[\log \frac{r_{\boldsymbol{\alpha}}(\mathbf{x})}{M + r_{\boldsymbol{\alpha}}(\mathbf{x})}\right] + M\mathbb{E}_{q_0(\mathbf{x})}\left[\log \frac{M}{M + r_{\boldsymbol{\alpha}}(\mathbf{x})}\right]. \tag{2.5}$$

Here, $M = 1$ recovers the standard NCE objective, while larger M corresponds to amplifying the effective weight of the noise distribution. Intuitively, this adjustment can be seen as replacing q_0 with a virtual noise-to-data sample ratio of M: the negative distribution remains q_0, while its contribution to the objective is reweighted. As we show below, increasing M has a striking effect: under mild conditions, the gradient of $\mathcal{L}_M(\boldsymbol{\alpha})$ approaches that of MLE, *i.e.*, Eq. (2.4). This not only establishes a principled bridge between NCE and MLE, but also provides a simple and effective remedy for the convergence issues of standard NCE.

2.3.2 Scaling Effect of Noise Magnitude

Proposition 2.3.1 (Gradient approximation)**.** *Under mild regularity conditions,*

$$\lim_{M\to\infty} \nabla_{\boldsymbol{\alpha}} \mathcal{L}_M(\boldsymbol{\alpha}) = \mathbb{E}_{q_*}[\nabla_{\boldsymbol{\alpha}} f_{\boldsymbol{\alpha}}(\mathbf{x})] - \mathbb{E}_{p_{\boldsymbol{\alpha}}}[\nabla_{\boldsymbol{\alpha}} f_{\boldsymbol{\alpha}}(\mathbf{x})].$$

Sketch of Proof. The gradient of $\mathcal{L}_M$ can be expressed as

$$\nabla_{\boldsymbol{\alpha}} \mathcal{L}_M(\boldsymbol{\alpha}) = \int \frac{M}{M + r_{\boldsymbol{\alpha}}(\mathbf{x})} \big(q_*(\mathbf{x}) - p_{\boldsymbol{\alpha}}(\mathbf{x})\big) \nabla_{\boldsymbol{\alpha}} \log r_{\boldsymbol{\alpha}}(\mathbf{x})\, d\mathbf{x}.$$

This makes clear that, with smooth densities and ratio functions, as M grows, the weight converges to 1 and the gradient approaches the MLE form. Additional derivation details appear in the Sec. 2.9.1.

Remark. The gradient approximation in Proposition 2.3.1 concerns normalized densities. The optimization theorem below instead uses the *extended* exponential-family parameterization, in which the log-normalizing constant is treated as an independent coordinate. For this parameterization the negative population N^2CE objective is globally convex. Following the

normalized-gradient framework of Liu et al. [53], we establish a global iteration bound from arbitrary feasible initializations under local exponential-family regularity and sufficiently large but finite noise weight M.

Theorem 2.3.2 (Global normalized-gradient convergence for large finite M)**.** *Let*

$$p_{\boldsymbol{\tau}}(\mathbf{x}) = h(\mathbf{x}) \exp\{\boldsymbol{\theta}^\top S(\mathbf{x}) - a\}, \qquad \boldsymbol{\tau} = (\boldsymbol{\theta}, a), \qquad T(\mathbf{x}) = (S(\mathbf{x}), -1),$$

be an extended exponential-family model, where $p_{\boldsymbol{\tau}}$ need not be normalized away from the optimum. Let $A(\boldsymbol{\theta}) = \log \int h(\mathbf{x}) e^{\boldsymbol{\theta}^\top S(\mathbf{x})} d\mathbf{x}$, and $p_{\boldsymbol{\theta}} = h e^{\boldsymbol{\theta}^\top S - A(\boldsymbol{\theta})}$. Assume the following:

1. ***Correct specification and coverage:*** *$q_* = p_{\boldsymbol{\theta}^*} = p_{\boldsymbol{\tau}^*}$, where $\boldsymbol{\tau}^* = (\boldsymbol{\theta}^*, A(\boldsymbol{\theta}^*))$; $q_0(\mathbf{x}) > 0$ wherever $h(\mathbf{x}) > 0$.*

2. ***A regular parameter neighborhood:*** *$\boldsymbol{\theta}^*$ is in the interior of the natural parameter domain $\mathcal{N} = \{\boldsymbol{\theta} : A(\boldsymbol{\theta}) < \infty\}$. There exist $r > 0$, $\beta_Z < \infty$, and $0 < \lambda_{\min} \leqslant \lambda_{\max} < \infty$ such that the $\boldsymbol{\theta}$-projection of a neighborhood of the closed ball $B_r(\boldsymbol{\tau}^*)$ lies in $\mathcal{N}$, A is β_Z-Lipschitz there, and*

$$\lambda_{\min} I \preceq F(\boldsymbol{\theta}) := \mathbb{E}_{p_{\boldsymbol{\theta}}}[T(\mathbf{x}) T(\mathbf{x})^\top] \preceq \lambda_{\max} I.$$

   Take $r \leqslant (1 + \beta_Z)^{-1}$.

3. ***Feasible iterations:*** *$\mathcal{C}$ is a closed convex subset of the extended natural parameter domain containing $B_r(\boldsymbol{\tau}^*)$ and the initialization $\boldsymbol{\tau}_0$. A projection onto $\mathcal{C}$ is available.*

Let $F_M = -\mathcal{L}_M$ be the population *negative log-contrastive objective, with $r_{\boldsymbol{\tau}} = p_{\boldsymbol{\tau}}/q_0$ in Eq.* (2.5)*. There exists $M_0 < \infty$ (possibly depending on q_*, q_0 and the neighborhood) such that for every fixed $M \geqslant M_0$,*

$$\mu I \preceq \nabla^2 F_M(\boldsymbol{\tau}) \preceq L I \quad (\boldsymbol{\tau} \in B_r(\boldsymbol{\tau}^*)), \qquad \mu = \frac{\lambda_{\min}}{2e}, \quad L = (e+1)\lambda_{\max}.$$

For any $0 < \delta \leqslant r$, projected normalized gradient descent,

$$\boldsymbol{\tau}_{t+1} = \Pi_{\mathcal{C}}\left(\boldsymbol{\tau}_t - \eta \frac{\nabla F_M(\boldsymbol{\tau}_t)}{\|\nabla F_M(\boldsymbol{\tau}_t)\|_2}\right), \qquad \eta = \delta\sqrt{\mu/L},$$

(where the algorithm stops if the gradient vanishes) reaches an iterate $\|\boldsymbol{\tau}_t - \boldsymbol{\tau}^*\|_2 \leqslant \delta$ within

$$\boxed{T \leqslant \left\lceil 2e(e+1)\frac{\lambda_{\max}}{\lambda_{\min}}\frac{\|\boldsymbol{\tau}_0 - \boldsymbol{\tau}^*\|_2^2}{\delta^2}\right\rceil}$$

iterations. In particular, this is an $O(\kappa_F\|\boldsymbol{\tau}_0 - \boldsymbol{\tau}^*\|_2^2/\delta^2)$ global iteration bound *in the feasible set, with $\kappa_F = \lambda_{\max}/\lambda_{\min}$. If the entire extended natural parameter domain is Euclidean space, projection can be omitted.*

Scope and comparison. The theorem gives a *global population-optimization guarantee* from any feasible initialization, even though its quantitative Hessian bounds are required only near $\boldsymbol{\tau}^*$. Its proof uses global convexity and a normalized gradient step, not ordinary gradient ascent or stochastic gradient descent. Unlike exponential-loss NCE (eNCE) in Liu et al. [53], N^2CE retains the logistic classification objective and increases the effective noise-to-data weighting. Our direct neighborhood-curvature argument gives a linear dependence on the stated local Fisher-type condition number; this does not imply a uniformly smaller computational cost than eNCE under different assumptions or implementations. Critically, the threshold M_0 is not shown to scale polynomially with the data–noise distributional gap, and drawing Mn negative samples per iteration can itself be expensive. The result does not establish a global complexity bound for finite-sample optimization, nonconvex neural parameterizations, or arbitrary fixed M. A complete proof appears in Sec. 2.9.3.

2.3.3 Practical Error Analysis, N^2CE Family and Regularization

The population results above assume exact expectations and arbitrarily large M. In practice, both M and the sample sizes are finite. We therefore ask: how well does the empirical

"noisier" NCE gradient approximate the MLE gradient under these practical constraints? Proposition 2.3.3 provides an error decomposition that makes this trade-off explicit:

Proposition 2.3.3 (Finite-M and finite-sample gradient error)**.** *Fix a parameter $\boldsymbol{\alpha}$ and write $g_{\boldsymbol{\alpha}}(\mathbf{x}) = \nabla_{\boldsymbol{\alpha}} \log r_{\boldsymbol{\alpha}}(\mathbf{x})$. Let $\widehat{\mathcal{L}}_M$ use n_* independent samples from q_* and n_0 independent samples from q_0, with each expectation approximated by its own sample average. Assume the relevant second moments are finite and*

$$C_b(\boldsymbol{\alpha}) := \mathbb{E}_{q_*}[r_{\boldsymbol{\alpha}} \|g_{\boldsymbol{\alpha}}\|_2] + \mathbb{E}_{p_{\boldsymbol{\alpha}}}[r_{\boldsymbol{\alpha}} \|g_{\boldsymbol{\alpha}}\|_2] < \infty.$$

For the normalized *model $r_{\boldsymbol{\alpha}} = p_{\boldsymbol{\alpha}}/q_0$, the mean-squared error relative to the maximum-likelihood gradient obeys*

$$\begin{aligned}
&\mathbb{E}\big\|\nabla_{\boldsymbol{\alpha}} \widehat{\mathcal{L}}_M(\boldsymbol{\alpha}) - \nabla_{\boldsymbol{\alpha}} \mathcal{J}^{\mathrm{MLE}}(\boldsymbol{\alpha})\big\|_2^2 \\
&\quad \leqslant \frac{C_b(\boldsymbol{\alpha})^2}{M^2} + \frac{1}{n_*} \mathbb{E}_{q_*} \|g_{\boldsymbol{\alpha}}\|_2^2 \\
&\qquad + \frac{1}{n_0} \min\big\{ M^2 \mathbb{E}_{q_0} \|g_{\boldsymbol{\alpha}}\|_2^2,\ \mathbb{E}_{q_0} \|\nabla_{\boldsymbol{\alpha}} r_{\boldsymbol{\alpha}}\|_2^2 \big\}.
\end{aligned}$$

Remark. The first term is a bound on the *squared* gradient bias, $O(M^{-2})$; the unsquared bias is at most $O(M^{-1})$. The remaining terms bound Monte Carlo variance and depend on the actual number of positive and negative samples. Under a fixed $n_* = n_0 = n$ design, the growing branch of this upper bound has the form $aM^2/n + b/M^2$ and is balanced at $M \asymp n^{1/4}$, not at $\sqrt{n}$. If instead $n_0 = Mn_*$ as in some of our implementations, the sample-cost and variance scaling are different. The minimum in the variance bound also permits saturation, so the bound alone does not guarantee a U-shaped empirical risk curve or a unique finite optimum. See Sec. 2.9.2 for a derivation.

The N^{2}CE family and empirical U-shaped patterns In controlled 5-dimensional Gaussian experiments and high-dimensional neural ablations we observe that intermediate values of M often give better finite-sample performance than either vanilla NCE or extremely

large values of M (Sec. 2.11.1 and Tabs. 2.16, 2.17 and 2.26). These observations are consistent with a bias–variance trade-off, but they do not imply a universal scaling law for the optimal M. In practice, we select M using a validation sweep, while recognizing that the most useful range depends on model, dimensionality, sample allocation and computational budget. We next describe two approaches to stabilizing the estimated ratios.

Multi-stage ratio estimation One way to stabilize the gradient approximation is to adopt the multi-stage estimation strategy of Rhodes et al. [87]. This is especially useful when (i) a convenient base noise distribution is available, (ii) we prefer to use the density model directly as a discriminator or for decision making rather than as a smooth reward, and (iii) the problem scale is low- or moderately high-dimensional. The key idea is to decompose the ratio between q_* and q_0 into a telescoping product, $\frac{q_*}{q_0} = \frac{q_*}{q_K}\frac{q_K}{q_{K-1}}\cdots\frac{q_1}{q_0}$, where $\{q_k\}_{k=1}^K$ are *pre-specified* intermediate distributions. Each ratio now involves a pair (q_{k+1}, q_k) with greater overlap by design, yielding smaller values and more controllable variance at each stage. The trade-off is computational: in high-dimensional settings many stages may be needed, increasing overhead. For this reason, we mainly apply this technique in lower-dimensional tasks such as latent-space modeling.

Direct ratio regularization A more general and convenient strategy is to add a penalty on the ratio itself, for example $\mathbb{E}\|\log r_{\boldsymbol{\alpha}}\|_2^2$, directly to the "noisier" NCE objective. This penalty controls the magnitude of the log-ratio on the chosen evaluation distribution; by itself, it does not bound input-space derivatives or establish smoothness of the density ratio. It is broadly applicable and does not require intermediate distributions, but may introduce bias in the learned ratio and its gradients. Nonetheless, as we show in Sec. 2.4, this regularization is highly effective in practice, particularly for training reward models or critics in high-dimensional settings such as on ImageNet64×64 datasets.

2.3.4 An Information-Theoretic Perspective

The family of objectives also connects variational representations of Jensen–Shannon and Kullback–Leibler divergences. For $T = \log r$, the Nguyen–Wainwright–Jordan representation is

$$D_{\mathrm{KL}}(q_*\|q_0) = 1 + \sup_T \left\{\mathbb{E}_{q_*}[T] - \mathbb{E}_{q_0}[e^T]\right\} = 1 + \sup_r \left\{\mathbb{E}_{q_*}[\log r] - \mathbb{E}_{q_0}[r]\right\}. \tag{2.6}$$

For the equal-weight Jensen–Shannon divergence, the normalization of our logistic objective in Eq. (2.1) gives

$$D_{\mathrm{JS}}(q_*\|q_0) = \log 2 + \frac{1}{2}\sup_r \left\{\mathbb{E}_{q_*}\left[\log \frac{r}{1+r}\right] + \mathbb{E}_{q_0}\left[\log \frac{1}{1+r}\right]\right\}. \tag{2.7}$$

The optimum in both variational representations is attained at $r^* = q_*/q_0$ when the ratio is included in the variational family.

Interpolating through a weighted Jensen–Shannon divergence. Let $\alpha = M/(1+M)$ and $m_\alpha = (1-\alpha)q_* + \alpha q_0$. Define

$$D_\alpha(q_*\|q_0) = (1-\alpha)D_{\mathrm{KL}}(q_*\|m_\alpha) + \alpha D_{\mathrm{KL}}(q_0\|m_\alpha).$$

Writing $h(\alpha) = -\alpha\log\alpha - (1-\alpha)\log(1-\alpha)$, the exact relation with Eq. (2.5) is

$$D_\alpha(q_*\|q_0) = h(\alpha) + \frac{1}{M+1}\sup_r \mathcal{L}_M(r). \tag{2.8}$$

At $M = 1$ this recovers the Jensen–Shannon representation. However, $D_\alpha \to 0$, not $D_{\mathrm{KL}}(q_*\|q_0)$, as $\alpha \to 1$. Under suitable integrability conditions the correct rescaled limit is

$$\lim_{M\to\infty}(M+1)D_{M/(1+M)}(q_*\|q_0) = D_{\mathrm{KL}}(q_*\|q_0).$$

Equivalently, for a fixed positive ratio r with sufficient integrability,

$$\lim_{M\to\infty} \big(\mathcal{L}_M(r) + \log M \big) = \mathbb{E}_{q_*}[\log r] - \mathbb{E}_{q_0}[r],$$

which is the NWJ objective. The divergent $-\log M$ contribution is constant in the parameters and therefore does not affect optimization. Thus the population objective interpolates between logistic ratio estimation and a KL variational objective, consistent with the MLE-gradient limit in Proposition 2.3.1.

2.4 Experiments

We organize our empirical study around three main questions: (i) How does the proposed objective compare with standard baselines such as pure MLE and vanilla NCE? (ii) Do the advantages of "noisier" NCE transfer to downstream tasks? (iii) How do key hyperparameters, in particular the noise magnitude M, affect performance?

To answer these, we conduct experiments across a broad range of datasets and benchmarks, spanning diverse model families. The results consistently show that our method outperforms existing approaches, while providing new insights into the role of noise scaling. Full experimental settings, implementation details and baseline descriptions are provided in Secs. 2.10 and 2.11.

2.4.1 Learning LEBM with N^2CE

As a proof of concept, we first conduct a set of lightweight experiments on latent energy-based models (LEBMs) across image datasets including CIFAR-10 [44], MNIST [49], SVHN [67], CelebA64 [56], and CelebAMask-HQ [51]. Learning an LEBM [74, 117] can be viewed as fitting an unnormalized density model in the latent space of a VAE (Sec. 2.10.1). Prior approaches largely rely on MCMC-based MLE-style training [12, 74], where convergence and stability are often problematic. This makes LEBMs a natural and tractable testbed

Table 2.1: **FID(↓) on different datasets**. We highlight our model, the **1**st and 2nd performances; tables henceforth follow this format. Numbers from the first six rows are from Yu et al. [118]. `nz` denotes the latent dimension. `M` and `K` denote the noise magnitude and num. of stages for ratio estimation, respectively.

Model	SVHN `nz=100`	CelebA `nz=100`	CIFAR10 `nz=128`	CelebAHQ `nz=512`
ABP	49.71	51.50	90.30	160.21
ABP-LEBM*	29.44	37.87	**70.15**	133.07
SRI	44.86	61.03	-	-
SRI (L=5)	35.32	47.95	-	-
2s-VAE	42.81	44.40	72.90	-
RAE	40.02	40.95	74.16	-
VAE	34.81	47.84	110.37	154.60
w/ MLE-LEBM	32.74	40.24	90.54	111.11
w/ NCE-LEBM	30.71	39.61	92.83	118.84
w/ N^{2}CE-LEBM				
`M=100,K=1`	26.84	33.05	77.35	101.71
`M=100,K=3`	**25.63**	**31.09**	77.05	**95.66**

Table 2.2: **AUPRC(↑) scores for unsupervised anomaly detection on MNIST**. Baseline numbers are taken from Yoon et al. [114], Yu et al. [118]. Full results with variances in found in Sec. 2.11.3.

Heldout Digit	1	4	5	7	9
AE	0.062	0.204	0.259	0.125	0.113
VAE	0.063	0.337	0.325	0.148	0.104
ABP	0.095	0.138	0.147	0.138	0.102
IGEBM	0.101	0.106	0.205	0.100	0.079
MEG	0.281	0.401	0.402	0.290	0.342
BiGAN-σ	0.287	0.443	0.514	0.347	0.307
ABP-LEBM	0.336	0.630	0.619	0.463	0.413
JVAEBM	0.297	0.723	0.676	0.490	0.383
Adaptive CE	0.531	0.729	0.742	0.620	0.499
NAE	0.802	0.648	0.716	0.789	0.441
MPDR-S	0.764	0.823	0.741	**0.857**	0.478
MPDR-R	0.844	0.711	0.757	0.850	0.569
DAMC	0.684	0.911	0.939	0.801	0.705
DAMC-NCE	0.702	0.829	0.764	0.605	0.502
DAMC-N^{2}CE					
`M=100,K=1`	0.910	0.911	0.935	0.779	0.699
`M=100,K=3`	**0.959**	**0.935**	**0.959**	0.845	**0.854**

for prototyping our method. In these experiments, we apply ratio decomposition as a regularization (Sec. 2.3.3) and follow the training and evaluation protocols in Pang et al. [74] (see Sec. 2.10.2).

Generative image modeling We evaluate the quality of the learned LEBM with generated images measured by FID scores [30] (see Tab. 2.1). For image generation with LEBM, we perform 100-step short-run LD to draw latent vectors from the learned models, and map the latent vectors to the image space with the decoder network.

We observe that: first, LEBMs trained with N^{2}CE (*i.e.*, rows with `M=100`) consistently outperform those trained with vanilla NCE, with multi-stage estimation providing further gains that validate our analysis. Second, LEBMs trained with N^{2}CE show substantial improvements over those trained with MLE and short-run LD (denoted MLE-LEBM). These results confirm that N^{2}CE is a more reliable and effective objective that addresses the issues inherent to vanilla NCE and MCMC-based MLE. Finally, the improvement becomes more pronounced as the latent dimension `nz` increases, especially on the CelebAMask-HQ dataset

with `nz=512`. This suggests that our proposal scales more gracefully to high-dimensional, multimodal targets.

Anomaly Detection To further highlight the practical advantages of our approach in highly multimodal settings, we evaluate anomaly detection on MNIST following the setup in Zenati et al. [121]. Specifically, we train models with one digit held out as anomalous, focusing on digits 1, 4, 5, 7, and 9—cases known to be especially challenging. We build on DAMC [118] for posterior inference, replacing its original prior with an LEBM learned via N^2CE. Since empirical posteriors from DAMC are sharp and highly multimodal, this serves as a demanding testbed. Table 2.2 reports AUPRC scores averaged over 10 trials. Our method yields consistent and often substantial improvements over baselines, achieving a strong overall performance across the hardest digits. Full experimental details appear in Secs. 2.10.2 and 2.11.3.

2.4.2 Reward and Critic Learning for Diffusion Distillation

Building on the success of latent-space modeling, we next consider more challenging high-dimensional settings in the ambient image space. In particular, we study two representative scenarios for distilling diffusion samplers: (i) learning energy-based rewards for diffusion fine-tuning (DxMI) [115], and (ii) learning critics for adversarial distillation (SiD^{2}A) [123]. In both cases, we simply replace the original objectives with our "noisier" NCE combined with direct ratio regularization (Sec. 2.3.3), while using the same architectures and critical hyperparameters as baseline methods. Full implementation details appear in Sec. 2.10.3.

In the DxMI setup [115], N^2CE yields substantial improvements over vanilla NCE-like variants, previous methods and even the teacher models with full NFE (Tab. 2.3), underscoring its effectiveness in training rewards for diffusion distillation. In the SiD^{2}A setting [123], our method greatly outperforms vanilla NCE variants, and not only matches but can surpass strong baselines (Tabs. 2.4 and 2.5), notably with up to *half the training iterations*. Together, these results confirm that N^2CE scales to high-dimensional tasks while improving training

Table 2.3: **CIFAR-10 (DDPM backbone) and ImageNet64×64 (EDM backbone) results** shown side-by-side. First six rows are from [115]. † highlights the starting point of DxMI fine-tuning.

– **Backbone: DDPM**				– **Backbone: EDM**				
Method	NFE	FID (↓)	Rec. (↑)	Method	NFE	FID (↓)	Prec. (↑)	Rec. (↑)
DDPM	1000	3.21	0.57	EDM (Heun)	79	<u>2.44</u>	0.71	0.67
FastDPM†	10	35.85	0.29	EDM (Ancestral)†	10	50.27	0.37	0.35
DDIM	10	13.36	–	Consistency Model	2	4.70	0.69	0.64
SFT-PG	10	4.82	0.606	Consistency Model	1	6.20	0.68	0.63
DxMI	10	3.19	0.625	DxMI	10	2.68	<u>0.777</u>	0.574
DxMI + Value Guidance	10	<u>3.17</u>	<u>0.623</u>	DxMI + Value Guidance	10	2.67	**0.780**	0.574
DxMI + NCE (`M=1`)	10	3.93	0.623	DxMI + NCE (`M=1`)	10	2.69	0.756	<u>0.585</u>
DxMI + N^{2}CE (`M=100`)	10	**2.99**	**0.638**	DxMI + N^{2}CE (`M=100`)	10	**2.23**	0.757	**0.599**

efficiency across distinct distillation regimes. We provide additional ablation studies on M on CIFAR-10 in Sec. 2.11.4.

2.4.3 Offline Black-Box Optimization

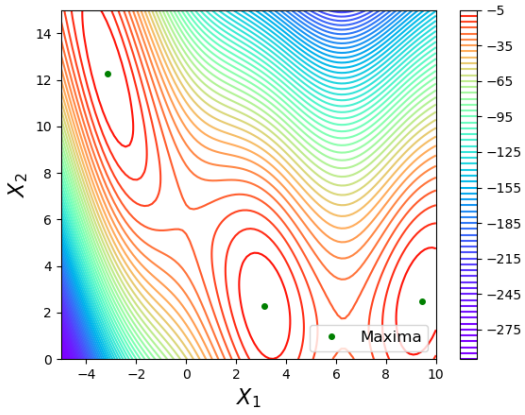
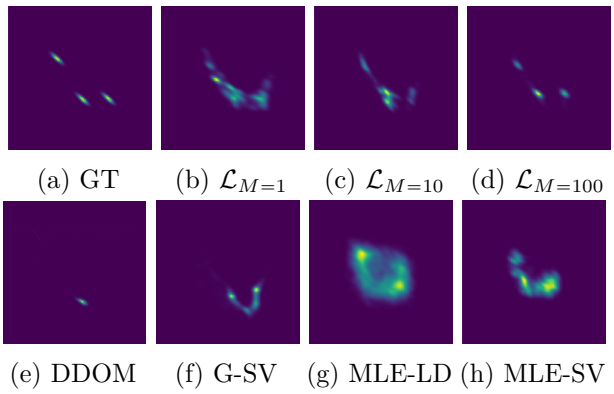


Figure 2.2: **Branin function level sets.**



(a) GT (b) $\mathcal{L}_{M=1}$ (c) $\mathcal{L}_{M=10}$ (d) $\mathcal{L}_{M=100}$

(e) DDOM (f) G-SV (g) MLE-LD (h) MLE-SV

Figure 2.3: **Viz. of Branin optimal samples.** (b–d) are results of our method. G-SV denotes the Gaussian prior model sampled with SVGD. MLE-LD and MLE-SV denote the model trained by MLE sampled with LD and SVGD, respectively.

Beyond image modeling, we also explore the broader impact of our proposed technique through the lens of offline BBO. This task evaluates not only how well a model captures the training distribution, but also its ability to internalize structural regularities and generalize to unseen queries. In the *offline* BBO setting [102], one is given a pre-collected dataset $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$ of inputs and their black-box function values $h(\mathbf{x}_i) = y_i$. At test time,

Table 2.4: **CIFAR-10 results**. "FID-U/C (iters)" shows uncond./cond. FID and, when provided, corresponding training iterations in parentheses. Baseline numbers from [122, 123].

Method	NFE	FID-U ↓	IS ↑	FID-C ↓
Training From Scratch				
DDPM	1000	3.17	–	–
DDIM	100	4.16	–	–
Score SDE	2000	–	–	2.20
DPM-Solve-3	48	–	–	2.65
EDM	35	1.98	–	1.79
BigGAN	1	–	–	14.73
StyleGAN2-ADA	1	2.92	9.82	2.42
SAN	1	**1.36**	–	–
iCT	1	2.83	9.54	–
iCT	2	2.46	9.80	–
iCT-deep	1	2.51	9.76	–
iCT-deep	2	2.24	9.89	–
Post-training				
PD	1	9.12	–	–
DFNO	1	3.78	–	–
CD	1	3.55	–	–
CD	2	2.93	–	–
CTM	1	1.98	–	1.73
CTM	2	1.87	–	1.63
DMD	1	2.62	–	–
D2O-F	1	1.54	10.10	1.44
SiD	1	1.92	9.98	1.71
SiD^{2}A	1	1.50 *(30K)*	10.19	1.40 *(50K)*
SiD + NCE (`M=1`)	1	1.53 *(30K)*	<u>10.20</u>	1.46 *(30K)*
SiD + N^{2}CE (`M=50`)	1	**1.45** ***(20K)***	**10.23**	**1.39 (20K)**

Table 2.5: **Conditional ImageNet 64×64 results.**

Method	NFE	FID ↓ (iters)	Prec. ↑	Rec. ↑
Training From Scratch				
RIN	1000	1.23	–	–
DDPM	250	11.00	0.67	0.58
ADM	250	2.07	0.74	0.63
EDM	79	2.64	–	–
iCT	1	4.02	0.70	0.63
iCT	2	3.20	0.73	0.63
iCT-deep	1	3.25	0.72	0.63
iCT-deep	2	2.77	0.74	0.62
BigGAN-deep	1	4.06	**0.79**	0.48
StyleGAN2-XL	1	1.51	–	–
Post-training				
PD	1	15.39	–	–
BOOT	1	16.3	0.68	0.36
DFNO	1	7.83	–	0.61
CD	1	6.20	0.68	0.63
CD	2	4.79	0.69	**0.64**
CTM	1	1.92	0.70	0.57
CTM	2	1.73	0.64	0.57
DMD	1	2.62	–	–
sCD-S	1	2.97	–	–
sCD-S	2	2.07	–	–
DMD2	1	1.51	–	–
MSD (DM)	1	2.37	–	–
MSD (ADM)	1	1.20	–	–
D2O-F	1	<u>1.16</u>	<u>0.75</u>	0.60
SiD	1	1.52	0.74	<u>0.63</u>
SiD^{2}A	1	**1.11** ***(20K)***	<u>0.75</u>	0.62
SiD + NCE (`M=1`)	1	1.28 *(15K)*	<u>0.75</u>	0.62
SiD + N^{2}CE (`M=50`)	1	**1.11** ***(10K)***	<u>0.75</u>	<u>0.63</u>

the optimizer may access a limited budget Q of evaluations of the unknown function h and must return candidates with high observed values. This setup is particularly well-suited for assessing generative approaches to optimization, where the function value y acts as a conditioning signal [5, 43, 45].

We instantiate this view by parameterizing a conditional model $p_{\boldsymbol{\theta}}(\mathbf{x} \mid y)$ via latent variables $\mathbf{z}$, using

$$p_{\boldsymbol{\theta}}(\mathbf{x} \mid y) \propto \mathbb{E}_{p(\mathbf{z}|y)}\big[p_{\boldsymbol{\beta},\mathbf{x}}(\mathbf{x} \mid \mathbf{z})\big], \quad p(\mathbf{z} \mid y) \propto p_{\boldsymbol{\beta},y}(y \mid \mathbf{z})\, p_{\boldsymbol{\alpha}}(\mathbf{z}),$$

and employing stochastic samplers such as Langevin Dynamics (LD) or Stein Variational Gradient Descent (SVGD) [54] for conditional sampling. To train this model, we use a VAE

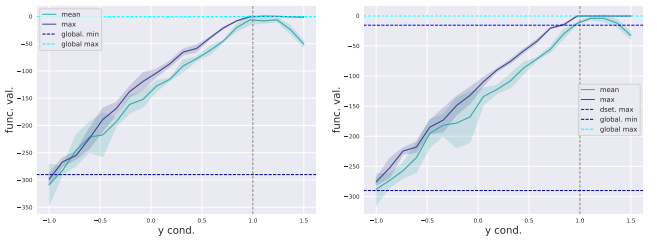
with an LEBM prior that jointly models $(\mathbf{x}, y)$. Full problem statement, implementation details, and extended results are deferred to Sec. 2.11.5.

2.4.3.1 2D Branin Function

We begin with the 2D Branin function (Fig. 2.2) as a proof of concept. This task serves as a sanity check to validate that our method can both faithfully capture high-value modes and generalize beyond the best points observed in the offline dataset (see Sec. 2.11.6 for details). Following DDOM [43], we perform a held-out offline optimization experiment. Specifically, we uniformly sample $N = 5000$ points from the domain $[-5, 10] \times [0, 15]$ to construct the offline dataset, then remove the top 10% by function value. This ensures that high-value contours are retained, but the exact global optima are excluded during training. For evaluation, we adopt a query budget $Q = 128$ and use SVGD for sampling. Table 2.6 shows that our method generalizes well beyond the maximum value in the dataset $\mathcal{D}_{\max}$, approaching the true optimum and outperforming gradient-based GA as well as modern generative baselines BONET [59] and DDOM [43].

Table 2.6: **Results on the top-10%-tile-removed Branin task (avg. over 5 runs).** OPT denotes the global optimum.

$\mathcal{D}_{\max}$/Opt.	GA	BONET	DDOM	Ours
-6.1/-0.4	$-4.0_{\pm 4.3}$	$-1.8_{\pm 0.8}$	$\underline{-1.6_{\pm 0.1}}$	$\mathbf{-0.4}_{\pm 0.1}$



(a) w/ top-10% points (b) w/o top-10% points

Figure 2.4: **Results on uniformly sampled Branin w/ and w/o top-10% points.** Zoom in for details.

We further visualize conditional samples corresponding to high function values (Fig. 2.3), and examine the correlation between target conditions (desired function values) and achieved function values (Fig. 2.4). For visualization, we draw 2560 samples from each model. In Fig. 2.3a, the distribution of true optima is emulated with a 3-component GMM centered at the global minima. As shown in the first row of Fig. 2.3, increasing M in Eq. (2.5) produces samples that better approximate the ground-truth distribution, while models trained with

MCMC-based MLE or vanilla NCE (`M=1`) lag behind. This again supports our analysis that a sufficiently large M is critical for accurate gradient approximation.

Table 2.7: **Normalized results on design-bench with $Q = 256$ (avg. over 5 runs).**

BASELINE	TFBIND8	TFBIND10	CHEMBL	SUPERCON.	ANT	D'KITTY	MEAN SCORE↑	MEAN RANK↓
$\mathcal{D}$ (best)	0.439	0.467	0.605	0.399	0.565	0.884	-	-
GP-qEI	0.824 ± 0.086	0.635 ± 0.011	0.633 ± 0.000	0.501 ± 0.021	0.887 ± 0.000	0.896 ± 0.000	0.729 ± 0.019	7.8
CMA-ES	0.933 ± 0.035	0.679 ± 0.034	0.636 ± 0.004	0.491 ± 0.004	$\mathbf{1.436 \pm 0.928}$	0.725 ± 0.002	$\underline{0.816 \pm 0.168}$	5.8
REINFORCE	0.959 ± 0.013	0.640 ± 0.028	0.636 ± 0.023	0.481 ± 0.017	0.261 ± 0.042	0.474 ± 0.202	0.575 ± 0.054	8.3
Gradient Ascent	$\underline{0.981 \pm 0.015}$	0.659 ± 0.039	0.647 ± 0.020	0.504 ± 0.005	0.340 ± 0.034	0.906 ± 0.017	0.672 ± 0.021	5.0
COMs	0.964 ± 0.020	0.654 ± 0.020	0.648 ± 0.005	0.423 ± 0.033	0.949 ± 0.021	0.948 ± 0.006	0.764 ± 0.018	5.8
BONET	0.975 ± 0.004	0.681 ± 0.035	$\underline{0.654 \pm 0.019}$	0.437 ± 0.022	0.976 ± 0.012	$\underline{0.954 \pm 0.012}$	0.780 ± 0.022	$\underline{3.7}$
CbAS	0.958 ± 0.018	0.657 ± 0.017	0.640 ± 0.005	0.450 ± 0.083	0.876 ± 0.015	0.896 ± 0.016	0.746 ± 0.003	7.3
MINs	0.938 ± 0.047	0.659 ± 0.044	0.653 ± 0.002	0.484 ± 0.017	0.942 ± 0.018	0.944 ± 0.009	0.770 ± 0.023	5.5
DDOM	0.971 ± 0.005	$\underline{0.688 \pm 0.092}$	0.633 ± 0.007	$\underline{0.560 \pm 0.044}$	0.957 ± 0.012	0.926 ± 0.009	0.787 ± 0.034	4.5
Ours	$\mathbf{0.990 \pm 0.003}$	$\mathbf{0.803 \pm 0.085}$	$\mathbf{0.661 \pm 0.025}$	$\mathbf{0.567 \pm 0.017}$	$\underline{0.982 \pm 0.012}$	$\mathbf{0.961 \pm 0.006}$	$\mathbf{0.827 \pm 0.021}$	**1.2**

Table 2.8: **Ablation studies on design-bench (avg. over 5 runs)**, results with a budget $Q = 256$.

BASELINE	TFBIND8	TFBIND10	CHEMBL	SUPERCON.	ANT	D'KITTY	MEAN SCORE↑
$\mathcal{D}$ (best)	0.439	0.467	0.605	0.399	0.565	0.884	-
LD sampler							
MLE-LEBM	0.524 ± 0.228	0.667 ± 0.000	0.633 ± 0.000	0.362 ± 0.006	0.512 ± 0.000	0.672 ± 0.049	0.562 ± 0.033
N^2CE-LEBM (`M=100,K=6`)	0.834 ± 0.073	$\underline{0.739 \pm 0.095}$	0.639 ± 0.013	0.363 ± 0.006	$\underline{0.957 \pm 0.023}$	0.955 ± 0.004	0.748 ± 0.025
SVGD sampler							
MLE-LEBM	0.941 ± 0.012	0.681 ± 0.079	0.634 ± 0.002	0.296 ± 0.004	0.926 ± 0.016	0.915 ± 0.007	0.732 ± 0.014
NCE-LEBM (`M=1,K=6`)	$\underline{0.961 \pm 0.024}$	0.655 ± 0.034	0.640 ± 0.011	$\underline{0.497 \pm 0.044}$	0.907 ± 0.007	$\underline{0.959 \pm 0.002}$	0.770 ± 0.007
N^2CE-LEBM (`M=100,K=1`)	0.945 ± 0.025	0.722 ± 0.043	0.648 ± 0.003	0.421 ± 0.020	0.893 ± 0.008	0.953 ± 0.008	0.764 ± 0.005
N^2CE-LEBM (`M=100,K=6`)	$\mathbf{0.990 \pm 0.003}$	$\mathbf{0.803 \pm 0.085}$	$\mathbf{0.661 \pm 0.025}$	$\mathbf{0.567 \pm 0.017}$	$\mathbf{0.982 \pm 0.012}$	$\mathbf{0.961 \pm 0.006}$	$\mathbf{0.827 \pm 0.021}$

2.4.3.2 Offline BBO on Design-Bench

Task setup and evaluation. We next evaluate on higher-dimensional real-world tasks[1] from Trabucco et al. [102], grouped into **discrete optimization** (TF-Bind-8, TF-Bind-10, ChEMBL) and **continuous optimization** (D'Kitty, Ant Morphology, Superconductor). We compare against three categories of baselines: **(i)** generative inverse models with different parameterizations, including CbAS [5], Auto.CbAS [14], MIN [45], and DDOM [43]; **(ii)** gradient(-like) updating from existing designs, such as gradient-ascent-based methods [7, 8, 15, 83, 101, 119, 120] and BONET [59]; **(iii)** additional baselines including REINFORCE and evolutionary algorithm reported in Trabucco et al. [102]. Further details are given in Secs. 2.11.7.1 and 2.11.7.3.

[1]NAS and Hopper tasks are excluded following [59]; see Sec. 2.11.7.2.

Following standard practice [102], we report normalized ground-truth function values $y_n = \frac{y - y^*_{\min}}{y^*_{\max} - y^*_{\min}}$, where $y^*_{\min}$ and $y^*_{\max}$ are the minimum and maximum values in the full dataset (unseen during training). We summarize performance using both mean scores and mean normalized ranks (MNR) across tasks. We provide two sets of results: (1) $Q = 256$ queries following Krishnamoorthy et al. [43]; (2) $Q = 128$ queries following Chen et al. [8], Trabucco et al. [101], excluding ChEMBL. In the main text we report 100-th percentile results with $Q = 256$, and defer $Q = 128$ results and additional percentiles (e.g., 50-th) to Sec. 2.11.7.4.

Main results. Tables 2.7 and 2.23 show that our method attains the best average ranks of 1.2 ($Q = 256$) and 2.0 ($Q = 128$), compared to runner-ups BONET [59] and Tri-mentoring [8], which achieve 3.7 and 3.0, respectively, in the corresponding tables. Our approach consistently yields the highest normalized mean values across tasks, outperforming all baselines—particularly generative inverse models—and confirming the effectiveness of our design. With $Q = 256$, we obtain the best performance on five of six tasks, with the sole exception being **ANT**. Notably, the best-performing baseline on this task, CMA-ES, exhibits a much higher variance (std. 0.928) compared to ours (0.012), underscoring its instability to initialization. With $Q = 128$, we achieve the best results on three out of five tasks. While ExPT [68] achieves competitive results on the remaining tasks, it heavily relies on pretraining from larger datasets and is not directly comparable to other baselines. Nevertheless, our method delivers top-tier performance across all tasks.

In Table 2.8, we further compare our method with MLE-LEBM and NCE (with ratio decomposition) under both LD and SVGD sampling. We find that the vanilla MLE paradigm, and even NCE, perform poorly in the offline BBO setting—likely due to the complex, highly multimodal joint latent space of $(\mathbf{x}, y)$. By contrast, our N^{2}CE-based model produces strong results even with basic LD sampling, in line with our analysis on the importance of noise magnitude. Extended results, including ablations over `M,K` (Tab. 2.26) and robustness to query budget Q (Tab. 2.27), are deferred to Sec. 2.11.7.4.

2.5 Conclusion

In this chapter, we revisited NCE through the effective weighting of noise samples, an often-overlooked factor in ratio estimation. Building on this view, we introduced the N^2CE framework, interpreting a "noisier" NCE objective as a controlled approximation to MLE and providing both theoretical and practical insights into its behavior. Our analysis shows that, under mild conditions, the N^2CE gradient aligns with the MLE gradient when the noise magnitude is sufficiently large, and we further examined finite-sample gradient error to motivate practical regularization and hyperparameter selection.

Empirically, N^2CE delivers consistent improvements across image modeling, anomaly detection, and offline black-box optimization, outperforming strong baselines and demonstrating its versatility. More broadly, our results suggest that N^2CE provides a practical and theoretically grounded framework for learning rewards and critics. Looking ahead, we see particular promise in extending this framework to discrete domains such as language modeling and to multimodal tasks like text-to-image and text-to-video generation, where traditional NCE remains limited.

2.6 Acknowledgments for This Research Chapter

This chapter is based on the coauthored article by Peiyu Yu, Dinghuai Zhang, Hengzhi He, Xiaojian Ma, Sirui Xie, Ruiyao Miao, Yifan Lu, Yasi Zhang, Deqian Kong, Ruiqi Gao, Jianwen Xie, Guang Cheng, and Ying Nian Wu, "Noisier" Noise Contrastive Estimation is (Almost) Maximum Likelihood, published at the *International Conference on Learning Representations (ICLR)*, 2026. The published article designates Peiyu Yu, Dinghuai Zhang, and Hengzhi He as equal contributors.

For the research presented in this chapter, Peiyu Yu conceived the method, designed and carried out the experiments, and wrote the manuscript. Hengzhi He participated in discussions on establishing the connection with the Nguyen–Wainwright–Jordan (NWJ)

objective and on the family-of-estimators interpretation. Dinghuai Zhang, Xiaojian Ma, Sirui Xie, Ruiyao Miao, Yifan Lu, Yasi Zhang, Deqian Kong, Ruiqi Gao, Jianwen Xie, Guang Cheng, and Ying Nian Wu contributed to general discussions of the research and manuscript.

Y. W. is partially supported by NSF DMS-2415226, DARPA W912CG25CA007, and research gift funds from Amazon and Qualcomm. We gratefully acknowledge Lambda, Inc. for computational resources, and the reviewers for discussions that clarified the connection with NWJ.

2.7 Reproducibility Details

We provide complete statements and proofs of all theoretical results in Sec. 2.9. Implementation details, including key hyperparameters, network architectures, and step-by-step training instructions, are described in Sec. 2.10. Comprehensive descriptions of our experimental setups, datasets, and evaluation metrics are available in Sec. 2.11.

2.8 Related Work

Noise contrastive estimation Noise Contrastive Estimation (NCE) [25, 29, 99] provides a handy interface bridging the gap between discriminative and generative learning. An interesting line of research has been devoted to estimating unnormalized density, *i.e.*, Energy-Based Models (EBMs) or LEBMs, with NCE-like objectives [1, 6, 16, 23, 25, 48, 103]. Particularly, Gao et al. [16] recruits a normalizing flow [40, 86] as the base distribution for density estimation. Aneja et al. [1] refine the prior distribution of a pre-trained powerful VAE with noise contrastive learning. However, vanilla NCE, even with multi-stage estimation [87], may demonstrate less desirable results due to large gaps between target and noise distributions, dubbed as *density-chasm*. In this chapter, we propose to study this fundamental problem from an often-neglected perspective: the magnitude of the effective noise-to-data weighting used for ratio estimation. Specifically, with an increased magnitude of the noise distribution, the gradient of the NCE objective can closely match the MLE gradient. We provide theoretical and empirical analysis showing the effectiveness of our proposed framework.

Energy-based prior model As an interesting branch of EBMs [11, 12, 71, 72, 111], Pang et al. [74] show that an EBM can serve as an informative prior model in the latent space, *i.e.*, LEBM; such a prior greatly improves the model expressivity over those with non-informative gaussian priors and brings strong performance on downstream tasks [75, 76, 116, 117]. However, learning LEBM requires MCMC sampling to estimate the learning gradients, which needs careful tuning and numerous iterations to converge when the target distributions are high-dimensional or highly multimodal. Commonly-used short-run MCMC [71] in practice can lead to malformed energy landscapes and instability in training [11, 12, 17, 72, 117, 118]. In this work, we consider inducing the informative latent space with N^2CE integrated into variational learning; the proposed method requires no MCMC for estimating the prior model, and shows reliable sampling quality in practice.

Diffusion distillation Diffusion distillation has recently emerged as a key direction for accelerating sampling while retaining the strong generative performance of diffusion models.

Early efforts such as consistency models [97] and progressive distillation [92] demonstrated that multi-step denoising trajectories could be compressed into a few steps, but these methods often faced trade-offs between speed and fidelity. More recent approaches refine this idea by explicitly distilling score information from pretrained diffusion models. For instance, Score Distillation Sampling (SDS) and its variants [58, 82, 106] leverage gradients of the diffused KL divergence to align student and teacher scores at different noise levels. Extensions such as Diffusion-GAN [105] and Distribution Matching Distillation (DMD) [112] incorporate adversarial or distribution-matching objectives to better handle mismatched supports in high-dimensional spaces. More recently, Zhou et al. [124] introduced Score identity Distillation (SiD), a data-free, Fisher-divergence-based method that enables one-step generation, later improved by SiDA [123], which integrates adversarial loss to surpass the teacher's performance. Complementary to these, reinforcement learning–based perspectives have been proposed: Yoon et al. [115] formulated diffusion distillation as a maximum entropy inverse reinforcement learning problem, jointly training diffusion models and energy-based models, and further developed dynamic programming techniques to enable efficient updates. Collectively, these works highlight a growing trend toward principled frameworks that unify score matching, adversarial training, and control-theoretic formulations to accelerate and strengthen diffusion generation. Our work adds to this line by revisiting noise-contrastive estimation from the perspective of noise magnitude. In particular, we introduce N^2CE, a framework that connects "noisier" objectives to maximum likelihood, offering both theoretical insights and practical improvements across domains.

Online BBO An important set-up for BBO problems is the online BBO, also referred to as active BBO in many literature [41–43, 59]. For online BBO, the proposed method can query the black-box function with limited times during training. A probably most well-known family is Bayesian Optimization (BO), with a large body of prior work in the area [10, 20, 21, 33, 47, 57, 65, 66, 69, 88, 93, 94, 98]. Typically, BO employs surrogates such as Gaussian Processes to model the underlying function, where the surrogates are

sequentially updated by querying the black-box function with newly proposed points from an uncertainty-aware acquisition function. Other branches include derivative free methods such as the cross-entropy method [91], methods derived from the REINFORCE trick [90, 108] and reward-weighted regression [78], *etc*. Several practical approaches have combined these methodologies with Bayesian neural networks [94, 95], neural processes [18, 19, 36], and ensembles of learned score models [2, 3, 62] with extensive efforts in developing advanced querying mechanisms or better approximation of the surrogates to address the online BBO problem.

Offline BBO Recent works have shown significant progress in the offline BBO set-up [5, 7, 8, 14, 15, 37, 43, 45, 59, 83, 101, 102, 119, 120]. Among them, [45] trains a stochastic inverse mapping $p_{\boldsymbol{\theta}}(\mathbf{x}|y) \propto \mathbb{E}_{p(\mathbf{z}|y)}[p(\mathbf{x}|\mathbf{z}, y)]$ with a conditional GAN-like model to instantiate $p(\mathbf{x}|\mathbf{z}, y)$ [22, 63]. The method optimizes over $\mathbf{z}$ given the offline dataset maximum $y_{\max}$ and map $\mathbf{z}$ back to $\mathbf{x}$ for BBO solutions. The optimization process over $\mathbf{z}$ draw approximate samples from $p(\mathbf{z}|y)$. DDOM [43] consider directly parameterizing the inverse mapping $p_{\boldsymbol{\theta}}(\mathbf{x}|y)$ with a conditional diffusion model in the input design space utilizing the expressiveness of DDPMs [31]. We show in this chapter that our framework demonstrates significant improvements over previous parameterizations of inverse model. Forward methods [7, 8, 15, 83, 101, 120] employ gradient ascent to optimize the learned surrogates to propose candidate solutions for BBO. One critical issue, however, exists for the surrogate: the forward model can assign the out-of-training-distribution input designs $\mathbf{x}$ with erroneously large values or underestimate the true support of the distribution [8, 45, 59], especially when the set of valid inputs lies in a low-dimensional manifold in the high-dimensional input design space. Several inspiring works have focus extensively on addressing this issue [7, 8, 83, 101, 119, 120]. For example, [101, 119] assign lower scores to identified outliers so that the approximated surrogate is expected to be robust. [59] instead mimic the optimization trajectories of black-box optimizers, and rolls out evaluation trajectories from the trained sequence model for optimization. Although the method circumvents the need of the surrogate, it assumes knowledge of the approximate

value of the true optima during optimization, and can struggle relative to other approaches on certain domains. Our method learns an informative latent space suitable for BBO, and delivers strong results in comparison to prior works.

2.9 Detailed Derivations and Proofs

2.9.1 Proof of Proposition 2.3.1

Assumption 2.9.1 (Well-defined ratio)**.** *The reference distribution q_0 does not depend on $\boldsymbol{\alpha}$ and its support covers that of $p_{\boldsymbol{\alpha}}$. Hence $r_{\boldsymbol{\alpha}}(\mathbf{x}) = p_{\boldsymbol{\alpha}}(\mathbf{x})/q_0(\mathbf{x})$ is well-defined for all $\mathbf{x}$.*

Assumption 2.9.2 (Differentiability and integrability)**.** *For each $\mathbf{x}$, $p_{\boldsymbol{\alpha}}(\mathbf{x})$ is differentiable in $\boldsymbol{\alpha}$, and the score function $\nabla_{\boldsymbol{\alpha}} \log p_{\boldsymbol{\alpha}}(\mathbf{x})$ is integrable with respect to both q_* and $p_{\boldsymbol{\alpha}}$.*

Proof. Recall the objective

$$\mathcal{L}_M(\boldsymbol{\alpha}) = \mathbb{E}_{q_*}\left[\log \frac{r_{\boldsymbol{\alpha}}}{M + r_{\boldsymbol{\alpha}}}\right] + M\,\mathbb{E}_{q_0}\left[\log \frac{M}{M + r_{\boldsymbol{\alpha}}}\right].$$

By Assumption 2.9.2, we may interchange $\nabla_{\boldsymbol{\alpha}}$ and $\mathbb{E}[\cdot]$. For the first term,

$$\nabla_{\boldsymbol{\alpha}} \log \frac{r_{\boldsymbol{\alpha}}}{M + r_{\boldsymbol{\alpha}}} = -\nabla_{\boldsymbol{\alpha}} \log\left(1 + \frac{M}{r_{\boldsymbol{\alpha}}}\right) = \frac{M}{M + r_{\boldsymbol{\alpha}}}\,\nabla_{\boldsymbol{\alpha}} \log r_{\boldsymbol{\alpha}}.$$

Since q_0 does not depend on $\boldsymbol{\alpha}$, $\nabla_{\boldsymbol{\alpha}} \log r_{\boldsymbol{\alpha}} = \nabla_{\boldsymbol{\alpha}} \log p_{\boldsymbol{\alpha}}$. Hence

$$\nabla_{\boldsymbol{\alpha}}\,\mathbb{E}_{q_*}\left[\log \frac{r_{\boldsymbol{\alpha}}}{M + r_{\boldsymbol{\alpha}}}\right] = \mathbb{E}_{q_*}\left[\frac{M}{M + r_{\boldsymbol{\alpha}}}\,\nabla_{\boldsymbol{\alpha}} \log p_{\boldsymbol{\alpha}}\right].$$

For the second term,

$$\nabla_{\boldsymbol{\alpha}} \log \frac{M}{M + r_{\boldsymbol{\alpha}}} = -\frac{1}{M + r_{\boldsymbol{\alpha}}} \nabla_{\boldsymbol{\alpha}} r_{\boldsymbol{\alpha}} = -\frac{r_{\boldsymbol{\alpha}}}{M + r_{\boldsymbol{\alpha}}}\,\nabla_{\boldsymbol{\alpha}} \log p_{\boldsymbol{\alpha}},$$

so

$$M\,\nabla_{\boldsymbol{\alpha}}\,\mathbb{E}_{q_0}\left[\log \frac{M}{M + r_{\boldsymbol{\alpha}}}\right] = -\,M\,\mathbb{E}_{q_0}\left[\frac{r_{\boldsymbol{\alpha}}}{M + r_{\boldsymbol{\alpha}}}\,\nabla_{\boldsymbol{\alpha}} \log p_{\boldsymbol{\alpha}}\right].$$

Using the change of measure $\mathbb{E}_{q_0}[r_{\boldsymbol{\alpha}} g] = \mathbb{E}_{p_{\boldsymbol{\alpha}}}[g]$ from Assumption 2.9.1, we get

$$M\,\mathbb{E}_{q_0}\left[\frac{r_{\boldsymbol{\alpha}}}{M + r_{\boldsymbol{\alpha}}}\,g\right] = \mathbb{E}_{p_{\boldsymbol{\alpha}}}\left[\frac{M}{M + r_{\boldsymbol{\alpha}}}\,g\right].$$

Applying this with $g = \nabla_{\boldsymbol{\alpha}} \log p_{\boldsymbol{\alpha}}$ yields the stated gradient identity.

For the limit, note that $\frac{M}{M+r_{\boldsymbol{\alpha}}} \to 1$ pointwise as $M \to \infty$ and $0 \leqslant \frac{M}{M+r_{\boldsymbol{\alpha}}} \leqslant 1$. By Assumption 2.9.2, dominated convergence applies, giving

$$\lim_{M\to\infty} \mathbb{E}_{q_*}\left[\frac{M}{M + r_{\boldsymbol{\alpha}}} \nabla_{\boldsymbol{\alpha}} \log p_{\boldsymbol{\alpha}}\right] = \mathbb{E}_{q_*}[\nabla_{\boldsymbol{\alpha}} \log p_{\boldsymbol{\alpha}}],$$

and similarly for the expectation under $p_{\boldsymbol{\alpha}}$. Combining finishes the proof. □

2.9.2 Proof of Proposition 2.3.3

Proof. Let $g(\mathbf{x}) = \nabla_{\boldsymbol{\alpha}} \log r_{\boldsymbol{\alpha}}(\mathbf{x})$. By direct differentiation,

$$\nabla_{\boldsymbol{\alpha}} \mathcal{L}_M = \mathbb{E}_{q_*}\left[\frac{M}{M+r} g\right] - \mathbb{E}_{q_0}\left[\frac{Mr}{M+r} g\right].$$

With independent samples of sizes n_* and n_0, the empirical gradient is unbiased for $\nabla \mathcal{L}_M$. Thus its mean-squared error relative to $G = \nabla \mathcal{J}^{\mathrm{MLE}}$ splits into variance and squared bias.

For the variance, independence and $\mathrm{Var}(X) \leqslant \mathbb{E}\|X\|_2^2$ give

$$\begin{aligned}\mathrm{Var}(\nabla \widehat{\mathcal{L}}_M) &\leqslant \frac{1}{n_*}\mathbb{E}_{q_*}\left\|\frac{M}{M+r} g\right\|_2^2 + \frac{1}{n_0}\mathbb{E}_{q_0}\left\|\frac{Mr}{M+r} g\right\|_2^2 \\ &\leqslant \frac{1}{n_*}\mathbb{E}_{q_*}\|g\|_2^2 + \frac{1}{n_0}\min\left\{M^2 \mathbb{E}_{q_0}\|g\|_2^2,\ \mathbb{E}_{q_0}\|\nabla r\|_2^2\right\}.\end{aligned}$$

Here the last step uses $Mr/(M+r) \leqslant M$ and $Mr/(M+r) \leqslant r$.

For the bias, the normalized model identity and the change of measure $\mathbb{E}_{q_0}[rg] = \mathbb{E}_{p_{\boldsymbol{\alpha}}}[g]$ yield

$$\nabla \mathcal{L}_M - G = -\mathbb{E}_{q_*}\left[\frac{r}{M+r} g\right] + \mathbb{E}_{p_{\boldsymbol{\alpha}}}\left[\frac{r}{M+r} g\right].$$

By the triangle inequality and $r/(M+r) \leqslant r/M$,

$$\|\nabla \mathcal{L}_M - G\|_2 \leqslant \frac{1}{M}\left(\mathbb{E}_{q_*}[r\|g\|_2] + \mathbb{E}_{p_{\boldsymbol{\alpha}}}[r\|g\|_2]\right) = \frac{C_b(\boldsymbol{\alpha})}{M}.$$

Squaring and adding the variance bound proves the proposition. The constants depend on the fixed model parameter and moment conditions; no uniform-in-parameter statement is implied. □

2.9.3 Proof of Theorem 2.3.2

We work throughout in the extended parameterization $p_{\boldsymbol{\tau}} = h \exp(\boldsymbol{\theta}^\top S - a)$. The free scalar a is not constrained to equal $A(\boldsymbol{\theta})$ away from $\boldsymbol{\tau}^*$. Let $F_M = -\mathcal{L}_M$ and $H_M = \nabla^2 F_M$.

Lemma 2.9.3 (Global convexity and uniform local Hessian limit)**.** *Under the conditions in Theorem 2.3.2, F_M is convex throughout the extended natural parameter domain. Moreover, for any closed ball $B_r(\boldsymbol{\tau}^*)$ as in the theorem,*

$$\lim_{M\to\infty} \sup_{\boldsymbol{\tau}\in B_r(\boldsymbol{\tau}^*)} \|H_M(\boldsymbol{\tau}) - H_\infty(\boldsymbol{\tau})\|_{\mathrm{op}} = 0, \qquad H_\infty(\boldsymbol{\tau}) = e^{A(\boldsymbol{\theta})-a} F(\boldsymbol{\theta}).$$

The conclusion requires no essential bound on q_*/q_0.

Proof. Differentiating the contrastive objective gives

$$\nabla_{\boldsymbol{\tau}} \mathcal{L}_M(\boldsymbol{\tau}) = \int \frac{M q_0(\mathbf{x})}{p_{\boldsymbol{\tau}}(\mathbf{x}) + M q_0(\mathbf{x})} \big(q_*(\mathbf{x}) - p_{\boldsymbol{\tau}}(\mathbf{x})\big) T(\mathbf{x})\, d\mathbf{x},$$

and

$$\begin{aligned} H_M(\boldsymbol{\tau}) &= \int p_{\boldsymbol{\tau}}(\mathbf{x})\, \phi_M(\mathbf{x};\boldsymbol{\tau})\, T(\mathbf{x}) T(\mathbf{x})^\top\, d\mathbf{x}, \\ \phi_M(\mathbf{x};\boldsymbol{\tau}) &= \frac{1 + q_*(\mathbf{x})/(M q_0(\mathbf{x}))}{(1 + p_{\boldsymbol{\tau}}(\mathbf{x})/(M q_0(\mathbf{x})))^2}. \end{aligned} \tag{2.9}$$

These derivatives are justified by the local exponential moments and the integrability bounds below. The integrand is positive semidefinite, hence F_M is convex.

For each fixed $\mathbf{x}$ on the common support, $\phi_M(\mathbf{x};\boldsymbol{\tau}) \to 1$ uniformly over $\boldsymbol{\tau} \in B_r(\boldsymbol{\tau}^*)$: indeed, on this compact ball, $\sup_{\boldsymbol{\tau}} p_{\boldsymbol{\tau}}(\mathbf{x}) < \infty$, while $q_*(\mathbf{x})$ and $q_0(\mathbf{x}) > 0$ are fixed. We next justify taking the supremum through the integral. Writing $u = p_{\boldsymbol{\tau}}(\mathbf{x})$ and $v = M q_0(\mathbf{x})$,

$$u\phi_M = \frac{uv^2}{(u+v)^2} + \frac{uv}{(u+v)^2} q_*(\mathbf{x}) \leqslant u + \tfrac{1}{4} q_*(\mathbf{x}).$$

Because the compact $\boldsymbol{\theta}$-projection of B_r is contained in the interior of the natural parameter

domain, it admits a finite covering by sufficiently small closed axis-aligned rectangles whose vertices remain in that domain. The pointwise supremum $\sup_{\boldsymbol{\tau}\in B_r} p_{\boldsymbol{\tau}}(\mathbf{x})$ is bounded by a finite sum of unnormalized exponential-family densities at all these vertices, multiplied by a fixed bound on e^{-a} (the exponential of an affine function attains its maximum over a rectangle at a vertex). All vertex densities have finite integrals of $\|T(\mathbf{x})\|_2^2$: the required moments follow from finiteness of the log partition function in an open neighborhood of each vertex. Consequently,

$$\left\|\left(p_{\boldsymbol{\tau}}\phi_M - p_{\boldsymbol{\tau}}\right)TT^\top\right\|_{\mathrm{op}} \leqslant \left(2\sup_{\boldsymbol{\tau}\in B_r} p_{\boldsymbol{\tau}} + \tfrac{1}{4}q_*\right)\|T\|_2^2$$

has an integrable envelope independent of M. The supremum of the left-hand side tends to zero pointwise. Dominated convergence therefore yields

$$\sup_{\boldsymbol{\tau}\in B_r}\|H_M(\boldsymbol{\tau}) - H_\infty(\boldsymbol{\tau})\|_{\mathrm{op}} \leqslant \int \sup_{\boldsymbol{\tau}\in B_r}|p_{\boldsymbol{\tau}}\phi_M - p_{\boldsymbol{\tau}}|\,\|T\|_2^2\,d\mathbf{x} \longrightarrow 0.$$

Finally, $\int p_{\boldsymbol{\tau}}TT^\top d\mathbf{x} = e^{A(\boldsymbol{\theta})-a}\mathbb{E}_{p_{\boldsymbol{\theta}}}[TT^\top]$, which gives the asserted H_∞. □

Proof of Theorem 2.3.2. By correct specification, $\nabla F_M(\boldsymbol{\tau}^*) = 0$, and global convexity implies that $\boldsymbol{\tau}^*$ is a global minimizer. On $B_r(\boldsymbol{\tau}^*)$,

$$|A(\boldsymbol{\theta}) - a| \leqslant |A(\boldsymbol{\theta}) - A(\boldsymbol{\theta}^*)| + |a - A(\boldsymbol{\theta}^*)| \leqslant (\beta_Z + 1)r \leqslant 1.$$

The Fisher-type matrix assumption therefore implies

$$e^{-1}\lambda_{\min}I \preceq H_\infty(\boldsymbol{\tau}) \preceq e\lambda_{\max}I.$$

By Lemma 2.9.3, choose finite M_0 so that $\sup_{\boldsymbol{\tau}\in B_r}\|H_M - H_\infty\|_{\mathrm{op}} \leqslant \lambda_{\min}/(2e)$ for all $M \geqslant M_0$. Weyl's inequality gives the stated uniform bounds

$$\mu I \preceq H_M(\boldsymbol{\tau}) \preceq LI, \qquad \mu = \lambda_{\min}/(2e), \quad L = (e+1)\lambda_{\max}.$$

It remains to derive the iteration complexity without requiring global strong convexity.

Fix $\delta \in (0, r]$, put $s = \delta\sqrt{\mu/L}$, and consider any $\boldsymbol{\tau} \in \mathcal{C}$ with $d = \|\boldsymbol{\tau} - \boldsymbol{\tau}^*\|_2 > \delta$. Let $g = \nabla F_M(\boldsymbol{\tau})$ and $v = g/\|g\|_2$; $g \neq 0$ because $\boldsymbol{\tau}^*$ is the unique minimizer under local strong convexity and global convexity. For $z = \boldsymbol{\tau}^* + \delta(\boldsymbol{\tau} - \boldsymbol{\tau}^*)/d$, local strong convexity gives $F_M(z) \geqslant F_M(\boldsymbol{\tau}^*) + \mu\delta^2/2$. Since z is on the segment from $\boldsymbol{\tau}^*$ to $\boldsymbol{\tau}$, global convexity implies $F_M(\boldsymbol{\tau}) \geqslant F_M(\boldsymbol{\tau}^*) + \mu\delta^2/2$. On the other hand, $y = \boldsymbol{\tau}^* + sv$ lies inside B_r, so local smoothness and $\nabla F_M(\boldsymbol{\tau}^*) = 0$ yield $F_M(y) \leqslant F_M(\boldsymbol{\tau}^*) + Ls^2/2 = F_M(\boldsymbol{\tau}^*) + \mu\delta^2/2$. Using the supporting-hyperplane inequality at $\boldsymbol{\tau}$ gives

$$\begin{aligned}\langle g, \boldsymbol{\tau} - \boldsymbol{\tau}^* \rangle &\geqslant F_M(\boldsymbol{\tau}) - F_M(y) + s\|g\|_2 \\ &\geqslant s\|g\|_2.\end{aligned}$$

Thus every iterate farther than δ from the optimum obeys

$$\left\langle \frac{\nabla F_M(\boldsymbol{\tau}_t)}{\|\nabla F_M(\boldsymbol{\tau}_t)\|_2}, \boldsymbol{\tau}_t - \boldsymbol{\tau}^* \right\rangle \geqslant s.$$

For an unprojected normalized gradient step of length s,

$$\left\| \boldsymbol{\tau}_t - s\frac{g_t}{\|g_t\|_2} - \boldsymbol{\tau}^* \right\|_2^2 \leqslant \|\boldsymbol{\tau}_t - \boldsymbol{\tau}^*\|_2^2 - s^2.$$

The Euclidean projection onto closed convex $\mathcal{C}$ is nonexpansive and leaves $\boldsymbol{\tau}^* \in \mathcal{C}$ fixed, so the same bound holds for the projected iterate. Consequently, as long as the tolerance has not been reached, each iteration decreases the squared parameter distance by at least s^2. Summation gives

$$T \leqslant \left\lceil \frac{\|\boldsymbol{\tau}_0 - \boldsymbol{\tau}^*\|_2^2}{s^2} \right\rceil = \left\lceil \frac{L}{\mu} \frac{\|\boldsymbol{\tau}_0 - \boldsymbol{\tau}^*\|_2^2}{\delta^2} \right\rceil \leqslant \left\lceil 2e(e+1) \frac{\lambda_{\max}}{\lambda_{\min}} \frac{\|\boldsymbol{\tau}_0 - \boldsymbol{\tau}^*\|_2^2}{\delta^2} \right\rceil.$$

This concludes the global *population* normalized-gradient iteration guarantee. It supplies no quantitative upper bound on the necessary M_0 and does not bound the number of examples

or computations used per step. □

2.10 Network Architecture and Implementation Details

Code Code and data are available at `https://github.com/yuPeiyu98/Noisier-NCE`.

2.10.1 How to Learn a Latent Space Model with N^{2}CE

ELBO We show how the proposed form can be integrated into the variational learning scheme to learn an accurate and explicit density $p_{\boldsymbol{\alpha}}$. We can first see that the target distribution for $p_{\boldsymbol{\alpha}}$ is the aggregated posterior $q_{K+1}(\mathbf{z}) = \int_{\mathbf{x}} q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})p(\mathbf{x})d\mathbf{x}$ [100], where $q_{\boldsymbol{\phi}}$ denotes the posterior distribution parameterized by the encoder network. For ratio decomposition we use the following KL-augmented variational criterion. If the ratio-weighted stage models are normalized, this criterion is a lower bound on the standard ELBO because the auxiliary KL divergences are nonnegative:

$$\widetilde{\mathrm{ELBO}}_{\boldsymbol{\theta},\boldsymbol{\phi}} = \mathbb{E}_{q_{\boldsymbol{\phi}}}[\log p_{\boldsymbol{\beta}}(\mathbf{x}|\mathbf{z})] - D_{\mathrm{KL}}(q_{\boldsymbol{\phi}} \| r_{\boldsymbol{\alpha}_K} q_K) - \sum_{k=0}^{K-1} D_{\mathrm{KL}}(q_{k+1} \| r_{\boldsymbol{\alpha}_k} q_k), \tag{2.10}$$

The standard ELBO omits the auxiliary sum of KL terms. $\boldsymbol{\beta}$ denotes the decoder network.

Training algorithms We directly learn the log-ratio estimator $\tilde{f}_{\boldsymbol{\alpha}_k} = \log r_{\boldsymbol{\alpha}_k}$ for implementation. Let $\sigma(\cdot)$ denote the sigmoid function, the gradient for learning each ratio estimator $\{p_{\boldsymbol{\alpha}_k}\}_{k=0}^K$ can be derived from Eq. (2.5) as:

$$\nabla_{\boldsymbol{\alpha}_k} \mathcal{L}_M = \nabla_{\boldsymbol{\alpha}_k} \mathbb{E}_{q_{k+1}} \left[\log \sigma\left(\tilde{f}_{\boldsymbol{\alpha}_k} - \log M\right)\right] + M \nabla_{\boldsymbol{\alpha}_k} \mathbb{E}_{q_k} \left[\log\left(1 - \sigma\left(\tilde{f}_{\boldsymbol{\alpha}_k} - \log M\right)\right)\right]. \tag{2.11}$$

Let $\boldsymbol{\psi} = \{\boldsymbol{\beta}, \boldsymbol{\phi}\}$ collect the parameters of the decoder and encoder networks. For $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ and $p_{\boldsymbol{\beta}}(\mathbf{x}|\mathbf{z})$, the standard ELBO gradient with respect to the encoder and decoder parameters takes the form:

$$\nabla_{\boldsymbol{\psi}} \mathrm{ELBO} = \nabla_{\boldsymbol{\psi}} \mathbb{E}_{q_{\boldsymbol{\phi}}}[\log p_{\boldsymbol{\beta}}] - \nabla_{\boldsymbol{\phi}} D_{\mathrm{KL}}(q_{\boldsymbol{\phi}} \| p_0) + \nabla_{\boldsymbol{\phi}} \mathbb{E}_{q_{\boldsymbol{\phi}}} \left[\sum_{k=0}^{K} \tilde{f}_{\boldsymbol{\alpha}_k}\right]. \tag{2.12}$$

Alg. 1 summarizes the training procedure. The expressions $r_{\boldsymbol{\alpha}_k} q_k$ in the KL terms presuppose normalized densities. For neural, unconstrained ratio estimators the corresponding energy-based objectives are understood up to normalizing constants; their KL/ELBO interpretation requires those constants or a separate justification. The NCE-based ratio updates are surrogate training steps and should not be confused with exact gradients of the standard ELBO or of the KL-augmented criterion. Dependence of intermediate distributions on encoder parameters must also be specified before interpreting their gradients exactly.

Implementation When using the auxiliary criterion in Eq. (2.10), we randomly choose one intermediate stage to optimize at each training iteration as in Ho et al. [31]. To construct these intermediate distributions, we follow Yu et al. [117] to interpolate between pairs of samples $\mathbf{z}_{K+1} \sim q_{K+1}$ and $\mathbf{z}_0 \sim q_0$; samples from the k-th intermediate distribution q_k are: $\mathbf{z}_k = \sqrt{1-\sigma_k^2}\mathbf{z}_0 + \sigma_k \mathbf{z}_{K+1}$, where the $\{\sigma_k\}_{k=1}^K$ form an increasing sequence from 0 to 1 controlling the distance between these implicitly defined distributions $\{q_k\}_{k=1}^K$. We use a linear schedule to specify σ_k^2 and construct the intermediate distributions $\{q_k\}_{k=0}^K$.

Algorithm 1 Variational Learning with N^2CE

Input: dataset $\mathcal{D} : \{\mathbf{x}^{(i)}\}_{i=1}^N$, params. $\left(\{\boldsymbol{\alpha}_k\}_{k=0}^K, \boldsymbol{\beta}, \boldsymbol{\phi}\right)$,
Output: updated params. $\left(\{\boldsymbol{\alpha}_k^*\}_{k=0}^K, \boldsymbol{\beta}^*, \boldsymbol{\phi}^*\right)$

repeat
  posterior sampling: For each pair of $\{\mathbf{x}^{(i)}\}$, sample $\mathbf{z}_{K+1}^{(i)} \sim q_\phi(\mathbf{z}|\mathbf{x}^{(i)})$ using inference network.
  forming $\{q_k\}_{k=1}^K$**:** For each sample $\mathbf{z}_{K+1}^{(i)}$, sample $k \sim \mathrm{Unif}(\{0, ..., K\})$ and obtain $\left(\mathbf{z}_k^{(i)}, \mathbf{z}_{k+1}^{(i)}\right) \sim (q_k, q_{k+1})$ using $\mathbf{z}_k = \sqrt{1-\sigma_k^2}\mathbf{z}_0 + \sigma_k \mathbf{z}_{K+1}$.
  learning $\{\boldsymbol{\alpha}_k\}_{k=0}^K$**:** Update $\{\boldsymbol{\alpha}_k\}_{k=0}^K$ with Eq. (2.11).
  learning $\boldsymbol{\psi} = (\boldsymbol{\beta}, \boldsymbol{\phi})$**:** Update $\boldsymbol{\psi}$ with Eq. (2.12).
until converged.

2.10.2 Image Modeling and Anomaly Detection

Architecture We provide detailed network architecture for the log-ratio estimator for each stage $\tilde{f}_{\boldsymbol{\alpha}_k}(\mathbf{z}, k)$ in Tab. 2.11. For the VAE model, the decoder network has a simple

deconvolution structure similar to DCGAN [84] shown in Tab. 2.9. The encoder network to embed the observed images has a fully convolutional structure, as shown in Tab. 2.10.

Table 2.9: **Network structures of the generator networks** used for the SVHN, CelebA, CIFAR-10, CelebA-HQ and MNIST (from top to bottom) datasets. ConvT(n) indicates a transposed convolutional operation with n output channels. We use `ngf=64` for the SVHN dataset and `ngf=128` for the rest. LReLU indicates the Leaky-ReLU activation function. The slope in Leaky ReLU is set to be 0.2.

Layers	**Out Size**	**Stride**
Input: **z**	1x1x100	-
4x4 ConvT(ngf x 8), LReLU	4x4x(ngf x 8)	1
4x4 ConvT(ngf x 4), LReLU	8x8x(ngf x 4)	2
4x4 ConvT(ngf x 2), LReLU	16x16x(ngf x 2)	2
4x4 ConvT(3), Tanh	32x32x3	2
Layers	**Out Size**	**Stride**
Input: **z**	1x1x100	-
4x4 ConvT(ngf x 8), LReLU	4x4x(ngf x 8)	1
4x4 ConvT(ngf x 4), LReLU	8x8x(ngf x 4)	2
4x4 ConvT(ngf x 2), LReLU	16x16x(ngf x 2)	2
4x4 ConvT(ngf x 1), LReLU	32x32x(ngf x 1)	2
4x4 ConvT(3), Tanh	64x64x3	2
Layers	**Out Size**	**Stride**
Input: **z**	1x1x128	-
8x8 ConvT(ngf x 8), LReLU	8x8x(ngf x 8)	1
4x4 ConvT(ngf x 4), LReLU	16x16x(ngf x 4)	2
4x4 ConvT(ngf x 2), LReLU	32x32x(ngf x 2)	2
3x3 ConvT(3), Tanh	32x32x3	1
Layers	**Out Size**	**Stride**
Input: **z**	1x1x128	-
4x4 ConvT(ngf x 16), LReLU	4x4x(ngf x 16)	1
4x4 ConvT(ngf x 8), LReLU	8x8x(ngf x 8)	2
4x4 ConvT(ngf x 4), LReLU	16x16x(ngf x 4)	2
4x4 ConvT(ngf x 4), LReLU	32x32x(ngf x 4)	2
4x4 ConvT(ngf x 2), LReLU	64x64x(ngf x 2)	2
4x4 ConvT(ngf x 1), LReLU	128x128x(ngfx1)	2
4x4 ConvT(3), Tanh	256x256x3	2
Layers	**Out Size**	**Stride**
Input: **z**	1x1x8	-
7x7 ConvT(ngf x 8), LReLU	7x7x(ngf x 8)	1
4x4 ConvT(ngf x 4), LReLU	14x14x(ngf x 4)	2
4x4 ConvT(ngf x 2), LReLU	28x28x(ngf x 2)	2
3x3 ConvT(1), Tanh	28x28x1	1

Hyperparameters and training details In the image modeling experiments, for the MLE-LEBM baseline, we run LD for $T = 60$ iterations for prior updates during training with a step size of $s = 0.4$. For test time sampling from LEBMs trained with different objectives, we consistently LD for $T = 100$ with the step size of $s = 0.4$ to draw samples from $p_{\boldsymbol{\alpha}}$. In the anomaly detection experiments, we follow the same experiment configuration as used in Yu et al. [118].

We use the linear schedule as in and Yu et al. [117] to specify σ_k^2 and construct the intermediate distributions $\{q_k\}_{k=0}^K$. We set the number of stages to 3. Specifically, the sequence of $\{\sigma_k^2\}_{k=0}^3$ is: $\{\sigma_k^2\}_{k=0}^3 = [0.01, 0.69175489, 0.92238785, 0.99974058]$. To estimate the Eq. (2.5), we can use Monte-Carlo average: $\mathcal{L}_M(\boldsymbol{\alpha}_k) \approx \frac{1}{n_1}\sum_{i=1}^{n_1}\left[\log \frac{r_{\boldsymbol{\alpha}_k}(\mathbf{z}_i)}{M+r_{\boldsymbol{\alpha}_k}(\mathbf{z}_i)}\right] + \frac{M}{n_2}\sum_{j=1}^{n_2}\left[\log \frac{M}{M+r_{\boldsymbol{\alpha}_k}(\mathbf{z}_j)}\right]$, where $\mathbf{z}_i \sim q_{k+1}, \mathbf{z}_j \sim q_k$. We follow Mnih & Teh [64] and set $n_2 = Mn_1$ in practice, where $M = 100$ for all experiments; n_1 is the batch size. When pre-training the VAE model, we set the weight for reconstruction to $[1, 1, 100, 1]$ on the four datasets and KL weights to 1; when training jointly with the LEBM, we set the weight for reconstructing $\mathbf{x}$ to $[5, 5, 2, 1]$ for the term $\mathbb{E}_{q_{\boldsymbol{\phi}}}[\log p_{\boldsymbol{\beta}}(\mathbf{x}|\mathbf{z})]$ on SVHN, CelebA64, CIFAR10 and CelebAHQ datasets, respectively. We set the weight for controlling the KL term between $q_{\boldsymbol{\phi}}$ and $p_{\boldsymbol{\alpha}}$ to 1, *i.e.*, $D_{\mathrm{KL}}(q_{\boldsymbol{\phi}} \| r_{\boldsymbol{\alpha}_K} q_K) - \sum_{k=0}^{K-1} D_{\mathrm{KL}}(q_{k+1} \| r_{\boldsymbol{\alpha}_k} q_k)$. Other hyperparameters are shared across all the experiments.

The parameters of all the networks are initialized with the default pytorch methods [77]. We use the Adam optimizer [38] with $\beta_1 = 0.5$ and $\beta_2 = 0.999$ to train the decoder and encoder networks; the log-ratio estimator network is trained with default (β_1, β_2). The initial learning rates of the generator, encoder and log-ratio networks are `2e-4`. We further perform gradient clipping by setting the maximal gradient norm as 100 for these networks. We run the experiments on a A6000 GPU with the batch size of 128. Training converges within 100K iterations on all the datasets.

2.10.3 Reward and Critic Learning

We adapt the objectives in prior work as follows:

(i) For DxMI [115], we replace the contrastive-divergence-style EBM update (Sec. 3.2 of [115]) with our objective in Eq. (2.5), treating the true data distribution as the target and diffusion samples as noise.

(ii) For SiD^{2}A [123], we substitute the discriminator loss in Eq. (10) in [123] with our objective in Eq. (2.5), again using true data as the target and diffusion samples as noise.

All experiments retain the original network architectures and hyperparameters unless noted:

(i) DxMI: direct ratio regularization weight is 1.5 on CIFAR-10 and 1.55 on ImageNet64×64; on ImageNet64×64, the value function learning rate is `3e-5`.

(ii) SiD^{2}A: direct ratio regularization weight is 0.5 for CIFAR-10 (conditional and unconditional) and 0.4 for ImageNet64×64; we set $\alpha = 1.2$ throughout.

(iii) All remaining hyperparameters (e.g., sampler learning rates, batch sizes) follow the defaults, making training iterations directly comparable as a measure of computational cost, particularly for SiD^{2}A.

2.10.4 Offline Black-Box Optimization

Architecture We provide detailed network architecture for the log-ratio estimator $\tilde{f}_{\boldsymbol{\alpha}_k}(\mathbf{z}, k)$ in Tab. 2.11; we adopt the same architecture as in Sec. 2.10.2 throughout the experiments. The generator networks for $\mathbf{x}$ and y is a 5-layer MLP structure shown in Tab. 2.12. We employ a single network that directly outputs the $(\mathbf{x}, y)$ pair to implement $g_{\boldsymbol{\beta},\mathbf{x}}$, $g_{\boldsymbol{\beta},y}$. For discrete tasks, the $g_{\boldsymbol{\beta},\mathbf{x}}$ head outputs logits for reconstruction. The encoder network to embed the input is a 3-layer MLP, as shown in Tab. 2.12. For discrete tasks,

we add an additional embedding layer to map discrete inputs to continuous vectors. The network architectures are shared across different task domains; we only adjust the input size for inputs from different tasks. For function values y and continuous $\mathbf{x}$, we normalize these values for training using the offline dataset maxium $\mathbf{x}_{\max}, y_{\max}$ and minimum $\mathbf{x}_{\min}, y_{\min}$: $\mathbf{x}_{\text{train}} = \frac{\mathbf{x}-\mathbf{x}_{\min}}{\mathbf{x}_{\max}-\mathbf{x}_{\min}}$. $y_{\text{train}} = \frac{y-y_{\min}}{y_{\max}-y_{\min}}$.

Hyperparameters and implementation details We use the linear schedule as in Ho et al. [31] and Yu et al. [117] to specify σ_k^2 and construct the intermediate distributions $\{q_k\}_{k=0}^K$. We set the number of stages to 6. Specifically, the sequence of $\{\sigma_k^2\}_{k=0}^5$ is: $\{\sigma_k^2\}_{k=0}^6 = [0.01, 0.3237, 0.5165, 0.6322, 0.7132, 0.7734, 0.9997]$. We re-weight the objective for each $\boldsymbol{\alpha}_k$ with $w_k = \sqrt{\sigma_K / \prod_{i=k}^K \sigma_i}$ to further emphasize stages closer to the final target distribution q_{K+1}, as in Yu et al. [117]. To estimate the Eq. (2.5), we can use Monte-Carlo average: $\mathcal{L}_M(\boldsymbol{\alpha}_k) \approx \frac{1}{n_1}\sum_{i=1}^{n_1}\left[\log \frac{r_{\boldsymbol{\alpha}_k}(\mathbf{z}_i)}{M+r_{\boldsymbol{\alpha}_k}(\mathbf{z}_i)}\right] + \frac{M}{n_2}\sum_{j=1}^{n_2}\left[\log \frac{M}{M+r_{\boldsymbol{\alpha}_k}(\mathbf{z}_j)}\right]$, where $\mathbf{z}_i \sim q_{k+1}, \mathbf{z}_j \sim q_k$. We follow Mnih & Teh [64] and set $n_2 = Mn_1$ in practice, where $M = 100$ for all experiments; n_1 is the batch size. When training with LEBM, we set the weight for reconstructing $\mathbf{x}$ and predicting y to 50 for the term $\mathbb{E}_{q_{\boldsymbol{\phi}}}[\log p_{\boldsymbol{\beta}}(\mathbf{x}, y|\mathbf{z})]$. We set the weight for KL term between $q_{\boldsymbol{\phi}}$ and $p_{\boldsymbol{\alpha}}$ to 0.75. We apply orthogonal regularization [4] to the parameters of decoder networks, with a weight of 0.001 to facilitate training. These hyperparameters are shared across all the experiments.

The parameters of all the networks are initialized with the default pytorch methods [77]. We use the Adam optimizer [38] with $\beta_1 = 0.5$ and $\beta_2 = 0.999$ to train the generator and encoder networks; the log-ratio estimator network is trained with default (β_1, β_2). The initial learning rates of the generator and encoder networks are `1e-4`, and `5e-5` for the log-ratio network. We further perform gradient clipping by setting the maximal gradient norm as 100 for these networks. We run the experiments on a A6000 GPU with the batch size of 128. Training typically converges within 30K iterations on all the datasets.

When sampling with SVGD for optimization, the number of SVGD iterations T is set to 500. For all experiments, we use RBF kernel $k(\mathbf{z}, \mathbf{z}_0) = \exp\left(-\frac{1}{2h^2}\|\mathbf{z}-\mathbf{z}_0\|_2^2\right)$ for SVGD,

and set the bandwidth to be $h^2 = \frac{\text{med}^2}{2\log(n+1)}$. med is the median of the pairwise distance between the current samples $\{\mathbf{z}_i^t\}_{i=1}^Q$; this follows the same heuristic as in Liu & Wang [54] that $\sum_j k(\mathbf{z}_i, \mathbf{z}_j) \approx n \exp(-\frac{1}{2h^2}\text{med}^2) = 1$. We can balance the contribution from its own gradient and the influence from the other points for each sample $\mathbf{z}_i$. Of note, the bandwidth changes adaptively across iterations with this heuristic. For stability, we use Adam [38] with default hyperparemters instead of AdaGrad [13] used in Liu & Wang [54] to allow for adaptive step size ϵ_t. The initial step size is within the range of $[0.3, 0.5]$ for different tasks.

Table 2.10: **Network structures of the encoder networks** used for the SVHN, CelebA, CIFAR-10, CelebA-HQ and MNIST (from top to bottom) datasets. Conv(n)Norm indicates a convolutional operation with n output channels followed by the Instance Normalization [104]. We use `nif=64` for all the datasets. LReLU indicates the Leaky-ReLU activation function. The slope in Leaky ReLU is set to be 0.2.

Layers	**Out Size**	**Stride**
Input: **x**	32x32x3	-
3x3 Conv(nif x 1)Norm, LReLU	32x32x(nif x 1)	1
4x4 Conv(nif x 2)Norm, LReLU	16x16x(nif x 2)	2
4x4 Conv(nif x 4)Norm, LReLU	8x8x(nif x 4)	2
4x4 Conv(nif x 8)Norm, LReLU	4x4x(nif x 8)	2
4x4 Conv(nemb)Norm, LReLU	1x1x(nz)	1
Layers	**Out Size**	**Stride**
Input: **x**	64x64x3	-
3x3 Conv(nif x 1)Norm, LReLU	64x64x(nif x 1)	1
4x4 Conv(nif x 2)Norm, LReLU	32x32x(nif x 2)	2
4x4 Conv(nif x 4)Norm, LReLU	16x16x(nif x 4)	2
4x4 Conv(nif x 8)Norm, LReLU	8x8x(nif x 8)	2
4x4 Conv(nif x 8)Norm, LReLU	4x4x(nif x 8)	2
4x4 Conv(nemb)Norm, LReLU	1x1x(nz)	1
Layers	**Out Size**	**Stride**
Input: **x**	32x32x3	-
3x3 Conv(nif x 1)Norm, LReLU	32x32x(nif x 1)	1
4x4 Conv(nif x 2)Norm, LReLU	16x16x(nif x 2)	2
4x4 Conv(nif x 4)Norm, LReLU	8x8x(nif x 4)	2
4x4 Conv(nif x 8)Norm, LReLU	4x4x(nif x 8)	2
4x4 Conv(nz)Norm, LReLU	1x1x(nz)	1
Layers	**Out Size**	**Stride**
Input: **x**	256x256x3	-
3x3 Conv(nif x 1)Norm, LReLU	256x256x(nif x 1)	1
4x4 Conv(nif x 2)Norm, LReLU	128x128x(nif x 2)	2
4x4 Conv(nif x 4)Norm, LReLU	64x64x(nif x 4)	2
4x4 Conv(nif x 4)Norm, LReLU	32x32x(nif x 4)	2
4x4 Conv(nif x 8)Norm, LReLU	16x16x(nif x 8)	2
4x4 Conv(nif x 8)Norm, LReLU	8x8x(nif x 8)	2
4x4 Conv(nif x 8)Norm, LReLU	4x4x(nif x 8)	2
4x4 Conv(nz)Norm, LReLU	1x1x(nz)	1
Layers	**Out Size**	**Stride**
Input: **x**	28x28x3	-
3x3 Conv(nif x 1)Norm, LReLU	28x28x(nif x 1)	1
4x4 Conv(nif x 2)Norm, LReLU	14x14x(nif x 2)	2
4x4 Conv(nif x 4)Norm, LReLU	7x7x(nif x 4)	2
4x4 Conv(nif x 8)Norm, LReLU	3x3x(nif x 8)	2
3x3 Conv(nz)Norm, LReLU	1x1x(nz)	1

Table 2.11: **Network architecture for the log-ratio estimator $\tilde{f}_{\boldsymbol{\alpha}_k}(\mathbf{z}, k)$.** N is set to 3 for image modeling and anomaly detection experiments, and 16 for the offline BBO experiments. LReLU denotes the Leaky ReLU activation function. The slope in Leaky ReLU is set to 0.2. For SVHN and CelebA datasets, we use `nz=100`. For the CIFAR-10 and CelebA-HQ datasets, we use `nz=128` and `nz=512`, respectively. We use `nz=8` for anomaly detection on the MNIST dataset; `nz=20` is the latent space dimension used for offline BBO experiments.

Layers	**Output size**	**Note**
	Indx. Emb.	
Input: k	1	stage indx. of N^2CE
Sin. emb.	128	
Linear, LReLU	128	neg_slope 0.2
Linear	128	
	Input Emb.	
Input: $\mathbf{z}$	$nz = 20$	
Linear, LReLU	128	neg_slope 0.2
Linear	128	
	$\tilde{\mathbf{f}}_{\boldsymbol{\alpha}_\mathbf{k}}(\mathbf{z}, \mathbf{k})$	
Input: $\mathbf{z}, t$	1, $nz = 20$	
Input & Indx. Emb.	128×2	Emb. of each input
Concat.	256	
LReLU, Linear	128	neg_slope 0.2
N ResBlocks	128	LReLU, Linear + Input Skip
LReLU, Linear	1	energy score

Table 2.12: **Generator and encoder network architectures.** LReLU indicates the Leaky-ReLU activation function. $d_\mathbf{x}$ is the dimension of input $\mathbf{x}$ mentioned in Tab. 2.18. For discrete tasks, $d_\mathbf{x}$ specifies the length of input, and L denotes number of possible discrete tokens.

Generator		
Layers	**Out Size**	**Note**
Input: $\mathbf{z}$	$nz = 20$	
Linear, LReLU	128	
Linear, LReLU	64	neg_slope
Linear, LReLU	32	0.2
Linear, LReLU	32	
Continuous Output		
Linear	$d_\mathbf{x} + 1$	output $(\mathbf{x}, y)$
Discrete Output		
Linear	$d_\mathbf{x} \times L + 1$	num. tokens. L

Encoder		
Layers	**Out Size**	**Note**
Embedding (for discrete $\mathbf{x}$ only)		
Input: $\mathbf{x}$	$32 \times d_\mathbf{x}$	emb. $\mathbf{x}$
concat $(\mathbf{x}, y)$	$(32\times)d_\mathbf{x} + 1$	input $(\mathbf{x}, y)$
Linear, LReLU	128	neg_slope
Linear, LReLU	128	0.2
Mean head		
Linear	20	
Var. head		
Linear	20	

2.11 Additional Experiment Details & Supplementary Results

2.11.1 Gaussian Simulation

We illustrate the gradient and conditioning phenomena in Proposition 2.3.3 and Theorem 2.3.2 on a Gaussian location family; these simulations use ordinary gradient updates and do not constitute a test of the global normalized-gradient complexity bound. Specifically, we consider $p_{\boldsymbol{\alpha}}(\mathbf{x}) = \mathcal{N}(\mathbf{x}; \boldsymbol{\alpha}, I_d)$, where $\boldsymbol{\alpha} \in \mathbb{R}^d$ denotes the mean parameter and I_d is the identity covariance. The target distribution is set to $q_* = \mathcal{N}(\boldsymbol{\alpha}^*, I_d)$, and the noise distribution to $q_0 = \mathcal{N}(0, I_d)$. Unless otherwise noted, we use $d = 2$ for visualization.

The log-ratio and its gradient admit closed forms:

$$f_{\boldsymbol{\alpha}}(\mathbf{x}) = \boldsymbol{\alpha}^\top \mathbf{x} - \tfrac{1}{2}\|\boldsymbol{\alpha}\|^2, \qquad \nabla_{\boldsymbol{\alpha}} f_{\boldsymbol{\alpha}}(\mathbf{x}) = \mathbf{x} - \boldsymbol{\alpha}.$$

Hence, the noisier NCE gradient is

$$\nabla_{\boldsymbol{\alpha}} \mathcal{L}_M(\boldsymbol{\alpha}) = \mathbb{E}_{q_*}\left[\tfrac{M}{M + r_{\boldsymbol{\alpha}}(\mathbf{x})}\ (\mathbf{x} - \boldsymbol{\alpha})\right] - \mathbb{E}_{p_{\boldsymbol{\alpha}}}\left[\tfrac{M}{M + r_{\boldsymbol{\alpha}}(\mathbf{x})}\ (\mathbf{x} - \boldsymbol{\alpha})\right],$$

where $r_{\boldsymbol{\alpha}}(\mathbf{x}) = \frac{p_{\boldsymbol{\alpha}}(\mathbf{x})}{q_0(\mathbf{x})}$. As $M \to \infty$, this reduces to the exact MLE gradient

$$\nabla_{\boldsymbol{\alpha}} \mathcal{L}_{\mathrm{MLE}}(\boldsymbol{\alpha}) = \mathbb{E}_{q_*}[\mathbf{x}] - \mathbb{E}_{p_{\boldsymbol{\alpha}}}[\mathbf{x}] = \boldsymbol{\alpha}^* - \boldsymbol{\alpha}.$$

Setup. We set $\boldsymbol{\alpha}^* = (1.5, -0.8)$ and initialize at $\boldsymbol{\alpha}_0 = (-2.0, 1.0)$. Gradients are approximated by Monte Carlo with $n = 4000$ samples per iteration. Optimization is performed with step size $\eta = 0.2$ for $T = 150$ iterations. We track several metrics during training, including the distance to the optimum $\|\boldsymbol{\alpha}_t - \boldsymbol{\alpha}^*\|_2$ and mean and median of absolute gradient errors. We report results across different noise magnitudes $M \in \{1, 2, 10, 50, 100, 1000\}$, with $M = 1$ corresponding to vanilla NCE and large M approximating MLE.

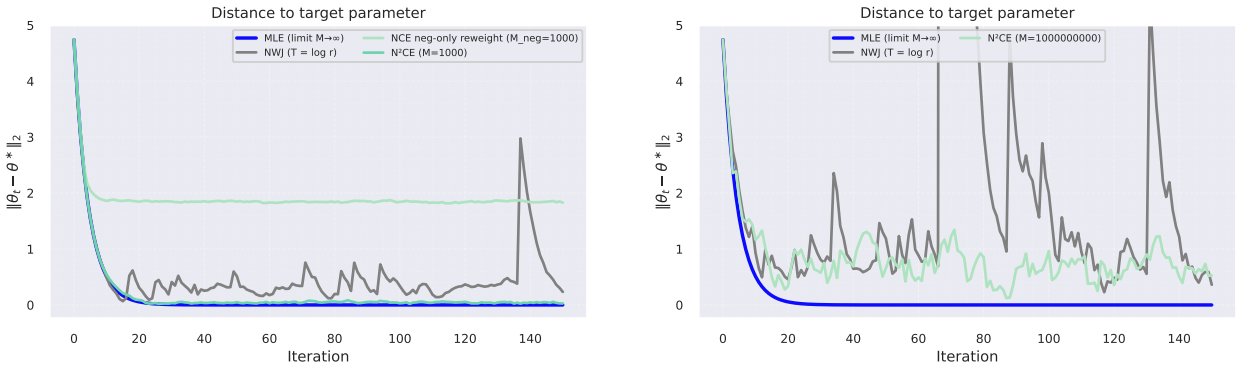
Further comparison with Nguyen-Wainwright-Jordan (NWJ) and simple reweighting. We also compare our method with the NWJ objective [70]:

$$\mathcal{L}_{\mathrm{NWJ}}(\boldsymbol{\alpha}) = \mathbb{E}_{q_*}[\log r_{\boldsymbol{\alpha}}] - \mathbb{E}_{q_0}[r_{\boldsymbol{\alpha}}] \implies \nabla_{\boldsymbol{\alpha}} \mathcal{L}_{\mathrm{NWJ}}(\boldsymbol{\alpha}) = \mathbb{E}_{q_*}[\mathbf{x} - \boldsymbol{\alpha}] - \mathbb{E}_{q_0}[r_{\boldsymbol{\alpha}}(\mathbf{x})\,(\mathbf{x} - \boldsymbol{\alpha})],$$

as well as a variant of our N^{2}CE objective, where we only use N^{2}CE-style weighting for the negative part of the objective and use the positive part of the vanilla N^{2}CE objective:

$$\mathcal{L}_{\mathrm{neg-reweight}}(\boldsymbol{\alpha}) = \mathbb{E}_{q_*(\mathbf{x})}\left[\log \frac{r_{\boldsymbol{\alpha}}(\mathbf{x})}{1 + r_{\boldsymbol{\alpha}}(\mathbf{x})}\right] + M\mathbb{E}_{q_0(\mathbf{x})}\left[\log \frac{M}{M + r_{\boldsymbol{\alpha}}(\mathbf{x})}\right].$$

Similar to our previous setting, we consider in this case 5-d gaussians. We set $\boldsymbol{\alpha}^* = ([-1.5, -0.75, 0., 0.75, 1.5])$ and initialize at $\boldsymbol{\alpha}_0 = -\boldsymbol{\alpha}^*$, *i.e.*, on the opposite side. We use different number of Monte Carlo samples to approximate different regimes and for better visual comparison: (i) Gradients are approximated by Monte Carlo with $n = 500$ samples per iteration, corresponding with low gradient estimation variance in Proposition 2.3.3, and (ii) with $n = 2$ samples per iteration, corresponding with high estimation variance. Optimization is performed with step size $\eta = 0.2$ for $T = 150$ iterations. We plot again the distance to the optimum $\|\boldsymbol{\alpha}_t - \boldsymbol{\alpha}^*\|_2$ to illustrate the training trajectories. We plot a single-run visualization of the optimization trajectory shown in Fig. 2.5. Further, for more systematic and scientific comparisons, we use the trajectory-wise average squared norm $\frac{1}{T}\sum_{t=0}^{T-1} \|\theta_t - \theta^*\|_2^2$ as a summary of one run. We repeat 100 independent experiments and report the means and stds in Tabs. 2.13 and 2.14. we observe that (i) in these $n = 2/500$ Monte Carlo experiments, a suitable finite M yields lower trajectory-wise error and variability than the NWJ baseline, although the population NWJ gradient agrees with the limiting MLE gradient. (ii) When M is sufficiently large, N^{2}CE empirically approaches the NWJ objective with $T = \log r$.



(a) (b)

Figure 2.5: **Finite-M N^{2}CE can exhibit more stable Monte Carlo optimization than NWJ and simple reweighting.** We plot trajectories represented by L_2 norms between model and target parameters.

Table 2.13: **MSE sweep for $n = 2$.** The finite-sample trade-off depends on the noise-sample allocation; see Proposition 2.3.3.

M	NWJ ($T = \log r$)	1	1.5	2	5
avg. runs MSE	54.548 ± 237.528	0.904 ± 0.084	$\mathbf{0.884 \pm 0.083}$	0.888 ± 0.084	0.983 ± 0.107
M	10	100	1000	10^9	
avg. runs MSE	1.139 ± 0.168	3.158 ± 2.683	17.564 ± 42.751	61.291 ± 277.336	

Table 2.14: **MSE sweep for $n = 500$.** The finite-sample trade-off depends on the noise-sample allocation; see Proposition 2.3.3.

M	NWJ ($T = \log r$)	1	10	50	100
avg. runs MSE	1.359 ± 5.454	0.678 ± 0.004	0.489 ± 0.004	0.456 ± 0.006	$\mathbf{0.453 \pm 0.007}$
M	1000	10^4	2×10^4	10^9	
avg. runs MSE	0.489 ± 0.020	0.641 ± 0.324	0.750 ± 0.889	1.909 ± 11.201	

2.11.2 Image Modeling

For image modeling in Sec. 2.4.1, we include the following datasets to study our method: SVHN ($32 \times 32 \times 3$), CIFAR-10 ($32 \times 32 \times 3$), CelebA ($64 \times 64 \times 3$) and CeleAMask-HQ (256 x 256 x 3). Following Pang et al. [74], we use the full training set of SVHN (73,257) and CIFAR-10 (50,000), and take 40,000 samples of CelebA as the training data. We take 29,500 samples from the CelebAMask-HQ dataset as the training data. The images are scaled to $[-1, 1]$ and are randomly horizontally flipped with a prob. of .5 for training.

Table 2.15: **AUPRC(↑) scores for unsupervised anomaly detection on MNIST**. Baseline numbers are taken from Yoon et al. [114], Yu et al. [118]. Results of our model are averaged over 10 trials.

Heldout Digit	1	4	5	7	9
AE	0.062 ± 0.00	0.204 ± 0.00	0.259 ± 0.01	0.125 ± 0.00	0.113 ± 0.00
VAE	0.063	0.337	0.325	0.148	0.104
ABP	0.095 ± 0.03	0.138 ± 0.04	0.147 ± 0.03	0.138 ± 0.02	0.102 ± 0.03
IGEBM	0.101 ± 0.02	0.106 ± 0.02	0.205 ± 0.11	0.100 ± 0.04	0.079 ± 0.02
MEG	0.281 ± 0.04	0.401 ± 0.06	0.402 ± 0.06	0.290 ± 0.04	0.342 ± 0.03
BiGAN-σ	0.287 ± 0.02	0.443 ± 0.03	0.514 ± 0.03	0.347 ± 0.02	0.307 ± 0.03
ABP-LEBM	0.336 ± 0.01	0.630 ± 0.02	0.619 ± 0.01	0.463 ± 0.01	0.413 ± 0.01
JVAEBM	0.297 ± 0.03	0.723 ± 0.04	0.676 ± 0.04	0.490 ± 0.04	0.383 ± 0.03
Adaptive CE	0.531 ± 0.02	0.729 ± 0.02	0.742 ± 0.01	0.620 ± 0.02	0.499 ± 0.01
NAE	0.802 ± 0.08	0.648 ± 0.05	0.716 ± 0.03	0.789 ± 0.04	0.441 ± 0.07
MPDR-S	0.764 ± 0.05	0.823 ± 0.02	0.741 ± 0.04	$\mathbf{0.857} \pm 0.02$	0.478 ± 0.05
MPDR-R	0.844 ± 0.03	0.711 ± 0.03	0.757 ± 0.02	$\underline{0.850} \pm 0.01$	0.569 ± 0.04
DAMC	0.684 ± 0.02	0.911 ± 0.01	0.939 ± 0.02	0.801 ± 0.01	$\underline{0.705} \pm 0.01$
DAMC-NCE	0.702 ± 0.01	0.829 ± 0.02	0.764 ± 0.01	0.605 ± 0.01	0.502 ± 0.02
DAMC-N^{2}CE					
`M=100,K=1`	$\underline{0.910} \pm 0.02$	$\underline{0.911} \pm 0.01$	$\underline{0.935} \pm 0.01$	0.779 ± 0.01	0.699 ± 0.02
`M=100,K=3`	$\mathbf{0.959} \pm 0.01$	$\mathbf{0.935} \pm 0.01$	$\mathbf{0.959} \pm 0.02$	0.845 ± 0.01	$\mathbf{0.854} \pm 0.01$

2.11.3 Anomaly Detection

Experiment settings For anomaly detection in Sec. 2.4.1, we consider the MNIST (28 x 28 x 1) dataset; we follow the experimental settings in [118, 121] and use 80% of the in-domain data to train the model. We base our method upon the DAMC [118] model for posterior inference, where we use the LEBM learned by N^{2}CE as a plug-in replacement of the original prior model. With properly learned models, the posterior $p_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ could form a discriminative latent space that has separated probability densities for normal and anomalous data. Given the testing sample $\mathbf{x}$, we use log of unnormalized joint density $p_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) \propto p_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathbf{x},\mathbf{z}) \approx p_{\boldsymbol{\beta}}(\mathbf{x}|\mathbf{z}) p_{\boldsymbol{\alpha}}(\mathbf{z})|_{\mathbf{z}\sim q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}$ as our decision function; we draw samples from $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ and compare the corresponding reconstruction errors and energy scores. A higher value of log joint density indicates a higher probability of the test sample being a normal sample.

Baselines We benchmark our method against three groups of strong counterparts: i) data-space energy-based models, including IGEBM [11], MEG [46], JVAEBM [27] and MPDR-S/R [114]; ii) latent variable models, including AE, VAE [39], ABP [26], ABP-LEBM [74], Adaptive CE [110], NAE [113] and DAMC [118], and iii) GAN-Based method like BiGAN-σ [121].

2.11.4 Reward and Critic Learning for Diffusion Distillation

Table 2.16: **Ablative FID scores of M on CIFAR-10 under [115] setting.**

$M = 1$	$M = 50$	$M = 100$	$M = 200$
3.93	3.05	**2.99**	3.04

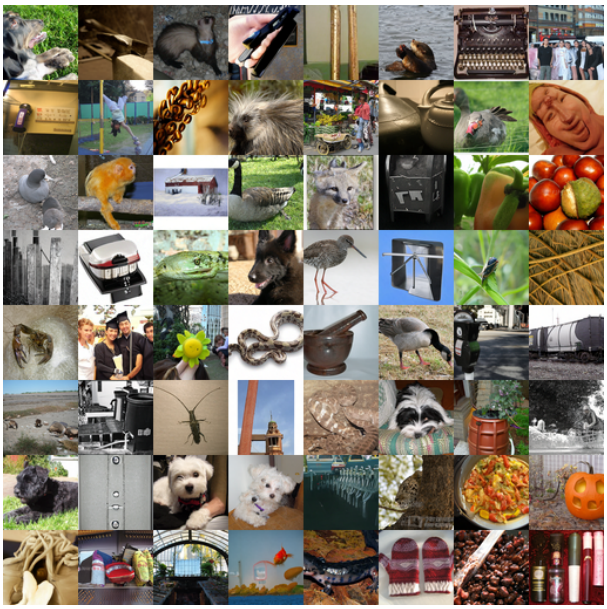
Table 2.17: **Ablative uncond. FID scores of M on CIFAR-10 under [123] setting.**

$M = 1$	$M = 10$	$M = 50$	$M = 100$
1.53	1.51	**1.45**	1.50

We further provide ablation of M on CIFAR-10 in Tabs. 2.16 and 2.17. We can see that greater M, *e.g.*, $M = 50$ or $M = 100$ is significantly better than smaller M, especially $M = 1$. However, larger M may also incur larger approximation variance based on our analysis in Proposition 2.3.3. We therefore choose $M = 100$ and $M = 50$ respectively after the sweep for these experiments. These results further confirm our assumption about the noise magnitude.



(a)



(b)

Figure 2.6: **Uncurated 1-step generation results from our distilled EDM models.** We present 1-step generation results from our distilled models on CIFAR-10 (left) and ImageNet64x64 (right) datasets.

2.11.5 Offline Black-Box Optimization

2.11.5.1 Problem Statement

Let $h : \mathcal{X} \to \mathbb{R}$ be a scalar function; the domain $\mathcal{X}$ is an arbitrary subset of $\mathbb{R}^d$. In Black-Box Optimization (BBO), h is an unknown black-box function; we are interested in finding $\mathbf{x}^*$ that maximizes h:

$$\mathbf{x}^* \in \arg\max_{\mathbf{x} \in \mathcal{X}} h(\mathbf{x}). \tag{2.13}$$

Typically, h is expensive to evaluate. In offline BBO, we assume no direct access to h during training, and are thus **not allowed** to actively query the black-box function during optimization, unlike in online BBO where most approaches would employ iterative online solving schemes [41, 42, 93, 94]. Specifically, in the *offline* BBO setting [102], one has only access to a pre-collected dataset of observations, $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$, where $h(\mathbf{x}_i) = y_i$. During evaluation, one may query the function h for a small budget of Q queries to output candidates with the best function value obtained.

2.11.5.2 Learning Latent Variable Models for Offline Black-Box Optimization

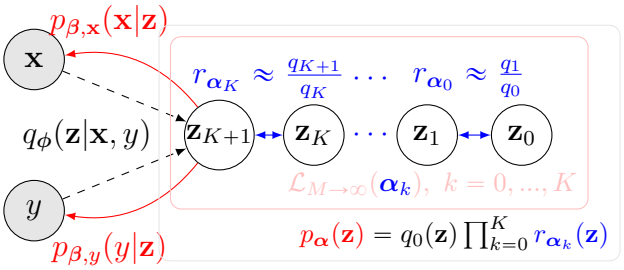


Figure 2.7: **Graphical illustration of our BBO framework.** We construct an energy-based latent space model $p_{\boldsymbol{\alpha}}$ for offline BBO via learning a series of ratio estimators $\{r_{\boldsymbol{\alpha}_k}\}_{k=0}^K$ with the N^{2}CE objective $\mathcal{L}_{M\to\infty}$ to optimize the ELBO *without* MCMC. After training, we employ stochastic samplers like LD or SVGD to perform BBO by sampling from the implicit inverse model $p_{\boldsymbol{\theta}}(\mathbf{x}|y)\propto\mathbb{E}_{p_{\boldsymbol{\theta}}(\mathbf{z}|y)}[p_{\beta,\mathbf{x}}(\mathbf{x}|\mathbf{z})]$, where $p_{\boldsymbol{\theta}}(\mathbf{z}|y)\propto p_{\beta,y}(y|\mathbf{z})p_{\boldsymbol{\alpha}}(\mathbf{z})$ given y. Best viewed in color.

We consider again variational learning of the latent variable model. Specifically, the ELBO in this setting would be

$$\begin{aligned}\text{ELBO}_{\boldsymbol{\theta},\phi} &= \log p_{\boldsymbol{\theta}}(\mathbf{x},y) - D_{\text{KL}}(q_\phi(\mathbf{z}|\mathbf{x},y)\|p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x},y)) \\ &= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x},y)}[\log p_\beta(\mathbf{x},y|\mathbf{z})] - D_{\text{KL}}(q_\phi(\mathbf{z}|\mathbf{x},y)\|p_{\boldsymbol{\alpha}}(\mathbf{z})),\end{aligned} \tag{2.14}$$

where in $p_{\boldsymbol{\theta}}(\mathbf{x},y)$, $p_{\beta,\mathbf{x}}(\mathbf{x}|\mathbf{z}) = \mathcal{N}(g_{\beta,\mathbf{x}}(\mathbf{z}),\sigma_x^2\mathbf{I})$ for continuous $\mathbf{x}$ and multinomial for discrete $\mathbf{x}$, and $p_{\beta,y}(y|\mathbf{z}) = \mathcal{N}(g_{\beta,y}(\mathbf{z}),\sigma_y^2\mathbf{I})$. $g_{\beta,\mathbf{x}}$ and $g_{\beta,y}$ is the generator network for $\mathbf{x}$ and prediction network for y, respectively. σ_x^2,σ_y^2 take pre-specified values. $q_\phi(\mathbf{z}|\mathbf{x},y) = \mathcal{N}(\boldsymbol{\mu}_\phi,\boldsymbol{\sigma}_\phi\mathbf{I})$ is the amortized posterior. See Fig. 2.7 for the graphical illustration.

Optimization as gradient-based sampling Once the latent variable model with $p_{\boldsymbol{\alpha}}$ is properly trained with N^{2}CE, we can sample from the parameterized inverse model $p_{\boldsymbol{\theta}}(\mathbf{x}|y)\propto\mathbb{E}_{p_{\boldsymbol{\theta}}(\mathbf{z}|y)}[p_{\beta,\mathbf{x}}(\mathbf{x}|\mathbf{z})]$ to solve the BBO problem. Specifically, we **i)** sample $\mathbf{z}$ from $p_{\boldsymbol{\theta}}(\mathbf{z}|y)\propto p_{\beta,y}(y|\mathbf{z})p_{\boldsymbol{\alpha}}(\mathbf{z})$ with LD or SVGD, where y is set to the offline dataset maximum $y_{\max}$ as in [43]; and **ii)** feed them into the generator network $g_{\beta,\mathbf{x}}$ to obtain the proposed input designs $\mathbf{x}$. Intuitively, $p_{\boldsymbol{\theta}}(\mathbf{z}|y=y_{\max})$ exploits the offline high value input design modes, while sampling from $p_{\boldsymbol{\theta}}$ and mapping from z to x with $p_{\beta,\mathbf{x}}(\mathbf{x}|\mathbf{z})$ achieves expanded latent exploration guided by the latent space model $p_{\boldsymbol{\alpha}}$. We provide an interpretation of latent

exploration in Sec. 2.11.5.3.

Stein Variational Gradient Descent (SVGD) A well-established counterpart of MCMC for approximating the target distribution is SVGD, first proposed in the seminal work of Liu & Wang [54]. Specifically, it tackles the sampling problem with a set of particles for approximation, on which a form of (functional) gradient descent is performed to minimize the Kullbeck-Leibler Divergence (KLD) and drive the particles to fit the true target distribution. Given an unnormalized log-density function $l(\mathbf{z})$, and an initial set of samples $\{\mathbf{z}_i^0\}_{i=1}^n$, we can iteratively update these samples as follows to approximate the distribution specified by l:

$$\begin{aligned} \mathbf{z}_i^{t+1} &= \mathbf{z}_i^t + \epsilon_t \hat{\phi}^*(\mathbf{z}_i^t),\ t = 0, 1, ..., T-1 \text{ where} \\ \hat{\phi}^*(\mathbf{z}) &= \frac{1}{n} \sum_{j=1}^{n} \left[k(\mathbf{z}_j^t, \mathbf{z}) \nabla_{\mathbf{z}_j^t} l(\mathbf{z}_j^t) + \nabla_{\mathbf{z}_j^t} k(\mathbf{z}_j^t, \mathbf{z}) \right], \end{aligned} \tag{2.15}$$

where $k(\cdot, \cdot)$ is a positive definite kernel (*e.g.*, RBF kernel) and ϵ_t the step size at the t-th iteration. We formulate the optimization process as sampling from an unnormalized log-density and utilize the mode-exploration ability of SVGD to propose high-quality candidates for the BBO problem.

2.11.5.3 Interpretation of Optimization as Gradient-Based Sampling

In order to extrapolate as far as possible, while still staying on the actual manifold of high-value observations, we need to measure the validity of the generated samples $\mathbf{x}$ as in Trabucco et al. [101]. We identify the value of y from $p_{\beta,y}(y|\mathbf{z})$ (forward model) serves as one good indicator as in [45]. The generated samples $\mathbf{x}$, associated with function values similar to existing y values in the offline dataset, are likely in-distribution; those where the $p_{\beta,y}(y|\mathbf{z})$ predicts a very different score (often erroneously large, see Fig. 2.4 when $y > 1$) can be too far outside the training distribution [43, 45, 59]. In addition, ideally, after training we have a well-learned and informative prior model $p_{\boldsymbol{\alpha}}$ that i) captures the multiple possible modes of the one-to-many inverse mapping $p_{\boldsymbol{\theta}}(\mathbf{x}|y)$, and ii) faithfully assigns the function

values to observations through the joint latent space (Fig. 2.3 and 2.4). Taking together, we can optimize over $\mathbf{z}$ in the latent space to find the best, most trustworthy $\mathbf{x}$ subject to the following constraints: i) $\mathbf{z}$ has a high likelihood under the prior and ii) the predicted function value associated with $\mathbf{z}$ is not too different from the value $y_{\max}$. This can be formulated as the following optimization problem:

$$\arg\max_{y(\mathbf{z})} y = g_{\boldsymbol{\beta}}(\mathbf{z}) \; s.t. \; p_{\boldsymbol{\beta}}(y_{\max}|\mathbf{z}) > \epsilon_1, \; p_{\boldsymbol{\alpha}}(\mathbf{z}) > \epsilon_2. \tag{2.16}$$

The above constrained optimization problem is conceptually similar to the formulation mentioned in Kumar & Levine [45] Sec. 3.2. The $p_{\boldsymbol{\beta}}(y_{\max}|\mathbf{z}) > \epsilon_1$ constraint enforces that $\mathbf{z}$ stays on the manifold of high-value observations as we discussed; the $p_{\boldsymbol{\alpha}}(\mathbf{z}) > \epsilon_2$ constraint encourages extrapolating beyond the best score in the offline dataset $\mathcal{D}$ by providing meaningful gradient during the optimization.

In our set-up, the unnormalized log-density of $p_{\boldsymbol{\beta},y}(y|\mathbf{z})p_{\boldsymbol{\alpha}}(\mathbf{z})$ given the best offline function value $y_{\max}$ can be written as $-\lambda_1\|y_{\max} - g_{\boldsymbol{\beta},y}(\mathbf{z})\|_2^2 + \lambda_2 \left(\sum_{k=0}^{K} \tilde{f}_{\boldsymbol{\alpha}_k}(\mathbf{z}) - \frac{1}{2}\|\mathbf{z}\|_2^2\right)$, where λ_1 acts as an inverse-variance weight associated with $p_{\boldsymbol{\beta},y}(y|\mathbf{z}) = \mathcal{N}(g_{\boldsymbol{\beta},y}(\mathbf{z}), \sigma_y^2\mathbf{I})$ and λ_2 re-weights the prior term. This is equivalent to optimizing the Lagrangian of Eq. (2.16), with the density constraints modified to be log-densities. In practice, we employ LD or SVGD to fully utilize the informative latent space during optimization. Ideally, the dual variables, λ_1 and λ_2 are supposed to be optimized together with y and $\mathbf{z}$, however, we find it convenient to choose them to be fixed as a constant throughout, at $\lambda_1 = 20.0$ and $\lambda_2 = 1.0$, analogous to penalty methods in constrained optimization. The same values of λ_1 and λ_2 are used for all domains in our experiments.

2.11.6 2D Branin Function

Branin is a well-known function for benchmarking optimization methods. We consider the negative of the standard 2D Branin function in the range $x_1 \in [-5, 10], x_2 \in [0, 15]$:

$$f_{br}(x_1, x_2) = -a(x_2 - bx_1^2 + cx_1 - r)^2 - s(1-t)\cos x_1 - s, \tag{2.17}$$

where $a = 1$, $b = \frac{5.1}{4\pi^2}$, $c = \frac{5}{\pi}$, $r = 6$, $s = 10$, and $t = \frac{1}{8\pi}$. In this square region, f_{br} has three global maximas, $(-\pi, 12.275)$, $(\pi, 2.275)$, and $(9.425, 2.475)$; with the maximum value of -0.398 (Fig. 2.2). The Gradient Ascent (GA) baseline uses the offline dataset to train a forward prediction model parameterized by a 2-layer MLP mapping $\mathbf{x}$ to y and then performs gradient ascent on $\mathbf{x}$ to infer its optima. DDOM and BONET follow their default implementation. For the typical forward method, GA in the data space, the problem of assigning an input $\mathbf{x}$ with erroneously large value is clear even in this 2D toy example; the proposed points often violates the square domain constraints. This can be partly seen in the largest std. in Tab. 2.6.

2.11.7 Design-Bench

2.11.7.1 Task Overview

The goal of **TF-Bind-8** and **TF-Bind-10** is to optimize for a DNA sequence that has a maximum affinity to bind with a particular transcription factor. The sequences are of length 8 for **TF-Bind-8** and 10 for **TF-Bind-10**, where each element in the sequence is one of 4 bases. **ChEMBL** optimizes drugs for specific chemical properties. In **D'Kitty** and **Ant Morphology**, one need to optimize for the morphology of robots. In **Superconductor**, the goal is to find a material with a high critical temperature. These tasks contain neither personally identifiable nor offensive information. We present detailed information on the tasks we evaluate on in Design-Bench [102] as below in Tab. 2.18. **SIZE** denotes the training dataset size, and **NUM. TOKS** denotes the number of tokens for discrete tasks.

Table 2.18: **Design-Bench dataset statistics.**

TASK	SIZE	INPUT DIM.	NUM. TOKS	TASK MAX
TF-Bind-8	32898	8	4	1.0
TF-Bind-10	10000	10	4	2.128
ChEMBL	441	31	185	443000.0
D'Kitty	10004	56	-	340.0
Ant	10004	60	-	590.0
Superconductor	17014	86	-	185.0

2.11.7.2 HopperController & NAS

As in Krishnamoorthy et al. [43] and Mashkaria et al. [59], we exclude the HopperController task in our results due to significant inconsistencies between the offline dataset values and the values obtained when running the oracle (see Fig. 2.8 and 2.9). This is a known bug in Design-bench (see details in the link). Therefore, we decided not to include Hopper in our evaluation and analysis. Following Krishnamoorthy et al. [43], we exclude NAS due to excessive computation cost required beyond our budget for evaluating across multiple seeds.

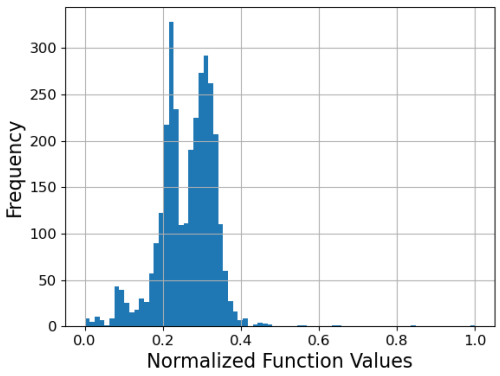


Figure 2.8: Histogram of normalized function values in the Hopper dataset. The distribution is highly skewed towards low function values (plot from Mashkaria et al. [59]).

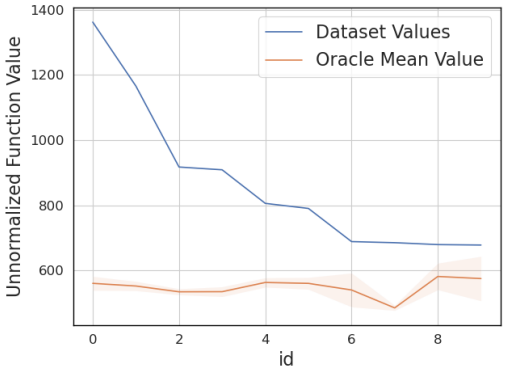


Figure 2.9: Dataset values vs Oracle values for top 10 points. We can see from mean and standard deviation over 20 runs that oracle is noisy (plot from Mashkaria et al. [59]).

2.11.7.3 Baselines

As mentioned in Sec. 2.4.3.2, we compare our method with three groups of baselines: **i)** sampling via generative inverse models with different parameterizations, including CbAS

[5], Auto.CbAS [14], MIN [45] and DDOM to validate our model design; **ii)** gradient(-like) updating from existing designs, *e.g.*, GA-based [7, 8, 15, 83, 101, 120] and BONET and **iii)** baselines mentioned in [102]. The generative inverse models learn $p_{\boldsymbol{\theta}}(\mathbf{x}|y)$ with different parameterizations and then perform conditional sampling of the high-value designs for BBO. CbAS [5] directly models $p_{\boldsymbol{\theta}}(\mathbf{x}|y)$, while introducing the joint distribution of $\mathbf{x}$ and $\mathbf{z}$ to facilitate importance sampling. It trains a VAE model to parameterize the distribution using identified high-quality inputs $\mathbf{x}$ (by thresholding); it gradually adapts the distribution to the high-value part by refining the threshold. MIN [45] learns $p_{\boldsymbol{\theta}}(\mathbf{x}|y) \propto \mathbb{E}_{p(\mathbf{z}|y)}[p(\mathbf{x}|\mathbf{z}, y)]$ with a conditional GAN-like model to instantiate $p(\mathbf{x}|\mathbf{z}, y)$ [22, 63]. The method optimizes over $\mathbf{z}$ given the offline dataset maximum $y_{\max}$ and map $\mathbf{z}$ back to $\mathbf{x}$ for BBO solutions. The optimization process over $\mathbf{z}$ draw approximate samples from $p(\mathbf{z}|y)$. DDOM [43] consider directly parameterizing the inverse mapping $p_{\boldsymbol{\theta}}(\mathbf{x}|y)$ with a conditional diffusion model in the input design space utilizing the expressiveness of DDPMs [31]. The second group includes the gradient(-like) updating methods, and has the flavor of forward methods in general. The baseline methods include: (i) Gradient Ascent: operates directly on the learned surrogates to propose new input designs via simple gradient ascent; (ii) COMs [101]: regularizes the NN-parameterized surrogate by assigning lower function values to designs obtained during the gradient ascent process; (iii) ROMA [119]: incorporates prior knowledge about function smoothness into the surrogate and optimizes the design against the proxy; (iv) NEMO [15]: leverages normalized maximum likelihood to constrain the distance between the surrogate and the ground-truth; (v) BDI [7]: proposes to distill the information from the static dataset into the high-scoring design; (vi) IOM [83]: enforces the invariance between the representations of the static dataset and generated designs to achieve a natural trade-off; (vii) ICT [120] explores using a pseudo-labeler to generate valuable data for fine-tuning the surrogate. BONET [59] trains a transformer-based model to mimic the optimization trajectories of black-box optimizers, and rolls out evaluation trajectories from the trained model for optimization.

In addition to these recent works, we also compare with several baseline methods described in Trabucco et al. [102]: 1) GP-qEI [109]: instantiates BO using a Gaussian Process as the uncertainty quantifier and the quasi-Expected Improvement (q-EI) acquisition function, 2) CMA-ES [28]: an evolutionary algorithm that maintains a belief distribution over the optimal candidates; it gradually refines the distribution by adapting the covariance matrix towards optimal candidates using feedback from the learned surrogate, and 3) REINFORCE [108]: first learns a proxy function and then optimizes the input space distribution by leveraging the proxy and the policy-gradient estimator. Since we are in the offline setting, for active methods like BO, we follow the procedure of [102] and optimize on a surrogate model trained on the offline dataset.

For $Q = 128$ results in Tab. 2.23, we additionally include ExPT [68] and BOOTGEN [37] for rough reference. Of note, these two methods have slightly different set-ups, and are therefore not directly comparable to other baselines and our method. ExPT [68] focuses extensively on few-shot learning scenarios with models pre-trained on larger datasets. BOOTGEN [37] focuses specifcially on optimizing biological sequences.

2.11.7.4 Additional Results & Analysis

Proof-of-concept results for data efficiency in BBO First, we uniformly sample $N = 5000$ and $N = 50$ points from the branin function domain for training. We observe that our method i) performs better in these experiments than BONET [59] and gradient ascent, and ii) demonstrates less performance drop compared with BONET when dataset size is significantly reduced. Further, we use the D'Kitty dataset and withhold an $x\%$ size subsection of data, *i.e.*, 0% represents the full dataset. We have similar observation on these datasets summarized in Tabs. 2.19 and 2.20, where our method shows much less performance drop compared with the transformer-based method BONET.

Table 2.19: **Results on (reduced size) Branin function dataset.**

N	$\mathcal{D}$ (best)	Grad. Ascent	BONET	Ours
5000	-6.199	-3.95 ± 4.26	-1.79 ± 0.84	$\mathbf{-0.41 \pm 0.13}$
50	-6.231	-4.64 ± 3.17	-2.13 ± 0.15	$\mathbf{-0.43 \pm 0.12}$

Table 2.20: **Results on (reduced size) D'Kitty dataset.** $x\%$ indicates the proportion of training data withheld, *i.e.*, 0% represents the full dataset.

	0%	90%	99%
BONET	285.11 ± 15.13	274.11 ± 7.57	$\mathbf{241.17 \pm 18.07}$
Ours	293.33 ± 7.45	289.67 ± 7.86	$\mathbf{283.65 \pm 9.23}$

2.11.7.5 Additional results for $Q = 256$

Unnormalized results For reference, we present the unnormalized results of Tab. 2.7 in Tab. 2.21.

Table 2.21: **Results on design-bench. Unnormalized** results with a budget $Q = 256$.

BASELINE	TFBIND8	TFBIND10	CHEMBL	SUPERCON.	ANT	D'KITTY
$\mathcal{D}$ (best)	0.439	0.00532	383700.000	74.0	165.326	199.363
GP-qEI	0.824 ± 0.086	0.675 ± 0.043	387950.000 ± 0.000	92.686 ± 3.944	480.049 ± 0.000	213.816 ± 0.000
CMA-ES	0.933 ± 0.035	0.848 ± 0.136	388400.000 ± 400.000	90.821 ± 0.661	$\mathbf{1016.409 \pm 906.407}$	4.700 ± 2.230
REINFORCE	0.959 ± 0.013	0.692 ± 0.113	388400.000 ± 2100.000	89.027 ± 3.093	-131.907 ± 41.003	-301.866 ± 246.284
Gradient Ascent	$\underline{0.981 \pm 0.015}$	0.770 ± 0.154	390050.000 ± 2000.000	93.252 ± 0.886	-54.955 ± 33.482	226.491 ± 21.120
COMs	0.964 ± 0.020	0.750 ± 0.078	390200.000 ± 500.000	78.178 ± 6.179	540.603 ± 20.205	277.888 ± 7.799
BONET	0.975 ± 0.004	0.855 ± 0.139	$\underline{391000.000 \pm 1900.000}$	80.845 ± 4.087	567.042 ± 11.653	$\underline{285.110 \pm 15.130}$
CbAS	0.958 ± 0.018	0.761 ± 0.067	389000.000 ± 500.000	83.178 ± 15.372	468.711 ± 14.593	213.917 ± 19.863
MINs	0.938 ± 0.047	0.770 ± 0.177	390950.000 ± 200.000	89.469 ± 3.227	533.636 ± 17.938	272.675 ± 11.069
DDOM	0.971 ± 0.005	$\underline{0.885 \pm 0.367}$	387950.000 ± 1050.000	$\underline{103.600 \pm 8.139}$	548.227 ± 11.725	250.529 ± 10.992
Ours	$\mathbf{0.990 \pm 0.003}$	$\mathbf{1.344 \pm 0.340}$	$\mathbf{392149.225 \pm 3821.853959}$	$\mathbf{104.848 \pm 3.113}$	$\underline{572.490 \pm 7.277}$	$\mathbf{293.327 \pm 7.452}$

50-th percentile results We present the normalized results with $Q = 256$ at 50-th percentile to further demonstrate the effectiveness of our approach in Tab. 2.22.

Table 2.22: **Normalized results on design-bench.** 50-th percentile results with $Q = 256$. Baseline numbers from [59]. DDOM [43] does not report 50-th percentile results; we are unable to find public model weights corresponding with the results in Tab. 2.7 for DDOM and therefore omit DDOM in this table.

BASELINE	TFBIND8	TFBIND10	CHEMBL	SUPERCON.	ANT	D'KITTY	MEAN SCORE$^{\uparrow}$	MNR$^{\downarrow}$
GP-qEI	0.443 ± 0.004	0.494 ± 0.002	0.299 ± 0.002	0.272 ± 0.006	0.754 ± 0.004	0.633 ± 0.000	0.491 ± 0.016	4.8
CMA-ES	0.543 ± 0.007	0.483 ± 0.011	0.376 ± 0.004	-0.051 ± 0.004	0.685 ± 0.018	0.633 ± 0.000	0.466 ± 0.020	5.2
REINFORCE	0.450 ± 0.003	0.472 ± 0.000	0.470 ± 0.017	0.146 ± 0.009	0.307 ± 0.002	0.633 ± 0.000	0.083 ± 0.004	5.7
Gradient Ascent	0.572 ± 0.024	0.470 ± 0.004	0.463 ± 0.022	0.141 ± 0.010	0.637 ± 0.148	0.633 ± 0.000	0.478 ± 0.029	5.1
COMs	0.492 ± 0.009	0.472 ± 0.012	0.365 ± 0.026	0.525 ± 0.018	0.885 ± 0.002	0.633 ± 0.000	0.522 ± 0.034	4.6
BONET	0.505 ± 0.055	0.496 ± 0.037	0.369 ± 0.015	$\mathbf{0.819 \pm 0.032}$	$\mathbf{0.907 \pm 0.020}$	0.630 ± 0.000	0.614 ± 0.035	3.4
CbAS	0.422 ± 0.007	0.458 ± 0.001	0.111 ± 0.009	0.384 ± 0.010	0.752 ± 0.003	0.633 ± 0.000	0.436 ± 0.008	6.5
MINs	0.425 ± 0.011	0.471 ± 0.004	0.330 ± 0.011	0.651 ± 0.010	0.890 ± 0.003	0.633 ± 0.000	0.547 ± 0.005	4.8
Ours	$\mathbf{0.896 \pm 0.033}$	$\mathbf{0.507 \pm 0.012}$	$\mathbf{0.373 \pm 0.002}$	0.715 ± 0.004	$\mathbf{0.907 \pm 0.008}$	$\mathbf{0.630 \pm 0.000}$	$\mathbf{0.672 \pm 0.009}$	**1.2**

2.11.7.6 Additional results for $Q = 128$

25-th, 50-th, 75-th and 100-th percentile results We present the normalized results with $Q = 128$ at 100-th and 50-th percentile to further demonstrate the effectiveness of our approach in Tabs. 2.23 and 2.25. Additionally, we provide results at 25-th and 75-th percentiles for reference in Tab. 2.24.

Table 2.23: **Normalized design-bench results with $Q = 128$ at 100-th percentile.** Baseline numbers from [8, 37, 68, 120]. **ChEMBL** dataset is excluded following baseline set-ups. * indicates that the baseline has slightly different set-ups; the numbers are for rough reference (see Sec. 2.11.7.3 for more details).

BASELINE	TFBIND8	TFBIND10	SUPERCON.	ANT	D'KITTY	MEAN SCORE$^{\uparrow}$	MNR$^{\downarrow}$
$\mathcal{D}$ (best)	0.439	0.467	0.399	0.565	0.884	-	-
GP-qEI	0.798 ± 0.083	0.652 ± 0.038	0.402 ± 0.034	0.819 ± 0.000	0.896 ± 0.000	0.713	14.2
CMA-ES	0.953 ± 0.022	0.670 ± 0.023	0.465 ± 0.024	$\mathbf{1.214 \pm 0.732}$	0.724 ± 0.001	0.805	8.6
REINFORCE	0.948 ± 0.028	0.663 ± 0.034	0.481 ± 0.013	0.266 ± 0.032	0.562 ± 0.196	0.584	12.4
*BOOTGEN	$\underline{0.979 \pm 0.001}$	-	-	-	-	-	-
*ExPT	0.933 ± 0.036	0.677 ± 0.048	-	$\underline{0.970 \pm 0.004}$	$\mathbf{0.973 \pm 0.005}$	-	-
Grad. Ascent	0.886 ± 0.035	0.647 ± 0.021	0.495 ± 0.011	0.934 ± 0.011	0.944 ± 0.017	0.781	10.2
COMS	0.496 ± 0.065	0.622 ± 0.003	0.491 ± 0.028	0.856 ± 0.040	0.938 ± 0.015	0.681	13.4
BONET	0.911 ± 0.005	$\underline{0.756 \pm 0.006}$	0.500 ± 0.002	0.927 ± 0.002	0.954 ± 0.000	0.810	6.8
ROMA	0.924 ± 0.040	0.666 ± 0.035	0.510 ± 0.015	0.917 ± 0.030	0.927 ± 0.013	0.789	8.2
NEMO	0.943 ± 0.005	0.711 ± 0.021	0.502 ± 0.002	0.958 ± 0.011	0.954 ± 0.007	0.814	5.2
BDI	0.870 ± 0.000	0.605 ± 0.000	0.513 ± 0.000	0.906 ± 0.000	0.919 ± 0.000	0.763	11.6
IOM	0.878 ± 0.069	0.648 ± 0.023	$\underline{0.520 \pm 0.018}$	0.918 ± 0.031	0.945 ± 0.012	0.782	8.6
ICT	0.958 ± 0.008	0.691 ± 0.023	0.503 ± 0.017	0.961 ± 0.007	$\underline{0.968 \pm 0.020}$	0.816	3.4
Tri-mtring	$\underline{0.970 \pm 0.001}$	0.722 ± 0.017	0.514 ± 0.018	0.948 ± 0.014	0.966 ± 0.010	$\underline{0.824}$	$\underline{3}$
CbAS	0.927 ± 0.051	0.651 ± 0.060	0.503 ± 0.069	0.876 ± 0.031	0.892 ± 0.008	0.770	10.8
Auto.CbAS	0.910 ± 0.044	0.630 ± 0.045	0.421 ± 0.045	0.882 ± 0.045	0.906 ± 0.006	0.745	13
MINs	0.905 ± 0.052	0.616 ± 0.021	0.499 ± 0.017	0.445 ± 0.080	0.892 ± 0.011	0.671	13.6
DDOM	0.957 ± 0.006	0.657 ± 0.006	0.495 ± 0.012	0.940 ± 0.004	0.935 ± 0.001	0.797	8
Ours	$\mathbf{0.983 \pm 0.011}$	$\mathbf{0.771 \pm 0.096}$	$\mathbf{0.559 \pm 0.024}$	0.954 ± 0.008	0.954 ± 0.004	**0.844**	**2**

Table 2.24: **Normalized design-bench 25-th and 75-th results with $Q = 128$.**

Percentile	TFBIND8	TFBIND10	SUPERCON.	ANT	D'KITTY
25-th	0.635 ± 0.020	0.453 ± 0.007	0.351 ± 0.019	0.527 ± 0.033	0.878 ± 0.002
75-th	0.873 ± 0.006	0.945 ± 0.012	0.409 ± 0.013	0.832 ± 0.012	0.910 ± 0.002

2.11.7.7 Extended discussion for ablation study

As mentioned in Sec. 2.4.3, we provide ablative results on noise magnitude M, number of intermediate stages K and query budget Q in Tabs. 2.8, 2.26 and 2.27. Of note, the

Table 2.25: **Normalized design-bench 50-th percentile results with $Q = 128$.** Baseline numbers from [8, 37, 68, 120]. **ChEMBL** dataset is excluded following baseline set-ups. * indicates that the baseline has slightly different set-ups; the numbers are for rough reference (see Sec. 2.11.7.3 for more details).

BASELINE	TFBIND8	TFBIND10	SUPERCON.	ANT	D'KITTY	MEAN SCORE$^{\uparrow}$	MNR$^{\downarrow}$
$\mathcal{D}$ (best)	0.439	0.467	0.399	0.565	0.884	-	-
GP-qEI	0.439 ± 0.000	0.467 ± 0.000	0.300 ± 0.015	0.567 ± 0.000	0.883 ± 0.000	0.531	11.2
CMA-ES	0.537 ± 0.014	0.484 ± 0.014	0.379 ± 0.003	-0.045 ± 0.004	0.684 ± 0.016	0.408	10.2
REINFORCE	0.462 ± 0.021	0.475 ± 0.008	0.463 ± 0.016	0.138 ± 0.032	0.356 ± 0.131	0.379	10.8
*BOOTGEN	0.833 ± 0.007	-	-	-	-	-	-
*ExPT	0.473 ± 0.014	0.477 ± 0.014	-	$\mathbf{0.705 \pm 0.018}$	$\mathbf{0.902 \pm 0.006}$	-	-
Grad. Ascent	0.532 ± 0.017	0.529 ± 0.027	0.339 ± 0.015	0.564 ± 0.014	0.877 ± 0.005	0.568	7.4
COMS	0.439 ± 0.000	0.466 ± 0.002	0.316 ± 0.022	0.568 ± 0.002	0.883 ± 0.002	0.534	10.6
BONET	0.505 ± 0.004	0.465 ± 0.002	$\mathbf{0.470 \pm 0.004}$	0.620 ± 0.003	0.897 ± 0.000	0.591	5.4
ROMA	0.555 ± 0.020	0.512 ± 0.020	0.372 ± 0.019	0.479 ± 0.041	0.853 ± 0.007	0.554	7.6
NEMO	0.548 ± 0.017	0.516 ± 0.020	0.322 ± 0.008	0.593 ± 0.000	0.885 ± 0.000	0.539	10
BDI	0.439 ± 0.000	0.476 ± 0.000	0.412 ± 0.000	0.474 ± 0.000	0.855 ± 0.000	0.546	9.4
IOM	0.439 ± 0.000	0.477 ± 0.010	0.352 ± 0.021	0.509 ± 0.033	0.876 ± 0.006	0.531	9.8
ICT	0.551 ± 0.013	$\mathbf{0.541 \pm 0.004}$	0.399 ± 0.012	0.592 ± 0.025	0.874 ± 0.005	0.591	5.2
Tri-mtring	0.609 ± 0.021	0.527 ± 0.008	0.355 ± 0.003	0.606 ± 0.007	0.886 ± 0.001	0.597	4.2
CbAS	0.428 ± 0.010	0.463 ± 0.007	0.111 ± 0.017	0.384 ± 0.016	0.753 ± 0.008	0.428	15
Auto.CbAS	0.419 ± 0.007	0.461 ± 0.007	0.131 ± 0.010	0.364 ± 0.014	0.736 ± 0.025	0.422	15.8
MINs	0.421 ± 0.015	0.468 ± 0.006	0.336 ± 0.016	0.618 ± 0.040	0.887 ± 0.004	0.546	8.8
DDOM	0.553 ± 0.002	0.488 ± 0.001	0.295 ± 0.001	0.590 ± 0.003	0.870 ± 0.001	0.559	8.6
Ours	$\mathbf{0.873 \pm 0.006}$	0.499 ± 0.002	0.370 ± 0.004	$\mathbf{0.704 \pm 0.013}$	$\mathbf{0.897 \pm 0.003}$	**0.669**	**3**

$K = 1, M = 100$ row in Tab. 2.8 means learning a LEBM w/ Eq. (2.5) alone, w/o ratio decomposition. We can see that the variant delivers significantly better results than MLE-LEBM w/ SVGD sampler. This indicates the effectiveness of the noise-intensified objective. $K = 6, M = 1$ row means learning with the original NCE objective but with ratio decomposition, w/o intensifying the noise distributions. We can see that simply combining the NCE objective with variational learning of LEBM only achieves inferior performance compared with learning LEBM with the N^2CE objective.

We further provide detailed ablation of M in Tab. 2.26. We can see that greater M, *e.g.*, $M = 100$ or $M = 1000$ is significantly better than smaller M, *e.g.*, $M = 1$ or $M = 10$. However, larger M incurs larger memory consumption based on our implementation in Sec. 2.10. It also has a pitfall of leading to larger approximation variance based on our analysis in Proposition 2.3.3. We therefore choose $M = 100$ for all experiments. These results further confirm our assumption about the noise magnitude. We can see from Tab. 2.26 that larger number of intermediate stages K typically brings better performance. These results

demonstrate the need for ratio decomposition, aligned with our analysis in Sec. 2.3.3. In addition, since we are using the same network architecture for ablating different Ks, the network capacity is fixed. Larger number of intermediate stages may require a larger network to perform well (circumventing competition between stages), as we are learning the ensemble of ratio estimators using the shared network. SVGD in the data space typically requires a relatively large population of initial samples to perform well. In our set-up, we set the query budget Q to the population size and run SVGD in the latent space. We can see from Tab. 2.27 that our method is robust to Q by pulling SVGD sampling back to the latent space.

Table 2.26: **Ablation of K, M on SPRCON. (avg. over 5 runs).**

$M = 100$	$K = 2$	$K = 4$	$K = 6$	$K = 8$
	$0.427_{\pm 0.013}$	$0.557_{\pm 0.029}$	$\mathbf{0.567}_{\pm 0.017}$	$0.527_{\pm 0.049}$
$K = 6$	$M = 1$	$M = 10$	$M = 100$	$M = 1000$
	$0.497_{\pm 0.044}$	$0.486_{\pm 0.023}$	$\mathbf{0.567}_{\pm 0.017}$	$0.548_{\pm 0.046}$

Table 2.27: **Ablation of Q on D'Kitty (avg. over 5 runs).**

$Q = 2$	$Q = 8$	$Q = 32$	$Q = 128$	$Q = 256$
$0.906_{\pm 0.009}$	$0.925_{\pm 0.013}$	$0.937_{\pm 0.006}$	$0.954_{\pm 0.004}$	$\mathbf{0.961}_{\pm 0.006}$

Random baselines One valid concern is that our framework solves the offline BBO problem by simply memorizing the best points in the offline dataset and proposing new points close to those best points during evaluation utilizing the randomness in the sampling process. To see whether this is the case, we follow Mashkaria et al. [59] to perform a simple experiment on the **D'Kitty** task. We include random baselines by similarly choosing a small hypercube domain around the optimal point in the offline dataset. These baselines then uniformly sample 256 random points in the cube as the proposed candidates; this can be seen as a offline dataset oracle w/ noise model. In Tab. 2.28, we show the results for different widths of this hypercube ranging from 0 to 0.1. 0 width means returning the best point in the offline dataset. We can see that the best result (226.10) from these noisy oracle models is significantly lower than the result from our model (304.36). We can also see from the trend in max, mean and std. values that simply extending the width of cube for random search is hopeless for solving the offline BBO task. This is very likely due to the high-dimensionality and highly-multimodal nature of the black-box function input space.

The comparison between these random baselines and our method suggests that our method does find an informative latent space for effectively extrapolating beyond the offline dataset best designs. The prior model in our formulation provides meaningful gradient for exploring the underlying high-function-value manifold.

Table 2.28: **Random baseline results on the D'Kitty dataset.** Results from 5 runs.

Cube Width	0.00	0.005	0.01	0.05	0.1	Ours
Max$^{\uparrow}$	199.23	212.66	222.44	226.10	209.00	**304.36**
Mean$^{\uparrow}$	199.23	190.68	182.13	−169.62	−368.71	**206.76**
Std.$^{\downarrow}$	0.00	9.28	12.21	331.00	261.37	**63.26**

Use of Large Language Models

During the preparation of this manuscript, we used Large Language Models (LLMs) to assist with language polishing, organization of sections, and improving clarity of exposition. All technical contributions, including theoretical results, experimental design, and analysis, were conceived and carried out by the authors. The LLMs were not used to generate novel research content, proofs, or experimental results.

Bibliography

[1] Jyoti Aneja, Alex Schwing, Jan Kautz, and Arash Vahdat. A contrastive learning approach for training variational autoencoder priors. *Advances in neural information processing systems*, 34:480–493, 2021.

[2] Christof Angermueller, David Dohan, David Belanger, Ramya Deshpande, Kevin Murphy, and Lucy Colwell. Model-based reinforcement learning for biological sequence design. In *International conference on learning representations*, 2019.

[3] Christof Angermueller, David Belanger, Andreea Gane, Zelda Mariet, David Dohan, Kevin Murphy, Lucy Colwell, and D Sculley. Population-based black-box optimization for biological sequence design. In *International Conference on Machine Learning*, pp. 324–334. PMLR, 2020.

[4] Andrew Brock, Theodore Lim, James M Ritchie, and Nick Weston. Neural photo editing with introspective adversarial networks. *arXiv preprint arXiv:1609.07093*, 2016.

[5] David Brookes, Hahnbeom Park, and Jennifer Listgarten. Conditioning by adaptive sampling for robust design. In *International conference on machine learning*, pp. 773–782. PMLR, 2019.

[6] Ciwan Ceylan and Michael U Gutmann. Conditional noise-contrastive estimation of unnormalised models. In *International Conference on Machine Learning*, pp. 726–734. PMLR, 2018.

[7] Can Chen, Yingxueff Zhang, Jie Fu, Xue Steve Liu, and Mark Coates. Bidirectional learning for offline infinite-width model-based optimization. *Advances in Neural Information Processing Systems*, 35:29454–29467, 2022.

[8] Can Sam Chen, Christopher Beckham, Zixuan Liu, Xue Steve Liu, and Chris Pal. Parallel-mentoring for offline model-based optimization. *Advances in Neural Information Processing Systems*, 36, 2024.

[9] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pp. 1597–1607. PMLR, 2020.

[10] Nando de Freitas and Ziyu Wang. Bayesian optimization in high dimensions via random embeddings. 2013.

[11] Yilun Du and Igor Mordatch. Implicit generation and generalization in energy-based models. *arXiv preprint arXiv:1903.08689*, 2019.

[12] Yilun Du, Shuang Li, Joshua Tenenbaum, and Igor Mordatch. Improved contrastive divergence training of energy based models. In *International Conference on Machine Learning (ICML)*, 2021.

[13] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7), 2011.

[14] Clara Fannjiang and Jennifer Listgarten. Autofocused oracles for model-based design. *Advances in Neural Information Processing Systems*, 33:12945–12956, 2020.

[15] Justin Fu and Sergey Levine. Offline model-based optimization via normalized maximum likelihood estimation. In *International Conference on Learning Representations*, 2020.

[16] Ruiqi Gao, Erik Nijkamp, Diederik P Kingma, Zhen Xu, Andrew M Dai, and Ying Nian Wu. Flow contrastive estimation of energy-based models. In *Proceedings of the*

IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 7518–7528, 2020.

[17] Ruiqi Gao, Yang Song, Ben Poole, Ying Nian Wu, and Diederik P Kingma. Learning energy-based models by diffusion recovery likelihood. In *International Conference on Learning Representations (ICLR)*, 2020.

[18] Marta Garnelo, Dan Rosenbaum, Christopher Maddison, Tiago Ramalho, David Saxton, Murray Shanahan, Yee Whye Teh, Danilo Rezende, and SM Ali Eslami. Conditional neural processes. In *International conference on machine learning*, pp. 1704–1713. PMLR, 2018.

[19] Marta Garnelo, Jonathan Schwarz, Dan Rosenbaum, Fabio Viola, Danilo J Rezende, SM Eslami, and Yee Whye Teh. Neural processes. *arXiv preprint arXiv:1807.01622*, 2018.

[20] R Garnett, M Osborne, and P Hennig. Active learning of linear embeddings for gaussian processes. In *30th Conference on Uncertainty in Artificial Intelligence (UAI 2014)*, pp. 230–239. AUAI Press, 2014.

[21] Rafael Gómez-Bombarelli, Jennifer N Wei, David Duvenaud, José Miguel Hernández-Lobato, Benjamín Sánchez-Lengeling, Dennis Sheberla, Jorge Aguilera-Iparraguirre, Timothy D Hirzel, Ryan P Adams, and Alán Aspuru-Guzik. Automatic chemical design using a data-driven continuous representation of molecules. *ACS central science*, 4(2): 268–276, 2018.

[22] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.

[23] Aditya Grover and Stefano Ermon. Boosted generative models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[24] Michael Gutmann and Aapo Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pp. 297–304. JMLR Workshop and Conference Proceedings, 2010.

[25] Michael U Gutmann and Aapo Hyvärinen. Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics. *Journal of machine learning research*, 13(2), 2012.

[26] Tian Han, Yang Lu, Song-Chun Zhu, and Ying Nian Wu. Alternating back-propagation for generator network. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2017.

[27] Tian Han, Erik Nijkamp, Linqi Zhou, Bo Pang, Song-Chun Zhu, and Ying Nian Wu. Joint training of variational auto-encoder and latent energy-based model. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[28] Nikolaus Hansen. The cma evolution strategy: a comparing review. *Towards a new evolutionary computation: Advances in the estimation of distribution algorithms*, pp. 75–102, 2006.

[29] Trevor Hastie, Robert Tibshirani, Jerome H Friedman, and Jerome H Friedman. *The elements of statistical learning: data mining, inference, and prediction*, volume 2. Springer, 2009.

[30] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[31] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[32] Yiwen Huang, Aaron Gokaslan, Volodymyr Kuleshov, and James Tompkin. The gan is dead; long live the gan! a modern gan baseline. *arXiv preprint arXiv:2501.05441*, 2025.

[33] Kirthevasan Kandasamy, Jeff Schneider, and Barnabás Póczos. High dimensional bayesian optimisation and bandits via additive models. In *International conference on machine learning*, pp. 295–304. PMLR, 2015.

[34] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 4401–4410, 2019.

[35] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. *Advances in neural information processing systems*, 35:26565–26577, 2022.

[36] Hyunjik Kim, Andriy Mnih, Jonathan Schwarz, Marta Garnelo, Ali Eslami, Dan Rosenbaum, Oriol Vinyals, and Yee Whye Teh. Attentive neural processes. In *International Conference on Learning Representations*, 2018.

[37] Minsu Kim, Federico Berto, Sungsoo Ahn, and Jinkyoo Park. Bootstrapped training of score-conditioned generator for offline design of biological sequences. *Advances in Neural Information Processing Systems*, 36, 2024.

[38] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[39] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[40] Durk P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. *Advances in neural information processing systems*, 31, 2018.

[41] Deqian Kong, Yuhao Huang, Jianwen Xie, and Ying Nian Wu. Molecule design by latent prompt transformer. *arXiv preprint arXiv:2310.03253*, 2023.

[42] Deqian Kong, Bo Pang, Tian Han, and Ying Nian Wu. Molecule design by latent space energy-based modeling and gradual distribution shifting. In *Uncertainty in Artificial Intelligence*, pp. 1109–1120. PMLR, 2023.

[43] Siddarth Krishnamoorthy, Satvik Mehul Mashkaria, and Aditya Grover. Diffusion models for black-box optimization. In *International Conference on Machine Learning*, 2023.

[44] Alex Krizhevsky et al. Learning multiple layers of features from tiny images. 2009.

[45] Aviral Kumar and Sergey Levine. Model inversion networks for model-based optimization. *Advances in Neural Information Processing Systems*, 33:5126–5137, 2020.

[46] Rithesh Kumar, Sherjil Ozair, Anirudh Goyal, Aaron Courville, and Yoshua Bengio. Maximum entropy generators for energy-based models. *arXiv preprint arXiv:1901.08508*, 2019.

[47] Matt J Kusner, Brooks Paige, and José Miguel Hernández-Lobato. Grammar variational autoencoder. In *International conference on machine learning*, pp. 1945–1954. PMLR, 2017.

[48] Justin Lazarow, Long Jin, and Zhuowen Tu. Introspective neural networks for generative modeling. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2774–2783, 2017.

[49] Yann LeCun. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998.

[50] Yann LeCun, Sumit Chopra, Raia Hadsell, M Ranzato, Fujie Huang, et al. A tutorial on energy-based learning. *Predicting structured data*, 1(0), 2006.

[51] Cheng-Han Lee, Ziwei Liu, Lingyun Wu, and Ping Luo. Maskgan: Towards diverse and interactive facial image manipulation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[52] Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

[53] Bingbin Liu, Elan Rosenfeld, Pradeep Kumar Ravikumar, and Andrej Risteski. Analyzing and improving the optimization landscape of noise-contrastive estimation. In *International Conference on Learning Representations*, 2022.

[54] Qiang Liu and Dilin Wang. Stein variational gradient descent: A general purpose bayesian inference algorithm. *Advances in neural information processing systems*, 29, 2016.

[55] Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. *arXiv preprint arXiv:2209.03003*, 2022.

[56] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of the IEEE international conference on computer vision*, pp. 3730–3738, 2015.

[57] Xiaoyu Lu, Javier Gonzalez, Zhenwen Dai, and Neil D Lawrence. Structured variationally auto-encoded optimization. In *International conference on machine learning*, pp. 3267–3275. PMLR, 2018.

[58] Weijian Luo, Tianyang Hu, Shifeng Zhang, Jiacheng Sun, Zhenguo Li, and Zhihua Zhang. Diff-instruct: A universal approach for transferring knowledge from pre-trained diffusion models. *Advances in Neural Information Processing Systems*, 36:76525–76546, 2023.

[59] Satvik Mehul Mashkaria, Siddarth Krishnamoorthy, and Aditya Grover. Generative pre-training for black-box optimization. In *International Conference on Machine Learning*, pp. 24173–24197. PMLR, 2023.

[60] David McAllester and Karl Stratos. Formal limitations on the measurement of mutual information. In *International Conference on Artificial Intelligence and Statistics*, pp. 875–884. PMLR, 2020.

[61] Aditya Menon and Cheng Soon Ong. Linking losses for density ratio and class-probability estimation. In *International Conference on Machine Learning*, pp. 304–313. PMLR, 2016.

[62] Azalia Mirhoseini, Anna Goldie, Mustafa Yazgan, Joe Jiang, Ebrahim Songhori, Shen Wang, Young-Joon Lee, Eric Johnson, Omkar Pathak, Sungmin Bae, et al. Chip placement with deep reinforcement learning. *arXiv preprint arXiv:2004.10746*, 2020.

[63] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[64] A Mnih and Y Teh. A fast and simple algorithm for training neural probabilistic language models. In *International Conference on Machine Learning (ICML)*, volume 2, 2012.

[65] Riccardo Moriconi, Marc Peter Deisenroth, and KS Sesh Kumar. High-dimensional bayesian optimization using low-dimensional feature spaces. *Machine Learning*, 109: 1925–1943, 2020.

[66] Riccardo Moriconi, KS Sesh Kumar, and Marc Peter Deisenroth. High-dimensional bayesian optimization with projections using quantile gaussian processes. *Optimization Letters*, 14:51–64, 2020.

[67] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. 2011.

[68] Tung Nguyen, Sudhanshu Agrawal, and Aditya Grover. Expt: Synthetic pretraining for few-shot experimental design. *Advances in Neural Information Processing Systems*, 36, 2024.

[69] Vu Nguyen and Michael A Osborne. Knowing the what but not the where in bayesian optimization. In *International Conference on Machine Learning*, pp. 7317–7326. PMLR, 2020.

[70] XuanLong Nguyen, Martin J Wainwright, and Michael I Jordan. Estimating divergence functionals and the likelihood ratio by convex risk minimization. *IEEE Transactions on Information Theory*, 56(11):5847–5861, 2010.

[71] Erik Nijkamp, Mitch Hill, Song-Chun Zhu, and Ying Nian Wu. Learning non-convergent non-persistent short-run mcmc toward energy-based model. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[72] Erik Nijkamp, Mitch Hill, Tian Han, Song-Chun Zhu, and Ying Nian Wu. On the anatomy of mcmc-based maximum likelihood learning of energy-based models. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2020.

[73] Sebastian Nowozin, Botond Cseke, and Ryota Tomioka. f-gan: Training generative neural samplers using variational divergence minimization. *Advances in neural information processing systems*, 29, 2016.

[74] Bo Pang, Tian Han, Erik Nijkamp, Song-Chun Zhu, and Ying Nian Wu. Learning latent space energy-based prior model. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[75] Bo Pang, Tian Han, and Ying Nian Wu. Learning latent space energy-based prior model for molecule generation. *arXiv preprint arXiv:2010.09351*, 2020.

[76] Bo Pang, Tianyang Zhao, Xu Xie, and Ying Nian Wu. Trajectory prediction with latent belief energy-based model. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[77] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[78] Jan Peters and Stefan Schaal. Reinforcement learning by reward-weighted regression for operational space control. In *Proceedings of the 24th international conference on Machine learning*, pp. 745–750, 2007.

[79] Miika Pihlaja, Michael Gutmann, and Aapo Hyvärinen. A family of computationally efficient and simple estimators for unnormalized statistical models. In *Proceedings of the Twenty-Sixth Conference on Uncertainty in Artificial Intelligence*, pp. 442–449, 2010.

[80] Yury Polyanskiy and Yihong Wu. *Information theory: From coding to learning*. Cambridge university press, 2025.

[81] Ben Poole, Sherjil Ozair, Aaron Van Den Oord, Alex Alemi, and George Tucker. On variational bounds of mutual information. In *International Conference on Machine Learning*, pp. 5171–5180. PMLR, 2019.

[82] Ben Poole, Ajay Jain, Jonathan T Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. *arXiv preprint arXiv:2209.14988*, 2022.

[83] Han Qi, Yi Su, Aviral Kumar, and Sergey Levine. Data-driven offline decision-making via invariant representation learning. *Advances in Neural Information Processing Systems*, 35:13226–13237, 2022.

[84] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[85] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pp. 8748–8763. PMLR, 2021.

[86] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic back-propagation and approximate inference in deep generative models. In *International Conference on Machine Learning (ICML)*, 2014.

[87] Benjamin Rhodes, Kai Xu, and Michael U Gutmann. Telescoping density-ratio estimation. *Advances in neural information processing systems*, 33:4905–4916, 2020.

[88] Paul Rolland, Jonathan Scarlett, Ilija Bogunovic, and Volkan Cevher. High-dimensional bayesian optimization via additive models with overlapping groups. In *International conference on artificial intelligence and statistics*, pp. 298–307. PMLR, 2018.

[89] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10684–10695, 2022.

[90] Reuven Y Rubinstein. Optimization of computer simulation models with rare events. *European Journal of Operational Research*, 99(1):89–112, 1997.

[91] Reuven Y Rubinstein and Dirk P Kroese. *The cross-entropy method: a unified approach to combinatorial optimization, Monte-Carlo simulation, and machine learning*, volume 133. Springer, 2004.

[92] Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. *arXiv preprint arXiv:2202.00512*, 2022.

[93] Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P Adams, and Nando De Freitas. Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175, 2015.

[94] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. *Advances in neural information processing systems*, 25, 2012.

[95] Jasper Snoek, Oren Rippel, Kevin Swersky, Ryan Kiros, Nadathur Satish, Narayanan Sundaram, Mostofa Patwary, Mr Prabhat, and Ryan Adams. Scalable bayesian optimization using deep neural networks. In *International conference on machine learning*, pp. 2171–2180. PMLR, 2015.

[96] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations (ICLR)*, 2020.

[97] Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. In *International Conference on Machine Learning*, pp. 32211–32252. PMLR, 2023.

[98] Niranjan Srinivas, Andreas Krause, Sham Kakade, and Matthias Seeger. Gaussian process optimization in the bandit setting: no regret and experimental design. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, pp. 1015–1022, 2010.

[99] Masashi Sugiyama, Taiji Suzuki, and Takafumi Kanamori. *Density ratio estimation in machine learning*. Cambridge University Press, 2012.

[100] Jakub Tomczak and Max Welling. Vae with a vampprior. In *International Conference on Artificial Intelligence and Statistics*, pp. 1214–1223. PMLR, 2018.

[101] Brandon Trabucco, Aviral Kumar, Xinyang Geng, and Sergey Levine. Conservative objective models for effective offline model-based optimization. In *International Conference on Machine Learning*, pp. 10358–10368. PMLR, 2021.

[102] Brandon Trabucco, Xinyang Geng, Aviral Kumar, and Sergey Levine. Design-bench: Benchmarks for data-driven offline model-based optimization. In *International Conference on Machine Learning*, pp. 21658–21676. PMLR, 2022.

[103] Zhuowen Tu. Learning generative models via discriminative approaches. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–8. IEEE, 2007.

[104] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*, 2016.

[105] Zhendong Wang, Huangjie Zheng, Pengcheng He, Weizhu Chen, and Mingyuan Zhou. Diffusion-gan: Training gans with diffusion. *arXiv preprint arXiv:2206.02262*, 2022.

[106] Zhengyi Wang, Cheng Lu, Yikai Wang, Fan Bao, Chongxuan Li, Hang Su, and Jun Zhu. Prolificdreamer: High-fidelity and diverse text-to-3d generation with variational score distillation. *Advances in neural information processing systems*, 36:8406–8441, 2023.

[107] Max Welling and Yee W Teh. Bayesian learning via stochastic gradient langevin dynamics. In *International Conference on Machine Learning (ICML)*, 2011.

[108] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8:229–256, 1992.

[109] James T Wilson, Riccardo Moriconi, Frank Hutter, and Marc Peter Deisenroth. The reparameterization trick for acquisition functions. *arXiv preprint arXiv:1712.00424*, 2017.

[110] Zhisheng Xiao and Tian Han. Adaptive multi-stage density ratio estimation for learning latent space energy-based model. *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[111] Jianwen Xie, Yang Lu, Song-Chun Zhu, and Yingnian Wu. A theory of generative convnet. In *International Conference on Machine Learning*, pp. 2635–2644, 2016.

[112] Tianwei Yin, Michaël Gharbi, Richard Zhang, Eli Shechtman, Fredo Durand, William T Freeman, and Taesung Park. One-step diffusion with distribution matching distillation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 6613–6623, 2024.

[113] Sangwoong Yoon, Yung-Kyun Noh, and Frank Park. Autoencoding under normalization constraints. In *International Conference on Machine Learning*, pp. 12087–12097. PMLR, 2021.

[114] Sangwoong Yoon, Young-Uk Jin, Yung-Kyun Noh, and Frank Park. Energy-based models for anomaly detection: A manifold diffusion recovery approach. *Advances in Neural Information Processing Systems*, 36:49445–49466, 2023.

[115] Sangwoong Yoon, Himchan Hwang, Dohyun Kwon, Yung-Kyun Noh, and Frank Park. Maximum entropy inverse reinforcement learning of diffusion models with energy-based models. *Advances in Neural Information Processing Systems*, 37:24601–24624, 2024.

[116] Peiyu Yu, Sirui Xie, Xiaojian Ma, Yixin Zhu, Ying Nian Wu, and Song-Chun Zhu. Unsupervised foreground extraction via deep region competition. *Advances in Neural Information Processing Systems*, 34:14264–14279, 2021.

[117] Peiyu Yu, Sirui Xie, Xiaojian Ma, Baoxiong Jia, Bo Pang, Ruiqi Gao, Yixin Zhu, Song-Chun Zhu, and Ying Nian Wu. Latent diffusion energy-based model for interpretable text modeling. *arXiv preprint arXiv:2206.05895*, 2022.

[118] Peiyu Yu, Yaxuan Zhu, Sirui Xie, Xiaojian Shawn Ma, Ruiqi Gao, Song-Chun Zhu, and Ying Nian Wu. Learning energy-based prior model with diffusion-amortized mcmc. *Advances in Neural Information Processing Systems*, 36, 2024.

[119] Sihyun Yu, Sungsoo Ahn, Le Song, and Jinwoo Shin. Roma: Robust model adaptation for offline model-based optimization. *Advances in Neural Information Processing Systems*, 34:4619–4631, 2021.

[120] Ye Yuan, Can Sam Chen, Zixuan Liu, Willie Neiswanger, and Xue Steve Liu. Importance-aware co-teaching for offline model-based optimization. *Advances in Neural Information Processing Systems*, 36, 2024.

[121] Houssam Zenati, Chuan Sheng Foo, Bruno Lecouat, Gaurav Manek, and Vijay Ramaseshan Chandrasekhar. Efficient gan-based anomaly detection. *Proceedings of the 20th IEEE International Conference on Data Mining (ICDM), 2018*, 2018.

[122] Bowen Zheng and Tianming Yang. Revisiting diffusion models: From generative pre-training to one-step generation. *arXiv preprint arXiv:2506.09376*, 2025.

[123] Mingyuan Zhou, Huangjie Zheng, Yi Gu, Zhendong Wang, and Hai Huang. Adversarial score identity distillation: Rapidly surpassing the teacher in one step. *arXiv preprint arXiv:2410.14919*, 2024.

[124] Mingyuan Zhou, Huangjie Zheng, Zhendong Wang, Mingzhang Yin, and Hai Huang. Score identity distillation: Exponentially fast distillation of pretrained diffusion models for one-step generation. In *Forty-first International Conference on Machine Learning*, 2024.

Chapter 3

Learning Energy-Based Prior Model with Diffusion-Amortized MCMC

3.1 Introduction

This chapter addresses the sampling bottleneck in learning latent-space energy-based models. In contrast to the density-estimation perspective of Chapter 2, the focus here is the interaction between approximate maximum-likelihood learning and the Markov chains used to estimate its gradients. Diffusion-amortized MCMC (DAMC) compresses repeated short-run transitions into a learned sampler, providing an alternative to directly executing a prohibitively long chain at each training iteration.

Generative modeling of data distributions has achieved impressive progress with the fast development of deep generative models in recent years [1–9]. It provides a powerful framework that allows successful applications in synthesizing data of different modalities [10–15], extracting semantically meaningful data representation [16–18] as well as other important domains of unsupervised or semi-supervised learning [19–21]. A fundamental and powerful branch of generative modeling is the DLVM. Typically, DLVM assumes that the observation (*e.g.*, a piece of text or images) is generated by its corresponding low-dimensional latent

variables via a top-down generator network [1–3]. The latent variables are often assumed to follow a non-informative prior distribution, such as a uniform or isotropic Gaussian distribution. While one can directly learn a deep top-down generator network to faithfully map the non-informative prior distribution to the data distribution, learning an informative prior model in the latent space could further improve the expressive power of the DLVM with significantly less parameters [22]. In this chapter, we specifically consider learning an EBM in the latent space as an informative prior for the model.

Learning energy-based prior can be challenging, as it typically requires computationally expensive MCMC sampling to estimate learning gradients. The difficulty of MCMC-based sampling is non-negligible when the target distribution is highly multi-modal or high-dimensional. In these situations, MCMC sampling can take a long time to converge and perform poorly on traversing modes with limited iterations [23]. Consequently, training models with samples from non-convergent short-run MCMC [24], which is a common choice for learning latent space EBMs [22], often results in malformed energy landscapes [15, 24, 25] and biased estimation of the model parameter. One possible solution is to follow the variational learning scheme [1], which however requires non-trivial extra efforts on model design to deal with problems like posterior collapse [26–28] and limited expressivity induced by model assumptions [1, 29, 30].

To remedy this sampling issue and further unleash the expressive power of the prior model, we propose to shift attention to an economical compromise between unrealistically expensive long-run MCMC and biased short-run MCMC: *we consider learning an amortized approximation to the potentially long-run MCMC for learning energy-based priors*. Specifically, inspired by the connection between MCMC sampling and denoising diffusion process [7, 8, 31], in this chapter we propose a diffusion-based amortization method suitable for long-run MCMC sampling in learning latent space EBMs. The learning algorithm derived from it breaks the long-run chain into consecutive affordable short-run segments that can be iteratively distilled by a diffusion-based sampler. The core idea is simple and can be summarized by a one-liner (Fig. 3.1). We provide theoretical and empirical evidence that the resulting sampler

approximates the long-run chain (see the proof-of-concept toy examples in Section 3.12.1), and brings significant performance improvement for learning latent space EBMs on several tasks. We believe that this proposal is a notable attempt to address the learning issues of energy-based priors and is new to the best of our knowledge. We kindly refer to Section 3.5 for a comprehensive discussion of the related work.

Contributions i) We propose a diffusion-based amortization method for MCMC sampling and develop a novel learning algorithm for the latent space EBM. ii) We analyze KL contraction under an idealized stationary Markov kernel and exact amortization, while using experiments to assess the practical finite-step sampler. iii) Our experiments demonstrate empirically that the proposed method brings higher sampling quality, a better-learned model and stronger performance on several image modeling benchmark datasets.

3.2 Background

3.2.1 Energy-Based Prior Model

We assume that for the observed sample $\boldsymbol{x} \in \mathbb{R}^D$, there exists $\boldsymbol{z} \in \mathbb{R}^d$ as its unobserved latent variable vector. The complete-data distribution is

$$p_{\boldsymbol{\theta}}(\boldsymbol{z}, \boldsymbol{x}) := p_{\boldsymbol{\alpha}}(\boldsymbol{z}) p_{\beta}(\boldsymbol{x}|\boldsymbol{z}),\ p_{\boldsymbol{\alpha}}(\boldsymbol{z}) := \frac{1}{Z_{\boldsymbol{\alpha}}} \exp\left(f_{\boldsymbol{\alpha}}(\boldsymbol{z})\right) p_0(\boldsymbol{z}), \tag{3.1}$$

where $p_{\boldsymbol{\alpha}}(\boldsymbol{z})$ is the prior model with parameters $\boldsymbol{\alpha}$, $p_{\beta}(\boldsymbol{x}|\boldsymbol{z})$ is the top-down generation model with parameters $\boldsymbol{\beta}$, and $\boldsymbol{\theta} = (\boldsymbol{\alpha}, \boldsymbol{\beta})$. The prior model $p_{\boldsymbol{\alpha}}(\boldsymbol{z})$ can be formulated as an energy-based model, which we refer to as the LEBM [22] throughout this chapter. In this formulation, $f_{\boldsymbol{\alpha}}(\boldsymbol{z})$ is parameterized by a neural network with scalar output, $Z_{\boldsymbol{\alpha}}$ is the partition function, and $p_0(\boldsymbol{z})$ is standard normal as a reference distribution. The prior model in Eq. (3.1) can be interpreted as an energy-based correction or exponential tilting of the original prior distribution p_0. The generation model follows $p_{\beta}(\boldsymbol{x}|\boldsymbol{z}) = \mathcal{N}(g_{\beta}(\boldsymbol{z}), \sigma^2 \mathbf{I}_D)$, where

g_β is the generator network and σ^2 takes a pre-specified value as in VAE [1]. This is equivalent to using l_2 error for reconstruction.

The parameters of LEBM and the generation model can be learned by MLE [22]. To be specific, given the training data $\boldsymbol{x}$, the gradients for updating $\boldsymbol{\alpha}, \boldsymbol{\beta}$ are,

$$\delta_{\boldsymbol{\alpha}}(\boldsymbol{x}) := \mathbb{E}_{p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})}\left[\nabla_{\boldsymbol{\alpha}} f_{\boldsymbol{\alpha}}(\boldsymbol{z})\right] - \mathbb{E}_{p_{\boldsymbol{\alpha}}(\boldsymbol{z})}\left[\nabla_{\boldsymbol{\alpha}} f_{\boldsymbol{\alpha}}(\boldsymbol{z})\right], \ \delta_{\beta}(\boldsymbol{x}) := \mathbb{E}_{p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})}\left[\nabla_{\beta} \log p_{\beta}(\boldsymbol{x}|\boldsymbol{z})\right]. \quad (3.2)$$

In practice, one may use the Monte-Carlo average to estimate the expectations in Eq. (3.2). This involves sampling from the prior $p_{\boldsymbol{\alpha}}(\boldsymbol{z})$ and the posterior $p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})$ distribution using MCMC, specifically LD [32], to estimate the expectations and hence the gradient. For a target distribution $\pi(\boldsymbol{z})$, the dynamics iterates

$$\boldsymbol{z}_{t+1} = \boldsymbol{z}_t + \frac{s^2}{2}\nabla_{z_t} \log \pi(\boldsymbol{z}_t) + s\boldsymbol{w}_t, \ t = 0, 1, ..., T-1, \ \boldsymbol{w}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_d), \quad (3.3)$$

where s is a small step size. One can draw $\boldsymbol{z}_0 \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_d)$ to initialize the chain. In the continuous-time Langevin limit, the target density π is stationary under standard regularity conditions. With a fixed nonzero step size s, however, the unadjusted discretization generally has a biased invariant distribution even when the chain is run for many steps [32]. It is prohibitively expensive to run LD until convergence in most cases, for which we may resort to limited iterations of LD for sampling in practice. Such non-convergent short chains generally produce a biased approximation to $\pi(\boldsymbol{z})$, which was dubbed as short-run LD [15, 23–25].

3.2.2 Denoising Diffusion Probabilistic Model

Closely related to EBMs are the DDPMs [5, 7, 8]. As pointed out in [5, 8], the sampling procedure of DDPM with $\boldsymbol{\epsilon}$-prediction parametrization resembles LD; $\boldsymbol{\epsilon}$ (predicted noise) plays a similar role to the gradient of the log density [8].

In the formulation proposed by Kingma et al. [33], the DDPM parameterized by $\boldsymbol{\phi}$ is specified by a noise schedule built upon $\lambda_s = \log[\beta_s^2/\sigma_s^2]$, *i.e.*, the log signal-to-noise-ratio,

that decreases monotonically with s. β_s and σ_s^2 are strictly positive scalar-valued functions of s. We use $\boldsymbol{z}_0$ to denote training data in $\mathbb{R}^d$. The forward-time diffusion process $q(\boldsymbol{z}|\boldsymbol{z}_0)$ is defined as:

$$q(\boldsymbol{z}_s|\boldsymbol{z}_0) = \mathcal{N}(\boldsymbol{z}_s; \beta_s \boldsymbol{z}_0, \sigma_s^2 \mathbf{I}_d), \quad q(\boldsymbol{z}'_s|\boldsymbol{z}_s) = \mathcal{N}(\boldsymbol{z}'_s; (\beta_{s'}/\beta_s)\boldsymbol{z}_s, \sigma_{s'|s}^2 \mathbf{I}_d), \tag{3.4}$$

where $0 \leqslant s < s' \leqslant S$ and $\sigma_{s'|s}^2 = (1 - e^{\lambda_{s'} - \lambda_s})\sigma_{s'}^2$. Noticing that the forward process can be reverted as $q(\boldsymbol{z}_s|\boldsymbol{z}_{s'}, \boldsymbol{z}_0) = \mathcal{N}(\boldsymbol{z}_s; \tilde{\boldsymbol{\mu}}_{s|s'}(\boldsymbol{z}_{s'}, \boldsymbol{z}_0), \tilde{\sigma}_{s|s'}^2 \mathbf{I}_d)$, an ancestral sampler $q_{\boldsymbol{\phi}}(\boldsymbol{z}_s|\boldsymbol{z}_{s'})$ [8] that starts at $\boldsymbol{z}_S \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_d)$ can be derived accordingly [33]:

$$\begin{aligned} \tilde{\boldsymbol{\mu}}_{s|s'}(\boldsymbol{z}_{s'}, \boldsymbol{z}_0) &= e^{\lambda_{s'} - \lambda_s}(\beta_s/\beta_{s'})\boldsymbol{z}_{s'} + (1 - e^{\lambda_{s'} - \lambda_s})\beta_s \boldsymbol{z}_0, \quad \tilde{\sigma}_{s|s'}^2 = (1 - e^{\lambda_{s'} - \lambda_s})\sigma_s^2, \\ \boldsymbol{z}_s &= \tilde{\boldsymbol{\mu}}_{s|s'}(\boldsymbol{z}_{s'}, \hat{\boldsymbol{z}}_0) + \sqrt{(\tilde{\sigma}_{s|s'}^2)^{1-\gamma}(\sigma_{s'|s}^2)^{\gamma}}\boldsymbol{\epsilon}, \end{aligned} \tag{3.5}$$

where $\boldsymbol{\epsilon}$ is standard Gaussian noise, $\hat{\boldsymbol{z}}_0$ is the prediction of $\boldsymbol{z}_0$ by the DDPM $\boldsymbol{\phi}$, and γ is a hyperparameter that controls the noise magnitude, following [34]. The goal of DDPM is to recover the distribution of $\boldsymbol{z}_0$ from the given Gaussian noise distribution. It can be trained by optimizing $\mathbb{E}_{\boldsymbol{\epsilon},\lambda}\left[\|\boldsymbol{\epsilon}(\boldsymbol{z}_\lambda) - \boldsymbol{\epsilon}\|_2^2\right]$, where $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_d)$ and λ is drawn from a distribution of log noise-to-signal ratio $p(\lambda)$ over uniformly sampled times $s \in [0, S]$. This denoising loss is related to variational lower-bound objectives for the data log-likelihood [8, 33] or as a variant of denoising score matching [7, 35]. We will exploit in this chapter the connection between DDPMs and LD sampling of EBMs, based upon which we achieve better sampling performance for LEBM compared with short-run LD.

3.3 Method

In this section, we introduce the diffusion-based amortization method for long-run MCMC sampling in learning LEBM in Section 3.3.1. The learning algorithm of LEBM based on the proposed method and details about implementation are then presented in Section 3.3.2 and

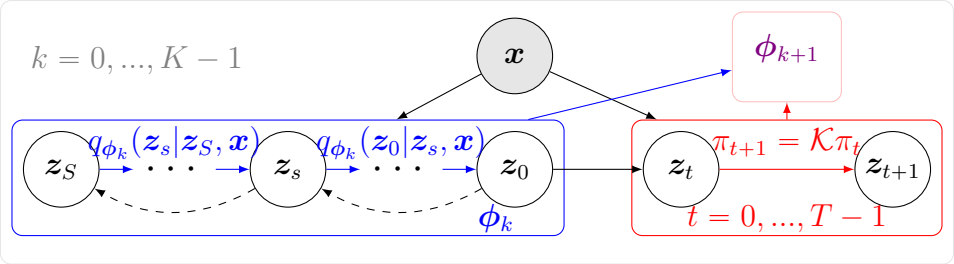


Figure 3.1: **Learning the DAMC sampler.** The training samples for updating the sampler to $\boldsymbol{\phi}_{k+1}$ is obtained by T-step short-run LD, initialized with the samples from the current learned sampler $\boldsymbol{\phi}_k$. The left block shows the diffusion-based sampler, the middle block shows Langevin transitions, and the upper block shows the updated sampler.

Section 3.3.3, respectively.

3.3.1 Amortizing MCMC with DDPM

Amortized MCMC Using the same notation as in Section 3.2, we denote the starting distribution of LD as $\pi_0(\boldsymbol{z})$, the distribution after t-th iteration as $\pi_t(\boldsymbol{z})$ and the target distribution as $\pi(\boldsymbol{z})$. The trajectory of LD in Eq. (3.3) is typically specified by its transition kernel $\mathcal{K}(\boldsymbol{z}|\boldsymbol{z}')$. The process starts with drawing $\boldsymbol{z}_0$ from $\pi_0(\boldsymbol{z})$ and iteratively sample $\boldsymbol{z}_t$ at the t-th iteration from the transition kernel conditioned on $\boldsymbol{z}_{t-1}$, *i.e.*, $\pi_t(\boldsymbol{z}) = \mathcal{K}\pi_{t-1}(\boldsymbol{z})$, where $\mathcal{K}\pi_{t-1}(\boldsymbol{z}) := \int \mathcal{K}(\boldsymbol{z}|\boldsymbol{z}')\pi_{t-1}(\boldsymbol{z}')d\boldsymbol{z}'$. Recursively, $\pi_t = \mathcal{K}_t\pi_0$, where $\mathcal{K}_t$ denotes the t-step transition kernel. LD can therefore be viewed as approximating a fixed point update in a non-parametric fashion since the target distribution π is a stationary distribution $\pi(\boldsymbol{z}) := \int \mathcal{K}(\boldsymbol{z}|\boldsymbol{z}')\pi(\boldsymbol{z}')d\boldsymbol{z}', \forall \boldsymbol{z}$. This motivates several works for more general approximations of this update [36–38] with the help of neural samplers.

Inspired by [36], we propose to use the following framework for amortizing the LD in learning the LEBM. Formally, let $\mathcal{Q} = \{q_{\boldsymbol{\phi}}\}$ be the set of amortized samplers parameterized by $\boldsymbol{\phi}$. Given the transition kernel $\mathcal{K}$, the goal is to find a sampler $q_{\boldsymbol{\phi}*}$ to closely approximate the target distribution π. This can be achieved by iteratively distill T-step transitions of LD into $q_{\boldsymbol{\phi}}$:

$$q_{\boldsymbol{\phi}_k} \leftarrow \arg\min_{q_{\boldsymbol{\phi}} \in \mathcal{Q}} \mathcal{D}[q_{\boldsymbol{\phi}_{k-1},T}||q_{\boldsymbol{\phi}}], \; q_{\boldsymbol{\phi}_{k-1},T} := \mathcal{K}_T q_{\boldsymbol{\phi}_{k-1}}, \; q_{\boldsymbol{\phi}_0} \approx \pi_0, \; k = 0, ..., K-1. \tag{3.6}$$

where $\mathcal{D}[\cdot||\cdot]$ is the KLD measure between distributions. Concretely, Eq. (3.6) means that to recover the target distribution π, instead of using long-run LD, we can repeat the following steps: i) employ a T-step short-run LD initialized with the current sampler $q_{\phi_{k-1}}$ to approximate $\mathcal{K}_T q_{\phi_{k-1}}$ as the target distribution of the current sampler, and ii) update the current sampler $q_{\phi_{k-1}}$ to q_{ϕ_k}. For a fixed Markov kernel that leaves π invariant, KL contraction holds under an idealized exact update $q_{\phi_k} = \mathcal{K}_T q_{\phi_{k-1}}$ [39], namely $\mathcal{D}[q_{\phi_k}||\pi] \leqslant \mathcal{D}[q_{\phi_{k-1}}||\pi]$. We refer to Section 3.8.1 for the assumptions and proof. This is not a monotonicity guarantee for finite-step unadjusted Langevin dynamics or inexact diffusion distillation. In practice, one can apply gradient-based methods to minimize $\mathcal{D}[q_{\phi_{k-1},T}||q_\phi]$ and approximate Eq. (3.6) for the update from $q_{\phi_{k-1}}$ to q_{ϕ_k}. The above formulation provides a generic and flexible framework for amortizing the potentially long MCMC.

Diffusion-based amortization To avoid clutter, we simply write $q_{\phi_{k-1},T}$ as q_T. We can see that

$$\arg\min_{q_\phi} \mathcal{D}[q_T||q_\phi] = \arg\min_{q_\phi} -\mathcal{H}(q_T) + \mathcal{H}(q_T, q_\phi) = \arg\min_{q_\phi} -\mathbb{E}_{q_T}\left[\log q_\phi\right], \tag{3.7}$$

where $\mathcal{H}$ represents the entropy of distributions. The selection of the sampler q_ϕ is a matter of design. According to Eq. (3.7), we may expect the following properties from q_ϕ: i) having analytically tractable expression of the exact value or lower bound of log-likelihood, ii) easy to draw samples from and iii) capable of close approximation to the given distribution $\{q_T\}$. In practice, iii) is important for the convergence of Eq. (3.6). If q_ϕ is far away from q_T in each iteration, then non-increasing KLD property $\mathcal{D}[q_{\phi_k}||\pi] \leqslant \mathcal{D}[q_{\phi_{k-1}}||\pi]$ might not hold, and the resulting amortized sampler would not converge to the true target distribution $\pi(\boldsymbol{z})$.

For the choice of q_ϕ, let us consider distilling the gradient field of q_T in each iteration, so that the resulting sampler is close to the q_T distribution. This naturally points to the DDPMs [8]. To be specific, learning a DDPM with $\boldsymbol{\epsilon}$-prediction parameterization is equivalent to fitting the finite-time marginal of a sampling chain resembling annealed Langevin dynamics

[7, 8, 31]. Moreover, it also fulfills i) and ii) of the desired properties mentioned above. We can plug in the objective of DDPM (Section 3.2.2), which is a lower bound of $\log q_{\boldsymbol{\phi}}$, to obtain the gradient-based update rule for $q_{\boldsymbol{\phi}}$:

$$\boldsymbol{\phi}_k^{(i+1)} \leftarrow \boldsymbol{\phi}_k^{(i)} - \eta \nabla_{\boldsymbol{\phi}} \mathbb{E}_{\boldsymbol{z}_0 \sim q_T, \boldsymbol{\epsilon}, \lambda} \left[\|\boldsymbol{\epsilon}_{\boldsymbol{\phi}}(\boldsymbol{z}_\lambda, \lambda) - \boldsymbol{\epsilon}\|_2^2 \right], \ \boldsymbol{\phi}_k^{(0)} \leftarrow \boldsymbol{\phi}_{k-1}, \ i = 0, 1, ..., M-1 \tag{3.8}$$

where $\boldsymbol{z}_\lambda = \beta_\lambda \boldsymbol{z}_0 + \sigma_\lambda \boldsymbol{\epsilon}$, $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_d)$, and $\boldsymbol{z}_0 \sim q_T$. λ is drawn from a distribution of log noise-to-signal ratio $p(\lambda)$. η is the step size for the update, and M is the number of iterations needed in Eq. (3.8). In practice, we find that when amortizing the LD sampling chain, a light-weight DDPM $q_{\boldsymbol{\phi}}$ updated with very small M, *i.e.*, few steps of Eq. (3.8) iteration, approximates Eq. (3.6) well. We provide a possible explanation using the Fisher information by scoping the asymptotic behavior of this update rule in the Section 3.8.2. We term the resulting sampler as DAMC.

3.3.2 Approximate MLE with DAMC

In this section, we show how to integrate the DAMC sampler into the learning framework of LEBM and form a symbiosis between these models. Given a set of N training samples $\{\boldsymbol{x}_i\}_{i=1}^N$ independently drawn from the unknown data distribution $p_{\text{data}}(\boldsymbol{x})$, the model $p_{\boldsymbol{\theta}}$ (Section 3.2.1) can be trained by maximizing the log-likelihood over training samples $\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{N}\sum_{i=1}^N \log p_{\boldsymbol{\theta}}(\boldsymbol{x}_i)$. Doing so typically requires computing the gradients of $\mathcal{L}(\boldsymbol{\theta})$, where for each $\boldsymbol{x}_i$ the learning gradient satisfies:

$$\begin{aligned}
\nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\boldsymbol{x}_i) &= \mathbb{E}_{p_{\boldsymbol{\theta}}(\boldsymbol{z}_i|\boldsymbol{x}_i)} \left[\nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\boldsymbol{z}_i, \boldsymbol{x}_i) \right] \\
&= (\underbrace{\mathbb{E}_{p_{\boldsymbol{\theta}}(\boldsymbol{z}_i|\boldsymbol{x}_i)} \left[\nabla_{\boldsymbol{\alpha}} f_{\boldsymbol{\alpha}}(\boldsymbol{z}_i) \right] - \mathbb{E}_{p_{\boldsymbol{\alpha}}(\boldsymbol{z}_i)} \left[\nabla_{\boldsymbol{\alpha}} f_{\boldsymbol{\alpha}}(\boldsymbol{z}_i) \right]}_{\delta_{\boldsymbol{\alpha}}(\boldsymbol{x}_i)}, \underbrace{\mathbb{E}_{p_{\boldsymbol{\theta}}(\boldsymbol{z}_i|\boldsymbol{x}_i)} \left[\nabla_{\boldsymbol{\beta}} \log p_{\boldsymbol{\beta}}(\boldsymbol{x}_i|\boldsymbol{z}_i) \right]}_{\delta_{\boldsymbol{\beta}}(\boldsymbol{x}_i)}).
\end{aligned} \tag{3.9}$$

Intuitively, based on the discussion in Section 3.2.1 and Section 3.3.1, we can approximate the distributions in Eq. (3.9) by drawing samples from $[\boldsymbol{z}_i|\boldsymbol{x}_i] \sim \mathcal{K}_{T,\boldsymbol{z}_i|\boldsymbol{x}_i} q_{\boldsymbol{\phi}_k}(\boldsymbol{z}_i|\boldsymbol{x}_i)$, $\boldsymbol{z}_i \sim$

$\mathcal{K}_{T,\boldsymbol{z}_i} q_{\phi_k}(\boldsymbol{z}_i)$, to estimate the expectations and hence the learning gradient. Here we learn the DAMC samplers $q_{\phi_k}(\boldsymbol{z}_i|\boldsymbol{x}_i)$ and $q_{\phi_k}(\boldsymbol{z}_i)$ for the posterior and prior sampling chain, respectively. $\boldsymbol{\phi}_k$ represents the current sampler as in Section 3.3.1; $\mathcal{K}_{T,\boldsymbol{z}_i|\boldsymbol{x}_i}$ and $\mathcal{K}_{T,\boldsymbol{z}_i}$ are the transition kernels for posterior and prior sampling chain. Equivalently, this means to better estimate the learning gradients we can i) first draw approximate posterior and prior MCMC samples from the current q_{ϕ_k} model, and ii) update the approximation of the prior $p_{\boldsymbol{\alpha}}(\boldsymbol{z})$ and posterior $p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})$ distributions with additional T-step LD initialized with q_{ϕ_k} samples. The additional transitions are intended to refine the sampler toward $p_{\boldsymbol{\theta}}(\boldsymbol{z}_i|\boldsymbol{x}_i)$ and $p_{\boldsymbol{\alpha}}(\boldsymbol{z}_i)$; the improvement is empirical rather than guaranteed for a finite unadjusted Langevin chain. We evaluate the resulting approximate learning gradients and model performance in Section 3.4. After updating $\boldsymbol{\theta} = (\boldsymbol{\alpha}, \boldsymbol{\beta})$ based on these approximate samples with Eq. (3.9), we can update q_{ϕ_k} with Eq. (3.8) to distill the sampling chain into $q_{\phi_{k+1}}$. As shown in Fig. 3.1, we can see that the whole learning procedure iterates between the approximate MLE of $p_{\boldsymbol{\theta}}$ and the amortization of MCMC with q_{ϕ}. We refer to Section 3.8.3 for an extended discussion of this procedure.

After learning the models, we can use either DAMC or LEBM for prior sampling. For DAMC, we may draw samples from $q_{\phi}(\boldsymbol{z}_i)$ with Eq. (3.5). Prior sampling with LEBM still requires short-run LD initialized from $\mathcal{N}(0, \mathbf{I}_d)$. For posterior sampling, we may sample from $\mathcal{K}_{T,\boldsymbol{z}_i|\boldsymbol{x}_i} q_{\phi_k}(\boldsymbol{z}_i|\boldsymbol{x}_i)$, *i.e.*, first draw samples from DAMC and then run few steps of LD to obtain posterior samples.

3.3.3 Implementation

In order to efficiently model both $q_{\phi_k}(\boldsymbol{z}_i|\boldsymbol{x}_i)$ and $q_{\phi_k}(\boldsymbol{z}_i)$, we follow the method of [40] to train a single neural network to parameterize both models, where $q_{\phi_k}(\boldsymbol{z}_i|\boldsymbol{x}_i)$ can be viewed as a conditional DDPM with the embedding of $\boldsymbol{x}_i$ produced by an encoder network as its condition, and $q_{\phi_k}(\boldsymbol{z}_i)$ an unconditional one. For $q_{\phi_k}(\boldsymbol{z}_i)$, we can simply input a null token $\varnothing$ as its condition when predicting the noise $\boldsymbol{\epsilon}$. We jointly train both models by randomly

Algorithm 1: Learning algorithm of DAMC.

Input: initial parameters $(\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\phi})$; learning rate $\eta = (\eta_{\boldsymbol{\alpha}}, \eta_{\boldsymbol{\beta}}, \eta_{\boldsymbol{\phi}})$; observed examples $\{\boldsymbol{x}^{(i)}\}_{i=1}^{N}$; prob. of uncond. training p_{uncond} for the DAMC sampler.

Output: $\big(\boldsymbol{\theta}^{(K)} = \{\boldsymbol{\alpha}^{(K)}, \boldsymbol{\beta}^{(K)}\}, \boldsymbol{\phi}^{(K)}\big)$.

for $k = 0 : K-1$ **do**

- Sample a minibatch of data $\{\boldsymbol{x}^{(i)}\}_{i=1}^{B}$;
- **Draw DAMC samples:** For each i, independently initialize $\boldsymbol{z}_{+}^{(i)}$ and $\boldsymbol{z}_{-}^{(i)}$ using the conditional sampler $q_{\boldsymbol{\phi}_k}(\boldsymbol{z} \mid \boldsymbol{x}^{(i)})$. The two chains are subsequently refined using different targets.
- **Prior LD update:** For each i, update $\boldsymbol{z}_{-}^{(i)}$ using Eq. (3.3), targeting $\pi(\boldsymbol{z}) = p_{\boldsymbol{\alpha}^{(k)}}(\boldsymbol{z})$;
- **Posterior LD update:** For each i, update $\boldsymbol{z}_{+}^{(i)}$ using Eq. (3.3), targeting $\pi(\boldsymbol{z}) = p_{\boldsymbol{\theta}^{(k)}}(\boldsymbol{z} \mid \boldsymbol{x}^{(i)}) \propto p_{\boldsymbol{\alpha}^{(k)}}(\boldsymbol{z})\, p_{\boldsymbol{\beta}^{(k)}}(\boldsymbol{x}^{(i)} \mid \boldsymbol{z})$;
- **Update $\boldsymbol{\theta}^{(k)}$:** Update $\boldsymbol{\alpha}^{(k)}$ and $\boldsymbol{\beta}^{(k)}$ using Monte-Carlo estimates (*i.e.*, Monte-Carlo average) of Eq. (3.9) with $\{\boldsymbol{z}_{+}^{(i)}\}_{i=1}^{B}$ and $\{\boldsymbol{z}_{-}^{(i)}\}_{i=1}^{B}$.
- **Update $\boldsymbol{\phi}^{(k)}$:** Update $\boldsymbol{\phi}^{(k)}$ using Eq. (3.8) with p_{uncond} and $\{\boldsymbol{z}_{+}^{(i)}\}_{i=1}^{B}$ as the target.

nullifying the inputs with the probability $p_{\text{uncond}} = 0.2$. During training, we use samples from $q_{\boldsymbol{\phi}_k}(\boldsymbol{z}_i|\boldsymbol{x}_i)$ to initialize both prior and posterior updates. For the posterior and prior DAMC samplers, we set the number of diffusion steps to 100. The number of iterations in Eq. (3.8) is set to $M = 6$ throughout the experiments. The LD runs $T = 30$ and $T = 60$ iterations for posterior and prior updates during training with a step size of $s = 0.1$. For test time sampling from $\mathcal{K}_{T,\boldsymbol{z}_i|\boldsymbol{x}_i} q_{\boldsymbol{\phi}_k}(\boldsymbol{z}_i|\boldsymbol{x}_i)$, $T = 10$ for the additional LD. For a fair comparison, we use the same LEBM and generator as in [22, 41] for all the experiments. We summarize the learning algorithm in Algorithm 1. Please see Sections 3.9 and 3.10 for network architecture and further training details, as well as the pytorch-style pseudocode of the algorithm.

3.4 Experiments

In this section, we are interested in the following questions: (i) How does the proposed method compare with its previous counterparts (*e.g.*, purely MCMC-based or variational methods)? (ii) How is the scalability of this method? (iii) How are the time and parameter efficiencies?

Table 3.1: **MSE(↓) and FID(↓) obtained from models trained on different datasets**. The FID scores are computed based on 50k generated images and training images for the first three datasets and 5k images for the CelebA-HQ dataset. The MSEs are computed based on unseen testing images. We highlight our model results in gray color. The best and second-best performances are marked in bold numbers and underlines, respectively; tables henceforth follow this format. *[41] uses a prior model with 4x parameters compared with [22] and ours.

Model	SVHN		CelebA		CIFAR-10		CelebA-HQ	
	MSE	FID	MSE	FID	MSE	FID	MSE	FID
VAE [1]	0.019	46.78	0.021	65.75	0.057	106.37	0.031	180.49
2s-VAE [48]	0.019	42.81	0.021	44.40	0.056	72.90	-	-
RAE [49]	0.014	40.02	0.018	40.95	0.027	74.16	-	-
NCP-VAE [50]	0.020	33.23	0.021	42.07	0.054	78.06	-	-
Adaptive CE* [41]	<u>0.004</u>	26.19	<u>0.009</u>	35.38	**0.008**	65.01	-	-
ABP [51]	-	49.71	-	51.50	0.018	90.30	0.025	160.21
SRI [24]	0.018	44.86	0.020	61.03	-	-	-	-
SRI (L=5) [24]	0.011	35.32	0.015	47.95	-	-	-	-
LEBM [22]	0.008	29.44	0.013	37.87	0.020	70.15	<u>0.025</u>	133.07
Ours-LEBM	**0.002**	<u>21.17</u>	**0.005**	<u>35.67</u>	<u>0.015</u>	<u>60.89</u>	**0.023**	<u>89.54</u>
Ours-DAMC		**18.76**		**30.83**		**57.52**		**85.88**

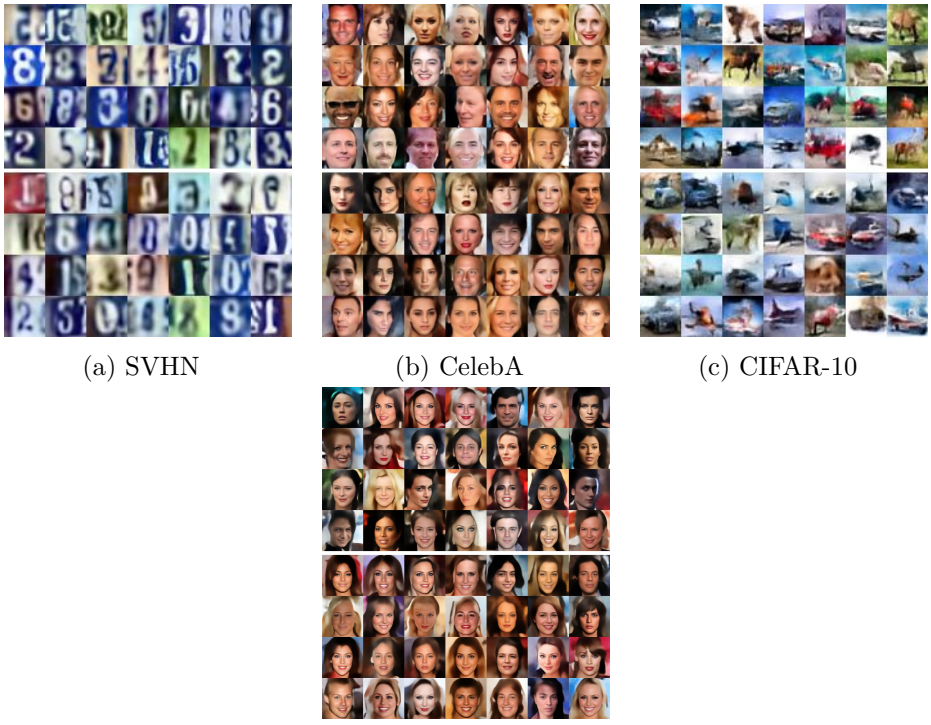
(iv) Does the proposed method provide a desirable latent space? To answer these questions, we present a series of experiments on benchmark datasets including MNIST [42], SVHN [43], CelebA64 [44], CIFAR-10 [45], CelebAMask-HQ [46], FFHQ [10] and LSUN-Tower [47]. As to be shown, the proposed method demonstrates consistently better performance in various experimental settings compared with previous methods. We refer to Section 3.11 for details about the experiments.

3.4.1 Generation and Inference: Prior and Posterior Sampling

Generation and reconstruction We evaluate the quality of the generated and reconstructed images to examine the sampling quality of DAMC. Specifically, we would like to check i) how well does DAMC fit the seen data, ii) does DAMC provide better MCMC samples for learning LEBM and iii) the generalizability of DAMC on unseen data. We check the goodness of fit of DAMC by evaluating the quality of the images generated with DAMC

prior samples. If DAMC does provide better MCMC samples for learning LEBM, we would expect better fitting of data and hence an improved generation quality of LEBM. We evaluate the performance of posterior sampling given unseen testing images by examining the reconstruction error on testing data. We benchmark our model against a variety of previous methods in two groups. The first group covers competing methods that adopt the variational learning scheme, including VAE [1], as well as recent two-stage methods such as 2-stage VAE [48], RAE [49] and NCP-VAE [50], whose prior distributions are learned with posterior samples in a second stage after the generator is trained. The second group includes methods that adopt MCMC-based sampling. It includes Alternating Back-Propogation (ABP) [51], Short-Run Inference (SRI) from [24] and the vanilla learning method of LEBM [22], which relies on short-run LD for both posterior and prior sampling. We also compare our method with the recently proposed Adaptive CE [41]. It learns a series of LEBMs adaptively during training, while these LEBMs are sequentially updated by density ratio estimation instead of MLE. To make fair comparisons, we follow the same evaluation protocol as in [22, 41].

For generation, we report the FID scores [52] in Table 3.1. We observe that i) the DAMC sampler, denoted as Ours-DAMC, provides superior generation performance compared to baseline models, and ii) the LEBM learned with samples from DAMC, denoted as Ours-LEBM, demonstrates significant performance improvement compared with the LEBM trained with short-run LD, denoted as LEBM. These results confirm that DAMC is a reliable sampler and indeed partly addresses the learning issue of LEBM caused by short-run LD. We would like to point out that the improvement is clearer on the CelebAMask-HQ dataset, where the input data is of higher dimension (256×256) and contains richer details compared with other datasets. This illustrates the superiority of DAMC sampler over short-run LD when the target distribution is potentially highly multi-modal. We show qualitative results of generated samples in Fig. 3.2, where we observe that our method can generate diverse, sharp and high-quality samples. For reconstruction, we compare our method with baseline methods in terms of MSE in Table 3.1. We observe that our method demonstrates competitive



(a) SVHN (b) CelebA (c) CIFAR-10

(d) CelebA-HQ

Figure 3.2: **Samples generated from the DAMC sampler and LEBM** trained on SVHN, CelebA, CIFAR-10 and CelebA-HQ datasets. In each sub-figure, the first four rows are generated by the DAMC sampler. The last four rows are generated by LEBM trained with the DAMC sampler.

reconstruction error, if not better, than competing methods do. Additional qualitative results of generation and reconstruction are presented in Sections 3.12.2 and 3.12.3.

GAN inversion We have examined the scalability of our method on the CelebAMask-HQ dataset. Next, we provide more results on high-dimensional and highly multi-modal data by performing GAN inversion [54] using the proposed method. Indeed, we may regard GAN inversion as an inference problem and a special case of posterior sampling. As a suitable testbed, the StyleGAN structure [10] is specifically considered as our generator in the experiments: [53] points out that to effectively infer the latent representation of a given image, the GAN inversion method needs to consider an extended latent space of StyleGAN, consisting of 14 different 512-dimensional latent vectors. We attempt to use the DAMC sampler for GAN inversion. We benchmark our method against i) learning an encoder that maps a given image to the latent space [16], which relates to the variational methods for posterior inference, ii) optimizing a random initial latent code by minimizing the reconstruction error and perceptual loss [53], which can be viewed as a variant of LD sampling, and iii) optimizing the latent code by minimizing both the objectives used in ii) and the energy score provided by LEBM. We use the pretrained weights provided by [18] for the experiments. Both the DAMC sampler and the encoder-based method are augmented with 100 post-processing optimization iterations. We refer to Section 3.11 for more experiment details. We test LEBM-based inversion with different optimization iterations. To be specific, 1x, 2x, and 4x represent 100, 200, and 400 iterations respectively. We can see in Table 3.2 that DAMC performs better than all the

Model	FFHQ		LSUN-T	
	MSE	FID	MSE	FID
Opt. [53]	0.055	149.39	0.080	240.11
Enc. [16]	<u>0.028</u>	<u>62.32</u>	0.079	132.41
[22] w/ 1x	0.054	149.21	0.072	239.51
[22] w/ 2x	0.039	101.59	0.066	163.20
[22] w/ 4x	0.032	84.64	<u>0.059</u>	<u>111.53</u>
Ours	**0.025**	**52.85**	**0.059**	**80.42**

Table 3.2: **MSE(↓) and FID(↓) for GAN inversion on different datasets**. Opt. and Enc. denotes the optimization-based and encoder-based methods.



(a) Observation



(b) Inversion

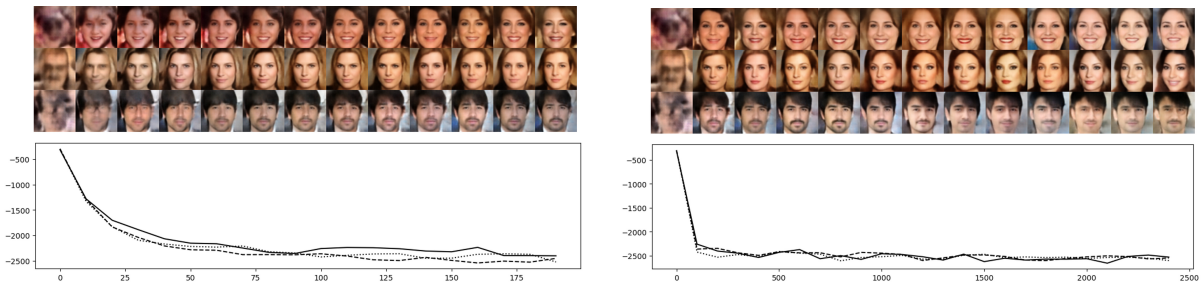
Figure 3.3: **Qualitative results of StyleGAN inversion using the DAMC sampler.** In each sub-figure, the left panel contain samples from the FFHQ dataset, and the right panel contains samples from the LSUN-T dataset.

baseline methods on the unseen testing data, which supports the efficacy of our method in high-dimensional settings. We provide qualitative results in Fig. 3.3.

Parameter efficiency and sampling time One potential disadvantage of our method is its parameter inefficiency for introducing an extra DDPM. Fortunately, our models are in the latent space so the network is lightweight. To be specific, on CIFAR-10 dataset the number of parameters in the DDPM is only around 10% (excluding the encoder) of those in the generator. The method has competitive time efficiency. With the batch size of 64, the DAMC prior sampling takes $0.3s$, while 100 steps of short-run LD with LEBM takes $0.2s$. The DAMC posterior sampling takes $1.0s$, while LEBM takes $8.0s$. Further discussions about the limitations can be found in the Section 3.14.1.

3.4.2 Analysis of Latent Space

Long-run langevin transition In this section, we examine the energy landscape induced by the learned LEBM. We expect that a well-trained $p_{\boldsymbol{\alpha}}(\boldsymbol{z})$ fueled by better prior and posterior samples from the DAMC sampler would lead to energy landscape with regular geometry. In Fig. 3.4, we visualize the transition of LD initialized from $\mathcal{N}(0, \mathbf{I}_d)$ towards $p_{\boldsymbol{\alpha}}(\boldsymbol{z})$ on the model trained on the CelebA dataset. Additional visualization of transitions on



(a) 200 steps (b) 2500 steps

Figure 3.4: **Transition of Markov chains initialized from $\mathcal{N}(0, \mathbf{I}_d)$ towards $p_{\boldsymbol{\alpha}}(\boldsymbol{z})$.** We present results by running LD for 200 and 2500 steps. In each sub-figure, the top panel displays the trajectory in the data space uniformly sampled along the chain. The bottom panel shows the energy score $f_{\boldsymbol{\alpha}}(\boldsymbol{z})$ over the iterations.

SVHN and CIFAR-10 datasets can be found in the Section 3.12.4. The LD iterates for 200 and 2500 steps, which is longer than the LD within each training iteration (60 steps). For the 200-step set-up, we can see that the generation quality quickly improves by exploring the local modes (demonstrating different facial features, *e.g.*, hairstyle, facial expression and lighting). For the 2500-step long-run set-up, we can see that the LD produces consistently valid results without the oversaturating issue of the long-run chain samples [23]. These observations provide empirical evidence that the LEBM is well-trained.

Anomaly detection We further evaluate how the LEBM learned by our method could benefit the anomaly detection task. With properly learned models, the posterior $p_{\boldsymbol{\theta},\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$ could form a discriminative latent space that has separated probability densities for in-distribution (normal) and out-of-distribution (anomalous) data. Given the testing sample $\boldsymbol{x}$, we use un-normalized log joint density $p_{\boldsymbol{\theta},\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) \propto p_{\boldsymbol{\theta},\boldsymbol{\phi}}(\boldsymbol{x},\boldsymbol{z}) \approx p_{\beta}(\boldsymbol{x}|\boldsymbol{z}) p_{\boldsymbol{\alpha}}(\boldsymbol{z})|_{\boldsymbol{z}\sim\mathcal{K}_{T,\boldsymbol{z}|\boldsymbol{x}} q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}$ as our decision function. This means that we draw samples from $\mathcal{K}_{T,\boldsymbol{z}|\boldsymbol{x}} q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$ and compare the corresponding reconstruction errors and energy scores. A higher value of log joint density indicates a higher probability of the test sample being a normal sample. To make fair comparisons, we follow the experimental settings in [22, 41, 55, 56] and train our model on MNIST with one class held out as an anomalous class. We consider the baseline models that employ MCMC-based or variational inferential mechanisms. Table 3.3 shows the results of AUPRC scores averaged over the last 10 trials. We observe significant improvements in our

Table 3.3: **AUPRC(↑) scores for unsupervised anomaly detection on MNIST**. Numbers are taken from [41]. Results of our model are averaged over the last 10 trials to account for variance.

Heldout Digit	1	4	5	7	9
VAE [1]	0.063	0.337	0.325	0.148	0.104
ABP [51]	0.095 ± 0.03	0.138 ± 0.04	0.147 ± 0.03	0.138 ± 0.02	0.102 ± 0.03
MEG [55]	0.281 ± 0.04	0.401 ± 0.06	0.402 ± 0.06	0.290 ± 0.04	0.342 ± 0.03
BiGAN-σ [56]	0.287 ± 0.02	0.443 ± 0.03	0.514 ± 0.03	0.347 ± 0.02	0.307 ± 0.03
LEBM [22]	0.336 ± 0.01	0.630 ± 0.02	0.619 ± 0.01	0.463 ± 0.01	0.413 ± 0.01
Adaptive CE [41]	$\underline{0.531 \pm 0.02}$	$\underline{0.729 \pm 0.02}$	$\underline{0.742 \pm 0.01}$	$\underline{0.620 \pm 0.02}$	$\underline{0.499 \pm 0.01}$
Ours	$\mathbf{0.684 \pm 0.02}$	$\mathbf{0.911 \pm 0.01}$	$\mathbf{0.939 \pm 0.02}$	$\mathbf{0.801 \pm 0.01}$	$\mathbf{0.705 \pm 0.01}$

method over the previous counterparts.

3.4.3 Ablation Study

In this section, we conduct ablation study on several variants of the proposed method. Specifically, we would like to know: i) what is the difference between the proposed method and directly training a DDPM in a fixed latent space? ii) What is the role of LEBM in this learning scheme? iii) Does DAMC effectively amortize the sampling chain? We use CIFAR-10 dataset for the ablative experiments to empirically answer these questions. More ablation studies can be found in Section 3.13.

Non-Amortized DDPM vs. DAMC We term directly training a DDPM in a fixed latent space as the non-amortized DDPM. To analyze the difference between non-amortized DDPM and DAMC, we first train a LEBM model with persistent long-run chain sampling [57] and use the trained model to obtain persistent samples for learning the non-amortized DDPM. In short, the non-amortized DDPM can be viewed as directly distilling the long-run MCMC sampling process, instead of progressively amortizing the chain. We present the FID and MSE of the non-amortized model (NALR) in Table 3.4. We observe that directly amortizing the long-run chain leads to degraded performance compared with the proposed method. The results are consistently worse for both posterior and prior sampling and the learned LEBMs, which verify the effectiveness of the proposed iterative learning scheme.

Table 3.4: **Ablation study on CIFAR-10 dataset**. VI denotes learning LEBM using variational methods. SR denotes learning LEBM with short-run LD. DAMC-G replaces the LEBM in DAMC-LEBM with a standard Gaussian distribution. NALR denotes the non-amortized DDPM setting. For each set-up, we provide results using the vanilla sampling method, denoted as V, and the ones using the DAMC sampler, denoted as D.

Model	VI-LEBM		SR-LEBM		DAMC-G		NALR-LEBM		DAMC-LEBM	
	V.	D.	V.	D.	V.	D.	V.	D.	V.	D.
MSE	0.054	-	0.020	-	0.018	0.015	0.028	0.016	0.021	**0.015**
FID	78.06	-	70.15	-	90.30	66.93	68.52	64.38	60.89	**57.52**

The contribution of LEBM One may argue that since we have the DAMC as a powerful sampler, it might not be necessary to jointly learn LEBM in the latent space. To demonstrate the necessity of this joint learning scheme, we train a variant of DAMC by replacing the LEBM with a Gaussian prior. The results are presented in Table 3.4. We observe that models trained with non-informative Gaussian prior obtain significantly worse generation results. It suggests that LEBM involved in the learning iteration provides positive feedback to the DAMC sampler. Therefore, we believe that it is crucial to jointly learn the DAMC and LEBM.

Vanilla sampling vs. DAMC We compare the vanilla sampling process of each model with DAMC. The vanilla sampling typically refers to short-run or long-run LD initialized with $\mathcal{N}(0, \mathbf{I}_d)$. We also provide results of learning LEBM using variational methods for comparison. We can see in Table 3.4 that sampling with DAMC shows significantly better scores of the listed models, compared with vanilla sampling. The result is even better than that of the persistent chain sampler (V. of NALR-LEBM). This indicates that DAMC effectively amortizes the sampling chain. Comparing DAMC sampler with the variational sampler also indicates that DAMC is different from general variational approximation: it benefits from its connection with LD and shows better expressive power.

3.5 Related Work

Energy-based prior model EBMs [23, 24, 58–60] play an important role in generative modeling. Pang et al. [22] propose to learn an EBM as a prior model in the latent space of DLVMs; it greatly improves the model expressivity over those with non-informative priors and brings strong performance on downstream tasks, *e.g.*, image segmentation, text modeling, molecule generation, and trajectory prediction [12, 15, 61, 62]. However, learning EBMs or latent space EBMs requires MCMC sampling to estimate the learning gradients, which needs numerous iterations to converge when the target distributions are high-dimensional or highly multi-modal. Typical choices of sampling with non-convergent short-run MCMC [24] in practice can lead to poor generation quality, malformed energy landscapes [15, 24, 25], biased estimation of the model parameter and instability in training [23, 25, 59, 60]. In this work, we consider learning valid amortization of the long-run MCMC for energy-based priors; the proposed model shows reliable sampling quality in practice.

Denoising diffusion probabilistic model DDPMs [7, 8, 31], originating from [5], learn the generative process by recovering the observed data from a sequence of noise-perturbed versions of the data. The learning objective can be viewed as a variant of the denoising score matching objective [35]. As pointed out in [5, 8], the sampling procedure of DDPM with $\boldsymbol{\epsilon}$-prediction parametrization resembles LD of an EBM; $\boldsymbol{\epsilon}$ (predicted noise) plays a similar role to the gradient of the log density [8]. To be specific, learning a DDPM with $\boldsymbol{\epsilon}$-prediction parameterization is equivalent to fitting the finite-time marginal of a sampling chain resembling annealed Langevin dynamics [7, 8, 31]. Inspired by this connection, we propose to amortize the long-run MCMC in learning energy-based prior by iteratively distilling the short-run chain segments with a DDPM-based sampler. We study its sampling behavior experimentally and give an idealized KL contraction argument for exact Markov updates and exact amortization.

Amortized MCMC The amortized MCMC technique is formally brought up by Li et al. [36], which incorporates feedback from MCMC back to the parameters of the amortizer

distribution q_ϕ. It is concurrently and independently proposed by Xie et al. [37] as the MCMC teaching framework. Methods under this umbrella term [36–38, 63–66] generally learns the amortizer by minimizing the divergence (typically the KLD) between the improved distribution and its initialization, i.e., $\mathcal{D}[\mathcal{K}_T q_{\phi_{k-1}} || q_\phi]$, where $\mathcal{K}_T$ represents T-step MCMC transition kernel and $q_{\phi_{k-1}}$ represents the current amortizer. The diffusion-based amortization proposed in this work can be viewed as an instantiation of this framework, while our focus is on learning the energy-based prior. Compared with previous methods, our method i) specifically exploits the connection between EBMs and DDPMs and is suitable for amortizing the prior and posterior sampling MCMC of energy-based prior, and ii) resides in the lower-dimensional latent space and enables faster sampling and better convergence.

More methods for learning EBM Several techniques other than short-run MCMC have been proposed to learn the EBM. In the seminal work, Hinton [67] proposes to initialize Markov chains using real data and run several steps of MCMC to obtain samples from the model distribution. Tieleman [57] proposes to start Markov chains from past samples in the previous sampling iteration, known as Persistent Contrastive Divergence (PCD) or persistent chain sampling, to mimic the long-run sampling chain. Nijkamp et al. [23] provide comprehensive discussions about tuning choices for LD such as the step size s and sampling steps T to obtain stable long-run samples for persistent training. [59, 60] employ a hybrid of persistent chain sampling and short-run sampling by maintaining a buffer of previous samples. The methods draw from the buffer or initialize the short-run chain with noise distribution with some pre-specified probability. Another branch of work, stemmed from [68], considers discriminative contrastive estimation to avoid MCMC sampling. Gao et al. [69] use a normalizing flow [4] as the base distribution for contrastive estimation. Aneja et al. [50] propose to estimate the energy-based prior model based on the prior of a pre-trained VAE [70] by noise contrastive estimation. More recently, Xiao and Han [41] learn a sequence of EBMs in the latent space with adaptive multi-stage NCE to further improve the expressive power of the model.

3.6 Conclusion

In this chapter, we presented the DAMC sampler and developed a learning algorithm for LEBM based on it. The idealized analysis and empirical results motivate amortizing Markov-chain computation with a diffusion sampler, although the finite-step implementation does not inherit an exact convergence guarantee. We notice that our method can be applied to amortizing MCMC sampling of unnormalized continuous densities in general. It can also be applied to sampling posterior distributions of continuous latent variables in general latent variable models. These extensions remain interesting directions for future work. Together with Chapter 2, this chapter illustrates complementary routes to learning energy-based generative distributions: improving density estimation objectives and improving the sampling used to estimate learning gradients.

3.7 Acknowledgments for This Research Chapter

This chapter is adapted from Peiyu Yu, Yaxuan Zhu, Sirui Xie, Xiaojian Ma, Ruiqi Gao, Song-Chun Zhu, and Ying Nian Wu, "Learning Energy-Based Prior Model with Diffusion-Amortized MCMC," *Advances in Neural Information Processing Systems* 36 (NeurIPS 2023), pp. 42717–42747, 2023, DOI: 10.52202/075280-1851.

Author contributions. Peiyu Yu conceived the method, designed and conducted the experiments, and wrote the original manuscript. Yaxuan Zhu, Sirui Xie, Xiaojian Ma, Ruiqi Gao, Song-Chun Zhu, and Ying Nian Wu contributed through research discussions and feedback. Peiyu Yu adapted the material for this dissertation.

Y. N. Wu was supported by NSF DMS-2015577. We thank the anonymous reviewers for their constructive comments.

3.8 Theoretical Discussion

The following analysis distinguishes the KL contraction of an invariant Markov kernel from the approximate distribution updates used in the implemented sampler. Additional derivations, algorithm details, and experiments are included in the remaining sections of this chapter.

3.8.1 KL Contraction under Idealized Assumptions

In the main text we use KL contraction only for a fixed transition kernel $\mathcal{K}$ with invariant density π and an idealized exact amortization step. The statement is conditional on $\mathcal{K}\pi = \pi$; finite-step unadjusted Langevin dynamics need not satisfy this condition for the intended continuous-time target. To show this, we first provide a proof of $\mathcal{D}[\pi_{t+T}||\pi] \leqslant \mathcal{D}[\pi_t||\pi]$, where π_t and π_{t+T} are the distributions of $\boldsymbol{z}$ at t-th and $(t+T)$-th iteration, respectively. This is a known result from [39], and we include it here for completeness.

$$\begin{aligned}
\mathcal{D}[\pi_t||\pi] &= \mathbb{E}_{\pi_t(\boldsymbol{z}_t)}\left[\log \frac{\pi_t(\boldsymbol{z}_t)}{\pi(\boldsymbol{z}_t)}\right] = \mathbb{E}_{\pi_t(\boldsymbol{z}_t),\mathcal{K}(\boldsymbol{z}_{t+1}|z_t)}\left[\log \frac{\pi_t(\boldsymbol{z}_t)\mathcal{K}(\boldsymbol{z}_{t+1}|z_t)}{\pi(\boldsymbol{z}_t)\mathcal{K}(\boldsymbol{z}_{t+1}|z_t)}\right] \\
&\overset{(i)}{=} \mathbb{E}_{\pi_{t+1}(\boldsymbol{z}_{t+1}),\mathcal{K'}_{\pi_{t+1}}(\boldsymbol{z}_t|z_{t+1})}\left[\log \frac{\mathcal{K'}_{\pi_{t+1}}(\boldsymbol{z}_t|z_{t+1})\pi_{t+1}(\boldsymbol{z}_{t+1})}{\mathcal{K'}_{\pi}(\boldsymbol{z}_t|z_{t+1})\pi(\boldsymbol{z}_{t+1})}\right] \\
&= \mathcal{D}[\pi_{t+1}||\pi] + \mathbb{E}_{\pi_{t+1}(\boldsymbol{z}_{t+1})}\mathcal{D}[\mathcal{K'}_{\pi_{t+1}}(\boldsymbol{z}_t|z_{t+1})||\mathcal{K'}_{\pi}(\boldsymbol{z}_t|z_{t+1})] \\
&\overset{(ii)}{\geqslant} \mathcal{D}[\pi_{t+1}||\pi],
\end{aligned} \tag{3.10}$$

where we denote the forward-time transition kernel as $\mathcal{K}$ and the reverse-time kernel as $\mathcal{K}'$. (i) holds because we are just re-factorizing the joint density of $[\boldsymbol{z}_t, \boldsymbol{z}_{t+1}]$: $\pi_t(\boldsymbol{z}_t)\mathcal{K}(\boldsymbol{z}_{t+1}|z_t) = \mathcal{K'}_{\pi_{t+1}}(\boldsymbol{z}_t|z_{t+1})\pi_{t+1}(\boldsymbol{z}_{t+1})$ and $\pi(\boldsymbol{z}_t)\mathcal{K}(\boldsymbol{z}_{t+1}|z_t) = \mathcal{K'}_{\pi}(\boldsymbol{z}_t|z_{t+1})\pi(\boldsymbol{z}_{t+1})$. (ii) holds because the KLD is non-negative. We can see that $\mathcal{D}[\pi_{t+T}||\pi] \leqslant \mathcal{D}[\pi_t||\pi]$ is a direct result from Eq. (3.10), and that $\pi_t \to \pi$ as $t \to \infty$ under proper conditions [32].

In the main text, we describe the update rule of the sampler $q_{\boldsymbol{\phi}}$ as follows:

$$q_{\boldsymbol{\phi}_k} \leftarrow \underset{q_{\boldsymbol{\phi}} \in \mathcal{Q}}{\arg\min}\, \mathcal{D}[q_{\boldsymbol{\phi}_{k-1},T}||q_{\boldsymbol{\phi}}],\ q_{\boldsymbol{\phi}_{k-1},T} := \mathcal{K}_T q_{\boldsymbol{\phi}_{k-1}},\ q_{\boldsymbol{\phi}_0} \approx \pi_0,\ k = 0, ..., K-1. \tag{3.11}$$

In the ideal case, we can assume that the objective in Eq. (3.6) is properly optimized and that $\{q_\phi\}$ is expressive enough to parameterize each $q_{\phi_{k-1},T}$. If the update is exact, $q_{\phi_k} = \mathcal{K}_T q_{\phi_{k-1}}$, then $\mathcal{D}[q_{\phi_k}||\pi] \leqslant \mathcal{D}[q_{\phi_{k-1}}||\pi]$ follows from Eq. (3.10). When the sampler only approximately fits $\mathcal{K}_T q_{\phi_{k-1}}$, this inequality does not follow without further control of approximation error. We will discuss in the following section the scenario where we apply gradient-based methods to minimize $\mathcal{D}[q_{\phi_{k-1},T}||q_\phi]$ and approximate Eq. (3.6) for the update from $q_{\phi_{k-1}}$ to q_{ϕ_k}.

3.8.2 Discussion about Diffusion-Based Amortization

We can see that the statement in Section 3.8.1 holds when q_{ϕ_k} is a close approximation of $q_T := \mathcal{K}_T q_{\phi_{k-1}}$. This motivates our choice of employing DDPM to amortize the LD transition, considering its capability of close approximation to the given distribution $\{q_T\}$. The optimization in Eq. (3.6) is equivalent to maximum-likelihood fitting to q_T, because q_T does not depend on the new sampler parameters:

$$\arg\min_{q_\phi} \mathcal{D}[q_T \| q_\phi] = \arg\min_{\phi} -\mathbb{E}_{\boldsymbol{z}_0 \sim q_T}[\log q_\phi(\boldsymbol{z}_0)]\,. \tag{3.12}$$

Rather than evaluating this likelihood directly, we train the diffusion sampler with the tractable denoising surrogate

$$\mathcal{L}_{\text{denoise}}(\boldsymbol{\phi}) = \mathbb{E}_{\boldsymbol{z}_0 \sim q_T,\, \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_d),\, \lambda}\left[\|\boldsymbol{\epsilon}_\phi(\boldsymbol{z}_\lambda, \lambda) - \boldsymbol{\epsilon}\|_2^2\right], \quad \boldsymbol{z}_\lambda = \beta_\lambda \boldsymbol{z}_0 + \sigma_\lambda \boldsymbol{\epsilon}. \tag{3.13}$$

This is the usual noise-prediction training criterion associated with diffusion variational objectives [8, 33]. The displayed surrogate and the exact KL objective should not be regarded as having identical minimizers without additional assumptions.

In practice, we use Monte-Carlo average to approximate the objective and employ a

gradient-based update rule for $q_{\boldsymbol{\phi}}$:

$$\boldsymbol{\phi}_k^{(i+1)} \leftarrow \boldsymbol{\phi}_k^{(i)} - \eta \nabla_{\boldsymbol{\phi}} \frac{1}{N} \sum_{j=1}^{N} \left[\| \boldsymbol{\epsilon}_{\boldsymbol{\phi}}(\boldsymbol{z}_{j,\lambda_j}, \lambda_j) - \boldsymbol{\epsilon}_j \|_2^2 \right], \ \boldsymbol{\phi}_k^{(0)} \leftarrow \boldsymbol{\phi}_{k-1}, \ i = 0, 1, ..., M-1. \quad (3.14)$$

The diffusion update can be interpreted as fitting the marginal distribution $q_T = \mathcal{K}_T q_{\boldsymbol{\phi}_{k-1}}$. To illustrate how statistical estimation accuracy might depend on transition complexity, consider separately an auxiliary parametric model $p_{\tilde{\boldsymbol{\phi}}}(\hat{\boldsymbol{z}} \mid \boldsymbol{z})$ for the Markov transition, fitted to pairs $(\boldsymbol{z}_j, \hat{\boldsymbol{z}}_j)$ with $\boldsymbol{z}_j \sim q_{\boldsymbol{\phi}_{k-1}}$ and $\hat{\boldsymbol{z}}_j \sim \mathcal{K}_T(\cdot \mid \boldsymbol{z}_j)$. The conditional maximum-likelihood criterion for this auxiliary model would be $N^{-1} \sum_{j=1}^{N} \log p_{\tilde{\boldsymbol{\phi}}}(\hat{\boldsymbol{z}}_j \mid \boldsymbol{z}_j)$. Importantly, the actual denoising objective Eq. (3.13) is not this conditional maximum-likelihood criterion.

Let $\tilde{\boldsymbol{\phi}}$ be the result of M-estimation and $\tilde{\boldsymbol{\phi}}^*$ be the true target parameter. Then based on the derivation in [25], asymptotically we have

$$\sqrt{N} \left(\tilde{\boldsymbol{\phi}} - \tilde{\boldsymbol{\phi}}^* \right) \rightarrow \mathcal{N} \left(0, \mathcal{I} \left(\tilde{\boldsymbol{\phi}}^* \right)^{-1} \right), \quad (3.15)$$

where $\mathcal{I} \left(\tilde{\boldsymbol{\phi}}^* \right) = \mathbb{E}_{\hat{\boldsymbol{z}}, \boldsymbol{z}} \left[-\nabla^2 \log p_{\tilde{\boldsymbol{\phi}}}(\hat{\boldsymbol{z}} | \boldsymbol{z}) \right]$ is the Fisher information matrix. This asymptotic statement holds only under the usual regularity, correct-specification, identifiability, and optimization assumptions for the auxiliary conditional likelihood. It does not directly establish the statistical properties of the diffusion denoising estimator or imply that a small number of optimizer steps suffices. The choice of the transition length T therefore remains a practical trade-off: larger transitions may be more informative but harder to amortize with a small number of gradient updates. This trade-off is an empirical design consideration, rather than a consequence of the Fisher-information limit above. We find that setting $T = 30$ and $M = 6$ works well in the experiments.

3.8.3 Further Discussion about the Learning Algorithm

For completeness, we first derive the learning gradients for updating $\boldsymbol{\theta}$.

$$\begin{aligned}
\nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\boldsymbol{x}) &= \frac{1}{p_{\boldsymbol{\theta}}(\boldsymbol{x})} \nabla_{\boldsymbol{\theta}} \int_z p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{z}) d\boldsymbol{z} = \int_z \frac{p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{z})}{p_{\boldsymbol{\theta}}(\boldsymbol{x})} \nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{z}) d\boldsymbol{z} \\
&= \mathbb{E}_{p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})} \left[\nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\boldsymbol{z}, \boldsymbol{x}) \right] \\
&= \left(\mathbb{E}_{p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})} \left[\nabla_{\boldsymbol{\alpha}} \log p_{\boldsymbol{\alpha}}(\boldsymbol{z}) \right], \mathbb{E}_{p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})} \left[\nabla_{\beta} \log p_{\beta}(\boldsymbol{x}|\boldsymbol{z}) \right] \right) \\
&= (\underbrace{\mathbb{E}_{p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})} \left[\nabla_{\boldsymbol{\alpha}} f_{\boldsymbol{\alpha}}(\boldsymbol{z}) \right] - \mathbb{E}_{p_{\boldsymbol{\alpha}}(\boldsymbol{z})} \left[\nabla_{\boldsymbol{\alpha}} f_{\boldsymbol{\alpha}}(\boldsymbol{z}) \right]}_{\delta_{\boldsymbol{\alpha}}(\boldsymbol{x})}, \underbrace{\mathbb{E}_{p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})} \left[\nabla_{\beta} \log p_{\beta}(\boldsymbol{x}|\boldsymbol{z}) \right]}_{\delta_{\beta}(\boldsymbol{x})}).
\end{aligned} \tag{3.16}$$

We can see that $\boldsymbol{\theta}$ is estimated in a MLE-by-EM style. The learning gradient is the same as that of directly maximizing the observed data likelihood, while we need to approximate the expectations in Eq. (3.16). Estimating the expectations is like the E-step, and update $\boldsymbol{\theta}$ with Eq. (3.16) is like the M-step in the EM algorithm. The proposed diffusion-based amortization brings better estimation of the expectations in the E-step, and incorporate the feedback from the M-step by running prior and posterior sampling LD as follows

$$\begin{aligned}
\boldsymbol{z}_{t+1} &= \boldsymbol{z}_t + \frac{s^2}{2} \nabla_{\boldsymbol{z}_t} \underbrace{\left(f_{\boldsymbol{\alpha}}(\boldsymbol{z}_t) - \frac{1}{2} \|\boldsymbol{z}_t\|_2^2 \right)}_{\log p_{\boldsymbol{\alpha}}(\boldsymbol{z}_t)} + s \boldsymbol{w}_t, \\
\boldsymbol{z}_{t+1} &= \boldsymbol{z}_t + \frac{s^2}{2} \nabla_{\boldsymbol{z}_t} \underbrace{\left(-\frac{\|\boldsymbol{x} - g_{\beta}(\boldsymbol{z}_t)\|_2^2}{2\sigma^2} + f_{\boldsymbol{\alpha}}(\boldsymbol{z}_t) - \frac{1}{2} \|\boldsymbol{z}_t\|_2^2 \right)}_{\log p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x}) = \log p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{z}) + C} + s \boldsymbol{w}_t,
\end{aligned} \tag{3.17}$$

to obtain training data. Here $t = 0, 1, ..., T$. $\boldsymbol{z}_0 \sim q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$ for posterior sampling and $\boldsymbol{z}_0 \sim q_{\boldsymbol{\phi}}(\boldsymbol{z})$ for prior sampling. $\boldsymbol{w}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_d)$. Note that we plug-in $p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{z})$ for the target distribution of posterior sampling LD. This is because given the observed data $\boldsymbol{x}$, by Bayes' rule we know that $p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x}) \propto p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{z}) = p_{\beta}(\boldsymbol{x}|\boldsymbol{z}) p_{\boldsymbol{\alpha}}(\boldsymbol{z})$. The whole learning iteration can be viewed as a variant of the variational EM algorithm [71].

3.9 Network Architecture and Training Details

Architecture The energy score network $f_{\boldsymbol{\alpha}}$ uses a simple fully-connected structure throughout the experiments. We describe the architecture in details in Table 3.5. The generator network has a simple deconvolution structure similar to DCGAN [72] shown in Table 3.6. The denoising diffusion model is implemented by a light-weight MLP-based U-Net [73] structure (Table 3.7). The encoder network to embed the observed images has a fully convolutional structure [74], as shown in Table 3.8.

Table 3.5: **Network structures of the energy score network.** LReLU denotes the Leaky ReLU activation function. The slope in Leaky ReLU is set to 0.2. For SVHN and CelebA datasets, we use `nz=100`. For the CIFAR-10 and CelebA-HQ datasets, we use `nz=128`. We use `nz=8` for anomaly detection on the MNIST dataset, and `nz=7168` for GAN inversion. We use `ndf=512` for GAN inversion and `ndf=200` for the rest experiments.

Layers	**Out Size**
Input: $\boldsymbol{z}$	nz $\in \{8, 100, 128, 7168\}$
Linear, LReLU	ndf $\in \{200, 512\}$
Linear, LReLU	ndf $\in \{200, 512\}$
Linear	1

Hyperparameters and training details As mentioned in the main text, for the posterior and prior DAMC samplers, we set the number of diffusion steps to 100. The number of iterations in Eq. (3.8) is set to $M = 6$ for the experiments. The LD runs $T = 30$ and $T = 60$ iterations for posterior and prior updates during training with a step size of $s = 0.1$. For test time sampling from $\mathcal{K}_{T,\boldsymbol{z}_i|\boldsymbol{x}_i} q_{\boldsymbol{\phi}_k}(\boldsymbol{z}_i|\boldsymbol{x}_i)$, we set $T = 10$ for the additional LD. For test time prior sampling of LEBM with LD, we follow [22, 41] and set $T = 100$. To further stabilize the training procedure, we i) perform gradient clipping by setting the maximal gradient norm as 100, ii) use a separate target diffusion network which is the EMA of the current diffusion network to initialize the prior and posterior updates and iii) add noise-initialized prior samples for the prior updates. These set-ups are identical across different datasets.

The parameters of all the networks are initialized with the default pytorch methods [77]. We use the Adam optimizer [78] with $\beta_1 = 0.5$ and $\beta_2 = 0.999$ to train the generator network and the energy score network. We use the AdamW optimizer [79] with $\beta_1 = 0.5$,

Table 3.6: **Network structures of the generator networks** used for the SVHN, CelebA, CIFAR-10, CelebA-HQ and MNIST (from top to bottom) datasets. For GAN inversion, we use the StyleGAN [10] structure as our generator network. ConvT(n) indicates a transposed convolutional operation with n output channels. We use `ngf=64` for the SVHN dataset and `ngf=128` for the rest. LReLU indicates the Leaky-ReLU activation function. The slope in Leaky ReLU is set to be 0.2.

Layers	**Out Size**	**Stride**
Input: $\boldsymbol{z}$	1x1x100	-
4x4 ConvT(ngf x 8), LReLU	4x4x(ngf x 8)	1
4x4 ConvT(ngf x 4), LReLU	8x8x(ngf x 4)	2
4x4 ConvT(ngf x 2), LReLU	16x16x(ngf x 2)	2
4x4 ConvT(3), Tanh	32x32x3	2

Layers	**Out Size**	**Stride**
Input: $\boldsymbol{z}$	1x1x100	-
4x4 ConvT(ngf x 8), LReLU	4x4x(ngf x 8)	1
4x4 ConvT(ngf x 4), LReLU	8x8x(ngf x 4)	2
4x4 ConvT(ngf x 2), LReLU	16x16x(ngf x 2)	2
4x4 ConvT(ngf x 1), LReLU	32x32x(ngf x 1)	2
4x4 ConvT(3), Tanh	64x64x3	2

Layers	**Out Size**	**Stride**
Input: $\boldsymbol{z}$	1x1x128	-
8x8 ConvT(ngf x 8), LReLU	8x8x(ngf x 8)	1
4x4 ConvT(ngf x 4), LReLU	16x16x(ngf x 4)	2
4x4 ConvT(ngf x 2), LReLU	32x32x(ngf x 2)	2
3x3 ConvT(3), Tanh	32x32x3	1

Layers	**Out Size**	**Stride**
Input: $\boldsymbol{z}$	1x1x128	-
4x4 ConvT(ngf x 16), LReLU	4x4x(ngf x 16)	1
4x4 ConvT(ngf x 8), LReLU	8x8x(ngf x 8)	2
4x4 ConvT(ngf x 4), LReLU	16x16x(ngf x 4)	2
4x4 ConvT(ngf x 4), LReLU	32x32x(ngf x 4)	2
4x4 ConvT(ngf x 2), LReLU	64x64x(ngf x 2)	2
4x4 ConvT(ngf x 1), LReLU	128x128x(ngfx1)	2
4x4 ConvT(3), Tanh	256x256x3	2

Layers	**Out Size**	**Stride**
Input: $\boldsymbol{z}$	1x1x8	-
7x7 ConvT(ngf x 8), LReLU	7x7x(ngf x 8)	1
4x4 ConvT(ngf x 4), LReLU	14x14x(ngf x 4)	2
4x4 ConvT(ngf x 2), LReLU	28x28x(ngf x 2)	2
3x3 ConvT(1), Tanh	28x28x1	1

$\beta_2 = 0.999$ and `weight_decay=1e-4` to train the diffusion network. The initial learning rates of the generator and diffusion networks are `2e-4`, and `1e-4` for the energy score network. The

Table 3.7: **Network structure of the denoising diffusion network.** a) We use the sinusoidal embedding to embed the time index as in [8, 25]. b) We use the learned fourier feature module [75] to embed the input $\boldsymbol{z}$. c) The merged time embedding and context embedding is used to produce a pair of bias and scale terms to shift and scale the embedding of input $\boldsymbol{z}$. `nz` is the input dimension, as in Table 3.5. `nemb` is the dimension of image embedding as in Table 3.8.

Layers	**Out size**	**Note**
Time Embedding		
Input: t	1	time index
Sin. emb.[a]	128	
Linear, SiLU	128	
Linear	128	
Input Embedding		
Input: $\boldsymbol{z}$	nz	
Fr. emb.[b]	2x(nz)	Fourier feature
Basic Block		
Input: $\boldsymbol{z}, \boldsymbol{z}_{\text{ctx}}, \boldsymbol{z}_t$	nzf, nemb, 128	$\boldsymbol{z}$, ctx. and t emb.
Cat, SiLU	nemb + 128	merge ctx. & t emb.
Linear, SiLU	nout	
Linear	nout	Input emb.
Scale-shift[c]	nout	scale-shift $\boldsymbol{z}$ emb. w/ merged emb.
Add $\boldsymbol{z}$	nout	skip connection from $\boldsymbol{z}$
Denoising Diffusion Network		
Input: $\boldsymbol{z}, \boldsymbol{z}_{\text{ctx}}, t$	nz, nemb, 128	Input
Embedding	2x(nz), 128	Input & t emb.
Basic Block	128	Encoding
Basic Block	256	
Basic Block	256	
Basic Block	256	Intermediate
Basic Block	256	Cat & Decoding
Basic Block	128	
Basic Block	nz	Output

learning rates are decayed with a factor of 0.99 every 1K training iterations, with a minimum learning rate of `1e-5`. We run the experiments on a A6000 GPU with the batch size of 128. For GAN inversion, we reduce the batch size to 64. Training typically converges within 200K iterations on all the datasets.

Table 3.8: **Network structures of the encoder networks** used for the SVHN, CelebA, CIFAR-10, CelebA-HQ and MNIST (from top to bottom) datasets. For GAN inversion, the encoder network structure is the same as in [18]. Conv(n)Norm indicates a convolutional operation with n output channels followed by the Instance Normalization [76]. We use `nif=64` and `nemb=1024` for all the datasets. LReLU indicates the Leaky-ReLU activation function. The slope in Leaky ReLU is set to be 0.2.

Layers	**Out Size**	**Stride**
Input: $\boldsymbol{x}$	32x32x3	-
3x3 Conv(nif x 1)Norm, LReLU	32x32x(nif x 1)	1
4x4 Conv(nif x 2)Norm, LReLU	16x16x(nif x 2)	2
4x4 Conv(nif x 4)Norm, LReLU	8x8x(nif x 4)	2
4x4 Conv(nif x 8)Norm, LReLU	4x4x(nif x 8)	2
4x4 Conv(nemb)Norm, LReLU	1x1x(nemb)	1

Layers	**Out Size**	**Stride**
Input: $\boldsymbol{x}$	64x64x3	-
3x3 Conv(nif x 1)Norm, LReLU	64x64x(nif x 1)	1
4x4 Conv(nif x 2)Norm, LReLU	32x32x(nif x 2)	2
4x4 Conv(nif x 4)Norm, LReLU	16x16x(nif x 4)	2
4x4 Conv(nif x 8)Norm, LReLU	8x8x(nif x 8)	2
4x4 Conv(nif x 8)Norm, LReLU	4x4x(nif x 8)	2
4x4 Conv(nemb)Norm, LReLU	1x1x(nemb)	1

Layers	**Out Size**	**Stride**
Input: $\boldsymbol{x}$	32x32x3	-
3x3 Conv(nif x 1)Norm, LReLU	32x32x(nif x 1)	1
4x4 Conv(nif x 2)Norm, LReLU	16x16x(nif x 2)	2
4x4 Conv(nif x 4)Norm, LReLU	8x8x(nif x 4)	2
4x4 Conv(nif x 8)Norm, LReLU	4x4x(nif x 8)	2
4x4 Conv(nemb)Norm, LReLU	1x1x(nemb)	1

Layers	**Out Size**	**Stride**
Input: $\boldsymbol{x}$	256x256x3	-
3x3 Conv(nif x 1)Norm, LReLU	256x256x(nif x 1)	1
4x4 Conv(nif x 2)Norm, LReLU	128x128x(nif x 2)	2
4x4 Conv(nif x 4)Norm, LReLU	64x64x(nif x 4)	2
4x4 Conv(nif x 4)Norm, LReLU	32x32x(nif x 4)	2
4x4 Conv(nif x 8)Norm, LReLU	16x16x(nif x 8)	2
4x4 Conv(nif x 8)Norm, LReLU	8x8x(nif x 8)	2
4x4 Conv(nif x 8)Norm, LReLU	4x4x(nif x 8)	2
4x4 Conv(nemb)Norm, LReLU	1x1x(nemb)	1

Layers	**Out Size**	**Stride**
Input: $\boldsymbol{x}$	28x28x3	-
3x3 Conv(nif x 1)Norm, LReLU	28x28x(nif x 1)	1
4x4 Conv(nif x 2)Norm, LReLU	14x14x(nif x 2)	2
4x4 Conv(nif x 4)Norm, LReLU	7x7x(nif x 4)	2
4x4 Conv(nif x 8)Norm, LReLU	3x3x(nif x 8)	2
3x3 Conv(nemb)Norm, LReLU	1x1x(nemb)	1

3.10 Pytorch-style Pseudocode

We provide pytorch-style pseudocode to help understand the proposed method. We denote the generator network as `G`, the energy score network as `E` and the diffusion network as `Q`. The first page sketches the prior and posterior sampling process. The second page outlines the learning procedure.

Listing 3.1: Prior and posterior LD sampling.

```
def sample_langevin_prior_z(z, netE):
    s = step_size

    for i in range(n_steps):
        en = netE(z).sum()
        z_norm = 1.0 / 2.0 * torch.sum(z**2)
        z_grad = torch.autograd.grad(en + z_norm, z)[0]
        w = torch.randn_like(z)

        # Prior LD Update
        z.data = z.data - 0.5 * (s ** 2) * z_grad + s * w

    return z.detach()

def sample_langevin_posterior_z(z, x, netG, netE):
    s = step_size
    sigma_inv = 1.0 / (2.0 * sigma ** 2)

    for i in range(n_steps):
        x_hat = netG(z)
```

```
        g_log_lkhd = sigma_inv * torch.sum((x_hat - x) ** 2)

        z_n = 1.0 / 2.0 * torch.sum(z**2)
        en = netE(z).sum()

        total_en = g_log_lkhd + en + z_n
        z_grad = torch.autograd.grad(total_en, z)[0]
        w = torch.randn_like(z)

        # Posterior LD Update
        z.data = z.data - 0.5 * (s ** 2) * z_grad + s * w
    return z.detach()
```

Listing 3.2: Learning LEBM with DAMC.

```
for x in dataset:
    # mask for unconditional learning of DAMC
    z_mask_prob = torch.rand((len(x),), device=x.device)
    z_mask = torch.ones(len(x), device=x.device)
    z_mask[z_mask_prob < 0.2] = 0.0
    z_mask = z_mask.unsqueeze(-1)

    # draw DAMC samples
    z0 = Q(x)
    zk_pos, zk_neg = z0.detach().clone(), z0.detach().clone()

    # prior and posterior updates
    zk_pos = sample_langevin_posterior_z(
            z=zk_pos, x=x, netG=G, netE=E)
    zk_neg = sample_langevin_prior_z(
            z=torch.cat(
            [zk_neg, torch.randn_like(zk_neg)], dim=0),
            netE=E)

    # update Q
    for __ in range(6):
        Q_optimizer.zero_grad()
        Q_loss = Q.calculate_loss(
        x=x, z=zk_pos, mask=z_mask).mean()
        Q_loss.backward()
        Q_optimizer.step()
```

```
# update G
G_optimizer.zero_grad()
x_hat = G(zk_pos)
g_loss = torch.sum((x_hat - x) ** 2, dim=[1,2,3]).mean()
g_loss.backward()
G_optimizer.step()

# update E
E_optimizer.zero_grad()
e_pos, e_neg = E(zk_pos), E(zk_neg)
E_loss = e_pos.mean() - e_neg.mean()
E_loss.backward()
E_optimizer.step()
```

3.11 Dataset and Experiment Settings

Datasets We include the following datasets to study our method: SVHN ($32 \times 32 \times 3$), CIFAR-10 ($32 \times 32 \times 3$), CelebA ($64 \times 64 \times 3$), CeleAMask-HQ (256 x 256 x 3) and MNIST (28 x 28 x 1). Following Pang et al. [22], we use the full training set of SVHN (73,257) and CIFAR-10 (50,000), and take 40,000 samples of CelebA as the training data. We take 29,500 samples from the CelebAMask-HQ dataset as the training data, and test the model on 500 held-out samples. For anomaly detection on MNIST dataset, we follow the experimental settings in [22, 41, 55, 56] and use 80% of the in-domain data to train the model. The images are scaled to $[-1, 1]$ and randomly horizontally flipped with a prob. of .5 for training.

GAN inversion settings We attempt to use the DAMC sampler for GAN version on the FFHQ (256 x 256 x 3) and LSUN-Tower (256 x 256 x 3) datasets. We take 69,500 samples from the CelebAMask-HQ dataset as the training data, and use the held-out 500 samples for testing. We follow the default data splits of the LSUN dataset.

For the LEBM-based inversion method, we train a LEBM in the 14 x 512 = 7168 dim. latent space. During training, we add l_2-regularization on the energy score of LEBM to stabilize training, as suggested in [59]. For the DAMC sampler and the encoder-based inversion method [16], we generate initial posterior samples of the training data using [18] and train the DAMC sampler and the encoder-based method for 5K iterations using these samples as a warm-up step. The encoder-based method is trained by minimizing the l_2 distance between the encoder output and the target samples. After that, these two methods are trained with the default learning algorithms.

3.12 Additional Qualitative Results

3.12.1 Toy Examples

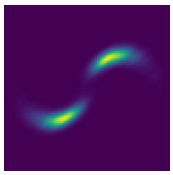
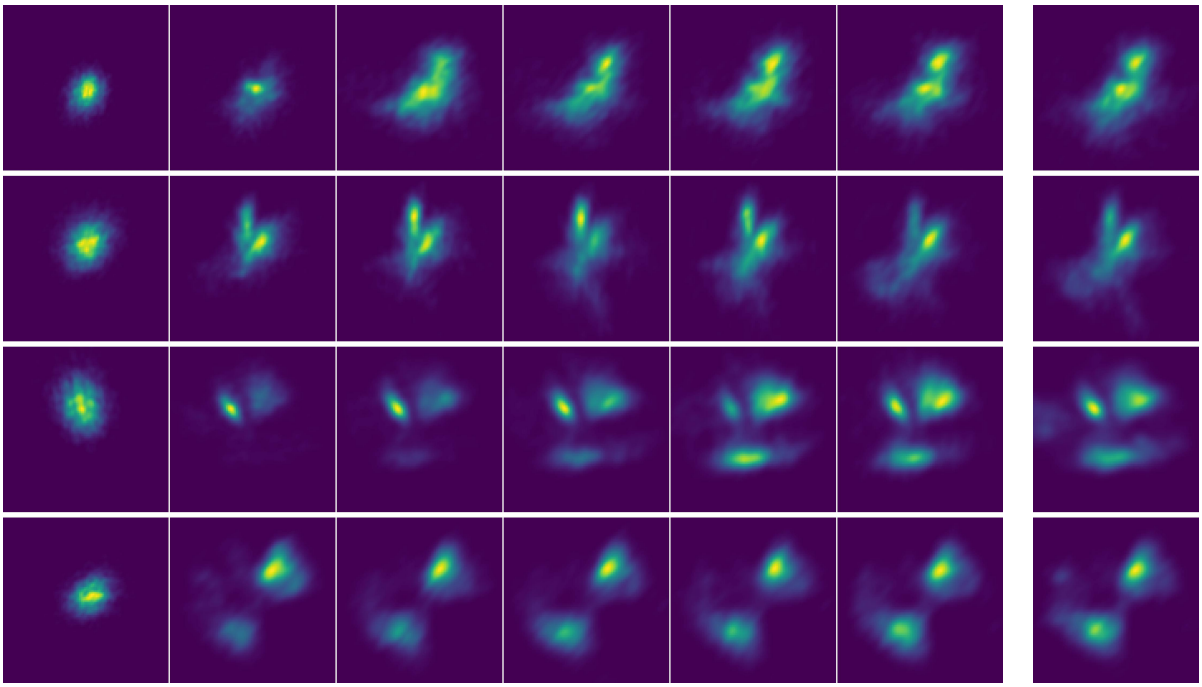


Figure 3.5: **The 2-arm pinwheel-shaped prior distribution used in the toy example.**



(a) Evolution of the learned posterior distributions (b) GT

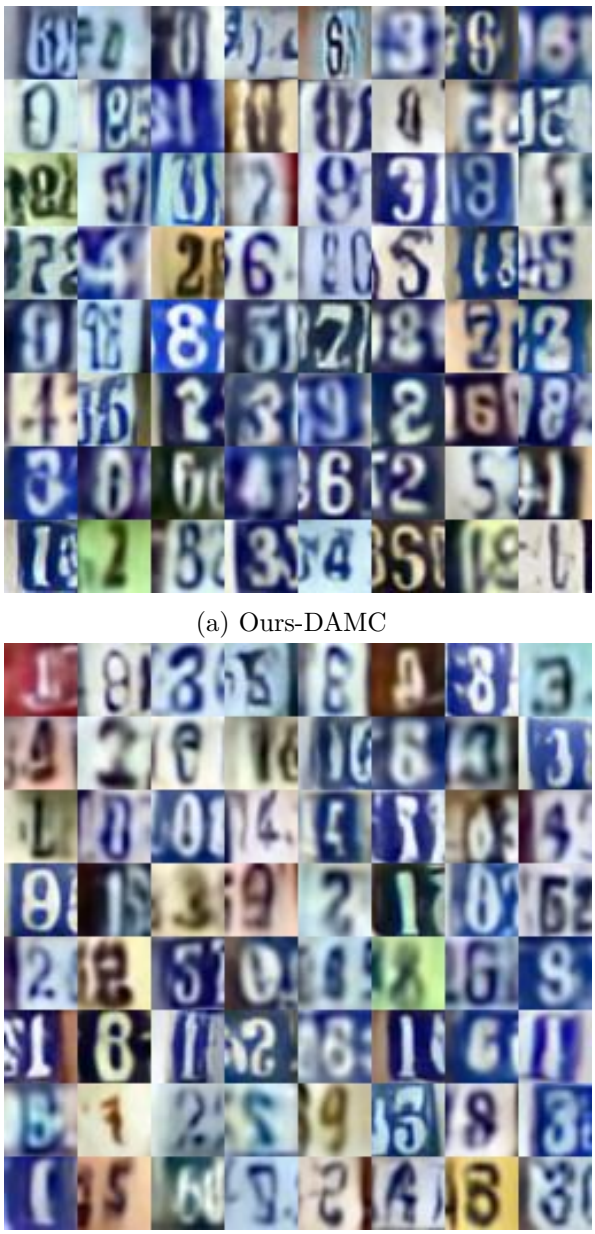
Figure 3.6: **Evolution of the posterior distributions learned by the DAMC sampler.** In each row, we display from left to right the evolution of the learned posterior distributions from the DAMC sampler through training iterations. The last column shows the corresponding ground-truth posterior distribution obtained by running 1K-3K steps of Langevin Dynamics until convergence for posterior sampling.

As the proof-of-concept toy examples, we implement the neural likelihood experiments following the same set-up mentioned in Section 5.1 and B.1 in [80]. We choose to use a more complex prior distribution, i.e., 2-arm pinwheel-shaped prior distribution (shown in Fig. 3.5) instead of a standard normal one to make sure that the true posterior distributions are multimodal. The ground-truth posterior distributions are obtained by performing long-run

LD sampling until convergence. We visualize the convergence trajectory of the posterior distributions learned by our model to the ground-truth ones. In Fig. 3.6, we can see that our model can faithfully reproduce the ground-truth distributions with sufficient training iterations, which indicates that the learned sampler successfully amortizes the long-run sampling chain with the length of $1,000$-$3,000$ iterations.

3.12.2 Generation

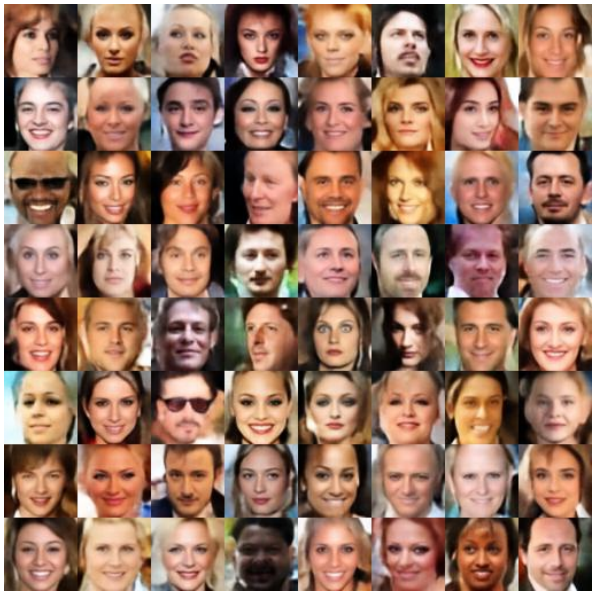
We provide additional generated samples from our models trained on SVHN (Fig. 3.7), CelebA (Fig. 3.8), CIFAR-10 (Fig. 3.9) and CelebA-HQ (Fig. 3.10).



(a) Ours-DAMC

(b) Ours-LEBM

Figure 3.7: **Samples generated from the DAMC sampler and LEBM** trained on the SVHN dataset.



(a) Ours-DAMC



(b) Ours-LEBM

Figure 3.8: **Samples generated from the DAMC sampler and LEBM** trained on the CelebA dataset.



(a) Ours-DAMC



(b) Ours-LEBM

Figure 3.9: **Samples generated from the DAMC sampler and LEBM** trained on the CIFAR-10 dataset.



(a) Ours-DAMC



(b) Ours-LEBM

Figure 3.10: **Samples generated from the DAMC sampler and LEBM** trained on the CelebA-HQ dataset.

3.12.3 Reconstruction

We provide qualitative examples about the reconstruction results from our models trained on SVHN (Fig. 3.11), CelebA (Fig. 3.12), CIFAR-10 (Fig. 3.13) and CelebA-HQ (Fig. 3.14). Observed images are sampled from the testing set unseen during training.

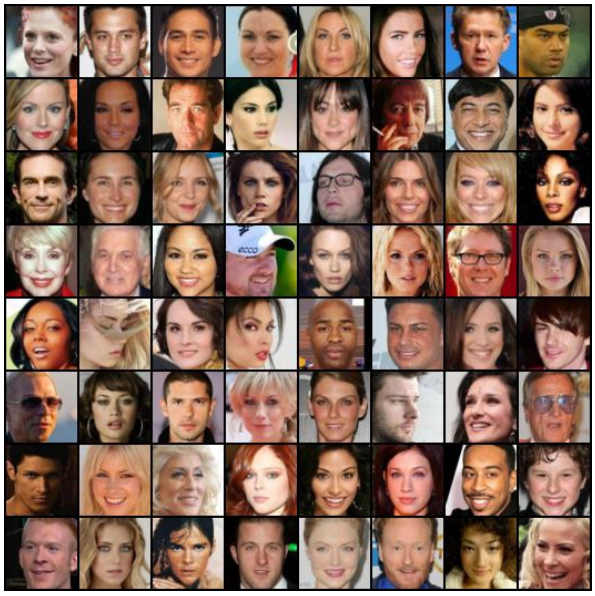


(a) Observation

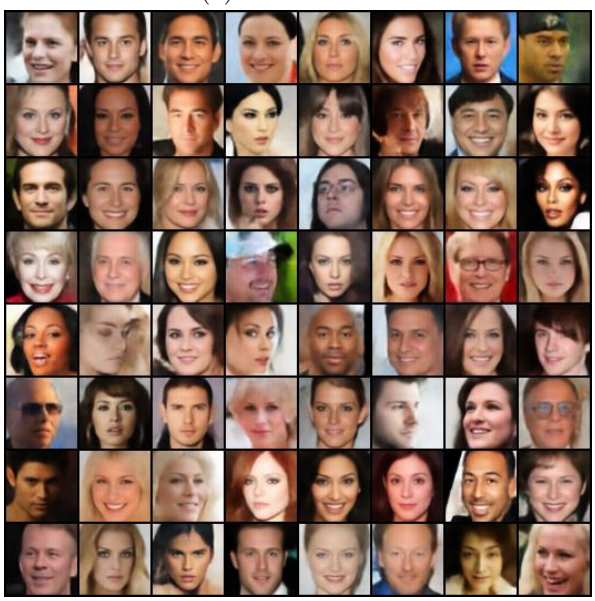


(b) Reconstruction

Figure 3.11: **Reconstructed samples from the posterior DAMC sampler** trained on the SVHN dataset.



(a) Observation



(b) Reconstruction

Figure 3.12: **Reconstructed samples from the posterior DAMC sampler** trained on the CelebA dataset.



(a) Observation



(b) Reconstruction

Figure 3.13: **Reconstructed samples from the posterior DAMC sampler** trained on the CIFAR-10 dataset.



(a) Observation

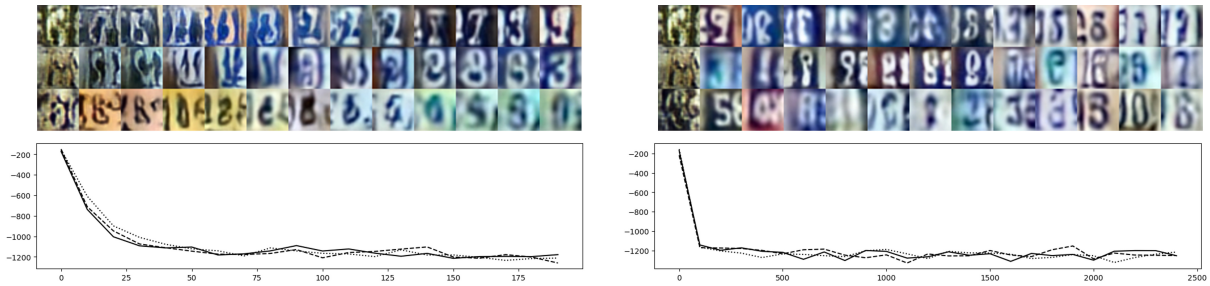


(b) Reconstruction

Figure 3.14: **Reconstructed samples from the posterior DAMC sampler** trained on the CelebA-HQ dataset.

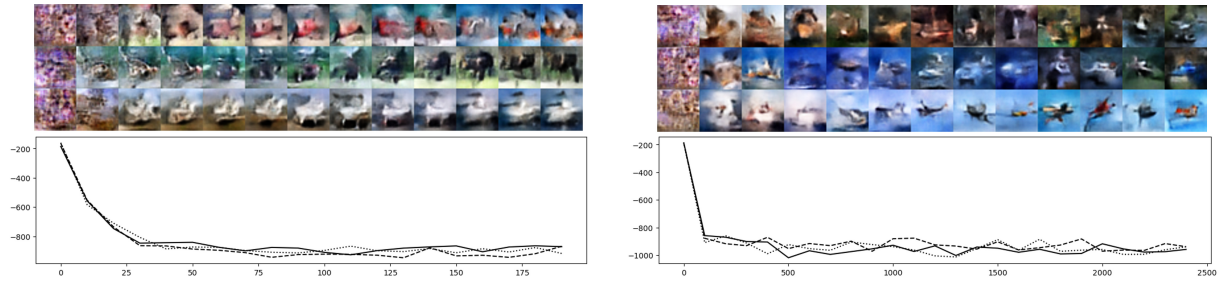
3.12.4 Visualization of Transitions

We provide additional visualization results of LD transitions initialized from $\mathcal{N}(0, \mathbf{I}_d)$ on SVHN (Fig. 3.15) and CIFAR-10 datasets (Fig. 3.16). For the 200-step set-up, we can see that the generation quality quickly improves by exploring the local modes with LD. For the 2500-step long-run set-up, we can see that the LD produces consistently valid results without the oversaturating issue of the long-run chain samples.



(a) 200 steps (b) 2500 steps

Figure 3.15: **Transition of Markov chains initialized from $\mathcal{N}(0, \mathbf{I}_d)$ towards $p_{\boldsymbol{\alpha}}(\boldsymbol{z})$ on SVHN.** We present results by running LD for 200 and 2500 steps. In each sub-figure, the top panel displays the trajectory in the data space uniformly sampled along the chain. The bottom panel shows the energy score $f_{\boldsymbol{\alpha}}(\boldsymbol{z})$ over the iterations.



(a) 200 steps (b) 2500 steps

Figure 3.16: **Transition of Markov chains initialized from $\mathcal{N}(0, \mathbf{I}_d)$ towards $p_{\boldsymbol{\alpha}}(\boldsymbol{z})$ on CIFAR-10.** We present results by running LD for 200 and 2500 steps. In each sub-figure, the top panel displays the trajectory in the data space uniformly sampled along the chain. The bottom panel shows the energy score $f_{\boldsymbol{\alpha}}(\boldsymbol{z})$ over the iterations.

3.13 Additional Quantitative Results

3.13.1 Learning DAMC in the Latent Space of Other DLVMs

We have compared our method with directly training diffusion models in the latent space. In the NALR-LEBM column in Table 3.4, we compare our method with learning a diffusion model in the pre-trained energy-based latent space on CIFAR-10 dataset. We further add experiments of learning this model in the pre-trained VAE [1] and ABP [51] model latent spaces summarized in Table 3.9. We can see that our method greatly outperforms these baselines using the same network architectures.

Table 3.9: **Further baseline results for learning DAMC on CIFAR-10 dataset.** These models are implemented using the same encoder and decoder network architectures for fair comparison

	VAE	ABP	NALR	Ours
FID	102.54	69.93	64.38	**57.52**
MSE	0.036	0.017	0.016	**0.015**

3.13.2 Further Ablation Studies

We conduct more ablation studies to inspect the impact of some hyper-parameters. For prior LD sampling, we follow [22] and set the sampling step as 60 throughout the experiments for fair comparison. We further add experiments on CIFAR-10 for training our model with different posterior LD sampling steps `T` and with different capacities of the amortizer q_ϕ, summarized in Table 3.10. We observe that larger posterior sampling steps and larger model capacities for training brings marginal improvement compared with our default set-up. Less sampling steps and lower model capacities, however, may have negative impact on the performances.

(a) Posterior sampling steps `T`

	`T=10`	`T=30`	`T=50`
FID	74.20	**57.52**	57.03
MSE	0.016	**0.015**	0.015

(b) Model capacity of q_ϕ

	`f=1/4`	`f=1/2`	`f=1`	`f=2`	`f=4`
FID	116.28	80.28	**57.52**	57.82	57.56
MSE	0.017	0.016	**0.015**	0.015	0.015

Table 3.10: **Ablation Studies for the choice of Langevin steps and model capacity**. We highlight the results of our set-up reported in the main text. `T` is the posterior sampling step. `f` stands for the factor for the model capacity, *e.g.*, `f=2` means 2x the size of the original model.

3.14 Further Discussion

3.14.1 Limitations

We mentioned in the main text that one potential disadvantage of our method is its parameter inefficiency for introducing an extra DDPM. Although fortunately, our models are in the latent space so the network is lightweight. To be specific, on SVHN, CelebA, CIFAR-10 and CelebA-HQ datasets the number of parameters in the diffusion network is around 10% of those in the generator.

Another issue is the time efficiency. We mentioned in the main text that the time efficiency for sampling is competitive. With the batch size of 64, on these datasets the DAMC prior sampling takes $0.3s$, while 100 steps of short-run LD with LEBM takes $0.2s$. The DAMC posterior sampling takes $1.0s$, while LEBM takes $8.0s$. However, during training we need to run 30 steps of posterior LD sampling and 60 steps of prior LD sampling in each training iteration. We observe that the proposed learning method takes 15.2 minutes per training epoch, while the short-run LD-based learning method takes 14.8 minutes per epoch. These methods are slower than the VAE-base method, which takes 5.5 minutes for an training epoch. We can see that the time efficiency for training is generally bottlenecked by the LD sampling process, and could be improved in future works.

3.14.2 Broader Impacts

Generative models could be misused for disinformation or faking profiles. Our work focuses on the learning algorithm of energy-based prior model. Though we consider our work to be foundational and not tied to particular applications or deployments, it is possible that more powerful energy-based generative models augmented with this method may be used maliciously. Work on the reliable detection of synthetic content could be important to address such harms from generative models.

Bibliography

[1] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[2] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *Proceedings of International Conference on Machine Learning (ICML)*, 2014.

[3] Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2014.

[4] Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *International conference on machine learning*, pages 1530–1538. PMLR, 2015.

[5] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *Proceedings of International Conference on Machine Learning (ICML)*, 2015.

[6] Ali Razavi, Aaron Van den Oord, and Oriol Vinyals. Generating diverse high-fidelity images with vq-vae-2. *Advances in neural information processing systems*, 2019.

[7] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[8] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[9] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations (ICLR)*, 2020.

[10] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4401–4410, 2019.

[11] Zhifeng Kong, Wei Ping, Jiaji Huang, Kexin Zhao, and Bryan Catanzaro. Diffwave: A versatile diffusion model for audio synthesis. *arXiv preprint arXiv:2009.09761*, 2020.

[12] Peiyu Yu, Sirui Xie, Xiaojian Ma, Yixin Zhu, Ying Nian Wu, and Song-Chun Zhu. Unsupervised foreground extraction via deep region competition. *Advances in Neural Information Processing Systems*, 2021.

[13] Shitong Luo and Wei Hu. Diffusion probabilistic models for 3d point cloud generation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[14] Arash Vahdat, Karsten Kreis, and Jan Kautz. Score-based generative modeling in latent space. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[15] Peiyu Yu, Sirui Xie, Xiaojian Ma, Baoxiong Jia, Bo Pang, Ruigi Gao, Yixin Zhu, Song-Chun Zhu, and Ying Nian Wu. Latent diffusion energy-based model for interpretable text modeling. *arXiv preprint arXiv:2206.05895*, 2022.

[16] Jun-Yan Zhu, Philipp Krähenbühl, Eli Shechtman, and Alexei A Efros. Generative visual manipulation on the natural image manifold. In *Proceedings of European Conference on Computer Vision (ECCV)*, 2016.

[17] Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. *Advances in neural information processing systems*, 2017.

[18] Jiapeng Zhu, Yujun Shen, Deli Zhao, and Bolei Zhou. In-domain gan inversion for real image editing. In *Proceedings of European Conference on Computer Vision (ECCV)*, 2020.

[19] Yilun Du, Shuang Li, and Igor Mordatch. Compositional visual generation with energy based models. *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[20] Will Grathwohl, Kuan-Chieh Wang, Jörn-Henrik Jacobsen, David Duvenaud, Mohammad Norouzi, and Kevin Swersky. Your classifier is secretly an energy based model and you should treat it like one. In *International Conference on Learning Representations (ICLR)*, 2020.

[21] Weili Nie, Arash Vahdat, and Anima Anandkumar. Controllable and compositional generation with latent-space energy-based models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[22] Bo Pang, Tian Han, Erik Nijkamp, Song-Chun Zhu, and Ying Nian Wu. Learning latent space energy-based prior model. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[23] Erik Nijkamp, Mitch Hill, Tian Han, Song-Chun Zhu, and Ying Nian Wu. On the anatomy of mcmc-based maximum likelihood learning of energy-based models. In *Proceedings of AAAI Conference on Artificial Intelligence (AAAI)*, 2020.

[24] Erik Nijkamp, Mitch Hill, Song-Chun Zhu, and Ying Nian Wu. Learning non-convergent non-persistent short-run mcmc toward energy-based model. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[25] Ruiqi Gao, Yang Song, Ben Poole, Ying Nian Wu, and Diederik P Kingma. Learning energy-based models by diffusion recovery likelihood. In *International Conference on Learning Representations (ICLR)*, 2020.

[26] Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew M Dai, Rafal Jozefowicz, and Samy Bengio. Generating sentences from a continuous space. In *Proceedings of the Conference on Computational Natural Language Learning (CoNLL)*, 2016.

[27] Ali Razavi, Aäron van den Oord, Ben Poole, and Oriol Vinyals. Preventing posterior collapse with delta-vaes. *arXiv preprint arXiv:1901.03416*, 2019.

[28] James Lucas, George Tucker, Roger B Grosse, and Mohammad Norouzi. Don't blame the elbo! a linear vae perspective on posterior collapse. *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[29] Rajesh Ranganath, Dustin Tran, and David Blei. Hierarchical variational models. In *Proceedings of International Conference on Machine Learning (ICML)*, 2016.

[30] Jakub Tomczak and Max Welling. Vae with a vampprior. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2018.

[31] Yang Song and Stefano Ermon. Improved techniques for training score-based generative models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[32] Max Welling and Yee W Teh. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of International Conference on Machine Learning (ICML)*, 2011.

[33] Diederik Kingma, Tim Salimans, Ben Poole, and Jonathan Ho. Variational diffusion models. *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[34] Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *Proceedings of International Conference on Machine Learning (ICML)*, 2021.

[35] Pascal Vincent. A connection between score matching and denoising autoencoders. *Neural computation*, 23(7):1661–1674, 2011.

[36] Yingzhen Li, Richard E Turner, and Qiang Liu. Approximate inference with amortised mcmc. *arXiv preprint arXiv:1702.08343*, 2017.

[37] Jianwen Xie, Yang Lu, Ruiqi Gao, and Ying Nian Wu. Cooperative learning of energy-based model and latent variable model via mcmc teaching. In *Proceedings of AAAI Conference on Artificial Intelligence (AAAI)*, 2018.

[38] Francisco Ruiz and Michalis Titsias. A contrastive divergence for combining variational inference and mcmc. In *Proceedings of International Conference on Machine Learning (ICML)*, 2019.

[39] Thomas M Cover. *Elements of information theory*. John Wiley & Sons, 1999.

[40] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. In *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*.

[41] Zhisheng Xiao and Tian Han. Adaptive multi-stage density ratio estimation for learning latent space energy-based model. *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[42] Yann LeCun. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998.

[43] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. 2011.

[44] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of the IEEE international conference on computer vision*, pages 3730–3738, 2015.

[45] Alex Krizhevsky et al. Learning multiple layers of features from tiny images. 2009.

[46] Cheng-Han Lee, Ziwei Liu, Lingyun Wu, and Ping Luo. Maskgan: Towards diverse and interactive facial image manipulation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[47] Fisher Yu, Ari Seff, Yinda Zhang, Shuran Song, Thomas Funkhouser, and Jianxiong Xiao. Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint arXiv:1506.03365*, 2015.

[48] Bin Dai and David Wipf. Diagnosing and enhancing vae models. *International Conference on Learning Representations (ICLR)*, 2019.

[49] Partha Ghosh, Mehdi SM Sajjadi, Antonio Vergari, Michael Black, and Bernhard Schölkopf. From variational to deterministic autoencoders. *International Conference on Learning Representations (ICLR)*, 2020.

[50] Jyoti Aneja, Alex Schwing, Jan Kautz, and Arash Vahdat. A contrastive learning approach for training variational autoencoder priors. *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[51] Tian Han, Yang Lu, Song-Chun Zhu, and Ying Nian Wu. Alternating back-propagation for generator network. In *Proceedings of AAAI Conference on Artificial Intelligence (AAAI)*, 2017.

[52] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[53] Rameen Abdal, Yipeng Qin, and Peter Wonka. Image2stylegan: How to embed images into the stylegan latent space? In *Proceedings of International Conference on Computer Vision (ICCV)*, 2019.

[54] Weihao Xia, Yulun Zhang, Yujiu Yang, Jing-Hao Xue, Bolei Zhou, and Ming-Hsuan Yang. Gan inversion: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[55] Rithesh Kumar, Sherjil Ozair, Anirudh Goyal, Aaron Courville, and Yoshua Bengio. Maximum entropy generators for energy-based models. *arXiv preprint arXiv:1901.08508*, 2019.

[56] Houssam Zenati, Chuan Sheng Foo, Bruno Lecouat, Gaurav Manek, and Vijay Ramaseshan Chandrasekhar. Efficient gan-based anomaly detection. *Proceedings of the 20th IEEE International Conference on Data Mining (ICDM), 2018*, 2018.

[57] Tijmen Tieleman. Training restricted boltzmann machines using approximations to the likelihood gradient. In *Proceedings of International Conference on Machine Learning (ICML)*, 2008.

[58] Jianwen Xie, Yang Lu, Song-Chun Zhu, and Yingnian Wu. A theory of generative convnet. In *Proceedings of International Conference on Machine Learning (ICML)*, 2016.

[59] Yilun Du and Igor Mordatch. Implicit generation and generalization in energy-based models. *arXiv preprint arXiv:1903.08689*, 2019.

[60] Yilun Du, Shuang Li, Joshua Tenenbaum, and Igor Mordatch. Improved contrastive divergence training of energy based models. In *Proceedings of International Conference on Machine Learning (ICML)*, 2021.

[61] Bo Pang, Tian Han, and Ying Nian Wu. Learning latent space energy-based prior model for molecule generation. *arXiv preprint arXiv:2010.09351*, 2020.

[62] Bo Pang, Tianyang Zhao, Xu Xie, and Ying Nian Wu. Trajectory prediction with latent belief energy-based model. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[63] Jianwen Xie, Zilong Zheng, Ruiqi Gao, Wenguan Wang, Song-Chun Zhu, and Ying Nian Wu. Generative voxelnet: learning energy-based models for 3d shape synthesis and analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(5): 2468–2484, 2020.

[64] Jianwen Xie, Zilong Zheng, and Ping Li. Learning energy-based model with variational auto-encoder as amortized sampler. In *Proceedings of AAAI Conference on Artificial Intelligence (AAAI)*, 2021.

[65] Jianwen Xie, Yaxuan Zhu, Jun Li, and Ping Li. A tale of two flows: cooperative learning of langevin flow and normalizing flow toward energy-based model. *arXiv preprint arXiv:2205.06924*, 2022.

[66] Yaxuan Zhu, Jianwen Xie, Yingnian Wu, and Ruiqi Gao. Learning energy-based models by cooperative diffusion recovery likelihood. *arXiv preprint arXiv:2309.05153*, 2023.

[67] Geoffrey E Hinton. Training products of experts by minimizing contrastive divergence. *Neural computation*, 14(8):1771–1800, 2002.

[68] Michael Gutmann and Aapo Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, 2010.

[69] Ruiqi Gao, Erik Nijkamp, Diederik P Kingma, Zhen Xu, Andrew M Dai, and Ying Nian Wu. Flow contrastive estimation of energy-based models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[70] Arash Vahdat and Jan Kautz. Nvae: A deep hierarchical variational autoencoder. *Advances in neural information processing systems*, 2020.

[71] JM Bernardo, MJ Bayarri, JO Berger, AP Dawid, D Heckerman, AFM Smith, M West,

et al. The variational bayesian em algorithm for incomplete data: with application to scoring graphical model structures. *Bayesian statistics*, 7(453-464):210, 2003.

[72] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[73] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, pages 234–241. Springer, 2015.

[74] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[75] Alexander Li and Deepak Pathak. Functional regularization for reinforcement learning via learned fourier features. *Advances in Neural Information Processing Systems*, 2021.

[76] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*, 2016.

[77] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[78] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[79] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[80] Shohei Taniguchi, Yusuke Iwasawa, Wataru Kumagai, and Yutaka Matsuo. Langevin autoencoders for learning deep latent variable models. *Advances in Neural Information Processing Systems*, 35:13277–13289, 2022.

Chapter 4

Learning Multi-Denoiser Guidance for Diffusion Sampling

4.1 Introduction

Diffusion models generate data by following a learned denoising process from a noise distribution toward a distribution of clean samples. Their practical success relies on a sequence of neural-network predictions that approximate the reverse-time evolution. Even when a model is trained on a large dataset, a finite-capacity denoiser may introduce systematic approximation errors. These errors can influence both the fidelity and diversity of the resulting samples. Consequently, the quality of a diffusion sampler depends not only on the underlying learned generative model but also on how the denoising predictions are used during inference.

A common approach to modifying inference is *guidance*, in which the prediction of a primary denoiser is combined with information from an additional model. Classifier-free guidance combines conditional and unconditional predictions [1], while auto-guidance uses a weaker version of the generating model as a guidance signal [2]. Both approaches modify the reverse dynamics through a weighted combination of predictions, making the auxiliary model and guidance strength consequential choices.

This chapter presents a multi-denoiser guidance framework disclosed in the publicly available U.S. patent application *Enhanced Diffusion Model Guidance via Multi-Denoiser Mixing* [4]. Rather than relying on a single auxiliary denoiser with manually tuned guidance strength, the framework combines several pretrained denoisers into a guidance model with learnable mixture weights. We develop its product-of-experts motivation, weighted denoiser predictions, reward-regularized formulation, and preference-based learning procedure related to diffusion direct preference optimization (DPO).

The resulting method addresses a sampling-time learning problem. The parameters of the constituent diffusion models may remain fixed, while a comparatively small collection of guidance coefficients determines how their predictions interact. Thus, the method separates the computationally expensive task of pretraining denoisers from the subsequent task of choosing a useful combined sampler. This distinction connects naturally to the broader thesis theme: learning the distribution represented by a neural model and designing an effective procedure for obtaining samples from it are related, but distinct, statistical problems.

4.2 Multi-Denoiser Guidance Formulation

4.2.1 Pretrained models and a formal product-of-experts target

Let c denote a conditioning variable, and let $p_k(x_0 \mid c)$ be the clean-data distribution associated with a pretrained diffusion model, indexed by $k = 0, \ldots, K$. We reserve $k = 0$ for a base model and consider the remaining models as candidates for guidance. The component models may differ in network capacity, training duration, or training task [4]. Combining their predictions assumes a common data space and compatible noise, conditioning, and output parameterizations, or an explicit mapping into such a common parameterization.

A distribution-level motivation for combining the models is the following product-of-

experts expression:

$$\widetilde{p}_{\boldsymbol{a}}(x_0 \mid c) = \prod_{k=0}^{K} p_k(x_0 \mid c)^{a_k}, \qquad p_{\boldsymbol{a}}(x_0 \mid c) = \frac{\widetilde{p}_{\boldsymbol{a}}(x_0 \mid c)}{Z_{\boldsymbol{a}}(c)}, \tag{4.1}$$

where $\boldsymbol{a} = (a_0, \ldots, a_K)$ represents combination weights and

$$Z_{\boldsymbol{a}}(c) = \int \widetilde{p}_{\boldsymbol{a}}(x_0 \mid c)\, \mathrm{d}x_0 \tag{4.2}$$

whenever the integral is finite and nonzero. Introducing $Z_{\boldsymbol{a}}$ simply makes explicit the normalizing factor omitted from the proportionality-form product. Equation (4.1) is a formal probabilistic interpretation; it does not imply that either its normalization constant or its exact sampling dynamics are available in closed form.

This viewpoint illustrates why combining several models can be more expressive than a single auxiliary-guidance direction. Each exponent changes the relative influence of a component distribution, and the vector $\boldsymbol{a}$ can vary over a family of candidate sampling behaviors. In particular, a component model need not be individually superior to the base model to contain useful information for guidance. The role of a component is determined by its interaction with the others, not solely by its standalone sample quality.

4.2.2 Combining scores and denoiser predictions

Actual diffusion sampling operates at intermediate noise levels. Denote the perturbed input by x_σ, the noise scale by σ, and the output of the k-th pretrained denoiser by $D_k(x_\sigma, \sigma, c)$. We construct a combined prediction using weights that may depend on the noise level:

$$D_{\boldsymbol{a}}(x_\sigma, \sigma, c) = \sum_{k=0}^{K} a_k(\sigma) D_k(x_\sigma, \sigma, c). \tag{4.3}$$

A corresponding mixture of score estimates can instead be formed as

$$s_{\boldsymbol{a}}(x_\sigma, \sigma, c) = \sum_{k=0}^{K} a_k(\sigma) s_k(x_\sigma, \sigma, c). \tag{4.4}$$

These are algorithmic combinations of pretrained predictions; whether the denoiser and score versions coincide depends on the underlying parameterization and weights. In particular, the weights in these time-dependent combinations need not define a single, time-independent clean-data product-of-experts distribution.

A related implementation constructs an auxiliary mixture separately from the base prediction. For example, with $\sum_{k=1}^{K} b_k(\sigma) = 1$, let

$$D_{\text{aux}} = \sum_{k=1}^{K} b_k(\sigma) D_k, \qquad D_{\text{guided}} = D_0 + \gamma(\sigma)(D_{\text{aux}} - D_0), \tag{4.5}$$

where all predictions are evaluated at the same (x_σ, σ, c). This is an instance of Equation (4.3) with effective coefficients $a_0 = 1 - \gamma$ and $a_k = \gamma b_k$ for $k \geqslant 1$. Thus, learning an auxiliary mixture and applying it to guide the base model can be expressed within the same affine-combination framework; the external guidance scale γ need not be a learned parameter.

For example, under additive Gaussian corruption and the usual denoiser-to-score conversion,

$$s_k(x_\sigma, \sigma, c) = \frac{D_k(x_\sigma, \sigma, c) - x_\sigma}{\sigma^2}, \tag{4.6}$$

we obtain

$$\sum_k a_k(\sigma) s_k = \frac{\sum_k a_k(\sigma) D_k - \big(\sum_k a_k(\sigma)\big) x_\sigma}{\sigma^2}. \tag{4.7}$$

In this parameterization, combining denoisers as in Equation (4.3) is equivalent to combining scores as in Equation (4.4) when $\sum_k a_k(\sigma) = 1$. Otherwise, an explicit correction involving x_σ is needed. This qualification is relevant when guidance uses extrapolative or otherwise unconstrained weights.

Distributional interpretation versus implemented guidance. Even if the clean-data product in Equation (4.1) is a normalized probability distribution, its noisy marginal is not generally the product of the noisy marginals of the component models. Therefore the exact score of the noise-perturbed product-of-experts distribution need not equal the weighted sum of the individual noisy scores. Equations (4.3) and (4.4) should be read as the guidance rules adopted in this formulation, with Equation (4.1) providing a useful modeling motivation rather than an exact identification at all noise levels.

4.3 Learning Mixture Weights

4.3.1 Reward-regularized distribution optimization

A practical challenge is choosing the combination weights. Manual selection becomes less attractive as the number of constituent denoisers increases, especially when their relative usefulness may vary with the noise level or condition. A natural approach uses a reward-regularized optimization perspective. Write $\pi_{\boldsymbol{a}}(x_0 \mid c)$ for the distribution induced by the *implemented sampler* with weights $\boldsymbol{a}$; this distribution need not coincide with the formal product-of-experts distribution $p_{\boldsymbol{a}}$ in Equation (4.1). For a conditioning distribution $\mathcal{D}_c$ and a reward function $r(c, x_0)$, consider

$$\max_{\boldsymbol{a}} \; \mathbb{E}_{c\sim\mathcal{D}_c}\left[\mathbb{E}_{x_0\sim\pi_{\boldsymbol{a}}(\cdot|c)}[r(c,x_0)] - \beta\, D_{\mathrm{KL}}(\pi_{\boldsymbol{a}}(\cdot \mid c)\|p_{\mathrm{ref}}(\cdot \mid c))\right], \tag{4.8}$$

where p_{ref} is a reference distribution and $\beta > 0$ controls the penalty on distributional deviation. Here only the guidance coefficients $\boldsymbol{a}$, rather than the pretrained denoisers, are optimized [4]. The KL term is a distribution-level objective, not a claim that the exact density of $\pi_{\boldsymbol{a}}$ can be evaluated.

The two terms in Equation (4.8) express different desiderata. The reward term prefers outputs scored highly by r, whereas the divergence term discourages departures from the

reference model. If the reward is imperfect, unconstrained optimization may favor regions with spuriously high scores. The reference penalty is therefore intended to keep the modified sampling distribution near one whose outputs are already meaningful. The formulation does not guarantee better visual quality for an arbitrary reward; improvements depend on the suitability of the reward, the available component models, and the implemented optimization procedure.

The parameterization in Equation (4.3) makes it possible to change the sampling behavior while keeping pretrained denoiser parameters fixed. In this formulation, the learned variables are the mixture weights, potentially indexed by noise level. This is different from retraining a new diffusion model from scratch: a family of guidance policies is searched over the outputs of existing models.

4.3.2 Preference comparisons instead of explicit scalar rewards

A numerical reward is not always readily available. A preference-based alternative can instead be used, in which examples are organized into tuples

$$\mathcal{D}_{\text{pref}} = \{(c_i, x_{0,i}^{+}, x_{0,i}^{-})\}_{i=1}^{N}, \tag{4.9}$$

where x_0^+ is preferred to x_0^- under condition c. For conditional image generation, observed images paired with their class labels can serve as preferred examples. Less-preferred examples can be obtained from the base diffusion model or from several component denoisers. These are two possible construction strategies; neither is universally optimal. Such dataset-versus-model pairs are *constructed preferences*, rather than necessarily human-annotated comparisons; the probabilistic preference model below is an optimization assumption.

For a scalar reward r, the Bradley–Terry preference model assigns probability

$$\Pr\big(x_0^+ \succ x_0^- \mid c\big) = \operatorname{sigmoid}\big(r(c, x_0^+) - r(c, x_0^-)\big)\,. \tag{4.10}$$

This relates pairwise preference observations to reward differences while avoiding the need to specify the reward value for each individual sample.

4.4 Preference Optimization for Diffusion Guidance

4.4.1 From reward regularization to direct preference optimization

For intuition, consider optimizing Equation (4.8) over unrestricted probability distributions, with a fixed condition c. Under the standard support and normalization assumptions, the corresponding Gibbs-form optimizer is

$$p^*(x_0 \mid c) = \frac{p_{\text{ref}}(x_0 \mid c)\exp\big(r(c, x_0)/\beta\big)}{Z(c)}. \tag{4.11}$$

Rearranging yields

$$r(c, x_0) = \beta \log \frac{p^*(x_0 \mid c)}{p_{\text{ref}}(x_0 \mid c)} + \beta \log Z(c). \tag{4.12}$$

When comparing samples sharing the same condition, the additive term $\beta \log Z(c)$ cancels. Define the model-to-reference log-density ratio

$$h_{\boldsymbol{a}}(c, x_0) = \log \frac{\pi_{\boldsymbol{a}}(x_0 \mid c)}{p_{\text{ref}}(x_0 \mid c)}. \tag{4.13}$$

Substitution into Equation (4.10) motivates the DPO-style objective

$$\mathcal{L}_{\text{DPO}}(\boldsymbol{a}) = -\mathbb{E}_{\mathcal{D}_{\text{pref}}} \log \text{sigmoid}\big(\beta\big[h_{\boldsymbol{a}}(c, x_0^+) - h_{\boldsymbol{a}}(c, x_0^-)\big]\big)\,. \tag{4.14}$$

The expectation is over $(c, x_0^+, x_0^-) \sim \mathcal{D}_{\text{pref}}$. This is the distribution-level preference objective used to motivate the training procedure. Here, $\boldsymbol{a}$ denotes the weights determining a candidate model, not all neural-network parameters. The derivation provides motivation for direct optimization but does not make the clean-data likelihoods of the resulting diffusion sampler

tractable.

4.4.2 Diffusion surrogate based on denoising losses

Evaluating the exact clean-data log-density ratios in Equation (4.14) is difficult for diffusion models because the observed sample probability involves latent intermediate states along the reverse trajectory. We therefore follow the diffusion-DPO approach of approximating intractable clean-data likelihood ratios using a denoising-loss surrogate [3, 4].

Write a forward perturbation of a clean example as

$$x_t = \alpha_t x_0 + \sigma_t \epsilon, \qquad \epsilon \sim \mathcal{N}(0, I), \tag{4.15}$$

with signal-to-noise ratio $\lambda_t = \alpha_t^2/\sigma_t^2$. Assume for clarity that all models admit a common clean-data prediction $D_k(x_t, t, c)$ and the associated noise prediction $\epsilon_k = (x_t - \alpha_t D_k)/\sigma_t$. Then the noise-prediction mixture is

$$\epsilon_{\boldsymbol{a}} = \sum_{k=0}^{K} a_k(t)\epsilon_k = \frac{\big(\sum_k a_k(t)\big)x_t - \alpha_t D_{\boldsymbol{a}}(x_t, t, c)}{\sigma_t}. \tag{4.16}$$

Thus the familiar conversion $\epsilon_{\boldsymbol{a}} = (x_t - \alpha_t D_{\boldsymbol{a}})/\sigma_t$ requires $\sum_k a_k(t) = 1$. Let ϵ_{ref} be a fixed reference prediction, and for either $j \in \{+, -\}$ define the relative denoising error

$$\begin{aligned} \Delta^j_{\boldsymbol{a}}(t) = \; & \big\|\epsilon^j - \epsilon_{\boldsymbol{a}}(x_t^j, t, c)\big\|_2^2 \\ & - \big\|\epsilon^j - \epsilon_{\text{ref}}(x_t^j, t, c)\big\|_2^2 . \end{aligned} \tag{4.17}$$

The resulting preference-training surrogate can be written as

$$\boxed{\mathcal{L}_{\text{diff-DPO}}(\boldsymbol{a}) = -\mathbb{E}\log \text{sigmoid}\big[-\beta T\, \omega(\lambda_t)\big(\Delta^+_{\boldsymbol{a}}(t) - \Delta^-_{\boldsymbol{a}}(t)\big)\big]\,.} \tag{4.18}$$

The expectation is over preference pairs, a training timestep t, and the corresponding forward

noises ϵ^+ and ϵ^-. With uniform sampling from T discrete timesteps, T is the associated scaling factor; $\omega(\lambda_t)$ is a specified timestep-weighting function. We write ω rather than w to distinguish this scalar weighting function from the trainable denoiser-mixture weights $\boldsymbol{a}$.

The sign structure of Equation (4.18) is interpretable. Minimization favors a smaller relative denoising error on the preferred example than on the less-preferred example. This is a comparison between two relative errors, not a requirement that either error move in a particular direction on its own. The loss is a *surrogate*: its use of forward perturbations and denoising prediction errors should not be interpreted as an exact evaluation of the clean-data distribution in Equation (4.1). This distinction is important because the surrogate is not an exact product-of-experts likelihood.

4.5 Training and Sampling Procedures

4.5.1 Learning the guidance parameters

The guidance coefficients can be trained while keeping the component denoisers fixed. The essential procedure is summarized below [4]:

1. **Select models.** Choose a base denoiser, additional pretrained denoisers, and a reference prediction model; initialize the noise-dependent mixture weights $\boldsymbol{a}(\sigma)$.

2. **Construct preference examples.** Use tuples (c, x_0^+, x_0^-), such as observed conditional samples paired with generations from the base or component models.

3. **Perturb both examples.** For each pair, sample a diffusion time or noise scale and generate noisy inputs using the forward process.

4. **Combine denoiser predictions.** Evaluate the frozen component models at the sampled noise level, and form $D_{\boldsymbol{a}}$ or its noise-prediction counterpart using the current weights.

5. **Evaluate relative errors.** Compare mixed and reference predictions on both members of the preference pair using Equation (4.17).

6. **Update mixture weights.** Minimize the surrogate preference loss in Equation (4.18) with respect to $\boldsymbol{a}$, and iterate.

The procedure emphasizes the optimization variables and the relationship between denoising predictions and preference comparisons. Its performance depends on the particular optimizer, datasets, and choice of component models.

4.5.2 Guided generation

After learning the weights, inference begins from the usual initialization of the base diffusion sampler. At each denoising step, the selected denoisers evaluate the same noisy state. One may apply their affine combination directly through Equation (4.3); alternatively, an auxiliary mixture may be formed and used to steer the base prediction through Equation (4.5). The resulting prediction is passed to the sampler's update rule, which is iterated to produce a sample. In either implementation, the pretrained denoiser parameters remain fixed while the learned combination controls the sampling dynamics.

4.6 Discussion and Limitations

Flexible guidance without complete model retraining. The central methodological feature is the separation of pretrained denoiser parameters from learnable guidance parameters. Learning the combination coefficients provides a way to select among sampling behaviors induced by several existing models. This is particularly attractive when training a full generative model is substantially more expensive than optimizing a small number of guidance variables.

More components are not automatically better. The product-of-experts motivation introduces additional degrees of freedom, but it does not guarantee improvement. Component denoisers can be strongly correlated, disagree in unhelpful ways, or be calibrated differently. The final performance depends on model selection, the learned coefficients, and the chosen preference objective. Learning combination coefficients does not imply a universal guarantee that adding component models improves sample quality.

Formal distributions versus finite-step sampling. The clean-data product-of-experts expression and the actual finite-step sampler are separate mathematical objects. Learned guidance coefficients can modify the reverse trajectory, but numerical discretization and score approximation influence the distribution ultimately produced. Consequently, the output of the implemented algorithm should not be assumed to equal the ideal distribution in Equation (4.1) without further analysis. This is also why evaluating a sampler requires attention to the actual generated samples, not only to its formal density motivation.

Computational cost. Using several denoisers may add forward-pass cost at inference, even if training only the mixture weights is relatively inexpensive. The trade-off between guidance quality and additional model evaluations depends on the selected components and the implementation. There is no general complexity guarantee relative to single-model guidance; the main methodological contribution is a more flexible, trainable guidance mechanism.

Evaluation and limitations. The practical effects of mixture guidance should be evaluated under a common sampling budget, including both sample-quality and diversity metrics. Such comparisons would help determine whether improvements from additional guidance directions justify their inference cost. This chapter develops the formulation and training objective; it does not report an independent quantitative benchmark study.

4.7 Chapter Summary

This chapter describes an approach for learning diffusion guidance from multiple pretrained denoisers. A product-of-experts expression provides a distributional motivation for combining component models; weighted denoiser or score predictions yield an inference-time mechanism; and reward-regularized or preference-based optimization provides a way to learn the combination weights. The diffusion-DPO surrogate permits the weights to be updated using relative denoising errors in place of intractable clean-data likelihood ratios. The resulting methodology illustrates a broader principle of generative modeling: even when the underlying model family is fixed, the sampling procedure remains an object of statistical optimization.

4.8 Acknowledgments for This Research Chapter

The research described in this chapter was conducted in collaboration with Morteza Mardani and Arash Vahdat, and is documented in the publicly available U.S. patent application *Enhanced Diffusion Model Guidance via Multi-Denoiser Mixing* [4]. Peiyu Yu contributed to the conceptualization of the method, experimental design, and execution of the experiments. Morteza Mardani and Arash Vahdat each actively participated in discussions during method development and contributed ideas to the proposed approach.

Bibliography

[1] J. Ho and T. Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022. https://arxiv.org/abs/2207.12598.

[2] T. Karras, M. Aittala, T. Kynkäänniemi, J. Lehtinen, T. Aila, and S. Laine. Guiding a diffusion model with a bad version of itself. In *Advances in Neural Information Processing Systems*, vol. 37, 2024. https://doi.org/10.52202/079017-1679.

[3] B. Wallace, M. Dang, R. Rafailov, L. Zhou, A. Lou, S. Purushwalkam, S. Ermon, C. Xiong, S. Joty, and N. Naik. Diffusion model alignment using direct preference optimization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8228–8238, 2024. https://doi.org/10.1109/CVPR52733.2024.00786.

[4] P. Yu, M. Mardani, and A. Vahdat. Enhanced diffusion model guidance via multi-denoiser mixing. U.S. Patent Application Publication No. US20260087199A1, published March 26, 2026. https://patents.google.com/patent/US20260087199A1/en.

Chapter 5

Designing Instance-Level Sampling Schedules via REINFORCE with James–Stein Shrinkage



(a) Few-step sampling

(b) Text-image alignment

(c) Text rendering

(d) Counting

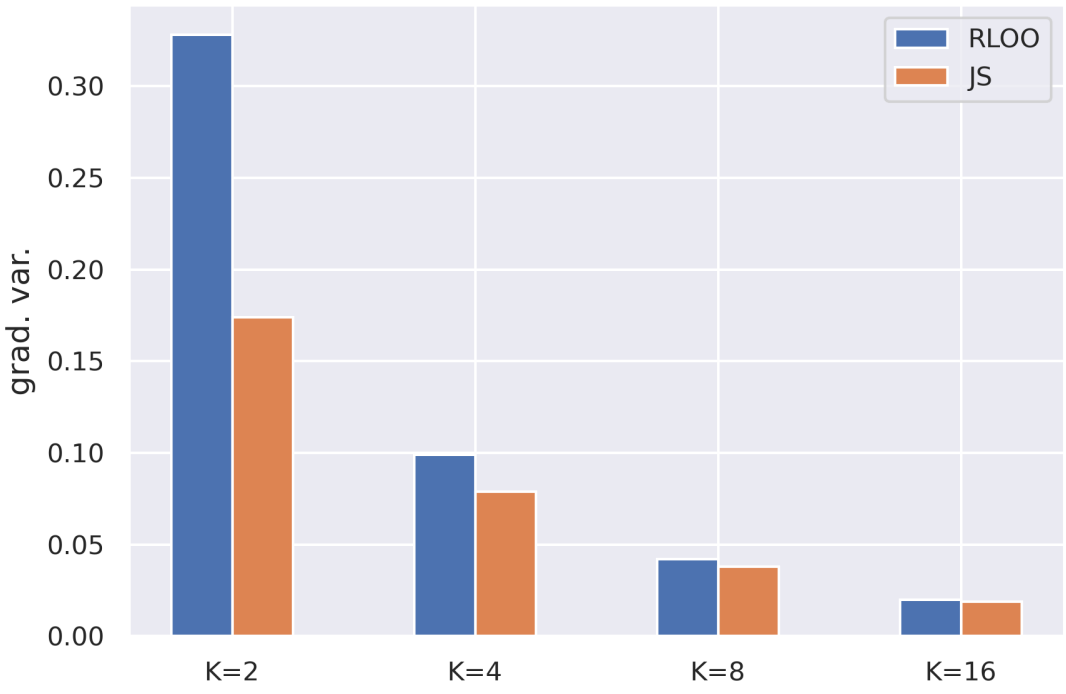
Figure 5.1: **Illustrations from the sampling-schedule study.** Paired images compare default (left) and learned instance-specific schedules (right), demonstrating few-step sampling, image-text alignment, text rendering, and object counting.

5.1 Introduction

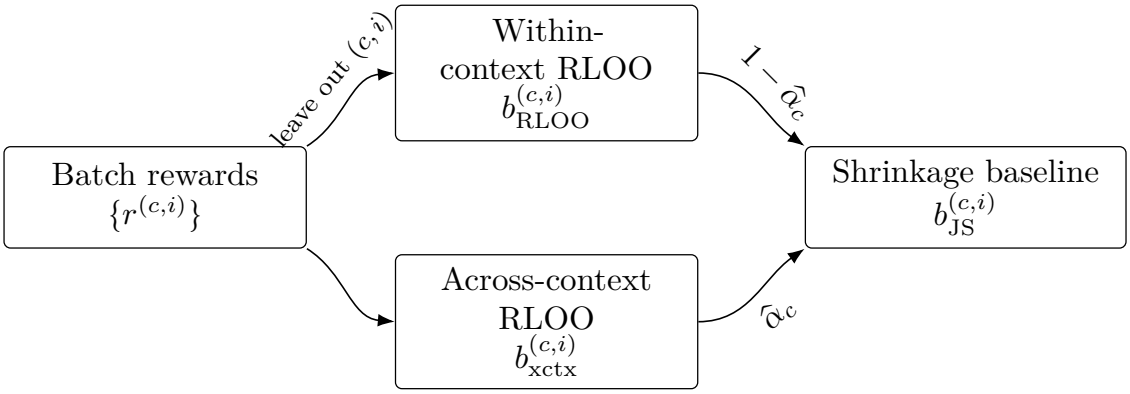
Building on the sampling-time guidance methods studied in Chapter 4, this chapter considers a complementary decision: when a frozen denoiser should be evaluated along its sampling trajectory. Diffusion and flow-matching models have achieved remarkable progress in prompt-based generation. By iteratively transforming an initial noise sample into a final image through multi-step denoising or probability flow trajectories, modern samplers demonstrate not only outstanding image fidelity but also impressive adherence to textual prompts [3, 10, 18]. At inference time, however, generation quality depends critically on the sampling schedule: how a limited step budget is allocated along the continuous sampling trajectory. Most production backbones adopt a single, globally fixed schedule for all inputs; while convenient, such a universal schedule is unlikely to be optimal across the diverse range of prompts encountered at test time.

Intuitively, different prompts demand different spatial or semantic details, and distinct noise seeds may induce varying initial conditions that affect image quality. Recent analyses of "golden noise" [31] further highlight such seed sensitivity, reinforcing the need for ***instance-level scheduling***. We therefore study schedule learning: adapting the timestep allocation per prompt and seed while keeping the total step budget fixed, to further exploit the generative potential of modern samplers. Prior work has begun to explore adjacent directions. One contemporaneous approach, TPDM [27], proposes the next timestep *autoregressively* from internal latents and current time. While TPDM primarily aims at early stopping and variable-length sampling to improve efficiency, our focus differs: we care about the Pareto frontier of the quality–vs.–steps tradeoff; we study reallocation of a fixed sampling budget to reach for the best possible generative quality per instance. A side-by-side comparison against a single-pass TPDM-style baseline under our settings further illustrates the distinctions (see Table 5.1).

Specifically, in this chapter we learn a non-autoregressive, single-pass policy that proposes the entire sampling schedule. One network evaluation replaces a sequence of timestep



(a) Simulated gradient estimation errors.



(b) James-Stein baseline workflow.

Figure 5.2: **James–Stein (JS) reward baseline.** a Simulation results from analytic policies illustrating lower measured gradient-estimator variance for the evaluated rollout counts K. The experimental setup is described in Section 5.7.3.1. b Diagram of JS baseline: combining b_{RLOO} and b_{xctx} into $b_{\text{JS}} = (1 - \widehat{\alpha}_c)b_{\text{RLOO}} + \widehat{\alpha}_c b_{\text{xctx}}$ (Eq. (5.7)).

proposals, although the dimensionality of the policy output still grows with the step budget. Training such a policy with terminal rewards requires reliable policy-gradient estimates. We study a James–Stein-inspired reward baseline that interpolates between rewards from repeated rollouts of one context and those pooled across contexts (Fig. 5.2). This shrinkage construction is computationally inexpensive and empirically reduces gradient-estimator variance in our evaluated settings. Its statistical motivation is an oracle Gaussian random-effects model; the guarantees for that oracle should not be conflated with guarantees for the data-dependent plug-in baseline used in training.

At larger sampling budgets, learned schedules can improve text–image alignment and fine-grained attributes such as text rendering and object counting. With as few as five steps, a learned schedule for the non-distilled Flux-Dev backbone attains a preference score close to that of Flux-schnell. These findings identify rescheduling as a complementary inference-time tool alongside guidance and distillation. The contributions of this chapter are:

- **Instance-level schedule learning.** We formulate prompt- and seed-conditioned scheduling as a single joint Dirichlet-policy action, keeping the pretrained image generator fixed.

- **Reward-baseline shrinkage.** We combine per-context leave-one-out and cross-context baselines, derive an oracle shrinkage result under an explicit random-effects model, and evaluate the empirical plug-in method without assuming a general unbiasedness or dominance guarantee.

- **Inference-time improvements.** We evaluate learned schedules over multiple backbones and step budgets, including few-step generation, text rendering, and fine-grained alignment, and identify where scheduling improves upon fixed schedules and where it does not.

5.2 Related Works

T2I samplers and inference-time schedules. Modern text-to-image diffusion and flow-matching models achieve high-fidelity synthesis by iteratively denoising with globally defined timestep schedules [3, 10, 12, 14, 18]. Considerable progress has been made on solver design and handcrafted timestep allocations (e.g., ODE solvers and higher-order variants) to improve efficiency and stability under limited step budgets [8, 16, 19, 21, 30]. However, these advances retain a *one-size-fits-all* schedule agnostic to instance-specific trajectory geometry. We instead learn a prompt-conditional schedule as a policy. Recent work such as TPDM [27] also explores RL-based optimization of diffusion sampler, but it autoregressively predicts next timesteps during sampling, aiming at early stopping and requiring $O(L)$ policy calls for inference; L denotes the inference steps. In contrast, our method issues a *single-pass* schedule proposal for the Pareto frontier of the quality–vs.–steps tradeoff, by leveraging a principled James–Stein control variate to facilitate training. Under the same compute budget, our design yields significantly stronger results than the previous counterpart.

Few-step sampling and distillation. Another active direction compresses long-run samplers into few-step generators through progressive or consistency-style distillation [20, 22, 26, 28, 29]. While highly effective, these methods retrain or distill the backbone often at additional training cost. We position our approach as an orthogonal yet compelling path: we do not retrain the backbone but reallocate an existing step budget via learned scheduling. Empirically, we find that simple rescheduling already unlocks strong few-step performance; pretrained samplers possess innate few-step capabilities even without distillation.

RL for generative alignment and variance reduction. Reinforcement learning has recently emerged as a versatile paradigm for aligning generative models with human or task-specific objectives. In text-to-image diffusion, this idea has been realized through policy-gradient and preference-based optimization frameworks such as DDPO, DPOK, and

Diffusion-DPO [2, 4, 23]. These methods fine-tune pretrained generators toward reward signals that capture aesthetic quality, faithfulness, or compositional alignment, often by treating the denoising latents throughout the generative process as a sequence of actions.

We adopt a complementary view: the *entire schedule* is treated as a high dimensional, open-loop and forward-looking plan proposed in one pass. This formulation exposes a key challenge common to sequence-level policy gradients—the high variance of gradient estimators. Classical variance-reduction techniques rely on hand-crafted heuristics or learned baselines [1, 9, 24]. For instance, RLOO [9] draws multiple samples per context and uses the average reward as a plug-in baseline. Although empirically effective, it remains brittle under heterogeneous reward scales, as we illustrate in Section 5.4. To address this, we relate the mean-reward baseline to the trace-variance-optimal control variate and introduce a James–Stein-inspired interpolation of contextual and pooled reward estimates. We give a precise oracle Bayes-risk result under Gaussian random effects and separately assess the variance of the practical plug-in gradient estimator. The observed gains suggest that shrinkage can improve reward-driven schedule learning without requiring a distribution-free dominance claim.

5.3 Method

We formulate instance-adaptive scheduling as a policy learning problem, derive a trace-variance-optimal scalar control variate, and construct a practical shrinkage baseline motivated by random-effects estimation and the classical James–Stein perspective [7]. We distinguish the oracle model, for which a precise Bayes-risk statement is available, from the estimated baseline used in experiments. The policy outputs a schedule in one pass without updating the pretrained diffusion backbone.

5.3.1 Problem Formulation

Let $\mathcal{S}_\phi$ denote a frozen pretrained sampler, and let $z = (\mathbf{x}_T, \mathbf{c})$ be a context containing an initial noise sample and a text condition. For a budget of L denoiser evaluations, we use a descending grid

$$1 = t_0 > t_1 > \cdots > t_L > t_{L+1} = 0.$$

The L scheduled evaluations traverse the first L grid intervals; the residual interval $(t_L, 0)$ is represented by a final, unevaluated stopping margin in the policy parameterization. Thus, L denotes the number of denoiser evaluations, not the number of simplex components. The precise terminal-state convention must match the implementation of the chosen solver.

For each context z, the stochastic schedule policy $\pi_\theta(\tau \mid z)$ proposes a grid $\tau = (t_1, \ldots, t_L)$. Executing the frozen sampler under that grid produces an image $\mathbf{x}_0(\tau; z)$ and a reward $r(\mathbf{x}_0(\tau; z); \mathbf{c})$. The training objective averages over contexts and schedules:

$$J(\theta) = \mathbb{E}_{z \sim p_{\text{ctx}}} \mathbb{E}_{\tau \sim \pi_\theta(\cdot|z)} \big[r(\mathbf{x}_0(\tau; z); \mathbf{c}) \big]. \tag{5.1}$$

Our goal is to optimize the policy parameters θ to maximize $J(\theta)$, enabling the model to learn instance-specific schedules that adapt to diverse text and noise conditions without modifying the frozen sampler backbone.

5.3.2 Policy Gradient with Baselines

Score-function estimator. To optimize the expected reward in Eq. (5.1), we apply the standard REINFORCE or score-function estimator:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta} \big[(r(\tau) - b) \, \nabla_\theta \log \pi_\theta(\tau \mid \mathbf{x}_T, \mathbf{c}) \big], \tag{5.2}$$

where $b = b(z)$ is independent of the sampled action τ conditional on the context z. It may depend on θ when treated as a stopped-gradient control variate. This conditional

action-independence, rather than parameter-independence, makes the baseline term have zero expectation. For a sample-estimated baseline, excluding information from the current action is a sufficient condition for this property.

Variance–optimal baseline. For fixed context and a scalar action-independent baseline, minimizing the trace of the gradient-estimator covariance yields the following familiar expression.

Proposition 5.3.1 (Variance–optimal baseline)**.** *Write* $g(\tau) = \nabla_\theta \log \pi_\theta(\tau)$*, and assume* $0 < \mathbb{E}[\|g(\tau)\|^2] < \infty$*. Among constant scalar baselines, the minimizer of the covariance trace* $\operatorname{tr} \operatorname{Cov}[(r - b)g(\tau)]$ *is*

$$b^* = \frac{\mathbb{E}[r(\tau)\, \|\nabla_\theta \log \pi_\theta(\tau)\|^2]}{\mathbb{E}[\|\nabla_\theta \log \pi_\theta(\tau)\|^2]}.$$

Sketch. Differentiate the variance w.r.t. b *and set the derivative to zero. A proof is given in Section 5.7.1.*

In practice, computing b^* exactly is infeasible for high-dimensional policies. If the squared score norm is only weakly correlated with reward within a context, the weighted mean approximately agrees with the contextual mean:

$$b^* \approx \mathbb{E}_{\tau \sim \pi_\theta}[r(\tau) \mid \mathbf{x}_T, \mathbf{c}], \tag{5.3}$$

which corresponds to the *contextual mean reward*. This observation motivates empirical estimators based on the contextual mean of sampled rewards.

Per-context RLOO baseline. A common design choice for instantiating the baseline b in (5.2) is the *per-context* RLOO estimator [1, 9], which can be viewed as a simple Monte Carlo approximation to the variance–optimal baseline b^*. In our problem setup, we collect a mini-batch of B independent contexts $\{(\mathbf{x}_T^{(c)}, \mathbf{c}^{(c)})\}_{c=1}^{B}$. For each context c, we draw K_c schedule samples $\{\tau^{(c,i)}\}_{i=1}^{K_c}$ with corresponding rewards $\{r^{(c,i)}\}_{i=1}^{K_c}$. The RLOO baseline estimates the

contextual mean reward as

$$b_{\mathrm{RLOO}}^{(c,i)} = \frac{1}{K_c - 1} \sum_{j=1, j\neq i}^{K_c} r^{(c,j)}. \tag{5.4}$$

The associated REINFORCE estimator is unbiased because $b_{\mathrm{RLOO}}^{(c,i)}$ uses no information from the current rollout i, conditional on the context. RLOO estimates the contextual mean reward from the other independent rollouts; it approximates the variance-optimal baseline when the score-norm weighting has little effect. However, when the number of rollouts per context K_c is small (e.g., 2–4) or when reward scales vary substantially across prompts, b_{RLOO} becomes a noisy estimator of the contextual mean (see Section 5.4 and Fig. 5.2).

Cross-context baseline. A natural extension is to exploit reward information *across* contexts by forming a shared baseline for the entire mini-batch:

$$b_{\mathrm{xctx}}^{(c,i)} = \frac{1}{(\sum_{c'} K_{c'}) - 1} \left(\sum_{c'=1}^{B} \sum_{j=1}^{K_{c'}} r^{(c',j)} - r^{(c,i)} \right), \tag{5.5}$$

which averages rewards over all rollouts except the current one. Like b_{RLOO}, this leave-one-out construction preserves the unbiasedness of the REINFORCE estimator, but now draws upon the entire batch to estimate the mean reward. Such pooling can substantially reduce estimator variance when reward magnitudes are comparable across prompts, effectively anchoring each contextual baseline to a global reference. However, this pooled baseline ignores systematic reward differences between prompts. We next consider shrinkage between the two estimators; the Gaussian oracle analysis motivates this interpolation, but does not automatically guarantee improvement for its data-dependent implementation.

5.3.3 James–Stein Shrinkage Reward Baseline

We now introduce a principled shrinkage formulation that unifies the per-context baseline b_{RLOO} and the cross-context baseline b_{xctx}. The key idea is to treat contextual rewards as random effects drawn from a shared distribution, and to shrink each context's baseline toward

the batch-level anchor in a data-dependent manner.

Random effects view. For statistical motivation, suppose that rewards satisfy a homoscedastic two-level Gaussian random-effects model with conditionally independent residuals:

$$\begin{aligned} r^{(c,i)} &= \mu_c + \varepsilon^{(c,i)}, \quad \varepsilon^{(c,i)} \sim \mathcal{N}(0, \sigma^2), \\ \mu_c &= \mu_0 + \xi^{(c)}, \qquad \xi^{(c)} \sim \mathcal{N}(0, \delta^2), \end{aligned} \tag{5.6}$$

where σ^2 represents within-context reward variance and δ^2 quantifies heterogeneity across contexts (e.g., varying prompt difficulty). For each context c, we denote by $\bar{r}_c = \frac{1}{K_c} \sum_i r^{(c,i)}$ its empirical mean reward. Under the model in (5.6), the posterior mean of the latent mean reward μ_c is

$$\mathbb{E}[\mu_c \mid \{r^{(c,i)}\}] = (1 - \alpha_c^*)\, \bar{r}_c + \alpha_c^*\, \mu_0, \quad \alpha_c^* = \frac{\sigma^2/K_c}{\sigma^2/K_c + \delta^2},$$

which adaptively shrinks $\bar{r}_c$ toward the global mean μ_0. The shrinkage strength α_c^* increases when the number of rollouts per context K_c is small or when contexts are relatively homogeneous (small δ^2).

Empirical James–Stein-inspired baseline. The two leave-one-out baselines provide complementary estimates of a contextual reward level. We interpolate between them with a plug-in shrinkage weight:

$$b_{\mathrm{JS}}^{(c,i)} = (1 - \widehat{\alpha}_c)\, b_{\mathrm{RLOO}}^{(c,i)} + \widehat{\alpha}_c\, b_{\mathrm{xctx}}^{(c,i)}, \tag{5.7}$$

with

$$\widehat{\alpha}_c = \frac{\widehat{\sigma}^2/(K_c - 1)}{\widehat{\sigma}^2/(K_c - 1) + \widehat{\delta}^2}. \tag{5.8}$$

Here $\widehat{\sigma}^2$ and $\widehat{\delta}^2$ are batch-estimated within- and between-context dispersion parameters. If both variance components vanish, the two baselines agree, so the coefficient can be set to zero without changing the baseline. Under small contextual noise relative to between-context variation, the interpolation approaches RLOO; when contextual means appear similar, it

favors the pooled estimate.

Estimating variance components. For reference, with $B \geqslant 2$ under the homoscedastic random-effects model (5.6), the standard pooled within-context moment estimator and its between-context counterpart are

$$\widehat{\sigma}^2_{\mathrm{MoM}} = \frac{\sum_{c=1}^{B}\sum_{i=1}^{K_c}(r^{(c,i)} - \bar{r}_c)^2}{\sum_{c=1}^{B}(K_c - 1)}, \tag{5.9}$$

$$S^2_{\mathrm{between}} = \frac{1}{B-1}\sum_{c=1}^{B}(\bar{r}_c - \bar{r})^2, \qquad \bar{r} = \frac{1}{B}\sum_{c=1}^{B}\bar{r}_c, \tag{5.10}$$

$$\widehat{\delta}^2_{\mathrm{MoM}} = \max\left\{0, S^2_{\mathrm{between}} - \frac{\widehat{\sigma}^2_{\mathrm{MoM}}}{B}\sum_{c=1}^{B}\frac{1}{K_c}\right\}. \tag{5.11}$$

Before truncation at zero, the between-context correction accounts for the variance of context means, including unequal K_c. This calibrated method-of-moments specification provides a statistical reference for the shrinkage coefficient. An alternative proxy based on $\sum_{c,i}(r^{(c,i)} - b^{(c,i)}_{\mathrm{RLOO}})^2/\sum_c(K_c - 1)$ would overestimate σ^2 by a factor $(K/(K-1))^2$ when $K_c = K$. As the original reported experiments depend on the estimator used in code, such implementation choices must be distinguished from this statistical reference.

Action-dependence and the plug-in limitation. Although $b^{(c,i)}_{\mathrm{RLOO}}$ and $b^{(c,i)}_{\mathrm{xctx}}$ exclude $r^{(c,i)}$, the common batch estimate $\widehat{\alpha}_c$ in (5.8) can depend on that reward. Consequently, the full plug-in baseline need not be independent of the current action, and stopping its computational gradient does not, by itself, establish an unbiased REINFORCE estimator. One unbiased alternative is to compute a separate $\widehat{\alpha}^{(-c,i)}_c$ using only rewards other than $r^{(c,i)}$; this cross-fitted variant is mathematically distinct and should not be assumed to generate the same empirical results without rerunning the experiments. We treat the batch-estimated baseline as an empirical variance-reduction procedure, rather than claiming a universal unbiasedness theorem for it.

Oracle posterior shrinkage. An exact statement is available for the idealized model and an independently specified population anchor.

Theorem 5.3.2 (Oracle Bayes-risk improvement)**.** *Assume* (5.6)*, with* $\sigma^2 > 0$*, finite* $\delta^2 \geqslant 0$*, known* μ_0*, and* $K_c \geqslant 2$*. For rollout* (c, i)*, let* $m_c^{(-i)} = b_{\mathrm{RLOO}}^{(c,i)}$ *and* $v_c = \sigma^2/(K_c - 1)$*. Define*

$$\alpha_c^\star = \frac{v_c}{v_c + \delta^2}, \qquad b_{\mathrm{oracle}}^{(c,i)} = (1 - \alpha_c^\star) m_c^{(-i)} + \alpha_c^\star \mu_0.$$

Then $b_{\mathrm{oracle}}^{(c,i)} = \mathbb{E}[\mu_c \mid \{r^{(c,j)} : j \neq i\}]$ *is the Bayes estimator under squared loss, and its integrated mean-squared error is*

$$\mathbb{E}[(b_{\mathrm{oracle}}^{(c,i)} - \mu_c)^2] = \frac{v_c \delta^2}{v_c + \delta^2} < v_c = \mathbb{E}[(b_{\mathrm{RLOO}}^{(c,i)} - \mu_c)^2].$$

Because the oracle baseline does not use the current rollout, it also yields an unbiased score-function gradient estimator conditional on the current context.

Scope of the result. Theorem 5.3.2 concerns an oracle with known hyperparameters and population anchor; it does not assert that the batch-estimated b_{JS} in (5.7) is a posterior mean, dominates RLOO in gradient variance, or is exactly unbiased. In particular, $b_{\mathrm{xctx}}^{(c,i)}$ contains other rewards from context c and is correlated with RLOO. The classical $B \geqslant 3$ James–Stein domination theorem [7] is related motivation, not an immediate theorem for this plug-in baseline. We evaluate its practical behavior empirically in Sections 5.4 and 5.7.3.1.

5.3.4 Single-Pass Scheduler as Dirichlet Policy

Dirichlet parameterization. We treat the entire denoising schedule as a *single, joint action* rather than a sequence of stepwise decisions. Concretely, we define $\tau \sim \mathrm{Dirichlet}(\boldsymbol{\alpha}_\theta(\mathbf{x}_T, \mathbf{c}))$, where the policy network outputs nonnegative parameters $\boldsymbol{\alpha}_\theta \in \mathbb{R}_{>0}^{L+1}$ conditioned on the initial noise $\mathbf{x}_T$ and text prompt $\mathbf{c}$. Each sampled component τ_t represents a positive interval almost surely, and the simplex constraint $\sum_{t=1}^{L+1} \tau_t = 1$ ensures that these intervals partition

the unit interval $[0, 1]$. We convert them into an ordered sampling schedule by cumulative summation:

$$\tilde{t}_\ell = \sum_{j=1}^{\ell} \tau_j, \quad \text{and} \quad t_\ell = 1 - \tilde{t}_\ell,$$

yielding a descending sequence $1 = t_0 > t_1 > \cdots > t_L > t_{L+1} = 0$. The residual interval τ_{L+1} is a learnable *stopping margin*: under the stated convention, no additional denoiser evaluation is performed over $[t_L, 0]$. Hence $L+1$ simplex components parameterize L denoiser evaluations. Any implementation that instead evaluates the final interval must adjust the stated budget accordingly.

This formulation naturally enforces simplex constraints, defines a smooth distribution over valid schedules, and avoids the need for per-step discrete decisions. Unlike a stepwise policy, the Dirichlet scheduler uses one policy-network forward pass regardless of L; its output dimensionality, likelihood evaluation, and the subsequent sampler still scale with the step budget.

Policy network. We implement the policy $\boldsymbol{\alpha}_\theta(\mathbf{x}_T, \mathbf{c})$ using a lightweight (approximately 20M parameters, less than 1% of the backbone size for the larger models studied) encoder–decoder architecture. The initial noise $\mathbf{x}_T$ is processed by multiscale convolutional blocks, whose spatial features are fused with the pretrained text embedding $\mathbf{c}$ via cross-attention layers. The fused representation is then projected through MLP layers to $(L+1)$ channels, corresponding to the Dirichlet parameters that define interval allocations along the sampling timeline. A final `softplus` activation followed by a small constant offset (10^{-3}) ensures strictly positive outputs, guaranteeing valid Dirichlet parameters and numerically stable gradients. The extra $(L+1)$-th channel naturally learns the residual "stopping" interval in the Dirichlet parameterization. Network architectures and compute costs are reported in Section 5.7.2.

Algorithm 2 Single-pass schedule-policy optimization with a batch-estimated shrinkage baseline

Inputs: frozen sampler $\mathcal{S}$, policy π_θ, reward R, optimizer Opt
repeat
  Sample contexts $\{c\}_1^B$; set $g \leftarrow 0$
  for $c = 1$ **to** B **do**
    for $i = 1$ **to** K_c **do**
      $\tau^{(c,i)} \sim \pi_\theta(\cdot \mid c)$
      $r^{(c,i)} \leftarrow R(\mathcal{S}(c, \tau^{(c,i)}), c)$
    end for
  end for
  Estimate the batch variance components and $\{\widehat{\alpha}_c\}_{c=1}^B$
  for $c = 1$ **to** B **do**
    for $i = 1$ **to** K_c **do**
      Compute $b_{\mathrm{RLOO}}^{(c,i)}$, $b_{\mathrm{xctx}}^{(c,i)}$, and $b_{\mathrm{JS}}^{(c,i)}$
      $g \leftarrow g + \frac{1}{BK_c}(r^{(c,i)} - \mathrm{sg}[b_{\mathrm{JS}}^{(c,i)}])\nabla_\theta \log \pi_\theta(\tau^{(c,i)} \mid c)$
    end for
  end for
  $\theta \leftarrow \mathsf{Opt.step}(\theta, g)$
until convergence

Training algorithm. At each iteration, we sample a mini-batch of B contexts and, for each context c, draw K_c schedules from the scheduler policy. After executing all sampled schedules through $\mathcal{S}$ and collecting the rewards $r^{(c,i)}$, we estimate the batch-level shrinkage parameters before forming the baselines $b_{\mathrm{JS}}^{(c,i)}$ (Eq. (5.7)). We stop gradients through the baseline coefficient; this is a computational convention and does not remove the possible statistical dependence described above. The policy is then updated by averaging the REINFORCE gradients with JS baselines across all samples. Of note, we (i) *exclude* KL constraints or other regularizers, and (ii) initialize the scheduler policies *from scratch*, in order to highlight the effectiveness of our framework and keep the experimental setting clean and interpretable. Overall procedure is summarized in Algorithm 2.

5.4 Experiments

5.4.1 General Text-to-Image Experiments

Dataset and protocol. We first conduct extensive experiments on the Human Preference Dataset (HPD) v2 text-to-image benchmark [25] to assess our method. We use the official train split $\sim 100K$ prompts to optimize the schedule policy and the held-out 3200 test split prompts for evaluation. We use the Human Preference Score (HPS) v2 model [25] as our reward for these experiments. For each backbone and sampling budget $L \in \{5, 10, 20, 40, 80\}$, we generate images for the same set of prompts and fixed noise seeds to ensure strict comparability across methods. We study four representative pretrained samplers: SD-XL [17], SD-3.5M/L [3], and Flux-Dev [10], covering both UNet- and MMDiT-based backbones. For all models we consistently use a rollout number of 2. Due to memory limit, we use a batch size of 32 for smaller models like SD-XL or SD-3.5M, and a batch size of 16 for larger ones like SD-3.5L and Flux-Dev. Further details are given in Section 5.7.3.

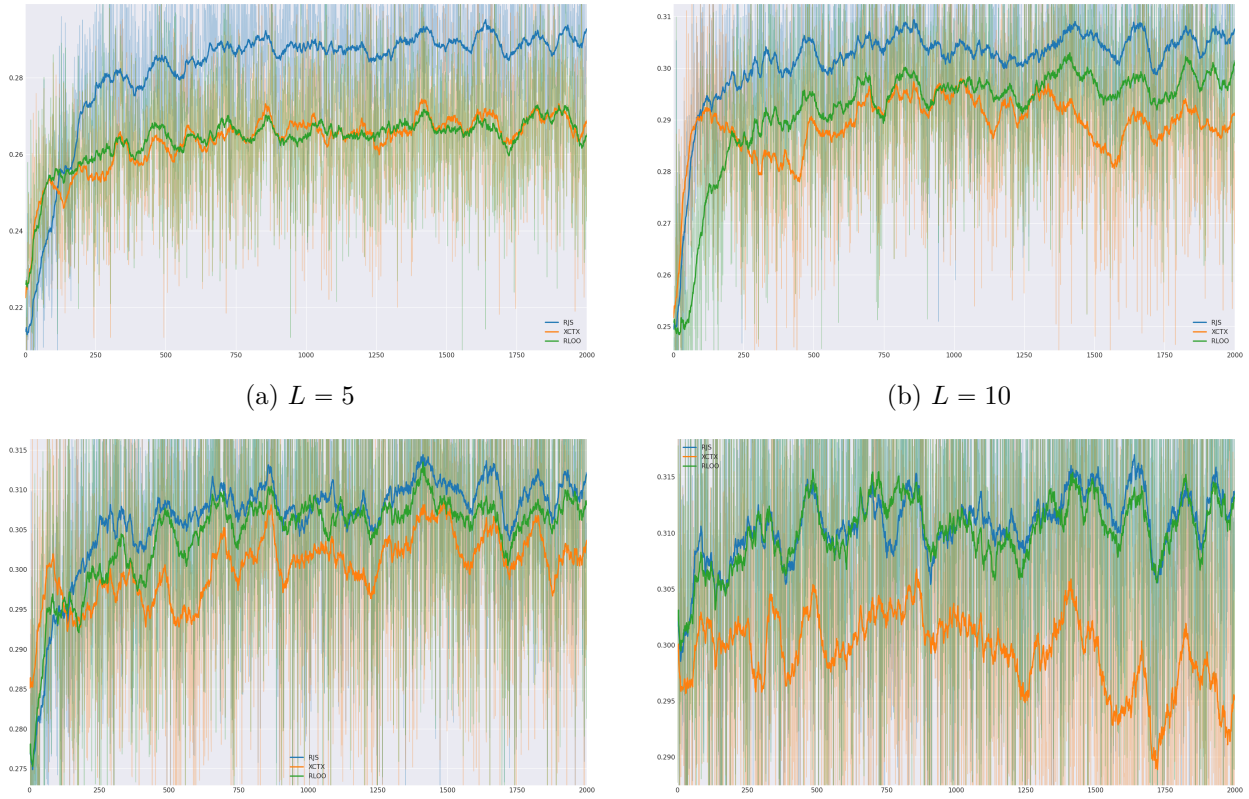
We compare scheduling strategies under a unified training setup: (1) the *Default* fixed schedule provided by each backbone, schedules optimized with the (2) *Cross-Context RLOO* baseline, (3) *RLOO* baseline, and (4) *Ours (JS)* shrinkage baseline. For Flux we additionally include a *TPDM-style PPO* variant using the same architecture for a direct comparison. All other configurations of TPDM follow its default setups. The implementation and differences from the original TPDM setting are discussed in Section 5.7.3. All methods share identical optimization settings and data pipelines, ensuring that performance differences arise solely from the choice of reward baseline and policy learning formulation.

Table 5.1: **HPSv2 scores on HPDv2 held-out prompts.** Scores by number of inference steps (L). Rows per backbone: Default, Cross-Context RLOO (XCTX), RLOO, TPDM-style PPO (Flux only), and Ours (JS). Best per column in **bold**; second-best underlined.

Backbone	Methods / Steps (L)	5	10	20	40	80
SD-XL	Default	18.25	25.47	27.69	28.52	28.55
	XCTX	24.03	26.44	27.30	27.59	28.19
	RLOO	24.10	26.59	27.85	28.42	28.50
	Ours (JS)	**24.22**	**26.89**	**27.98**	**28.53**	**28.66**
SD3.5-M	Default	21.59	26.31	28.66	**29.61**	**29.91**
	XCTX	23.96	25.32	27.53	26.70	28.14
	RLOO	24.05	27.20	28.52	29.03	29.36
	Ours (JS)	**24.31**	**27.31**	**28.87**	29.47	29.62
SD3.5-L	Default	24.24	28.04	29.85	**30.43**	30.61
	XCTX	24.92	27.39	28.44	28.03	29.30
	RLOO	25.55	28.61	29.37	29.75	30.27
	Ours (JS)	**26.28**	**28.88**	**29.98**	30.41	**30.64**
Flux-Dev	Default	23.73	28.06	29.88	30.84	31.04
	TPDM PPO	15.73	16.24	17.05	17.98	18.65
	XCTX	26.92	29.39	30.09	29.94	30.88
	RLOO	26.48	30.41	30.77	30.92	31.10
	Ours (JS)	**29.21**	**30.86**	**31.12**	**31.23**	**31.36**



Figure 5.3: **Rescheduling improves general T2I alignment.** Head-to-head comparisons between images generated with default schedules (upper) and our learned schedules (lower) from Flux-Dev with 40 steps. Figures henceforth follow the same format.



(a) $L = 5$ (b) $L = 10$

(c) $L = 20$ (d) $L = 40$

Figure 5.4: **Representative training curves (Flux-Dev) for different baselines and sampling budgets L.** The horizontal axis denotes training iterations and the vertical axis the aggregate reward. The plotted JS baseline yields stronger training rewards than the compared XCTX and RLOO baselines in these settings; the effect is most pronounced at small step budgets.

Findings. Among the learned scheduling methods in Table 5.1, the JS-based policy attains the highest HPSv2 score for each reported backbone and budget. Relative to each backbone's default fixed schedule, however, it does not win in every setting: SD3.5-M at 40 and 80 steps and SD3.5-L at 40 steps favor the default. Improvements over the fixed schedule are generally larger at small budgets, particularly for Flux-Dev. The table supports a comparison of resulting preference scores; the controlled estimator-variance results in Section 5.7.3.1 address variance separately.

Few-step sampling. The benefits of rescheduling are particularly striking under severe step constraints (Tables 5.1 and 5.2). With only $L = 5$ steps, *Ours (JS)* on Flux-Dev nearly matches the performance of the deliberately distilled Flux-schnell sampler, despite operating on the non-distilled backbone. This highlights that a learned, prompt-adaptive schedule can significantly improve the *effective* sampling efficiency, revealing that pretrained samplers possess meaningful few-step capabilities even without distillation. At five steps the JS-based policy outperforms the reported competing learned schedulers on each tested backbone; this supports the practical value of dynamic allocation in the low-budget regime without establishing a universal guarantee.

5.4.2 Text Rendering and Fine-Grained Alignment

Dataset and protocol. Building on the general text-to-image results in Section 5.4.1, we further assess the fine-grained alignment capability with our method. When the sampling budget is sufficiently large, the gap between the default schedule and our learned schedules may appear to narrow when measured by global preference scores; however, we find that our schedules yield substantially stronger fine-grained alignment capabilities. We focus on two representative scenarios: (i) prompt-specified text rendering, and (ii) object-centric generation tasks following GenEval [5].

For text rendering, we follow [13] and use its 20K text-centric training prompts (alphanu-

Table 5.2: Flux-Dev 5-step sampling on HPDv2.

Default	TPDM PPO	Cr. RLOO	RLOO	RJS	Schnell
23.73	15.73	26.92	26.48	29.21	**29.42**



Figure 5.5: **Rescheduling improves few-step sampling**. Comparisons between images generated with default schedules (upper) and our learned schedules (lower) from Flux-Dev, with 5 steps.

meric strings, short phrases, and proper nouns) and 500 held-out testing prompts. We use again the HPSv2 model as our reward. For GenEval tasks, we use the official script to generate 50K training prompts and 1816 held-out and deduplicated testing prompts. We use a rule-based reward as in [13] as our reward. We focus on two representative backbones: SD-3.5M and Flux-Dev, and compare the backbone's *Default* fixed schedule with **Ours (JS)** learned schedules. We set the sampling steps to 40 for all experiments. For text rendering, we report **HPSv2** (↑) for overall perceptual preference and three character-level text-specific OCR metrics: **OCR-A** (↑, accuracy), **OCR-P** (↑, precision) and **OCR-R** (↑, recall). Human preference (HPSv2) complements OCR, which can be brittle on stylized glyphs. For GenEval

Table 5.3: **Text rendering at a fixed step budget ($L = 40$).** HPSv2 (↑), OCR-A (↑), OCR-P (↑), OCR-R (↑). Best per backbone in **bold**.

SD3.5-M	**HPSv2 (↑)**	**OCR-A (↑)**	**OCR-P (↑)**	**OCR-R (↑)**
Default	**27.23**	19.08	53.69	53.34
Ours (JS)	27.07	**20.04**	**54.93**	**58.19**
Flux-Dev	**HPSv2 (↑)**	**OCR-A (↑)**	**OCR-P (↑)**	**OCR-R (↑)**
Default	28.70	20.06	52.63	49.77
Ours (JS)	**28.91**	**21.30**	**54.05**	**58.58**

Table 5.4: **GenEval** results by backbone and schedule. Metrics are fractions in $[0, 1]$. c.attrs. = Color Attribution.

Model	**Single**	**Two**	**Count**	**Colors**	**Pos.**	**C.Attrs.**	**Overall**
SD3.5-M	0.99	0.81	0.49	0.79	0.19	0.46	0.62
SD3.5-M + JS	**0.99**	**0.87**	**0.65**	**0.86**	**0.22**	**0.46**	**0.68**
Flux-Dev	0.98	0.83	0.58	0.79	0.22	0.45	0.64
Flux-Dev + JS	**0.99**	**0.88**	**0.77**	**0.86**	**0.24**	**0.46**	**0.70**

tasks, we report the corresponding scores on the held-out testing prompts with the official evaluation pipeline. Additional protocol details are given in Section 5.7.3.

Findings. For both SD3.5-M and Flux-Dev, the learned schedule improves all reported OCR metrics and the evaluated GenEval attributes (Tables 5.3 and 5.4). HPSv2 on the text-rendering subset increases for Flux-Dev (28.70 to 28.91) but slightly decreases for SD3.5-M (27.23 to 27.07); thus the gains in text accuracy need not coincide with a gain in the global preference score. The gains are visible in Figs. 5.6 and 5.7: adaptive timestep reallocation improves stroke completion and reduces character dropout in text rendering, and similarly enhances object-centric correctness in GenEval scenarios. Together, these findings indicate that variance-aware, prompt-adaptive scheduling yields meaningful fine-grained alignment benefits—even when the sampling budget is already sufficient.

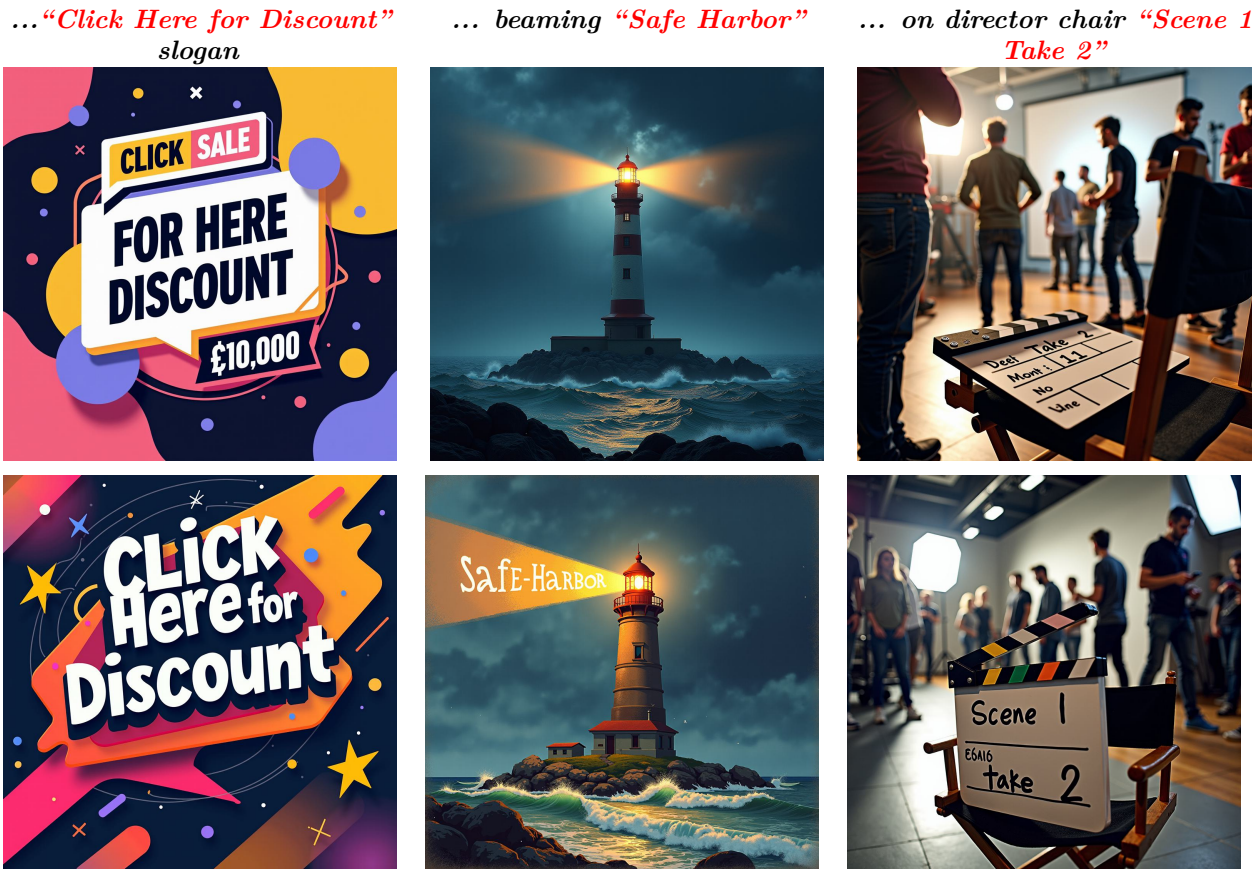


Figure 5.6: **Rescheduling improves text rendering**. We present comparisons between images generated with default schedules (upper) and our learned schedules (lower) from Flux-Dev.

5.5 Conclusion

We developed instance-level rescheduling for pretrained text-to-image samplers, using a single-pass policy to adapt the evaluation grid to each prompt and noise seed. A James–Stein-inspired interpolation between within-context and pooled reward baselines improves the measured policy-gradient variance in our synthetic tests and yields competitive learned schedules on the evaluated models. The accompanying oracle analysis explains the statistical appeal of shrinkage but does not imply a distribution-free variance or unbiasedness guarantee for the practical batch-estimated baseline. The experiments show clear improvements in several low-step and fine-grained alignment settings, alongside exceptions where default fixed schedules remain stronger.



Figure 5.7: **Rescheduling improves fine grained alignment**. Comparisons between images generated with default schedules (upper) and our learned schedules (lower) from Flux-Dev.

Limitations and outlook. We used simple scheduler architectures and limited reward types (HPSv2, Rule-based GenEval Rewards), leaving richer policies and broader objectives for future work. Promising extensions include: (i) adaptive early stopping for dynamic step budgets; (ii) multi-objective or process-based rewards (e.g., compositionality, aesthetics, safety); (iii) extending JS-based shrinkage to RLHF and other long-horizon policy settings and applications to video, flow, and 3D generative pipelines. These directions can further enhance the stability, efficiency, and controllability of post-training generative systems.

5.6 Acknowledgments for This Research Chapter

This chapter is based on the coauthored paper by Peiyu Yu, Suraj Kothawade, Sirui Xie, Ying Nian Wu, and Hongliang Fei, "Designing Instance-Level Sampling Schedules via REINFORCE with James–Stein Shrinkage," published in the *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026, pp. 36040–36050. The presentation in this dissertation adapts the published work and clarifies the assumptions underlying the shrinkage-baseline analysis.

For the research presented in this chapter, Peiyu Yu conceived the method, designed and carried out the experiments, and wrote the manuscript. Suraj Kothawade, Sirui Xie, Ying Nian Wu, and Hongliang Fei contributed to discussions of the proposed method.

5.7 Additional Analyses and Experimental Details

5.7.1 Proofs

5.7.1.1 Proof of Proposition 5.3.1

Proof. First, for a fixed context, let the baseline b be independent of the sampled action (and treated as constant under the score-function expectation). We abbreviate the policy as $\pi_\theta(\tau) \equiv \pi_\theta(\tau \mid \mathbf{x}_T, \mathbf{c})$,

$$g(\tau) \;=\; \nabla_\theta \log \pi_\theta(\tau), \qquad w(\tau) \;=\; \|g(\tau)\|^2 ,$$

and let $r(\tau)$ denote the scalar reward. The single–sample REINFORCE estimator with a scalar baseline b is

$$\widehat{G}(b) \;=\; (r(\tau) - b)\, g(\tau). \tag{5.12}$$

By swapping integral and gradient operators

$$\mathbb{E}_{\tau\sim\pi_\theta}[g(\tau)] = \mathbb{E}_{\tau\sim\pi_\theta}[\nabla_\theta \log \pi_\theta(\tau)] = \nabla_\theta \mathbb{E}_{\tau\sim\pi_\theta}[1] = 0. \tag{5.13}$$

It follows that the estimator is unbiased for any action-independent baseline b:

$$\mathbb{E}[\widehat{G}(b)] = \mathbb{E}[(r-b)g] = \mathbb{E}[rg] - b\,\mathbb{E}[g] = \mathbb{E}[rg]. \tag{5.14}$$

Next, we study the variances of the estimator. The covariance of $\widehat{G}(b)$ is

$$\mathrm{Var}[\widehat{G}(b)] = \mathbb{E}[(r-b)^2 g g^\top] - \mathbb{E}[(r-b)g]\,\mathbb{E}[(r-b)g]^\top.$$

Since $\mathbb{E}[(r-b)g] = \mathbb{E}[rg]$ does not depend on b, minimizing the total scalar variance (trace)

is equivalent to minimizing

$$V(b) \;=\; \mathbb{E}\big[(r-b)^2\, w(\tau)\big] \;-\; \text{const}, \;\; \text{Recall } w(\tau) = \|g(\tau)\|^2 .$$

Provided $\mathbb{E}[w] > 0$, the function $V(b)$ is differentiable and strictly convex in b. Differentiating and setting the derivative to zero yields

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}b}V(b) &= \frac{\mathrm{d}}{\mathrm{d}b}\,\mathbb{E}\big[(r-b)^2 w\big] = \mathbb{E}\big[2(b-r)w\big] \\ &= 2\Big(b\,\mathbb{E}[w] - \mathbb{E}[rw]\Big) = 0,\end{aligned}$$

which gives the unique minimizer

$$b^* = \frac{\mathbb{E}[r(\tau)\, w(\tau)]}{\mathbb{E}[w(\tau)]} = \frac{\mathbb{E}\big[r(\tau)\; \|\nabla_\theta \log \pi_\theta(\tau)\|^2\big]}{\mathbb{E}\big[\,\|\nabla_\theta \log \pi_\theta(\tau)\|^2\big]}. \tag{5.15}$$

Thus b^* minimizes the trace of $\mathrm{Var}[\widehat{G}(b)]$. □

Remarks. (i) Because b is a single scalar, the expression optimizes the trace of the covariance. Each individual gradient coordinate generally has a distinct variance-optimal scalar baseline. (ii) Our derivation shares the spirit of state-dependent baselines derived for GPOMDP in Greensmith et al. [6], which optimize a different variance criterion conditioned on Markov states. (iii) The contextual mean reward approximates Eq. (5.15) when $r(\tau)$ and the squared score norm are weakly correlated conditional on the context. No near-deterministic-policy assumption is required.

5.7.1.2 Proof of Theorem 5.3.2

Proof. Fix a context c and leave out the current rollout i. Write $m = m_c^{(-i)}$, $v = v_c = \sigma^2/(K_c - 1)$, and $\mu = \mu_c$. Under the stated Gaussian random-effects model,

$$\mu \sim \mathcal{N}(\mu_0, \delta^2), \qquad m \mid \mu \sim \mathcal{N}(\mu, v).$$

For $\delta^2 > 0$, conjugacy gives posterior precision $1/\delta^2 + 1/v$ and posterior mean

$$\mathbb{E}[\mu \mid m] = \frac{\delta^2}{v + \delta^2}\, m + \frac{v}{v + \delta^2}\, \mu_0 \tag{5.16}$$

$$= (1 - \alpha_c^\star)m + \alpha_c^\star \mu_0. \tag{5.17}$$

The posterior variance is the constant

$$\mathrm{Var}(\mu \mid m) = \frac{v\delta^2}{v + \delta^2}.$$

Since conditional expectation uniquely minimizes conditional squared loss, the integrated Bayes risk is this posterior variance. In contrast, m is conditionally unbiased for μ with integrated risk $\mathbb{E}[(m - \mu)^2] = v$. Their difference is

$$v - \frac{v\delta^2}{v + \delta^2} = \frac{v^2}{v + \delta^2} > 0,$$

for $v > 0$ and finite δ^2. If $\delta^2 = 0$, then $\mu = \mu_0$ almost surely and the oracle baseline has zero risk, so the claim still holds.

Finally, m is computed from rollouts $j \neq i$, and μ_0, σ^2, and δ^2 are fixed population quantities. Conditional on the context, the oracle baseline is therefore independent of the current sampled action. The score-function identity yields

$$\mathbb{E}[b_{\text{oracle}}^{(c,i)} \nabla_\theta \log \pi_\theta(\tau^{(c,i)} \mid c) \mid c] = 0,$$

under standard integrability and conditional independence assumptions. This conclusion does not extend automatically to the full-batch plug-in coefficient $\widehat{\alpha}_c$ or to a pooled anchor that reuses the current rollout. □

Relationship to classical James–Stein shrinkage. Classical James–Stein estimators can dominate the unshrunk sample-mean vector under a specified multivariate Gaussian normal-means model and an aggregate quadratic-risk criterion when the dimension is at least three [7]. Our theorem instead treats one context at a time under a hierarchical Bayes model with known parameters. The classical domination claim must not be used to justify per-context MSE dominance or unbiasedness of the empirical reward-baseline interpolation.

5.7.2 Scheduler Network Architectures

We summarize the network architecture we used for implementing the scheduler network in Table 5.5. The scheduler has approximately 20M parameters (about 0.1–1% of the backbone parameter count for the models considered) and incurs approximately 0.1–0.3% additional end-to-end sampling compute over the evaluated 5–80-step budgets.

Table 5.5: **Network architecture for the Dirichlet schedule policy $\alpha_\theta(x, c)$.** The network takes an image-shaped noise tensor x, a sequence of text embeddings e_{text}, and an optional pooled text embedding $\bar{e}_{\text{text}}$ (consistent with SD-XL, SD3.5M/L and FLUX models), and outputs Dirichlet parameters $\boldsymbol{\alpha} \in \mathbb{R}_+^{L+1}$ over L schedule intervals plus one skipped interval. Default network architecture hyperparameters: `text_embed_dim` $d_{\text{text}} = 2048$, `pooled_text_embed_dim` $d_{\text{pool}} = 1280$, `image_encoder_depth` $n_{\text{conv}} = 2$, `image_encoder_width` $d_{\text{conv}} = 32$, `attention_dim` $d_{\text{attn}} = 256$, `cross_attention_heads` $n_{\text{attn}} = 4$, `number_of_transformer_blocks` $N_b = 4$, `hidden_dim` $d_h = 256$, `num_mlp_layers` $n_{\text{mlp}} = 2$.

Layers	Output size	Note
	Inputs	
Input: x	$B \times H \times W \times C$	image-shaped noise
Input: e_{text}	$B \times L_{\text{text}} \times d_{\text{text}}$	token text emb.; d_{text}=2048
Input: $\bar{e}_{\text{text}}$	$B \times d_{\text{pool}}$	optional pooled emb.; d_{pool}=1280
	Image encoder & cross-attention blocks	
Image encoder conv stack (inside block i, layers $j = 0, \ldots, n_{\text{conv}}-1$)	$B \times H_i \times W_i \times C_i$	n_{conv} conv layers, 3×3, stride 1 GroupNorm, SiLU $C_i = d_{\text{conv}} \cdot 2^{\min(4,\, i+j)}$
Flatten spatial dims	$B \times (H_i W_i) \times C_i$	prepare for attention
Query projection (Dense)	$B \times (H_i W_i) \times C_i$	applied to image features (`query_projection`)
Key/Value projection (Dense)	$B \times L_{\text{text}} \times C_i$	applied to e_{text} (`key_value_projection`)
MultiHeadDotProductAttention	$B \times (H_i W_i) \times C_i$	cross-attn: query from image, key/value from text, n_{attn} heads
Residual + LayerNorm	$B \times (H_i W_i) \times C_i$	LayerNorm(query + attn_output) (`cross_attn_norm`)
Reshape to image grid	$B \times H_i \times W_i \times C_i$	output of `SchedulerTransformerBlock`
Global avg. pooling (per block)	$B \times C_i$	mean over (H_i, W_i) for feature fusion
Downsample Conv 3×3, stride 2 + GroupNorm + SiLU (between blocks)	$B \times H_{i+1} \times W_{i+1} \times C_{i+1}$	from reshaped image grid, for $i < N_b-1$; $C_{i+1} = d_{\text{conv}} \cdot 2^{\min(4,\,(i+1)\, n_{\text{conv}}-1)}$
	Feature pyramid fusion	
Concat. over blocks	$B \times \sum_i C_i$	feature pyramid $\{\text{block } i\}_{i=0}^{N_b-1}$
Concat. with $\bar{e}_{\text{text}}$	$B \times \left(\sum_i C_i + d_{\text{pool}}\right)$	only if pooled emb. used
	MLP Dirichlet head	
Dense, SiLU × $(n_{\text{mlp}} - 1)$	$B \times d_h$	default $d_h = 256$
Dense (Dirichlet logits head)	$B \times (L{+}1)$	L = `num_timesteps`
Softplus $+10^{-3}$	$B \times (L{+}1)$	$\boldsymbol{\alpha}$ = softplus$(\cdot)+10^{-3}$

5.7.3 Experiment Settings

5.7.3.1 Synthetic Sanity Check for Baseline Variance

To isolate the small-rollout regime, we construct a synthetic experiment with an analytic policy and spiky, heteroscedastic random-effects rewards. We report measured gradient-estimator variance for RLOO and the batch-estimated JS-style baseline. As explained below, the simulated reward is independent of the sampled action, so the true policy gradient is zero and the experiment measures zero-gradient estimator noise. The plug-in estimator is not covered by a general unbiasedness theorem.

Setup. We adopt an analytic Dirichlet policy $\pi(\tau) = \text{Dirichlet}(\boldsymbol{\alpha})$ with concentration $\boldsymbol{\alpha} = 2\mathbf{1}_L$ and horizons $L \in \{4, 16, 64\}$. Its score function has a closed form:

$$\nabla_{\alpha_j} \log \pi(\tau) = \psi\Big(\sum_t \alpha_t\Big) - \psi(\alpha_j) + \log \tau_j,$$

where $\psi(\cdot)$ denotes the digamma function and $\log \tau_j$ is the elementwise logarithm of the j-th coordinate of τ. To mimic the two-level random–effects model used in our theoretical analysis, we generate synthetic rewards as

$$\begin{aligned} &r^{(c,i)} = \mu_c + \varepsilon^{(c,i)} + \delta^{(c,i)}, \\ &\mu_c \sim \mathcal{N}(0,1), \quad \varepsilon^{(c,i)} \sim \mathcal{N}\big(0, s_c^2\big), \\ &s_c \sim \text{LogNormal}(0,1), \quad \delta^{(c,i)} \sim \text{Bernoulli}(0.15) \times 8, \end{aligned} \tag{5.18}$$

where μ_c acts as the context-level effect (analogous to $\mu_0 + \xi^{(c)}$ in Eq. (5.6)) and $\varepsilon^{(c,i)}$ represents within-context noise. The additional sparse term $\delta^{(c,i)}$ introduces occasional outliers, creating a heavy-tailed, heteroskedastic reward landscape that stresses RLOO in the small-K regime.

Here the synthetic rewards are generated independently of τ, so $\nabla_\alpha \mathbb{E}[r] = 0$. We sweep through different settings for (B, K, L). For each (B, K, L) we generate 500 i.i.d. batches,

Table 5.6: Synthetic variance sanity check. Average per-dimension variance of REINFORCE for an analytic Dirichlet policy. Rewards follow Eq. (5.18). Entries are RLOO → JS. JS consistently gives gain over RLOO baselines for various settings.

K	B	L=4	L=16	L=64
2	8	1.055 → **0.612**	1.445 → **0.725**	1.148 → **0.625**
	16	0.479 → **0.285**	0.663 → **0.339**	0.606 → **0.339**
	32	0.261 → **0.140**	0.328 → **0.174**	0.310 → **0.165**
4	8	0.393 → **0.284**	0.402 → **0.307**	0.386 → **0.299**
	16	0.175 → **0.143**	0.193 → **0.154**	0.210 → **0.163**
	32	0.086 → **0.066**	0.099 → **0.079**	0.106 → **0.084**
8	8	0.140 → **0.126**	0.191 → **0.170**	0.182 → **0.166**
	16	0.071 → **0.064**	0.099 → **0.090**	0.086 → **0.078**
	32	0.038 → **0.034**	0.042 → **0.038**	0.045 → **0.040**
16	8	0.072 → **0.070**	0.081 → **0.078**	0.094 → **0.089**
	16	0.031 → **0.030**	0.038 → **0.037**	0.039 → **0.038**
	32	0.017 → **0.016**	0.020 → **0.019**	0.020 → **0.019**

compute

$$\hat{g}(b) = \frac{1}{BK} \sum_{c=1}^{B} \sum_{i=1}^{K} \left(r^{(c,i)} - b^{(c,i)} \right) \nabla_\alpha \log \pi(\tau^{(c,i)}),$$

once with RLOO and once with JS, and report the empirical per-dimension variance $\frac{1}{L} \sum_{j=1}^{L} \mathrm{Var}(\hat{g}_j)$. We sweep $B \in \{8, 16, 32\}$ and $K \in \{2, 4, 8, 16\}$.

Observations. As shown in Table 5.6, the JS-style baseline has lower measured gradient-estimator variance than RLOO in every reported synthetic configuration. For the most challenging regime ($K = 2$), JS reduces variance by roughly 45–50%, without requiring additional sampler rollouts in this comparison. Even with moderate rollouts ($K = 4$), variance drops by 20–25%, demonstrating that the improvement is systematic rather than case-specific. Across all settings, JS remains consistently superior, reflecting its adaptive shrinkage across both within- and cross-context variability. These results support the use of shrinkage under the tested reward model; they do not prove the same ranking for every reward distribution or identify the bias of a full-batch plug-in estimator in general. We visualize a side-by-side comparison in Fig. 5.2a for the $B = 32, L = 16$ case with different number of rollouts K.

Table 5.7: Learning rates for schedule training on HPDv2 prompts.

Backbone	L=5	L=10	L=20	L=40	L=80
SD-XL	5e−5	5e−5	5e−5	5e−5	5e−5
SD3.5-M	5e−5	2.5e−5	2.5e−5	1.25e−5	5e−5
SD3.5-L	5e−5	5e−5	5e−5	5e−5	5e−5
Flux-Dev	2.5e−5	5e−5	2.5e−5	5e−5	5e−5

5.7.3.2 General Text-to-Image Experiments on HPDv2

General settings. For all the experiments, we consistently use the AdamW [15] optimizer for learning scheduler parameters. We set the `weight_decay` to $1e-4$ throughout experiments, tune the learning rates (see Table 5.7), and use default values for the remaining hyperparameters. We apply the gradient clipping with a maximum norm of 1. For all models we consistently use a rollout number of 2, and set the output resolution to 1024×1024. Due to memory limit, we use a batch size of 32 for smaller models like SD-XL or SD-3.5M, and a batch size of 16 for larger ones like SD-3.5L and Flux-Dev. We use a guidance weight of 7.5 for SD-XL and 5.0 for SD3.5-M/L and Flux models. For SD-XL model we use the DDIM solver [21] for per step denoising update; for the flow-matching-based SD3.5 and Flux models we use the Euler solver.

TPDM-style baseline. We reimplement the TPDM baseline using (i) the same network architecture we used, and (ii) the same set of seeds, hyperparameters and base model settings for training. We follow the customized PPO implementation in TPDM with no ratio clipping.

We would like to explicitly mention that (i) we *exclude* any regularizer including KL constraint between a reference schedule and the learned schedule during training to be consistent with our settings. This is different from the vanilla implementation of TPDM, where KL constraint is used. (ii) TPDM in its basic form *does not* involve text condition as its input. We experiment with different settings: (i) we faithfully nullify the textual inputs by setting the embeddings to zero vectors to train the model, and (ii) we include the textual inputs as in our settings. We report in Table 5.1 the results for (i). For (ii) the variant,

Table 5.8: Learning rates for schedule training on text rendering and geneval-style prompts. Num. of sampling steps is 40.

Backbone	**SD3.5-M**	**Flux-Dev**
Text Render	1.25e−5	5e−5
Geneval	1.25e−6	5e−5

especially in our setting, reduces to the RLOO baseline, which we also report in Table 5.1.

5.7.3.3 Text Rendering and Fine-Grained Alignment

We use the same settings as in Section 5.7.3.2, except for tuning the learning rates (see Table 5.8). For the OCR metrics, we adopt the commonly used mask textspotter v3 model [11] to calculate the character level accuracy, precision and recall scores for each image, and then calculate the average across the testing samples as the final results in Table 5.3.

5.7.4 Additional Qualitative Results

We provide further T2I results for 5-step sampling (Figs. 5.8 and 5.9), general T2I generation on HPDv2 prompts (Figs. 5.10 and 5.11), text rendering (Figs. 5.12 and 5.13) and fine-grained object-focused generation (Figs. 5.14 and 5.15). We follow the format of Fig. 5.1 and plot the corresponding schedules for generation at the top left corner.

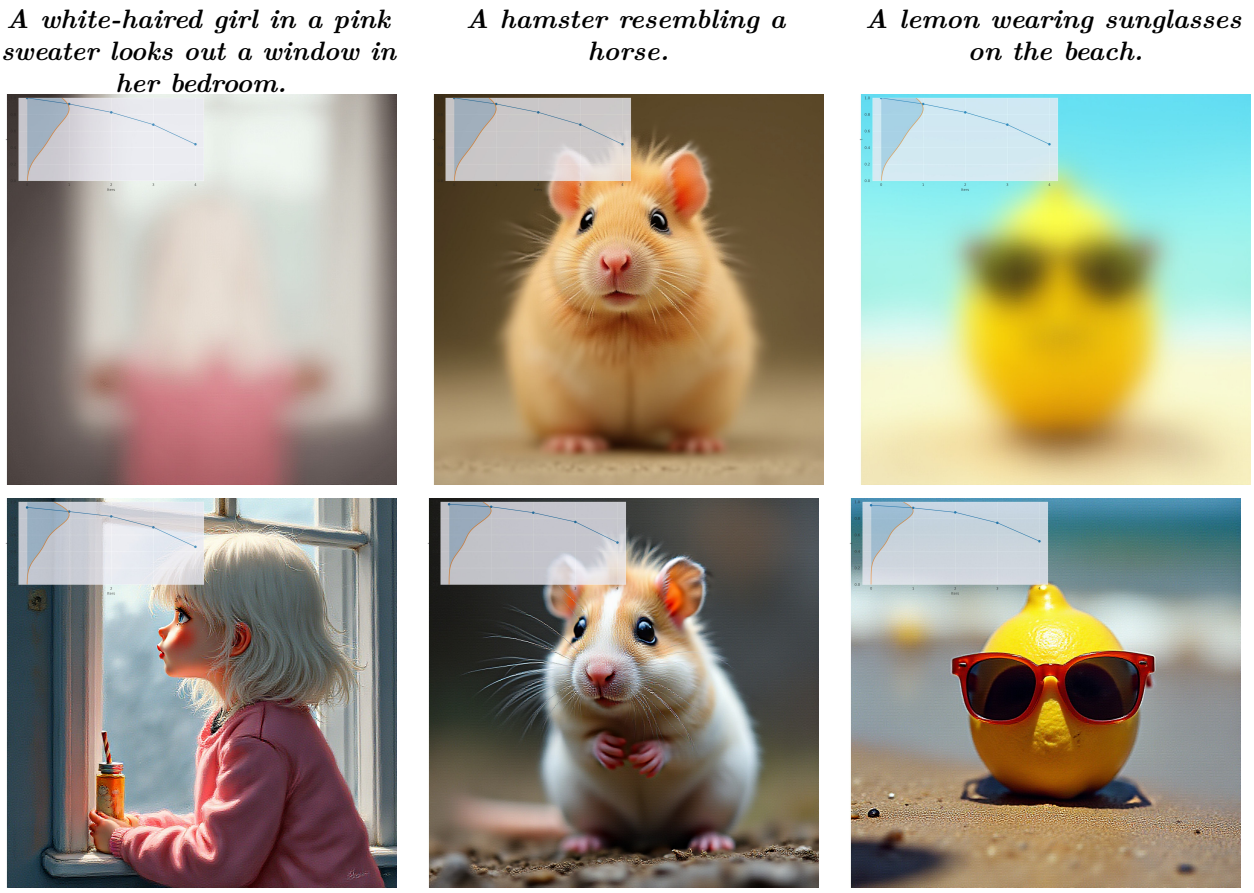


Figure 5.8: **Rescheduling improves few-step sampling**. Comparisons between images generated with default schedules (upper) and our learned schedules (lower) from Flux-Dev, with 5 steps.



Figure 5.9: **Rescheduling improves few-step sampling**. Comparisons between images generated with default schedules (upper) and our learned schedules (lower) from Flux-Dev, with 5 steps.

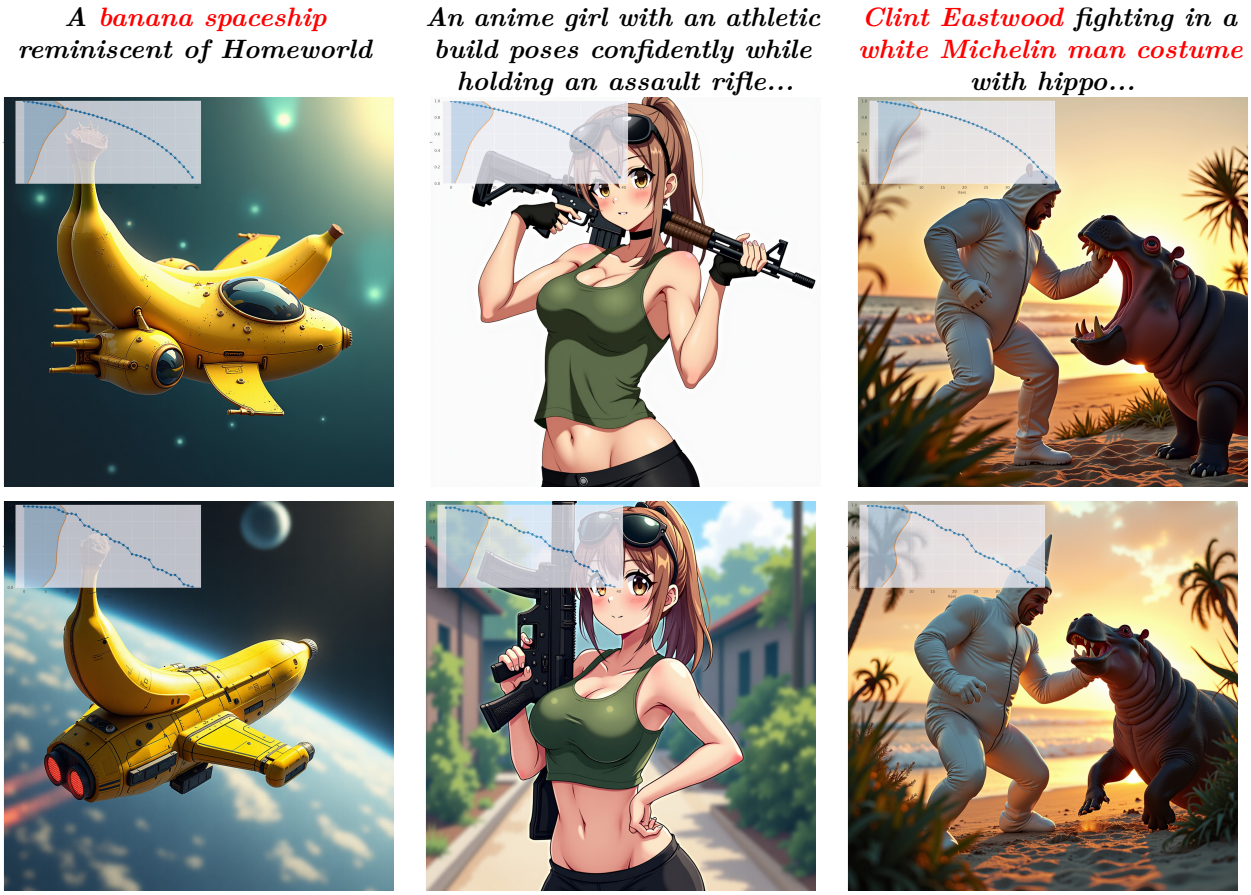


Figure 5.10: **Rescheduling improves general T2I alignment.** Head-to-head comparisons between images generated with default schedules (upper) and our learned schedules (lower) from Flux-Dev with 40 steps.



Figure 5.11: **Rescheduling improves general T2I alignment.** Head-to-head comparisons between images generated with default schedules (upper) and our learned schedules (lower) from Flux-Dev with 40 steps.



Figure 5.12: **Rescheduling improves text rendering**. We present comparisons between images generated with default schedules (upper) and our learned schedules (lower) from Flux-Dev.



Figure 5.13: **Rescheduling improves text rendering**. We present comparisons between images generated with default schedules (upper) and our learned schedules (lower) from Flux-Dev.



Figure 5.14: **Rescheduling improves fine grained alignment**. Comparisons between images generated with default schedules (upper) and our learned schedules (lower) from Flux-Dev.



Figure 5.15: **Rescheduling improves fine grained alignment**. Comparisons between images generated with default schedules (upper) and our learned schedules (lower) from Flux-Dev.

Bibliography

[1] Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. Back to basics: Revisiting reinforce-style optimization for learning from human feedback in llms. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12248–12267, 2024.

[2] Kevin Black, Michael Janner, Yilun Du, Ilya Kostrikov, and Sergey Levine. Training diffusion models with reinforcement learning. *arXiv preprint arXiv:2305.13301*, 2023.

[3] Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, et al. Scaling rectified flow transformers for high-resolution image synthesis. In *Forty-first international conference on machine learning*, 2024.

[4] Ying Fan, Olivia Watkins, Yuqing Du, Hao Liu, Moonkyung Ryu, Craig Boutilier, Pieter Abbeel, Mohammad Ghavamzadeh, Kangwook Lee, and Kimin Lee. Dpok: Reinforcement learning for fine-tuning text-to-image diffusion models. *Advances in Neural Information Processing Systems*, 36:79858–79885, 2023.

[5] Dhruba Ghosh, Hannaneh Hajishirzi, and Ludwig Schmidt. Geneval: An object-focused framework for evaluating text-to-image alignment. *Advances in Neural Information Processing Systems*, 36:52132–52152, 2023.

[6] Evan Greensmith, Peter L Bartlett, and Jonathan Baxter. Variance reduction techniques for gradient estimates in reinforcement learning. *Journal of Machine Learning Research*, 5(Nov):1471–1530, 2004.

[7] William James, Charles Stein, et al. Estimation with quadratic loss. In *Proceedings of the fourth Berkeley symposium on mathematical statistics and probability*, pages 361–379. University of California Press, 1961.

[8] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. *Advances in neural information processing systems*, 35:26565–26577, 2022.

[9] Wouter Kool, Herke van Hoof, and Max Welling. Buy 4 reinforce samples, get a baseline for free! 2019.

[10] Black Forest Labs. Flux. `https://github.com/black-forest-labs/flux`, 2024.

[11] Minghui Liao, Guan Pang, Jing Huang, Tal Hassner, and Xiang Bai. Mask textspotter v3: Segmentation proposal network for robust scene text spotting. In *European conference on computer vision*, pages 706–722. Springer, 2020.

[12] Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

[13] Jie Liu, Gongye Liu, Jiajun Liang, Yangguang Li, Jiaheng Liu, Xintao Wang, Pengfei Wan, Di Zhang, and Wanli Ouyang. Flow-grpo: Training flow matching models via online rl. *arXiv preprint arXiv:2505.05470*, 2025.

[14] Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. *arXiv preprint arXiv:2209.03003*, 2022.

[15] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[16] Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver: A fast ode solver for diffusion probabilistic model sampling in around 10 steps. *Advances in neural information processing systems*, 35:5775–5787, 2022.

[17] Dustin Podell, Zion English, Kyle Lacey, Andreas Blattmann, Tim Dockhorn, Jonas Müller, Joe Penna, and Robin Rombach. Sdxl: Improving latent diffusion models for high-resolution image synthesis. *arXiv preprint arXiv:2307.01952*, 2023.

[18] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022.

[19] Amirmojtaba Sabour, Sanja Fidler, and Karsten Kreis. Align your steps: Optimizing sampling schedules in diffusion models. *arXiv preprint arXiv:2404.14507*, 2024.

[20] Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. *arXiv preprint arXiv:2202.00512*, 2022.

[21] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020.

[22] Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. In *Proceedings of the 40th International Conference on Machine Learning*, pages 32211–32252, 2023.

[23] Bram Wallace, Meihua Dang, Rafael Rafailov, Linqi Zhou, Aaron Lou, Senthil Purushwalkam, Stefano Ermon, Caiming Xiong, Shafiq Joty, and Nikhil Naik. Diffusion model alignment using direct preference optimization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8228–8238, 2024.

[24] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256, 1992.

[25] Xiaoshi Wu, Yiming Hao, Keqiang Sun, Yixiong Chen, Feng Zhu, Rui Zhao, and Hongsheng Li. Human preference score v2: A solid benchmark for evaluating human preferences of text-to-image synthesis. *arXiv preprint arXiv:2306.09341*, 2023.

[26] Sirui Xie, Zhisheng Xiao, Diederik P Kingma, Tingbo Hou, Ying N Wu, Kevin Murphy, Tim Salimans, Ben Poole, and Ruiqi Gao. Em distillation for one-step diffusion models. *Advances in Neural Information Processing Systems*, 37:45073–45104, 2024.

[27] Zilyu Ye, Zhiyang Chen, Tiancheng Li, Zemin Huang, Weijian Luo, and Guo-Jun Qi. Schedule on the fly: Diffusion time prediction for faster and better image generation. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 23412–23422, 2025.

[28] Tianwei Yin, Michaël Gharbi, Richard Zhang, Eli Shechtman, Fredo Durand, William T Freeman, and Taesung Park. One-step diffusion with distribution matching distillation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6613–6623, 2024.

[29] Peiyu Yu, Dinghuai Zhang, Hengzhi He, Xiaojian Ma, Sirui Xie, Ruiyao Miao, Yifan Lu, Yasi Zhang, Deqian Kong, Ruiqi Gao, et al. "Noisier" noise contrastive estimation is (almost) maximum likelihood. In *The Fourteenth International Conference on Learning Representations*, 2026.

[30] Wenliang Zhao, Lujia Bai, Yongming Rao, Jie Zhou, and Jiwen Lu. Unipc: A unified predictor-corrector framework for fast sampling of diffusion models. *Advances in Neural Information Processing Systems*, 36:49842–49869, 2023.

[31] Zikai Zhou, Shitong Shao, Lichen Bai, Shufei Zhang, Zhiqiang Xu, Bo Han, and Zeke

Xie. Golden noise for diffusion models: A learning framework. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 17688–17697, 2025.

Chapter 6

Unsupervised Foreground Extraction via Deep Region Competition

6.1 Introduction

Foreground extraction, being a special case of generic image segmentation, aims for a binary partition of the given image with specific semantic meaning, *i.e.*, a foreground that typically contains identifiable objects and the possibly less structural remaining regions as the background. There is a rich literature on explicitly modeling and representing a given image as foreground and background (or more general visual regions), such that a generic inference algorithm can produce plausible segmentations ideally for any images without or with little supervision [1–8]. However, such methods essentially rely on low-level visual features (*e.g.*, edges, color, and texture), and some further require human intervention at initialization [4, 5], which largely limits their practical performance on modern datasets of complex natural images with rich semantic meanings [9, 10]. These datasets typically come with fine-grained semantic annotations, exploited by supervised methods that learn representation and inference algorithm as one monolithic network [11–16]. Despite the success of densely supervised learning, the unsupervised counterpart is still favored due to its

resemblance to how humans perceive the world [17, 18].

Attempting to combine unsupervised or weakly supervised learning with modern neural networks, three lines of work surge recently for foreground extraction: (1) deep networks as feature extractors for canonical segmentation algorithms, (2) GAN-based foreground-background disentanglement, and (3) compositional latent variable models with slot-based object modeling. Despite great successes of these methods, the challenge of unsupervised foreground extraction remains largely open.

Specifically, the first line of work trains designated deep feature extractors for canonical segmentation algorithms or metric networks as learned partitioning criteria [19–21]. These methods (*e.g.*, W-Net [19]) define foreground objects' properties using learned features or criteria and are thus generally bottlenecked by the selected post-processing segmentation algorithm [22, 23]. As a branch of pioneering work that moves beyond these limitations, Yang et al. [24, 25] have recently proposed a general contextual information separation principle and an efficient adversarial learning method that is generally applicable to unsupervised segmentation, separation and detection. GAN-based models [26–31] capture the foreground objectness with oversimplified assumptions or require additional supervision to achieve foreground-background disentanglement. For example, the segmentation model in ReDO [28] is trained by redrawing detected objects, which potentially limits its application to datasets with diverse object shapes. OneGAN [31] and its predecessors [29, 30], though producing impressive results on foreground extraction, require a set of background images without foreground objects as additional inputs. Lastly, compositional latent variable models [32–40] include the background as a "virtual object" and induce the independence of object representations using an identical generator for all object slots. Although these methods exhibit strong performance on synthetic multi-object datasets with simple backgrounds and foreground shapes, they may fail on complex real-world data or even synthetic datasets with more challenging backgrounds [37, 38]. In addition, few unsupervised learning methods have provided explicit identification of foreground objects and background regions. While they

can generate valid segmentation masks, most of these methods do not specify which output corresponds to the foreground objects. These deficiencies necessitate rethinking the problem of unsupervised foreground extraction. We propose to confront the challenges in formulating (1) a generic inductive bias for modeling foreground and background regions that can be baked into neural generators, and (2) an effective inference algorithm based on a principled criterion for foreground-background partition.

Inspired by Region Competition [1], a seminal approach that combines optimization-based inference [41–43] and probabilistic visual modeling [44, 45] by minimizing a generalized Bayes criterion [46], we propose to solve the foreground extraction problem by reconciling energy-based prior [47] with generative image modeling in the form of Mixture of Experts (MoE) [48, 49]. To generically describe background regions, we further introduce the learned pixel re-assignment as the essential inductive bias to capture their regularities. Fueled by our modeling, we propose to find the foreground-background partition through Expectation-Maximization (EM). Our algorithm effectively exploits the interaction between the mixture components during the partitioning process, echoing the intuition described in Region Competition [1]. We therefore coin our method Deep Region Competition (DRC). We summarize our **contributions** as follows:

1. We provide probabilistic foreground-background modeling by reconciling energy-based prior with generative image modeling in the form of MoE. With this modeling, the foreground-background partition can be naturally produced through EM. We further introduce an inductive bias, *pixel re-assignment*, to facilitate foreground-background disentanglement.

2. In experiments, we demonstrate that DRC exhibits more competitive performances on complex real-world data and challenging multi-object scenes compared with prior methods. Furthermore, we empirically show that using learned pixel re-assignment as the inductive bias helps to provide explicit identification for foreground and background regions.

3. We find that DRC can potentially generalize to novel foreground objects even from

categories unseen during training, which may provide some inspiration for the study of out-of-distribution (OOD) generalization in more general unsupervised disentanglement.

6.2 Related Work

A typical line of methods frames unsupervised or weakly supervised foreground segmentation within a generative modeling context. Several methods build upon generative adversarial networks (GAN) [26] to perform foreground segmentation. LR-GAN [27] learns to generate background regions and foreground objects separately and recursively, which simultaneously produces the foreground objects mask. ReDO (ReDrawing of Objects) [28] proposes a GAN-based object segmentation model, based on the assumption that replacing the foreground object in the image with a generated one does not alter the distribution of the training data, given that the foreground object is correctly discovered. Similarly, SEIGAN [29] learns to extract foreground objects by recombining the foreground objects with the generated background regions. FineGAN [30] hierarchically generates images (*i.e.*, first specifying the object shape and then the object texture) to disentangle the background and foreground object. Benny and Wolf [31] further hypothesize that a method solving an ensemble of unsupervised tasks altogether improves the model performance compared with the one that solves each individually. Therefore, they train a complex GAN-based model (OneGAN) to solve several tasks simultaneously, including foreground segmentation. Although LR-GAN and FineGAN do produce masks as part of their generative process, they cannot segment a given image. Despite SEIGAN and OneGAN achieving decent performance on foreground-background segmentation, these methods require a set of clean background images as additional inputs for weak supervision. ReDO captures the foreground objectness with possibly oversimplified assumptions, limiting its application to datasets with diverse object shapes.

On another front, compositional generative scene models [32–40], sharing the idea of scene decomposition stemming from DRAW [50], learn to represent foreground objects and

background regions in terms of a collection of latent variables with the same representational format. These methods typically exploit the spatial mixture model for generative modeling. Specifically, IODINE [37] proposes a slot-based object representation method and models the latent space using iterative amortized inference [51]. Slot-Attention [38], as a step forward, effectively incorporates the attention mechanism into the slot-based object representation for flexible foreground object binding. Both methods use fully shared parameters among individual mixture components to entail permutation invariance of the learned multi-object representation. Alternative models such as MONet [36] and GENESIS [39] use multiple encode-decode steps for scene decomposition and foreground object extraction. Although these methods exhibit strong performance on synthetic multi-object datasets with simple background and foreground shapes, they may fail when dealing with complex real-world data or even synthetic datasets with more challenging background [37, 38].

More closely related to the classical methods, another line of work focuses on utilizing image features extracted by deep neural networks or designing energy functions based on data-driven methods to define the desired property of foreground objects. Pham et al. [52] and Silberman et al. [53] obtain impressive results when depth images are accessible in addition to conventional RGB images, while such methods are not directly applicable for data with RGB images alone. W-Net [19] extracts image features via a deep auto-encoder jointly trained by minimizing reconstruction error and normalized cut. The learned features are further processed by CRF smoothing to perform hierarchical segmentation. Kanezaki [20] proposes to employ a neural network as part of the partitioning criterion (inspired by Ulyanov et al. [54]) to minimize the chosen intra-region pixel distance for segmentation directly. Ji et al. [21] propose to use Invariant Information Clustering as the objective for segmentation, where the network is trained to be part of the learned distance. As an interesting extension, one may also consider adapting methods that automatically discover object structures [55] to foreground extraction. Though being pioneering work in image segmentation, the aforementioned methods are generally bottlenecked by the selected post-processing segmentation algorithm or require

extra transformations to produce meaningful foreground segmentation masks. Yang et al. [24, 25] in their seminal work propose an information-theoretical principle and adversarial contextual model for unsupervised segmentation and detection by partitioning images into maximally independent sets, with the objective of minimizing the predictability of one set by the other sets. Additional efforts have also been devoted to weakly supervised foreground segmentation using image classification labels [56–58], bounding boxes [59, 60], or saliency maps [61–63].

6.3 Methodology

Foreground extraction performs a binary partition for the image $\mathbf{I}$ to extract the foreground region. Without explicit supervision, we propose to use learned pixel re-assignment as a generic inductive bias for background modeling, upon which we derive an EM-like partitioning algorithm. Compared with prior methods, our algorithm can handle images with more complex foreground shapes and background patterns, while providing explicit identification of foreground and background regions.

6.3.1 Preliminaries

Adopting the language of EM algorithm, we assume that for the observed sample $\mathbf{x} \in \mathbb{R}^D$, there exists $\mathbf{z} \in \mathbb{R}^d$ as its latent variables. The complete-data distribution is

$$p_\theta(\mathbf{z}, \mathbf{x}) = p_\alpha(\mathbf{z}) p_\beta(\mathbf{x}|\mathbf{z}), \tag{6.1}$$

where $p_\alpha(\mathbf{z})$ is the prior model with parameters α, $p_\beta(\mathbf{x}|\mathbf{z})$ is the top-down generative model with parameters β, and $\theta = (\alpha, \beta)$.

The prior model $p_\alpha(\mathbf{z})$ can be formulated as an energy-based model, which we refer to as

the LEBM [47] throughout this chapter:

$$p_\alpha(\mathbf{z}) = \frac{1}{Z_\alpha} \exp\left(f_\alpha(\mathbf{z})\right) p_0(\mathbf{z}), \tag{6.2}$$

where $f_\alpha(\mathbf{z})$ can be parameterized by a neural network, Z_α is the partition function, and $p_0(\mathbf{z})$ is a reference distribution, assumed to be isotropic Gaussian prior commonly used for the generative model. The prior model in Eq. (6.2) can be interpreted as an energy-based correction or exponential tilting of the original prior distribution p_0.

The LEBM can be learned by MLE. Given a training sample $\mathbf{x}$, the learning gradient for α is derived as shown by Pang et al. [47],

$$\delta_\alpha(\mathbf{x}) = \mathbf{E}_{p_\theta(\mathbf{z}|\mathbf{x})}\left[\nabla_\alpha f_\alpha(\mathbf{z})\right] - \mathbf{E}_{p_\alpha(\mathbf{z})}\left[\nabla_\alpha f_\alpha(\mathbf{z})\right]. \tag{6.3}$$

In practice, the above expectations can be approximated by Monte-Carlo average, which requires sampling from $p_\theta(\mathbf{z}|\mathbf{x})$ and $p_\alpha(\mathbf{z})$. This step can be done with stochastic gradient-based methods, such as Langevin dynamics [64] or Hamiltonian Monte Carlo [65].

An extension to LEBM is to further couple the vector representation $\mathbf{z}$ with a symbolic representation $\mathbf{y}$ [66]. Formally, $\mathbf{y}$ is a K-dimensional one-hot vector, where K is the number of possible $\mathbf{z}$ categories. Such symbol-vector duality can provide extra entries for auxiliary supervision; we will detail it in Section 6.3.4.

6.3.2 Generative Image Modeling

Mixture of Experts (MoE) for Image Generation Inspired by the regional homogeneity assumption proposed by Zhu and Yuille [1], we use separate priors and generative models for foreground and background regions, indexed as α_k and β_k, $k = 1, 2$, respectively; see Fig. 6.1. This design leads to the form of MoE [48, 49] for image modeling, as shown below.

Let us start by considering only the i-th pixel of the observed image $\mathbf{x}$, denoted as $\mathbf{x}_i$.

We use a binary one-hot random variable $\mathbf{w}_i$ to indicate whether the i-th pixel belongs to the foreground region. Formally, we have $\mathbf{w}_i = [w_{i1}, w_{i2}]$, $w_{ik} \in \{0, 1\}$ and $\sum_{k=1}^{2} w_{ik} = 1$. Let $w_{i1} = 1$ indicate that the i-th pixel $\mathbf{x}_i$ belongs to the foreground, and $w_{i2} = 1$ indicate the opposite.

We assume that the distribution of $\mathbf{w}_i$ is prior-dependent. Specifically, the mixture parameter $\pi_{ik},\ k = 1, 2$, is defined as the output of a gating function $\pi_{ik} = p_\beta(w_{ik} = 1|\mathbf{z}) = \frac{\exp(l_{ik})}{\sum_{j=1}^{2} \exp(l_{ij})}$; $l_{ik} = h_{\beta_k}(\mathbf{z}_k),\ k = 1, 2$ are the logit scores given by the foreground and background generative models respectively; $\beta = \{\beta_1, \beta_2\}, \mathbf{z} = \{\mathbf{z}_1, \mathbf{z}_2\}$. Taken together, the joint distribution of $\mathbf{w}_i$ is

$$p_\beta(\mathbf{w}_i|\mathbf{z}) = \prod_{k=1}^{2} \pi_{ik}^{w_{ik}}. \tag{6.4}$$

We model the foreground and background pixel values with Gaussian observation noise:

$$\mathbf{x}_i \mid (w_{ik} = 1, \mathbf{z}_k) \sim \mathcal{N}\big(g_{\beta_k}(\mathbf{z}_k)_i, \sigma^2\mathbf{I}\big),\ k = 1, 2 \tag{6.5}$$

where we assume that the generative model for region content, $p_{\beta_k}(\mathbf{x}_i|\mathbf{z}_k),\ k = 1, 2$, follows a Gaussian distribution parameterized by the generator network g_{β_k}. As in VAE, σ takes an assumed value. We follow the common practice and use a shared generator for parameterizing π_{ik} and $p_{\beta_k}(\mathbf{x}_i|\mathbf{z}_k)$. We use separate branches only at the output layer to generate logits and contents.

Generating $\mathbf{x}_i$ based on $\mathbf{w}_i$'s distribution involves two steps: (1) sample $\mathbf{w}_i$ from the distribution $p_\beta(\mathbf{w}_i|\mathbf{z})$, and (2) choose either the foreground model (*i.e.*, $p_{\beta_1}(\mathbf{x}_i|\mathbf{z}_1)$) or the background model (*i.e.*, $p_{\beta_2}(\mathbf{x}_i|\mathbf{z}_2)$) to generate $\mathbf{x}_i$ based on the sampled $\mathbf{w}_i$. As such, this distribution of $\mathbf{x}_i$ is a MoE,

$$p_\beta(\mathbf{x}_i|\mathbf{z}) = \sum_{k=1}^{2} p_\beta(w_{ik} = 1|\mathbf{z}) p_\beta(\mathbf{x}_i|w_{ik} = 1, \mathbf{z}_k) = \sum_{k=1}^{2} \pi_{ik} p_{\beta_k}(\mathbf{x}_i|\mathbf{z}_k), \tag{6.6}$$

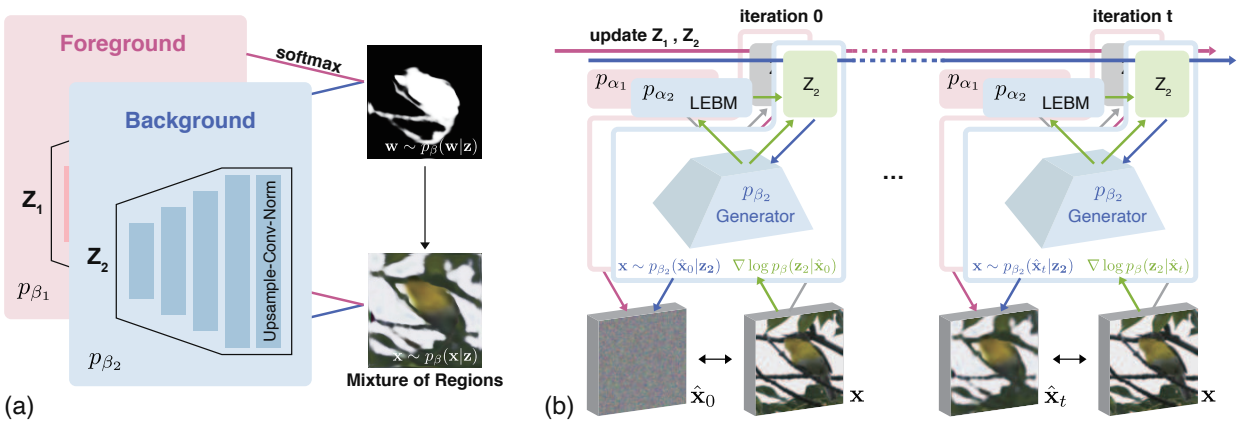


Figure 6.1: **Overview of DRC.** (a) The model generates foreground and background regions using sampled latent variables $\mathbf{z} = \{\mathbf{z}_1, \mathbf{z}_2\}$. $p_{\beta_k},\ k = 1, 2$ represents the generator for each region. Of note, the pixel re-assignment function is absorbed in the background generator; see Section 6.3.2 for details. (b) DRC samples the latent variables $\mathbf{z}$ in an iterative manner. Let $\mathbf{x}$ denote the observed image; we use $\hat{\mathbf{x}}_t,\ t = 0, 1, ...$ to represent the image generated by $p_\beta(\mathbf{x}|\mathbf{z})$ at the t-th sampling step. DRC has a two-step workflow for learning unsupervised foreground extractors that resembles the E- and M-step in the classic EM algorithm. In the E-step, it employs gradient-based MCMC sampling to infer the latent variables $\mathbf{z}$ as shown in (b). Of note, only the latent variables $\mathbf{z}$ are updated in this step. In the M-step, the sampled latent variables $\mathbf{z}$ are fed into the model for image generation as shown in (a), where the generators are updated to minimize the reconstruction error.

wherein the posterior responsibility of w_{ik} is

$$\gamma_{ik} = p(w_{ik} = 1|\mathbf{x}_i, \mathbf{z}) = \frac{\pi_{ik} p_{\beta_k}(\mathbf{x}_i|\mathbf{z}_k)}{\sum_{m=1}^{2} \pi_{im} p_{\beta_m}(\mathbf{x}_i|\mathbf{z}_m)},\ k = 1, 2. \tag{6.7}$$

Using a fully-factorized joint distribution of $\mathbf{x}$, we have $p_\beta(\mathbf{x}|\mathbf{z}) = \prod_{i=1}^{D} \sum_{k=1}^{2} \pi_{ik} p_{\beta_k}(\mathbf{x}_i|\mathbf{z}_k)$ as the generative modeling of $\mathbf{x} \in \mathbb{R}^D$.

Learning Pixel Re-assignment for Background Modeling We use pixel re-assignment in the background generative model as the essential inductive bias for modeling the background region. This is partially inspired by the concepts of "texture" and "texton" by Julesz [45, 67], where the textural part of an image may contain fewer structural elements in preattentive vision, which coincides with our intuitive observation of the background regions.

We use a separate pair of energy-based prior model $\alpha_{\texttt{pix}}$ and generative model $\beta_{\texttt{pix}}$ to learn the re-assignment. For simplicity, we absorb $\alpha_{\texttt{pix}}$ and $\beta_{\texttt{pix}}$ in the models for background modeling, *i.e.*, α_2 and β_2, respectively. In practice, the re-assignment follows the output of $\beta_{\texttt{pix}}$, a shuffling grid with the same size of the image $\mathbf{x}$. Its values indicate the re-assigned

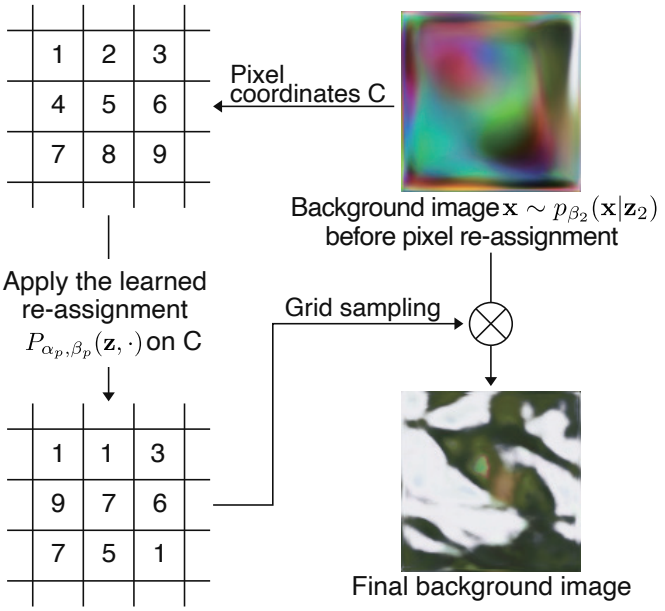


Figure 6.2: **Pixel re-assignment.** The output of β_p can be viewed as a learned re-assignment of the original background pixels that follows the mapped grid $P_{\alpha_p,\beta_p}(\mathbf{z}, C)$. Note that the re-assignment function $P_{\alpha_p,\beta_p}(\mathbf{z},\cdot)$ might not be injective. The final background image is generated via grid sampling.

pixel coordinates; see Fig. 6.2. We find that shuffling the background pixels using the learned re-assignment facilitates the model to capture the regularities of the background regions. Specifically, the proposed model with this essential inductive bias learns to constantly give the correct mask assignment, whereas most previous fully unsupervised methods do not provide explicit identification of the foreground and background regions; see discussion in Section 6.4.1 for more details.

6.3.3 Deep Region Competition: from Generative Modeling to Foreground Extraction

The complete-data distribution from the image modeling is

$$\begin{aligned} p_\theta(\mathbf{x},\mathbf{z},\mathbf{w}) &= p_\beta(\mathbf{x}|\mathbf{w},\mathbf{z})p_\beta(\mathbf{w}|\mathbf{z})p_\alpha(\mathbf{z}) \\ &= \left(\prod_{i=1}^{D}\prod_{k=1}^{2} p_{\beta_k}(\mathbf{x}_i|\mathbf{z}_k)^{w_{ik}}\right)\left(\prod_{i=1}^{D}\prod_{k=1}^{2}\pi_{ik}^{w_{ik}}\right)p_\alpha(\mathbf{z}) \\ &= p_\alpha(\mathbf{z})\prod_{i=1}^{D}\prod_{k=1}^{2}\left(\pi_{ik}p_{\beta_k}(\mathbf{x}_i|\mathbf{z}_k)\right)^{w_{ik}}, \end{aligned} \tag{6.8}$$

where $p_\alpha(\mathbf{z}) = p_{\alpha_1}(\mathbf{z}_1)p_{\alpha_2}(\mathbf{z}_2)$ is the prior model given by LEBMs. $\alpha = \{\alpha_1, \alpha_2\}$, and $\theta = \{\alpha, \beta\}$. $\mathbf{w}$ is the vector of $(\mathbf{w}_i), i = 1, ...D$, whose joint distribution is assumed to be fully-factorized.

Next, we derive the complete-data log-likelihood as our learning objective:

$$\mathcal{L}(\theta) = \log p_\theta(\mathbf{x}, \mathbf{z}, \mathbf{w}) = \log p_\alpha(\mathbf{z}) + \sum_{i=1}^{D}\sum_{k=1}^{2} w_{ik} \left(\log \pi_{ik} + \log p_{\beta_k}(\mathbf{x}_i|\mathbf{z}_k)\right). \tag{6.9}$$

Of note, $\mathbf{w}$ and $\mathbf{z}$ are unobserved variables in the modeling, which makes it impossible to learn the model directly through MLE. To calculate the gradients of θ, we instead optimize $\mathbf{E}_{\mathbf{z}\sim p(\mathbf{z}|\mathbf{x}),\mathbf{w}\sim p(\mathbf{w}|\mathbf{x},\mathbf{z})}[\mathcal{L}(\theta)]$ based on the fact that underlies the EM algorithm:

$$\begin{aligned} \nabla_\theta \log p_\theta(\mathbf{x}) &= \int_{\mathbf{z}} p_\theta(\mathbf{z}|\mathbf{x}) d\mathbf{z} \int_{\mathbf{w}} p_\theta(\mathbf{w}|\mathbf{z}, \mathbf{x}) \nabla_\theta \log p_\theta(\mathbf{x}, \mathbf{z}, \mathbf{w}) d\mathbf{w} \\ &= \mathbf{E}_{\mathbf{z}\sim p_\theta(\mathbf{z}|\mathbf{x}),\mathbf{w}\sim p_\theta(\mathbf{w}|\mathbf{x},\mathbf{z})}[\nabla_\theta \log p_\theta(\mathbf{x}, \mathbf{z}, \mathbf{w})]. \end{aligned} \tag{6.10}$$

Therefore, the derived surrogate learning objective becomes

$$\max_\theta \ \mathbf{E}_{\mathbf{z}\sim p_{\theta(t)}(\mathbf{z}|\mathbf{x})}\left[\mathcal{J}(\theta)\right], \text{ s.t. } \forall i, \sum_{k=1}^{2} \pi_{ik} = 1, \tag{6.11}$$

$$\mathcal{J}(\theta) = \underbrace{\log p_\alpha(\mathbf{z})}_{\text{objective for LEBM}} + \underbrace{\sum_{i=1}^{D}\sum_{k=1}^{2} \gamma_{ik} \log \pi_{ik}}_{\text{foreground-background partitioning}} + \underbrace{\sum_{i=1}^{D}\sum_{k=1}^{2} \gamma_{ik} \log p_{\beta_k}(\mathbf{x}_i|\mathbf{z}_k)}_{\text{objective for image generation}}, \tag{6.12}$$

where $\mathcal{J}(\theta)$ is the expectation of the complete-data log-likelihood over $\mathbf{w}$, using responsibilities estimated at the current iteration and held fixed during the model-parameter update. This is an EM-style surrogate, rather than an instruction to differentiate through the parameter-dependent posterior. See Section 6.7 for details.

Eq. (6.11) has an intuitive interpretation. We can decompose the learning objective into three components as in Eq. (6.12). In particular, the second term $\sum_{i=1}^{D}\sum_{k=1}^{2} \gamma_{ik} \log \pi_{ik}$ has a similar form to the cross-entropy loss commonly used for supervised segmentation task, where the posterior responsibility γ_{ik} serves as the target distribution. In this formulation, the foreground and background generative models compete with each other to fit the distribution

of each pixel $\mathbf{x}_i$. If the pixel value at $\mathbf{x}_i$ fits better to the distribution of foreground, $p_{\beta_1}(\mathbf{x}_i|\mathbf{z}_1)$, than to that of background, $p_{\beta_2}(\mathbf{x}_i|\mathbf{z}_2)$, the model tends to assign that pixel to the foreground region (see Eq. (6.7)), and vice versa. This mechanism is similar to the process derived in Zhu and Yuille [1], which is the reason why we coin our method Deep Region Competition (DRC).

Prior to our proposal, several methods [1, 37, 38] also employ mixture models and competition among the components to perform unsupervised foreground or image segmentation. The original Region Competition [1] combines several families of image modeling with Bayesian inference but is limited by the expressiveness of the pre-specified probability distributions. More recent methods, including IODINE [37] and Slot-attention [38], learn amortized inference networks for latent variables and induce the independence of foreground and background representations using an identical generator. Our method combines the best of the two worlds, reconciling the expressiveness of learned generators with the regularity of generic texture modeling under the framework of LEBM.

To optimize the learning objective in Eq. (6.11), we approximate the expectation by sampling from the prior $p_\alpha(\mathbf{z})$ and posterior model $p_\theta(\mathbf{z}|\mathbf{x}) \propto p_\alpha(\mathbf{z})p_\beta(\mathbf{x}|\mathbf{z})$, followed by calculating the Monte Carlo average. We use Langevin dynamics [64] to draw persistent MCMC samples, which iterates

$$\mathbf{z}_{t+1} = \mathbf{z}_t + s\nabla_{\mathbf{z}} \log Q(\mathbf{z}_t) + \sqrt{2s}\epsilon_t, \tag{6.13}$$

where t is the Langevin dynamics time step, s the step size, and ϵ_t the Gaussian noise. $Q(\mathbf{z})$ is the target distribution, being either $p_\alpha(\mathbf{z})$ or $p_\theta(\mathbf{z}|\mathbf{x})$. $\nabla_{\mathbf{z}} \log Q(\mathbf{z}_t)$ is efficiently computed via automatic differentiation in modern learning libraries [68]. We summarize the above process in Algorithm 2.

During inference, we initialize the latent variables $\mathbf{z}$ for MCMC sampling from Gaussian white noise and run only the posterior sampling step to obtain $\mathbf{z}_+$. The inferred mask and region images are then given by the outputs of generative models $p_{\beta_k}(\mathbf{w}|\mathbf{z}_+)$ and $p_{\beta_k}(\mathbf{x}|\mathbf{z}_+),\ k = 1, 2$, respectively.

Algorithm 2: Learning models of DRC via EM.

Input: Learning iterations T, initial parameters for LEBMs $\alpha^{(0)} = \{\alpha_1^{(0)}, \alpha_2^{(0)}\}$ and generators $\beta^{(0)} = \{\beta_1^{(0)}, \beta_2^{(0)}\}$, $\theta^{(0)} = \{\alpha^{(0)}, \beta^{(0)}\}$, learning rate η_α for LEBMs, η_β for foreground and background generators, observed examples $\{\mathbf{x}^{(i)}\}_{i=1}^N$, batch size M, and initial latent variables $\{\mathbf{z}_-^{(i)} = \{\mathbf{z}_{1-}^{(i)}, \mathbf{z}_{2-}^{(i)}\} \sim p_0(\mathbf{z})\}_{i=1}^N$ and $\{\mathbf{z}_+^{(i)} = \{\mathbf{z}_{1+}^{(i)}, \mathbf{z}_{2+}^{(i)}\} \sim p_0(\mathbf{z})\}_{i=1}^N$.

Output: $\theta^{(T)} = \{\alpha_1^{(T)}, \beta_1^{(T)}, \alpha_2^{(T)}, \beta_2^{(T)}\}$.

for $t = 0 : T-1$ **do**

- Sample a minibatch of data $\{\mathbf{x}^{(i)}\}_{i=1}^M$;
- **Prior sampling for learning LEBMs:** For each $\mathbf{x}^{(i)}$, update $\mathbf{z}_-^{(i)}$ using Eq. (6.13), with target distribution $Q(\mathbf{z}) = p_{\alpha^{(t)}}(\mathbf{z})$;
- **Posterior sampling for foreground and background generation:** For each $\mathbf{x}^{(i)}$, update $\mathbf{z}_+^{(i)}$ using Eq. (6.13), with target distribution $Q(\mathbf{z}) = p_{\theta^{(t)}}(\mathbf{z}|\mathbf{x})$;
- **Update LEBMs:** $\alpha^{(t+1)} = \alpha^{(t)} + \eta_\alpha \frac{1}{M} \sum_{i=1}^M [\nabla_\alpha f_{\alpha^{(t)}}(\mathbf{z}_+^{(i)}) - \nabla_\alpha f_{\alpha^{(t)}}(\mathbf{z}_-^{(i)})]$;
- **Update foreground and background generators:** $\beta^{(t+1)} = \beta^{(t)} + \eta_\beta \frac{1}{M} \sum_{i=1}^M \nabla_\beta \log p_{\beta^{(t)}}(\mathbf{x}^{(i)}|\mathbf{z}_+^{(i)})$;

6.3.4 Technical Details

Pseudo-label regularization Although the proposed DRC explicitly models the interaction between the regions, it is still possible that the model converges to a trivial extractor, which treats the entire image as the foreground or background region, leaving the other region null. We exploit the symbolic vector $\mathbf{y}$ emitted by the LEBM (see Section 6.3.1) for additional regularization. The strategy is similar to the mutual information maximization used in InfoGAN [69]. Specifically, we use the symbolic vector $\mathbf{y}$ inferred from $\mathbf{z}$ as the pseudo-class label for $\mathbf{z}$ and train an auxiliary classifier jointly with the above models; it ensures that the generated regions $\mathbf{x}_k$ contain similar symbolic information for $\mathbf{z}_k$. Intuitively, this loss prevents the regions from converging to null since the symbolic representation $\mathbf{y}_k$ would never be well retrieved if that did happen.

Implementation We adopt a similar architecture for the generator as in DCGAN [70] throughout the experiments and only change the dimension of the latent variables $\mathbf{z}$ for different datasets. The generator consists of a fully connected layer followed by five stacked upsample-conv-norm layers. We replace the batch-norm layers [71] with instance-norm [72] in

the architecture. The energy-term in LEBM is parameterized by a 3-layered MLP. We adopt orthogonal initialization [73] commonly used in generative models to initialize the networks and orthogonal regularization [74] to facilitate training. In addition, we observe performance improvement when adding Total-Variation norm [75] for the background generative model. More details, along with specifics of the implementation used in our experiments, are provided in the additional analyses below.

6.4 Experiments

We design experiments to answer three questions: (1) How does the proposed method compare to previous state-of-the-art competitors? (2) How do the proposed components contribute to the model performance? (3) Does the proposed method exhibit generalization on images containing unseen instances (*i.e.*, same category but not the same instance) and even objects from novel categories?

To answer these questions, we evaluate our method on five challenging datasets in two groups: (1) Birds [76], Dogs [77], and Cars [78] datasets; (2) CLEVR6 [79] and TM-dSprites [80] datasets. The first group of datasets covers complex real-world domains, whereas the second group features environments of the multi-object foreground with challenging spatial configurations or confounding backgrounds. As to be shown, the proposed method is generic to various kinds of input and produces more competitive foreground-background partition results than prior methods.

6.4.1 Results on Foreground Extraction

Single object in the wild In the first group of datasets, there is typically a single object in the foreground, varying in shapes, texture, and lighting conditions. Unsupervised foreground extraction on these datasets requires much more sophisticated visual cues than colors and shapes. Birds dataset consists of 11,788 images of 200 classes of birds annotated

Model	Single Object						Multi-Object			
	Birds		Dogs		Cars		CLEVR6		TM-dSprites	
	IoU	Dice	IoU	Dice	IoU	Dice	IoU	Dice	IoU	Dice
W-Net*	24.8	38.9	47.7	62.1	52.8	67.6	-	-	-	-
GrabCut	30.2	42.7	58.3	70.9	61.3	73.1	19.0	30.5	61.9	71.0
ReDO§	46.5	60.2	55.7	70.3	52.5	68.6	18.6	31.0	9.4	17.2
OneGAN*†	55.5	69.2	71.0	81.7	71.2	82.6	-	-	-	-
IODINE§	30.9	44.6	54.4	67.0	51.7	67.3	19.9	32.4	7.3	12.8
Slot-Attn.§	35.6	51.5	38.6	55.3	41.3	58.3	83.6	90.7	7.3	13.5
Ours	**56.4**	**70.9**	**71.7**	**83.2**	**72.4**	**83.7**	**84.7**	**91.5**	**78.8**	**87.5**

Table 6.1: **Foreground extraction results on training data measured in IoU and Dice.** Higher is better in all scores. *Results of W-Net and OneGAN are provided by Benny and Wolf [31]. Of note, results of these two models on Dogs and Cars datasets may **not** be directly comparable to other listed methods, as the data used for training and evaluation could be different. We include these results as a rough reference since no official implementation or pretrained model are publicly available. § indicates unfair baseline results obtained using extra ground-truth information, *i.e.*, we choose the best-matching scores from the permutation of foreground and background masks. †OneGAN is a strong **weakly supervised** baseline, which requires clean background images to provide additional supervision. We include this model as a potential upper bound of the fully unsupervised methods.

with high-quality segmentation masks, Dogs dataset consists of 20,580 images of 120 classes annotated with bounding boxes, and Cars dataset consists of 16,185 images of 196 classes. The latter two datasets are primarily made for fine-grained categorization. To evaluate foreground extraction, we follow the practice in Benny and Wolf [31], and approximate ground-truth masks for the images with Mask R-CNN [16], pre-trained on the MS COCO [9] dataset with a ResNet-101 [81] backend. The pre-trained model is acquired from the detectron2 [82] toolkit. This results in 5,024 dog images and 12,322 car images with a clear foreground-background setup and corresponding masks.

On datasets featuring a single foreground object, we use the 2-slot version of IODINE and Slot-attention. Since ReDO, IODINE, and Slot-Attention do not distinguish foreground and background in output regions, we choose the best-matching scores from the permutation of foreground and background masks as in [28]. We observe that the proposed method and Grabcut are the only two methods that provide explicit identification of foreground objects and background regions. While the Grabcut algorithm actually requires a predefined bounding

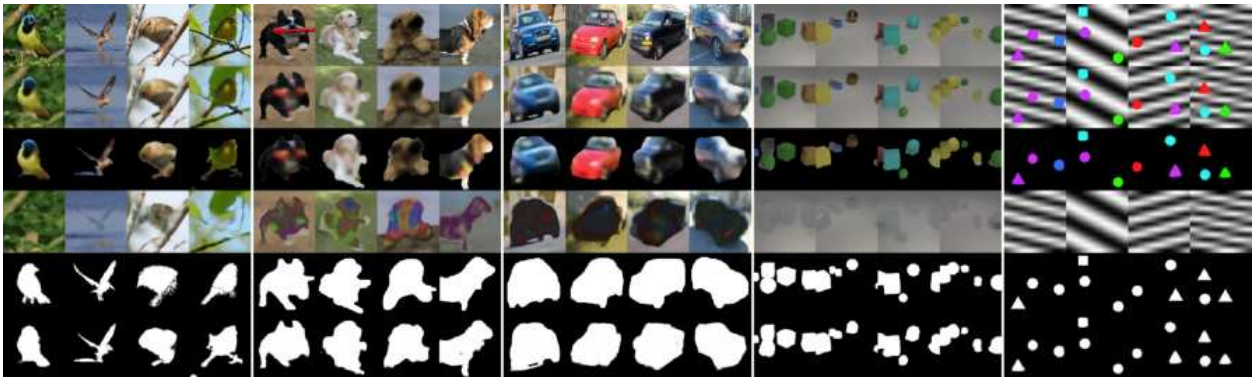


Figure 6.3: **Foreground extraction results for each dataset**; zoom in for better visibility. From top to bottom: (i) observed images, (ii) generated images, (iii) masked generated foregrounds, (iv) generated backgrounds, (v) ground-truth foreground masks, and (vi) inferred foreground masks. More samples and results of baselines can be found in the additional analyses below.

box as input that specifies the foreground region, our method, thanks to the learned pixel re-assignment (see Section 6.3.2), can achieve this in a fully unsupervised manner. Results in Table 6.1 show that our method outperforms all the unsupervised baselines by a large margin, exhibiting comparable performance even to the weakly supervised baseline that requires additional background information as inputs [31]. We provide samples of foreground extraction results as well as generated background and foreground regions in Fig. 6.3. Note that our final goal is not to synthesize appealing images but to learn foreground extractors in a fully unsupervised manner. As the limitation of our method, DRC generates foreground and background regions less realistic than those generated by state-of-the-art GANs, which hints a possible direction for future work. More detailed discussions of the limitation can be found in the additional analyses below.

Multi-object scenes The second group of datasets contains images with possibly simpler foreground objects but more challenging scene configurations or background parts. Visual scenes in the CLEVR6 dataset contain various objects and often with partial occlusions and truncations. Following the evaluation protocol in IODINE and Slot-attention, we use the first 70K samples from CLEVR [79] and filter the samples for scenes with at most 6 objects for training and evaluation, *i.e.*, CLEVR6. The TM-dSprites dataset is a variant of Multi-dSprites [80] but has strongly confounding backgrounds borrowed from Textured MNIST [32]. We generate 20K samples for the experiments. Similar to Greff et al. [37] and

Locatello et al. [38], we evaluate on a subset containing 1K samples for testing. Note that IODINE and Slot-attention are designed for segmenting complex multi-object scenes using slot-based object representations. Ideally, the output of these models consists of masks for each individual object, while the background is viewed as a virtual "object" as well. In practice, however, it is possible that the model distributes the background over all the slots as mentioned in Locatello et al. [38]. We therefore propose two corresponding approaches (see Section 6.7 for details) to convert the output object masks into a foreground-background partition and report the best results of these two options for IODINE and Slot-attention in Table 6.1.

On the CLEVR6 dataset, we use the publicly available pretrained model for IODINE, which achieves a reasonable ARI (excluding background pixels) of 94.4 on the testing data, close to the testing results in Greff et al. [37]. We observe that IODINE distributes the background over all the slots for some of the testing samples, resulting in much lower IoU and Dice scores. We re-train the Slot-attention model using the official implementation on CLEVR6, as no pretrained model is publicly available. The re-trained model achieves a foreground ARI of 98.0 on the 1K testing samples, which we consider as a sign of valid re-implementation. Results in Table 6.1 demonstrate that the proposed method can effectively process images of challenging multi-object scenes. To be specific, our method demonstrates competitive performance on the CLEVR6 dataset compared with the SOTA object discovery method. Moreover, as shown empirically in Fig. 6.3, the proposed method can handle the strongly confounding background introduced in Greff et al. [32], whereas previous methods are distracted by the background and mostly fail to capture the foreground objects.

6.4.2 Ablation Study

We provide ablation studies on the Birds dataset to inspect the contribution of each proposed component in our model. As shown in Table 6.2, we observe that replacing the LEBMs in the foreground and background models with amortized inference networks significantly harms

Model	IoU	Dice
amortized inference*	-	-
w/o pix. re-assign.	21.8	35.3
w/o pseudo label	48.7	64.2
w/o TV-norm reg.	53.0	68.1
w/o ortho. reg.	54.3	69.2
short-run chain†	52.5	67.7
Full model	56.4	70.9

Table 6.2: **Ablation study on Birds.** *We replace the LEBM with encoders to perform amortized inference for the latent variables $\mathbf{z}$ within a variational framework as in VAE [83]. †We explore the possibility of using short-run MCMC [84] instead of persistent chain sampling.

the performance of the proposed method. In particular, the modified model fails to generate any meaningful results (marked as - in Table 6.2). We conjecture that LEBM benefits from the low-dimensionality of the latent space [47] and therefore enjoys more efficient learning. However, the inference networks need to learn an extra mapping from the high-dimensional image space to the latent space and require more elaborate architecture and tuning for convergence. Furthermore, we observe that the model that does not learn pixel re-assignment for background can still generate meaningful images but only vaguely captures masks for foreground extraction.

6.4.3 Generalizable Foreground Extraction

Extracting novel foreground objects from the training categories We show results on generalizing to novel objects from the training classes. To evaluate our method, we split the Birds dataset following Chen et al. [28], resulting in 10K training images and 1K testing images. On Dogs and Cars datasets, we split the dataset based on the original train-test split [77, 78]. This split gives 3,286 dog images and 6,218 car images for training, and 1,738 dog images and 6,104 car images for testing, respectively. As summarized in Table 6.3, our method shows superior performances compared with baselines; the performance gap between training and testing is constantly small over all datasets.

Extracting novel foreground objects from unseen categories To investigate how well our method generalizes to categories unseen during training, we evaluate the

Model	Split	Birds		Dogs		Cars	
		IoU	Dice	IoU	Dice	IoU	Dice
GrabCut*	Train	30.2	42.7	58.3	70.8	60.9	72.7
	Test	30.3	42.8	57.9	70.5	61.6	73.5
ReDO	Train	46.8	61.4	54.3	69.2	52.6	68.7
	Test	47.1	61.7	52.8	67.9	52.5	68.6
Ours	Train	**54.8**	**69.5**	**71.6**	**83.2**	**71.9**	**83.3**
	Test	**54.6**	**69.4**	**72.3**	**83.6**	**70.8**	**82.5**

Table 6.3: **Performance of DRC on training and held-out testing data.** *GrabCut is a deterministic method that does not require training. We report the results of GrabCut [5] on these splits only for reference.

Test	Train	GrabCut		ReDO		Ours	
		IoU	Dice	IoU	Dice	IoU	Dice
Birds	Birds*	30.3	42.8	47.1	61.7	54.6	69.4
	Dogs			22.2	35.3	**41.3**	**57.4**
	Cars			16.4	27.7	39.2	55.3
Dogs	Dogs*	57.9	70.5	52.8	67.9	72.3	83.6
	Cars			44.5	61.2	**67.8**	**80.4**
	Birds			44.0	60.3	53.6	69.1
Cars	Cars*	61.6	73.5	52.5	68.6	70.8	82.5
	Dogs			51.6	67.1	**68.6**	**81.0**
	Birds			41.8	58.6	45.1	61.7

Table 6.4: **Performance of DRC on unseen testing categories.** *We include the testing results of models trained with data from the same categories for reference.

models trained in real-world single object datasets on the held-out testing data from different categories. We use the same training and testing splits on these datasets as in the previous experiments. Table 6.4 shows that our method outperforms the baselines on the Birds dataset when the model has trained on Dogs or Cars dataset, which have quite different distributions in foreground object shapes. Competitors like ReDO also exhibit such out-of-distribution generalization but only to a limited extent. Similar results are observed when using Dogs or Cars as the testing set. We can see that when the model is trained on Dogs and evaluated on Cars or vice versa, it still maintains comparable performances w.r.t. those are trained&tested on the same class. We hypothesize that these two datasets have similar distributions in foreground objects and background regions. In the light of this, we can further entail that the distribution of Dogs is most similar to that of Cars and less similar to that of Birds according to the results, which is consistent with our intuitive observation of the data. We provide a preliminary analysis of the statistics of these datasets in the additional analyses below.

6.5 Conclusion

We have presented the Deep Region Competition, an EM-based fully unsupervised foreground extraction algorithm fueled by energy-based prior and generative image modeling. We propose learned pixel re-assignment as an inductive bias to capture the background regularities. Experiments demonstrate that DRC exhibits more competitive performances on complex real-world data and challenging multi-object scenes. We show empirically that learned pixel re-assignment helps to provide explicit identification for foreground and background regions. Moreover, we find that DRC can potentially generalize to novel foreground objects even from categories unseen during training. We hope our work will inspire future research along this challenging but rewarding research direction.

6.6 Acknowledgments for This Research Chapter

This chapter is a version of the coauthored publication: Peiyu Yu, Sirui Xie, Xiaojian Ma, Yixin Zhu, Ying Nian Wu, and Song-Chun Zhu, "Unsupervised Foreground Extraction via Deep Region Competition," *Advances in Neural Information Processing Systems*, vol. 34 (NeurIPS 2021), 2021. NeurIPS proceedings.

Peiyu Yu was responsible for conceiving the method, designing and conducting the experiments, and writing the manuscript. Sirui Xie, Xiaojian Ma, Yixin Zhu, Ying Nian Wu, and Song-Chun Zhu participated in research discussions.

This research was supported by NSF DMS-2015577, ONR MURI project N00014-16-1-2007, and DARPA XAI project N66001-17-2-4029. We thank Bo Pang for insights on the latent-space energy-based model and the anonymous reviewers for their constructive comments.

6.7 Additional Analyses and Experimental Details

6.7.1 Dataset Details

6.7.1.1 Caltech-UCSD Birds-200-2011

Birds dataset consists of 11,788 images of 200 classes of birds annotated with high-quality segmentation masks. Each image is further annotated with 15 part locations, 312 binary attributes, and 1 bounding box. We use the provided bounding box to extract a center square from the image, and scale it to 128×128 pixels. Each scene contains exactly one foreground object.

6.7.1.2 Stanford Dogs

Dogs dataset consists of 20,580 images of 120 classes annotated with bounding boxes. We first use the provided bounding box to extract the center square, and then scale it to 128×128 pixels. As described in the main experiments, we approximate ground-truth masks for the pre-processed images with Mask R-CNN [16], pre-trained on the MS COCO [9] dataset with a ResNet-101 [81] backend. The pretrained model is acquired from the detectron2 [82] toolkit. We exclude the images where no dog is detected. We then manually exclude those images where the foreground object has occupied more than $\sim 90\%$ of the image, those with poor masks, and those with significant foreground distractors such as humans (see Fig. 6.4). The filtering strategy results in 5,024 images with a clear foreground-background setup and high-quality mask.

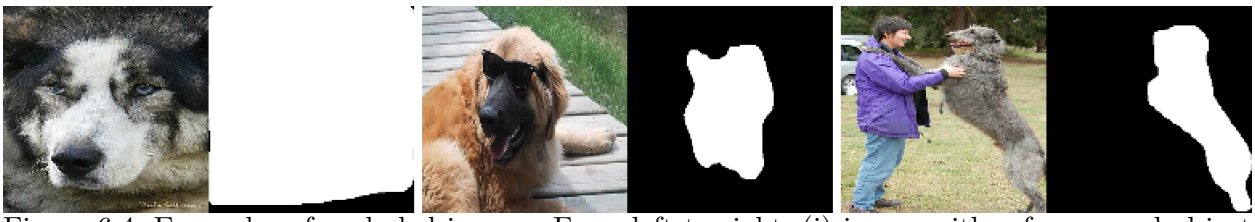


Figure 6.4: Examples of excluded images. From left to right: (i) image with a foreground object that occupied too much space, (ii) image with a low-quality mask, and (iii) image with significant foreground distractors.

6.7.1.3 Stanford Cars

Cars dataset consists of 16,185 images of 196 classes annotated with bounding boxes. Though also being primarily designed for fine-grained categorization, it has a much clearer foreground-background setup compared with the Dogs dataset. We employ a similar process as used for Dogs dataset to approximate the ground-truth masks, and only exclude those images where cars are not properly detected. It finally produces 12,322 images for our experiments.

6.7.1.4 CLEVR6

CLEVR6 dataset is a subset of the original CLEVR dataset [79] with masks, generated by Greff et al. [37]. We follow the evaluation protocol adopted by IODINE [37] and Slot-attention [38], which takes the first 70K samples from CLEVR. These samples are then filtered to only include scenes with at most 6 objects. Additionally, we perform a center square crop of 192×192 from the original 240×320 image, and scale it to 128×128 pixels. The resulting CLEVR6 dataset contains 3-6 foreground objects that could be with partial occlusion and truncation in each visual scene.

6.7.1.5 Textured Multi-dSprites

TM-dSprites dataset, which is based on the dSprites dataset [80] and Textured MNIST [32], consists of 20,000 images with a resolution of 128×128. Each image contains 2-3 random sprites, which vary in terms of shape (square, circle, or triangle), color (uniform saturated colors), and position (continuous). The background regions are borrowed from Textured MNIST dataset [32]. The textures for the background are randomly shifted samples from a bank of 20 sinusoidal textures with different frequencies and orientations. We adopt a simpler foreground setting compared with the vanilla Multi-dSprites dataset used in [37], *i.e.*, the foreground objects are not occluded as the dataset is designed to emphasize the background part. Some samples are presented in Fig. 6.5.

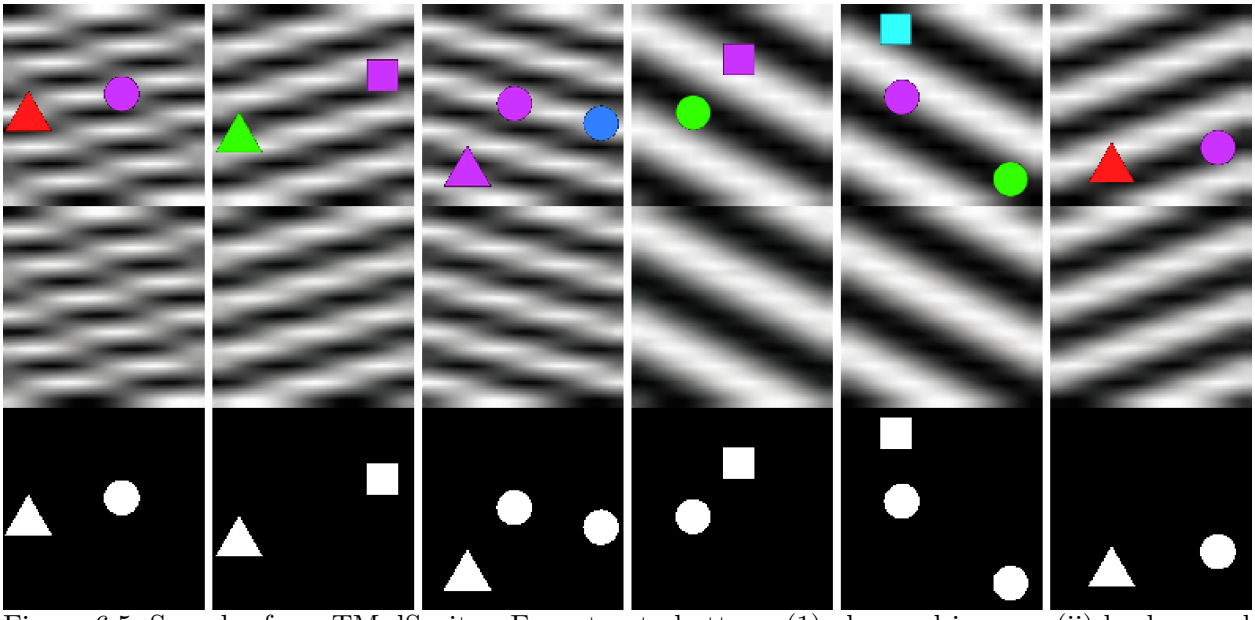


Figure 6.5: Samples from TM-dSprites. From top to bottom: (1) observed images, (ii) background textures, and (iii) ground-truth masks.

6.7.2 Details on Models and Hyperparameters

Architecture As described in the main text, we use the same overall architecture for different datasets (while the size of latent variables may vary). The details for the generators and LEBMs are summarized in the Table 6.5 and Table 6.6.

Dataset	Foreground	Background	Pixel Re-assignment
Birds	256	256	512
Dogs	256	256	512
Cars	256	192	512
CLEVR6	256	2	256
TM-dSprites	256	4	1024

Table 6.5: Dimension of latent variables on each dataset.

Hyperparameters and Training Details For the Langevin dynamics sampling [64], we use K_0 and K_1 to denote the number of prior and posterior sampling steps with step sizes s_0 and s_1 respectively. Our hyperparameter choices are: $K_0 = 60, K_1 = 40, s_0 = 0.4$ and $s_1 = 0.1$. These are identical across different datasets. During testing, we set the posterior sampling steps to 300 for Dogs and Cars, and 2.5K, 5K and 5K for Birds, CLEVR6 and TM-dSprites respectively. The parameters of the generators and LEBMs are initialized with orthogonal initialization [73]. The gain is set to 1.0 for all the models. We use the ADAM

Layers	In-Out size	Comment
LEBM for Foreground/Background Models		
Input: $\mathbf{z}$	D^*	
Linear, LReLU	200	
Linear, LReLU	200	
Linear	$K^\dagger$	
LEBM for Pixel Re-assignment Model		
Input: $\mathbf{z}$	D^*	
Linear, LReLU	200	
Linear, LReLU	200	
Linear, LReLU	200	
Linear	1	
Generator for Foreground/Background Model and Re-assignment Model		
Input: $\mathbf{z}$	D^*	
Linear, LReLU	$4 \times 4 \times 128$	reshaped output
UpConv3x3Norm, LReLU	$8 \times 8 \times 1024$	stride 1 & padding 1
UpConv3x3Norm, LReLU	$16 \times 16 \times 512$	stride 1 & padding 1
UpConv3x3Norm, LReLU	$32 \times 32 \times 256$	stride 1 & padding 1
UpConv3x3Norm, LReLU	$64 \times 64 \times 128$	stride 1 & padding 1
UpConv3x3Norm, LReLU	$128 \times 128 \times 64$	stride 1 & padding 1
Conv3x3	$128 \times 128 \times (3+1)$	RGB & Mask
	$128 \times 128 \times 2$	Re-assignment grid
Auxiliary classifier for Foreground/Background Model		
Input: $\mathbf{x}$	$128 \times 128 \times 3$	generated image
Conv4x4Norm, LReLU	$64 \times 64 \times 64$	stride 2 & padding 1
Conv4x4Norm, LReLU	$32 \times 32 \times 128$	stride 2 & padding 1
Conv4x4Norm, LReLU	$16 \times 16 \times 256$	stride 2 & padding 1
Conv4x4Norm, LReLU	$8 \times 8 \times 512$	stride 2 & padding 1
Conv4x4Norm, LReLU	$4 \times 4 \times 1024$	stride 2 & padding 1
Conv4x4	$1 \times 1 \times K^\dagger$	

Table 6.6: Architecture of the generators, LEBMs and auxiliary classifiers (see Section 6.7.3.2). UpConv3x3Norm denotes an Upsampling-Convolutional-InstanceNorm layer with a convolution kernel size of 3. Similarly, Conv4x4Norm denotes a Convolutional-InstanceNorm layer with a kernel size of 4. LReLU denotes the Leaky-ReLU activation function. The leak factor for LReLU is 0.2 in LEBMs and auxiliary classifiers, and 0.01 in generators. $^*D$ represents the dimensions of the latent variables for different datasets; see Table 6.5. $\dagger K$ represents the pre-specified category number for latent variables. We use 200 for both the foreground and background LEBMs on real-world datasets, and 30 and 10 in the foreground and background LEBMs on multi-object datasets respectively.

optimizer [85] with $\beta_1 = 0.5$ and $\beta_2 = 0.999$. Generators are trained with a constant learning rate of 0.0001, and LEBMs with 0.00002. We run experiments on a single V100 GPU with 16GB of RAM and with a batch size of 48. We set the maximum training iterations to 10K and run for at most 48hrs for each dataset.

6.7.3 Details on Learning Objective and Regularization

6.7.3.1 Learning Objective

Derivation of Surrogate Learning Objective $\mathcal{J}(\theta) = \mathbf{E}_{\mathbf{w}\sim p_\beta(\mathbf{w}|\mathbf{x},\mathbf{z})}\left[\mathcal{L}(\theta)\right]$ is the conditional expectation of $\mathbf{w}$,

$$\begin{aligned}\mathcal{J}(\theta) &= \mathbf{E}_{\mathbf{w}\sim p_\beta(\mathbf{w}|\mathbf{x},\mathbf{z})}\left[\mathcal{L}(\theta)\right] \\ &= \log p_\alpha(\mathbf{z}) + \mathbf{E}\left[\sum_{i=1}^{D}\sum_{k=1}^{2} w_{ik}\left(\log \pi_{ik} + \log p_{\beta_k}(\mathbf{x}_i|\mathbf{z}_k)\right)\right] \\ &= \log p_\alpha(\mathbf{z}) + \sum_{i=1}^{D}\sum_{k=1}^{2} \mathbf{E}\left[w_{ik}\right]\left(\log \pi_{ik} + \log p_{\beta_k}(\mathbf{x}_i|\mathbf{z}_k)\right),\end{aligned} \tag{6.14}$$

where $\mathbf{E}$ is the conditional expectation of $\mathbf{w}$. Recall that $w_{ik} \in \{0, 1\}$. The expectation becomes

$$\begin{aligned}\mathbf{E}\left[w_{ik}\right] &= 0 \times p(w_{ik} = 0|\mathbf{x}_i, \mathbf{z}) + 1 \times p(w_{ik} = 1|\mathbf{x}_i, \mathbf{z}) \\ &= \gamma_{ik},\end{aligned} \tag{6.15}$$

which is the posterior responsibility of w_{ik}. We can further decompose $\mathcal{J}(\theta)$ into

$$\mathcal{J}(\theta) = \underbrace{\log p_\alpha(\mathbf{z})}_{\text{objective for LEBM}} + \underbrace{\sum_{i=1}^{D}\sum_{k=1}^{2} \gamma_{ik} \log \pi_{ik}}_{\text{foreground-background partitioning}} + \underbrace{\sum_{i=1}^{D}\sum_{k=1}^{2} \gamma_{ik} \log p_{\beta_k}(\mathbf{x}_i|\mathbf{z}_k)}_{\text{objective for image generation}}, \tag{6.16}$$

as explained above.

Understanding the Optimization Process Note that the surrogate learning objective is an expectation w.r.t $\mathbf{z}$,

$$\max_{\theta} \ \mathbf{E}_{\mathbf{z}\sim p_{\theta(t)}(\mathbf{z}|\mathbf{x})}\left[\mathcal{J}(\theta)\right], \ \text{s.t. } \forall i, \sum_{k=1}^{2}\pi_{ik}=1, \tag{6.17}$$

which is generally intractable to calculate. We therefore need to approximate the expectation by sampling from the distributions, and calculating the Monte Carlo average. In practice, this can be done by gradient-based MCMC sampling method, such as Langevin Dynamics [64].

Given $\mathbf{x}$, we have $p_\theta(\mathbf{z}|\mathbf{x})\propto p_\beta(\mathbf{x}|\mathbf{z})p_\alpha(\mathbf{z})$. Note that

$$\begin{aligned}\nabla_{\mathbf{z}}\log p_\beta(\mathbf{x}|\mathbf{z}) &= \frac{1}{p_\beta(\mathbf{x}|\mathbf{z})}\nabla_{\mathbf{z}}p_\beta(\mathbf{x}|\mathbf{z})\\ &= \int_{\mathbf{w}} p_\beta(\mathbf{w}|\mathbf{x},\mathbf{z})\nabla_{\mathbf{z}}\log p_\beta(\mathbf{x},\mathbf{w}|\mathbf{z})d\mathbf{w}\\ &= \mathbf{E}_{\mathbf{w}\sim p_\beta(\mathbf{w}|\mathbf{x},\mathbf{z})}\left[\nabla_{\mathbf{z}}\log p_\beta(\mathbf{x},\mathbf{w}|\mathbf{z})\right].\end{aligned} \tag{6.18}$$

Therefore, An auxiliary expression used to compute the Langevin gradient at step t is

$$\begin{aligned}\log\tilde{Q}(\mathbf{z}_t) &= \log p_\alpha(\mathbf{z}_t) + \mathbf{E}_{\mathbf{w}\sim p_\beta(\mathbf{w}|\mathbf{x},\mathbf{z}_t)}\left[\sum_{i=1}^{D}\sum_{k=1}^{2} w_{ik}\left(\log\pi_{ik}+\log p_{\beta_k}(\mathbf{x}_i|\mathbf{z}_{k,t})\right)\right]\\ &= \log p_\alpha(\mathbf{z}_t) + \sum_{i=1}^{D}\sum_{k=1}^{2}\gamma_{ik,t}\left(\log\pi_{ik}+\log p_{\beta_k}(\mathbf{x}_i|\mathbf{z}_{k,t})\right),\end{aligned} \tag{6.19}$$

which has the same form as $\mathcal{J}(\theta)$. In evaluating its gradient with respect to $\mathbf{z}$, the responsibilities $\gamma_{ik,t}$ must be held fixed at their current values so that the resulting gradient agrees with the marginal log-likelihood score. Unlike the model-parameter update, Langevin dynamics updates the latent variables $\mathbf{z}$.

The two-step learning process of the DRC models can be understood as follows: (1) in the first step, the algorithm optimizes $\mathcal{J}$ by updating latent variables $\mathbf{z}$, where the posterior responsibility γ_{ik} inferred at each step serves to gradually disentangle the foreground and background components, and (2) in the second step, the updated $\mathbf{z}$ is fed again into the

models to generate the observation $\mathbf{x}$, where the algorithm optimizes $\mathcal{J}$ by updating the model parameters θ.

It is worth mentioning that learning LEBMs requires an extra sampling step [47], as the gradients are given by the following

$$\delta_\alpha(\mathbf{x}) = \mathbf{E}_{p_\theta(\mathbf{z}|\mathbf{x})}\left[\nabla_\alpha f_\alpha(\mathbf{z})\right] - \mathbf{E}_{p_\alpha(\mathbf{z})}\left[\nabla_\alpha f_\alpha(\mathbf{z})\right], \tag{6.20}$$

where the second terms should be computed by sampling with $p_\alpha(\mathbf{z})$. We term this as *prior sampling* in the main text.

Further Details on the Loss Functions For the generative models $p_{\beta_k}(\mathbf{x}|\mathbf{z}_k),\ k = 1, 2$, we assume that $\mathbf{x} = g_{\beta_k}(\mathbf{z}_k) + \epsilon$, where $g_{\beta_k}(\mathbf{z}_k),\ k = 1, 2$ are the generator networks for foreground and background regions, and ϵ is random noise sampled from a zero-mean Gaussian or Laplace distribution. Assuming a global fixed variance σ^2 for Gaussian, we have $\log p_{\beta_k}(\mathbf{x}|\mathbf{z}_k) = -\frac{1}{2\sigma^2}\|g_{\beta_k}(\mathbf{z}_k) - \mathbf{x}\|^2 + C,\ k = 1, 2$, where C is a constant unrelated to β_k and $\mathbf{z}_k$. Similarly for Laplace distribution, we have $\log p_{\beta_k}(\mathbf{x}|\mathbf{z}_k) = -\frac{1}{\lambda}|g_{\beta_k}(\mathbf{z}_k) - \mathbf{x}| + C,\ k = 1, 2$. These two log-likelihoods correspond to the MSE loss and L1 loss commonly used for image reconstruction, respectively.

6.7.3.2 Regularization

Pseudo Label Learning As described in the main text, we exploit the symbolic vector $\mathbf{y}$ emitted by the LEBM for additional regularization. Let the target distribution of $\mathbf{y}_k$ be P_k given by $p_{\alpha_k}(\mathbf{y}|\mathbf{z}_k),\ k = 1, 2$, which represents the distribution of symbolic vector for foreground and background regions respectively. We can optimize the following objective as a regularization to our original learning objective:

$$\min_{\beta,\tau}\ \mathcal{L}_{\text{pseudo-label}} = \sum_{k=1}^{2} H(P_k, Q_k), \tag{6.21}$$

$$H(P_k, Q_k) = -\langle p_{\alpha_k}(\mathbf{y}|\mathbf{z}_k), \log q_{\tau_k}(\mathbf{y}|g_{\beta_k}(\mathbf{z}_k))\rangle,\ k = 1, 2, \tag{6.22}$$

where $q_{\tau_k},\ k = 1, 2$ represents the jointly trained auxiliary classifier network (see Section 6.7.2 for architecture details) for foreground and background. $g_{\beta_k}(\mathbf{z}_k),\ k = 1, 2$ represents the output of generator network. We set the weight of this regularization term to 0.1 for all the models.

Total Variation norm (TV-norm) Total Variation norm [75] is commonly used for image denoising, and has been extended as an effective technique for in-painting. We use TV-norm as a regularization for learning the background generator.

$$\min_{\beta_2}\ \mathcal{L}_{\text{TV-norm}} = \sum_{h,w}\left(|\frac{\partial g_{\beta_2}(\mathbf{z}_2)}{\partial x}(h, w)| + |\frac{\partial g_{\beta_2}(\mathbf{z}_2)}{\partial y}(h, w)|\right), \tag{6.23}$$

where $\partial_x g_{\beta_2}(\mathbf{z}_2)(h, w)$ and $\partial_y g_{\beta_2}(\mathbf{z}_2)(h, w)$ represent the horizontal and vertical image gradients at the pixel coordinate (h, w) respectively. We set the weight of this regularization term to 0.01 for all the models.

Orthogonal Regularization We use orthogonal regularization [74] for the convolutional layers only. Let $\mathbf{W}$ be the flattened kernel weights of the convolutional layers, *i.e.*, the size of $\mathbf{W}$ is $C \times K$ where C is the output channel number. The orthogonal regularization is calculated according to

$$\min_{\beta}\ \mathcal{L}_{\text{orthogonal-reg}} = \|\mathbf{W}\mathbf{W}^T \odot (\mathbf{1} - \mathbf{I})\|_F, \tag{6.24}$$

where $\odot$ is the Hadamard product. $\mathbf{I}$ denotes the identity matrix, and $\mathbf{1}$ denotes the matrix filled with ones. We set the weight of this regularization term to 0.1 for Birds models, and 1.0 for the rest of the models.

6.7.4 PyTorch-style Code

The following PyTorch-style excerpts illustrate the main operations of the model. They omit surrounding module definitions, imports, and optimizer initialization, and are not intended to be a standalone executable program. The L1 reconstruction term is used here as a weighted loss; it should not be interpreted as the Gaussian log-likelihood without adjusting the noise model.

Forward Pass In the forward pass, the model takes latent variables $\mathbf{z}$, generates foreground and background regions separately, and mixes them into the final image. Note that the pixel re-assignment is applied to both background image and mask. We find it useful to feed the intermediate feature of background region into the generator for pixel re-assignment.

Listing 6.1: Forward pass of the DRC model.

```
def forward(z):
    zf = z[:, :ZF_DIM]
    zb = z[:, ZF_DIM:-ZS_DIM]
    zs = z[:, -ZS_DIM:]

    ### generating foreground
    fg, fm = fg_net(zf)

    ### generating background
    bg, bm, bg_feat = bg_net(zb)
    shuffling_grid = sp_net(zs, bg_feat.detach())
    bg_shuf = F.grid_sample(bg, shuffling_grid)
    bm_shuf = F.grid_sample(bm, shuffling_grid)

    ### generating foreground masks
    pi = torch.cat([fm, bm_shuf], dim=1).softmax(dim=1)
    pi_f, pi_b = pi[:,:1,...], pi[:,1:,...]

    ### mixing regions
    im_p = fg * pi_f + bg_shuf * pi_b
    return im_p, fg, bg_shuf, pi_f, pi_b, bg
```

Sampling Latent Variables We employ Langevin Dynamics for sampling latent variables, which iteratively updates the sample with the gradient computed against the likelihood. In the following code, `ebm_net` stands for the LEBMs, which outputs the energy and the distribution parameters of the symbolic vector $\mathbf{y}$ for the foreground and background regions.

Listing 6.2: Running Langevin Dynamics for prior and posterior sampling of the latent variables **z**.

```
def sample_langevin_prior(z):
    ### langevin prior inference
    # only latent variables 'z' are updated
    for __ in range(infer_step_K0):
        z = z.requires_grad_(True)

        en, __, __ = ebm_net(z)
        e_log_lkhd = en.sum() + .5 * z.square().sum()

        d_ebm_z = torch.autograd.grad(e_log_lkhd, z)[0]
        z = z - 0.5 * (self.delta_0 ** 2) * d_ebm_z \
                    + self.delta_0 * torch.randn_like(z)
    return z

def sample_langevin_posterior(z, im_t):
    ### langevin posterior inference
    # only latent variables 'z' are updated
    for __ in range(infer_step):
        z = z.requires_grad_(True)

        im_p, fg, bg_shuf, pi_f, pi_b, bg = forward(z)

        ### log-lkhd for LEBMs
        en, __, __ = ebm_net(z)

        ### log-lkhd for generators
        log_pf = (-F.l1_loss(fg, im_t, reduction='none')
                  / (2. * SIGMA ** 2))
        log_pb = (-F.l1_loss(bg_shuf, im_t, reduction='none')
                  / (2. * SIGMA ** 2))

        # posterior responsibility
        with torch.no_grad():
            ga_f = (pi_f * log_pf.exp() /
                (pi_f * log_pf.exp()
                 + pi_b * log_pb.exp() + 1e-8))
        # objective for image generation
        e_z_log_p = ga_f.detach() * \
                    ((pi_f + 1e-8).log() + log_pf) \
                  + (1. - ga_f.detach()) * \
                    ((pi_b + 1e-8).log() + log_pb)
        # regularization
        tv_norm = tv_loss(bg_shuf)

        j_log_lkhd = (-e_z_log_p.sum() + tv_norm * .01
                    + en.sum() + .5 * z.square().sum())

        d_j_z = torch.autograd.grad(j_log_lkhd, z)[0]
        z = z - 0.5 * (self.delta_1 ** 2) * d_j_z \
               + self.delta_1 * torch.randn_like(z)
        z = z.detach()
    return z
```

Updating Model Parameters Given the sampled latent variables $\mathbf{z}$, we optimize the model parameters by minimizing the reconstruction error.

Listing 6.3: Updating model parameters.

```
def update_G(im_t, fg, bg_shuf, pi_f, pi_b, bg,
             zf_logits, zb_logits):
    ### optimizers for generator networks
    fg_net_optimizer.zero_grad()
    bg_net_optimizer.zero_grad()
    sp_net_optimizer.zero_grad()
    ### optimizers for auxiliary classifiers
    fc_net_optimizer.zero_grad()
    bc_net_optimizer.zero_grad()

    ### Regularizations
    # Pseudo-label for additional regularization
    f_logits = fc_net(fg)
    b_logits = bc_net(bg)
    hpq_f = cross_ent(zf_logits, f_logits)
    hpq_b = cross_ent(zb_logits, b_logits)
    # orthogonal regularizations
    ortho_reg = orthogonal_reg(fg_net) + \
                    orthogonal_reg(bg_net)
    # TV-norm
    tv_norm = tv_loss(bg_shuf)

    ### log-lkhd for generators
    log_pf = (-F.l1_loss(fg, im_t, reduction='none')
              / (2. * SIGMA ** 2))
    log_pb = (-F.l1_loss(bg_shuf, im_t, reduction='none')
              / (2. * SIGMA ** 2))

    # posterior responsibility
    with torch.no_grad():
        ga_f = (pi_f * log_pf.exp() /
                (pi_f * log_pf.exp() + pi_b * log_pb.exp() + 1e-8))
    # objective for image generation
    e_z_log_p = ga_f.detach() * ((pi_f + 1e-8).log() + log_pf) \
       + (1. - ga_f.detach()) * ((pi_b + 1e-8).log() + log_pb)

    G_loss = - e_z_log_p.mean() + tv_norm * .01 + \
             hpq_f * .1 + hpq_b * .1 + ortho_reg * 1

    G_loss.backward()
    fg_net_optimizer.step()
    bg_net_optimizer.step()
    sp_net_optimizer.step()

    fc_net_optimizer.step()
    bc_net_optimizer.step()

def update_E(en_pos, en_neg):
```

```
    ebm_optimizer.zero_grad()

    ebm_loss = en_pos.mean() - en_neg.mean()
    ebm_loss.backward()

    ebm_optimizer.step()

def learn(zp, zn, im_t):
    ### 1. Sampling latent variables
    zp = sample_langevin_posterior(zp, im_t)
    zn = sample_langevin_prior(zn)

    ### 2. Updating the parameters
    en_pos, zpf_logits, zpb_logits = ebm_net(zp)
    en_neg, znf_logits, znb_logits = ebm_net(zn)
    # update LEBMs
    update_E(en_pos, en_neg)

    im_p, fg, bg_shuf, pi_f, pi_b, bg = forward(zp)
    # update the generators
    update_G(im_t, fg, bg_shuf, pi_f, pi_b, bg,
             zpf_logits, zpb_logits)
```

6.7.5 Evaluation Protocols

Intersection over Union (IoU) The IoU score measures the overlap of two regions A and B by calculating the ratio of intersection over union, according to

$$\text{IoU}(A, B) = \frac{|A \cap B|}{|A \cup B|}, \tag{6.25}$$

where we use the inferred mask and ground-truth mask as A and B respectively for evaluation.

Dice (F1) score Similarly, the Dice (F1) score is

$$\text{Dice}(A, B) = \frac{2|A \cap B|}{|A| + |B|}. \tag{6.26}$$

Higher is better for both scores.

Evaluation As described in the main text, IODINE [37] and Slot-attention [38] are designed for segmenting complex multi-object scenes using slot-based object representations. Ideally, the output of these models consists of masks for each individual object, while the

background is viewed as a virtual “object” as well. In practice, however, it is possible that the model distributes the background over all the slots as mentioned in Locatello et al. [38]. Taking both cases into consideration (see Fig. 6.6 and Fig. 6.7), we propose two approaches to convert the multiple output masks into a foreground-background partition, and report the best results of these two options: (1) we compute the scores by making each mask as the background mask at a time, and then choose the best one; this works better when the background is treated as a virtual "object"; (2) we threshold and combine all the masks into a foreground mask; this is for when background is distributed to all slots.



Figure 6.6: An example situation when using each individual mask as the background mask gives higher scores. Note that if we threshold the output of each individual slot and compose them, the result would be the mask shown in the last column.

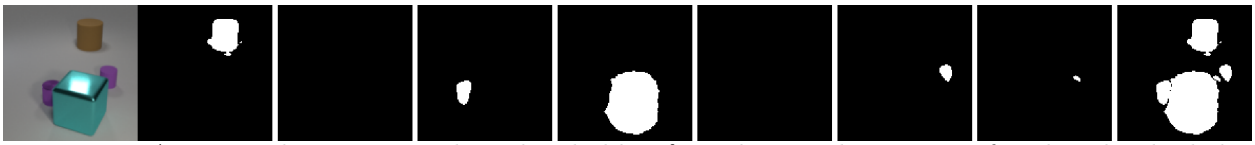


Figure 6.7: An example situation when thresholding&combining the output of each individual slot gives higher scores. We can see from the last column that the combined mask fits the foreground objects well.

6.7.6 Additional Illustrations and Baseline Results

6.7.6.1 More Examples

We provide more foreground extraction results of our model for each dataset; see Fig. 6.8, Fig. 6.9, Fig. 6.10 and Fig. 6.11. From top to bottom, we display: (i) observed images, (ii) generated images, (iii) masked generated foregrounds, (iv) generated backgrounds, (v) ground-truth foreground masks, and (vi) inferred foreground masks in each figure.

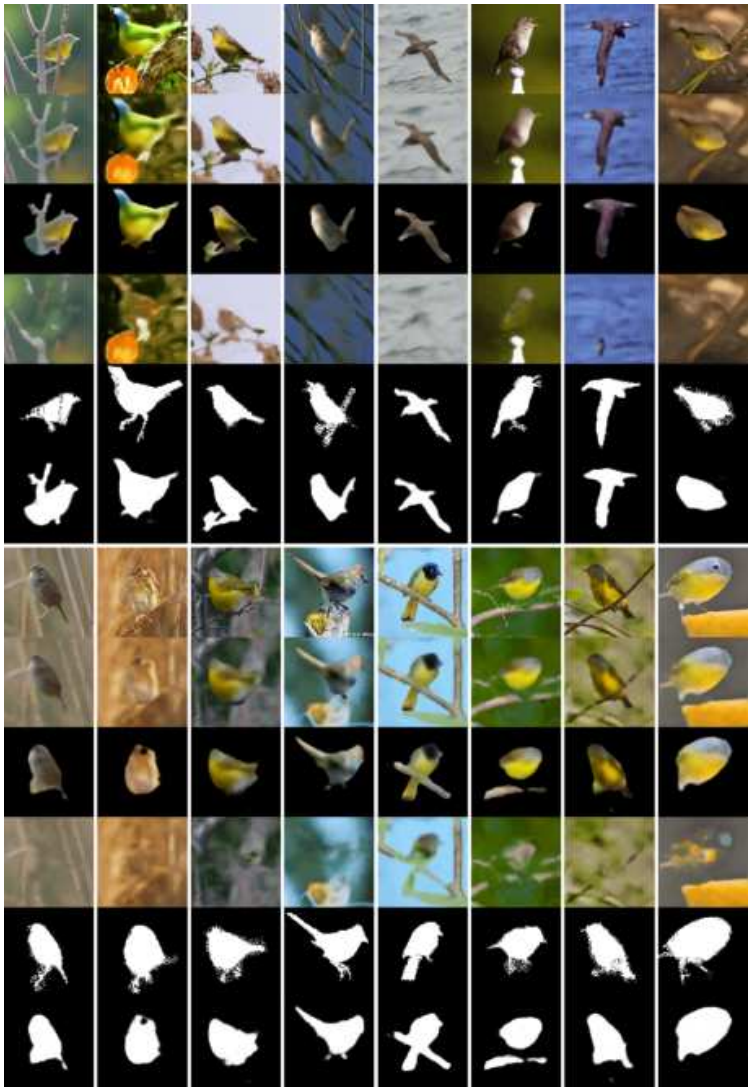


Figure 6.8: Additional foreground extraction results on Birds dataset.

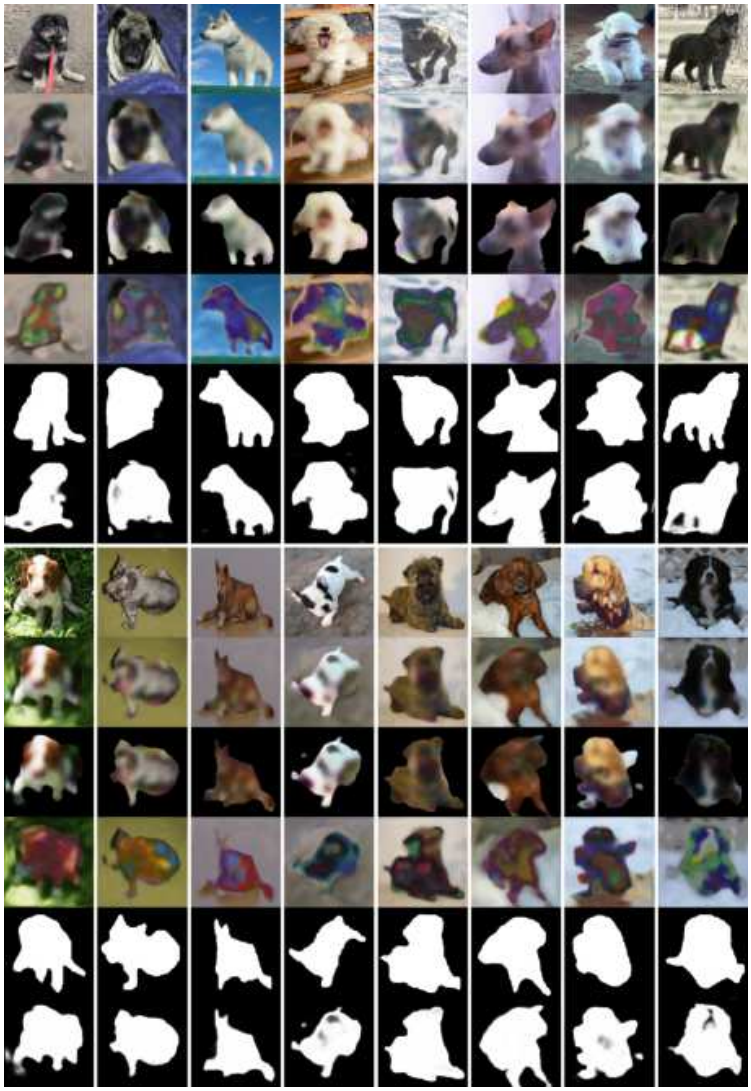


Figure 6.9: Additional foreground extraction results on Dogs dataset.

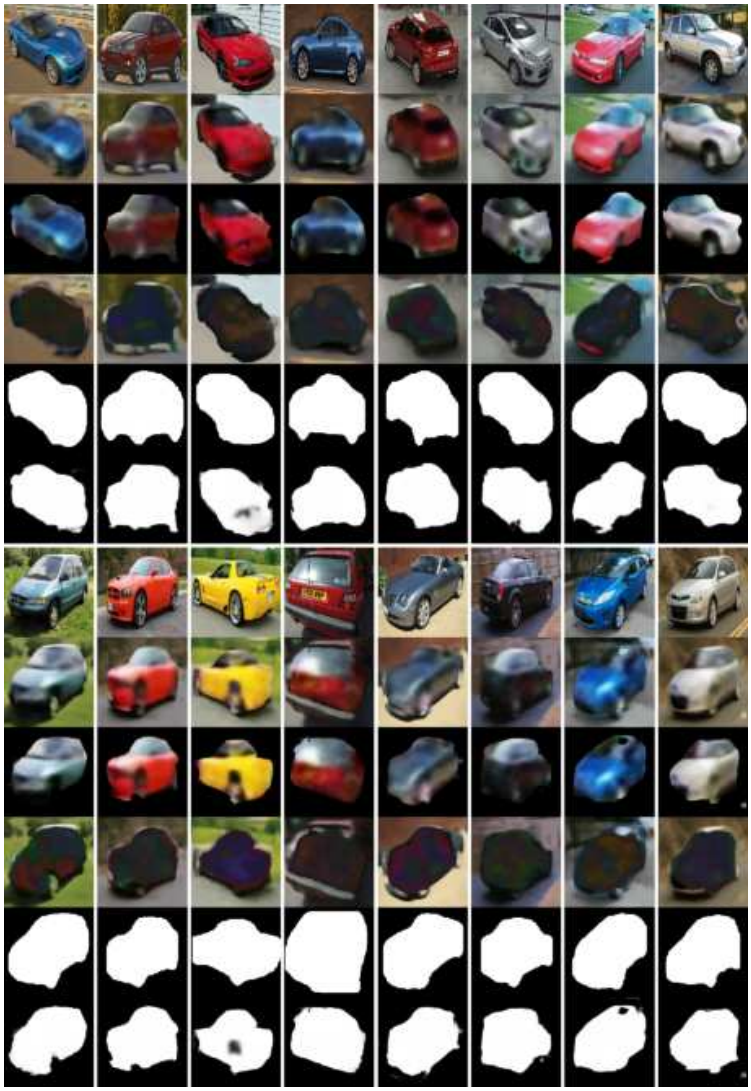


Figure 6.10: Additional foreground extraction results on Cars dataset.

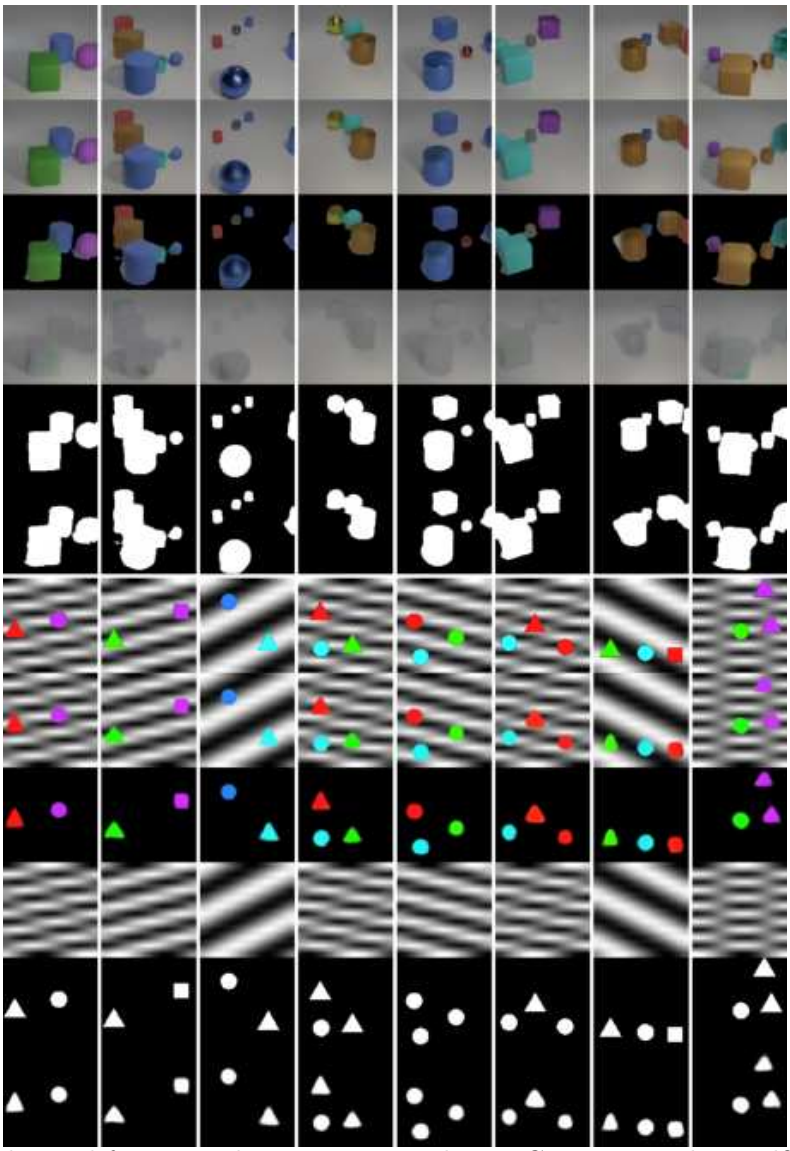


Figure 6.11: Additional foreground extraction results on CLEVR6 and TM-dSprites datasets.

6.7.6.2 Failure Modes

We provide examples for illustrating typical failure modes of the proposed model; see Fig. 6.12. On Birds dataset, we observe that the method can perform worse on samples where the foreground object has colors and textures quite similar to the background regions. Although the method can still capture the rough shape of the foreground object, some details can be missing. On TM-dSprites dataset, we observe that the method may occasionally miss one of the foreground objects. We conjecture that the problem can be mitigated with more powerful generator and further fine-tuning on this dataset.

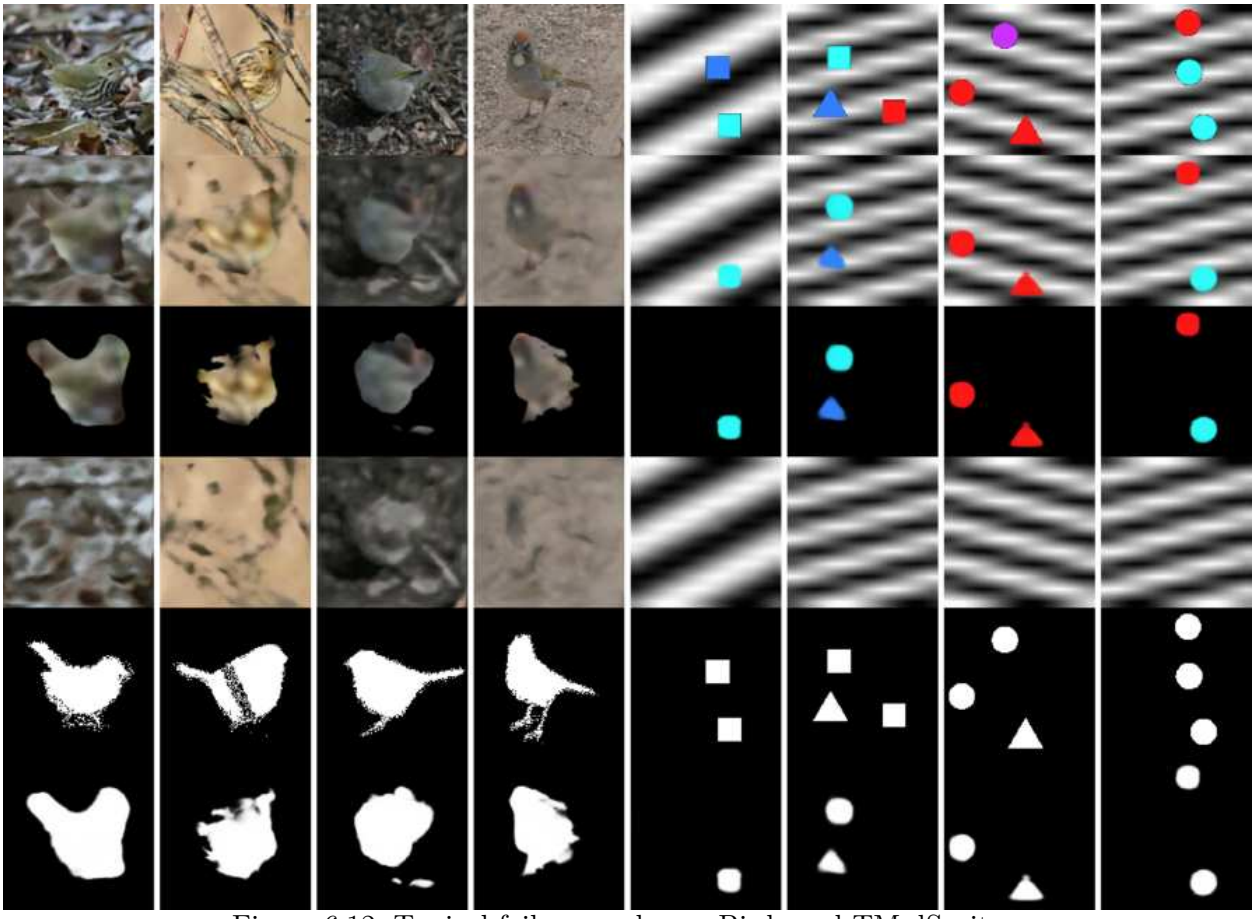


Figure 6.12: Typical failure modes on Birds and TM-dSprites.

6.7.6.3 Baseline Results

GrabCut We provide results of GrabCut [5] on Birds dataset and TM-dSprites dataset, shown in Fig. 6.13. We can see that GrabCut algorithm may fail when the foreground object

and background region have moderately similar colors and textures. On TM-dSprites dataset, GrabCut algorithm outperforms other baselines, but is still inferior to the proposed method and exhibits a similar failure pattern.

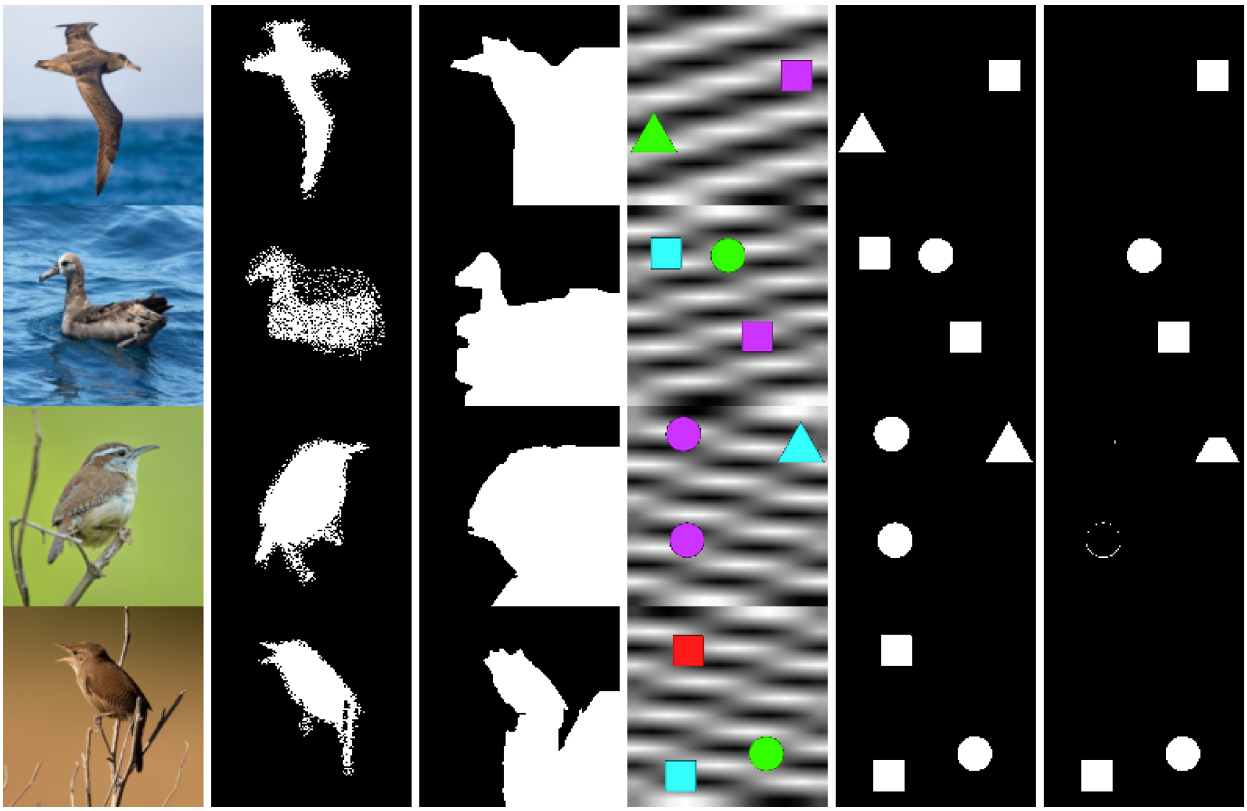


Figure 6.13: Results of GrabCut on Birds and TM-dSprites datasets. The first three columns are results from Birds dataset, and the last three are from TM-dSprites. From left to right, we display the observed image, ground-truth mask, and the foreground extraction results respectively.

ReDO We provide results of ReDO [28] on Birds dataset and TM-dSprites dataset, shown in Fig. 6.14. ReDO overall performs better than GrabCut on Birds dataset, while it may fail when the background regions become more complex. We can also observe that ReDO relies heavily on the pixel intensities for foreground-background grouping on TM-dSprites dataset.

IODINE On Birds dataset, we observe that IODINE [37] tends to use color as a strong cue for segmentation, see Fig. 6.15. On TM-dSprites dataset, IODINE is distracted by the background; see Fig. 6.16. These two findings are consistent with those reported in [37];

Slot-Attention On Birds dataset, Slot-attention learns to roughly locate the position of foreground objects, but mostly fails to provide foreground masks when the background

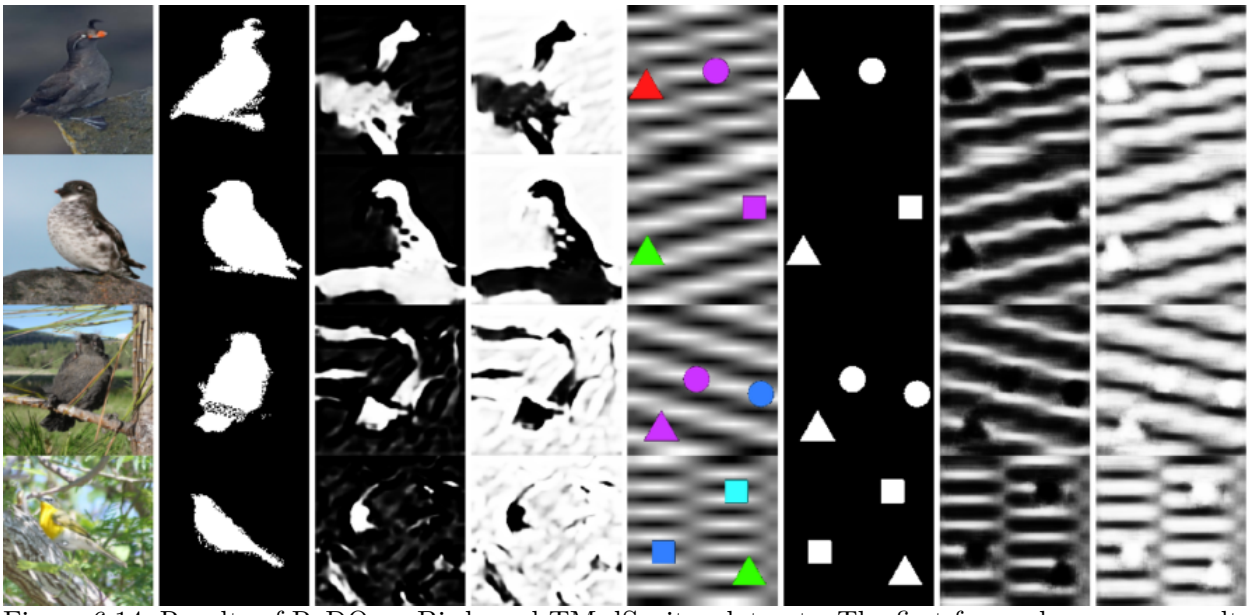


Figure 6.14: Results of ReDO on Birds and TM-dSprites datasets. The first four columns are results from Birds dataset, and the last four are from TM-dSprites. From left to right, we display the observed image, ground-truth mask, mask from the first output channel and from the second channel respectively.

region becomes complex; see Fig. 6.17. Similarly, we can observe that Slot-Attention tends to use color as a strong cue for segmentation. On TM-dSprites dataset, Slot-attention is distracted by the background; see Fig. 6.18.

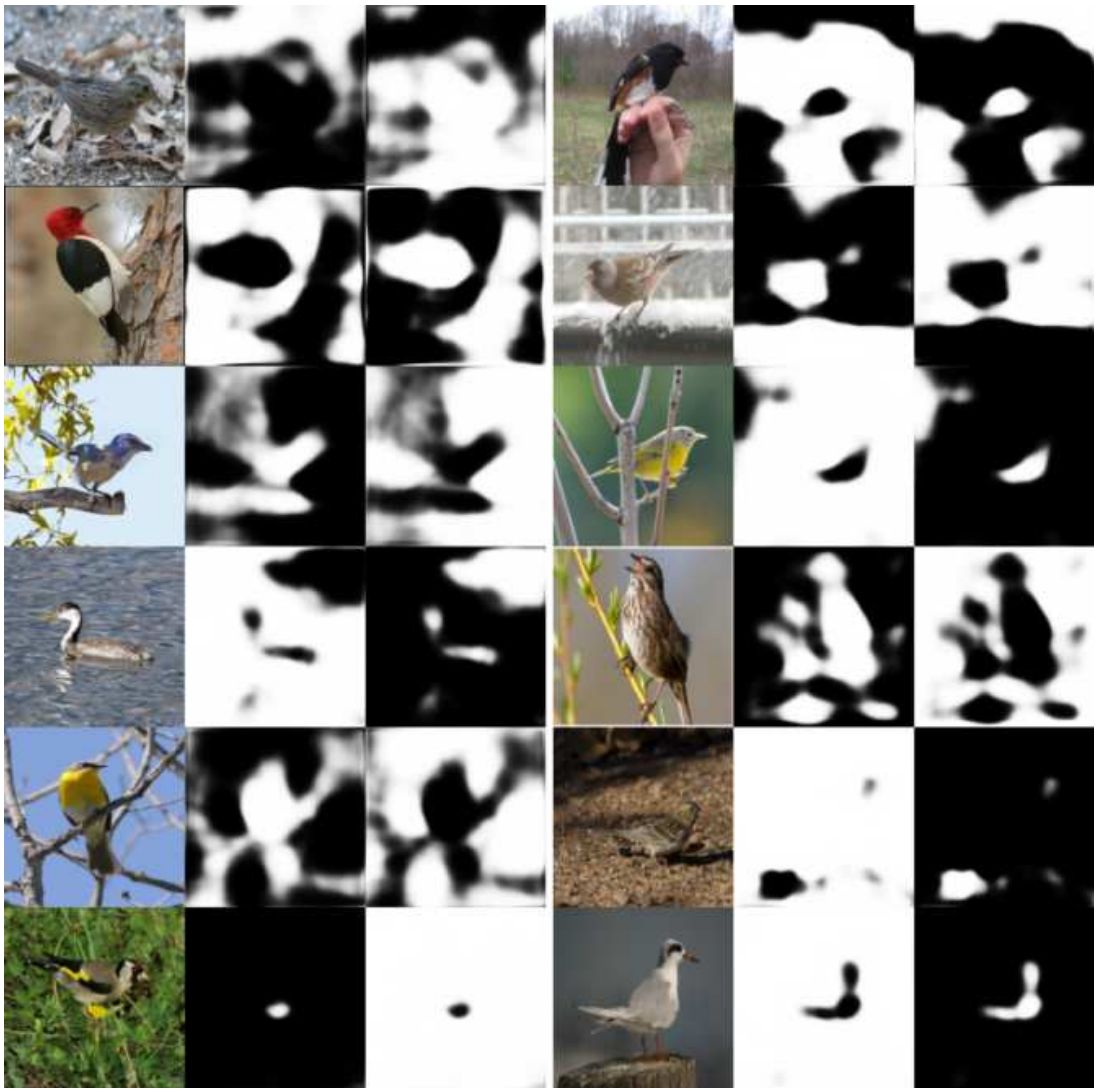


Figure 6.15: Results of IODINE on Birds datasets. We provide the observed image, mask from the first slot and from the second slot respectively.

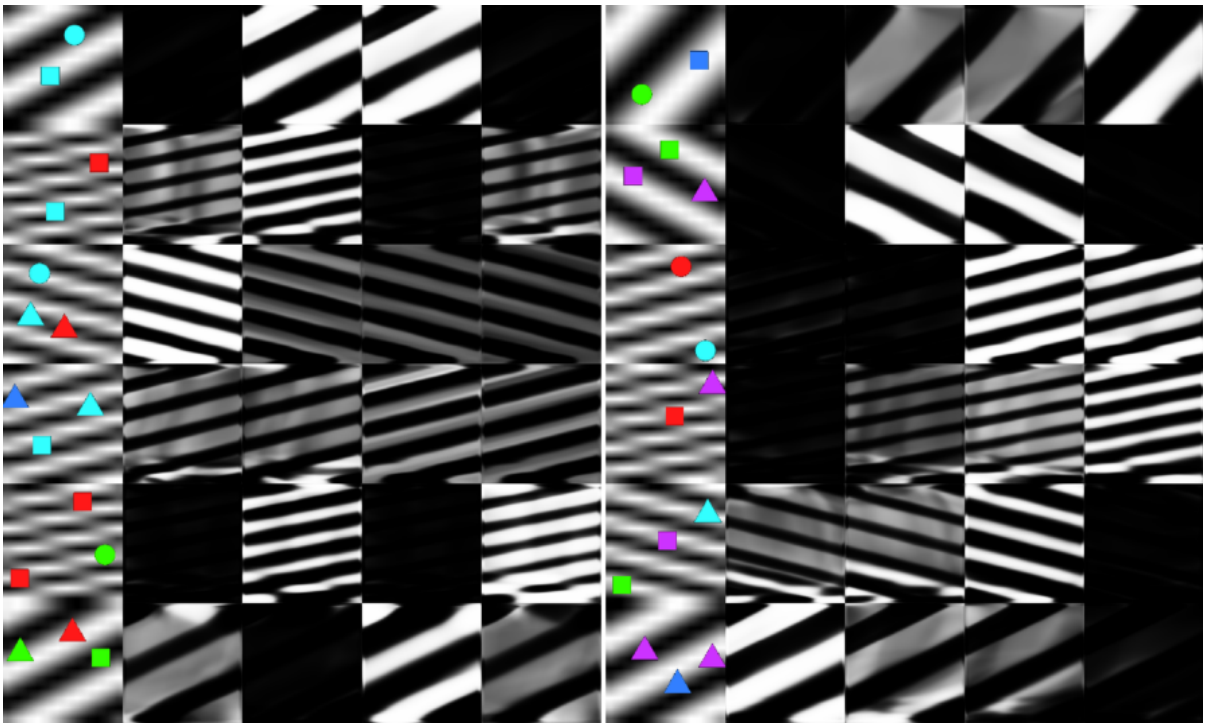


Figure 6.16: Results of IODINE on TM-dSprites datasets. We provide the observed image and masks from four object slots respectively.

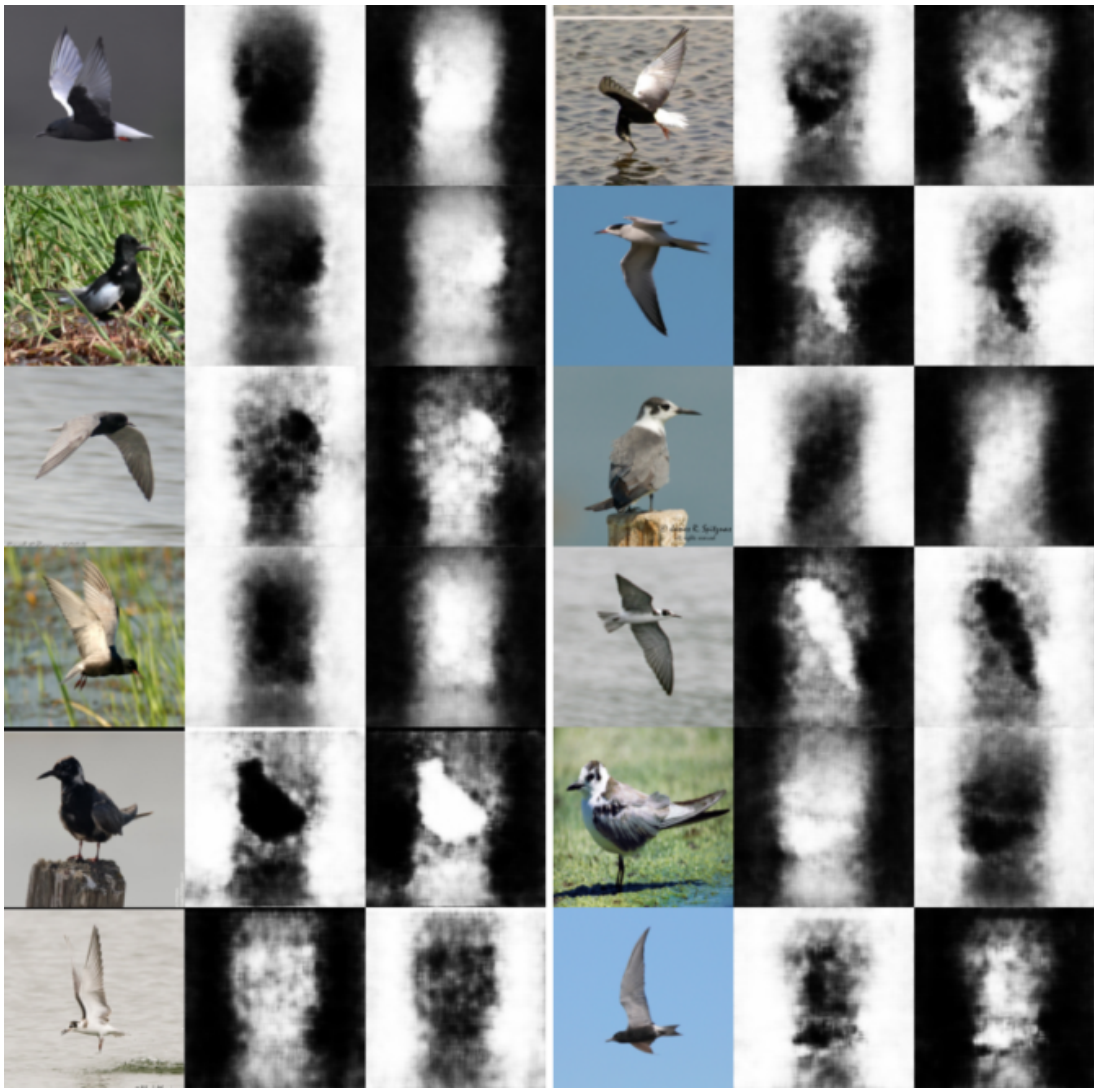


Figure 6.17: Results of Slot-Attention on Birds datasets. We provide the observed image, mask from the first slot and from the second slot respectively.

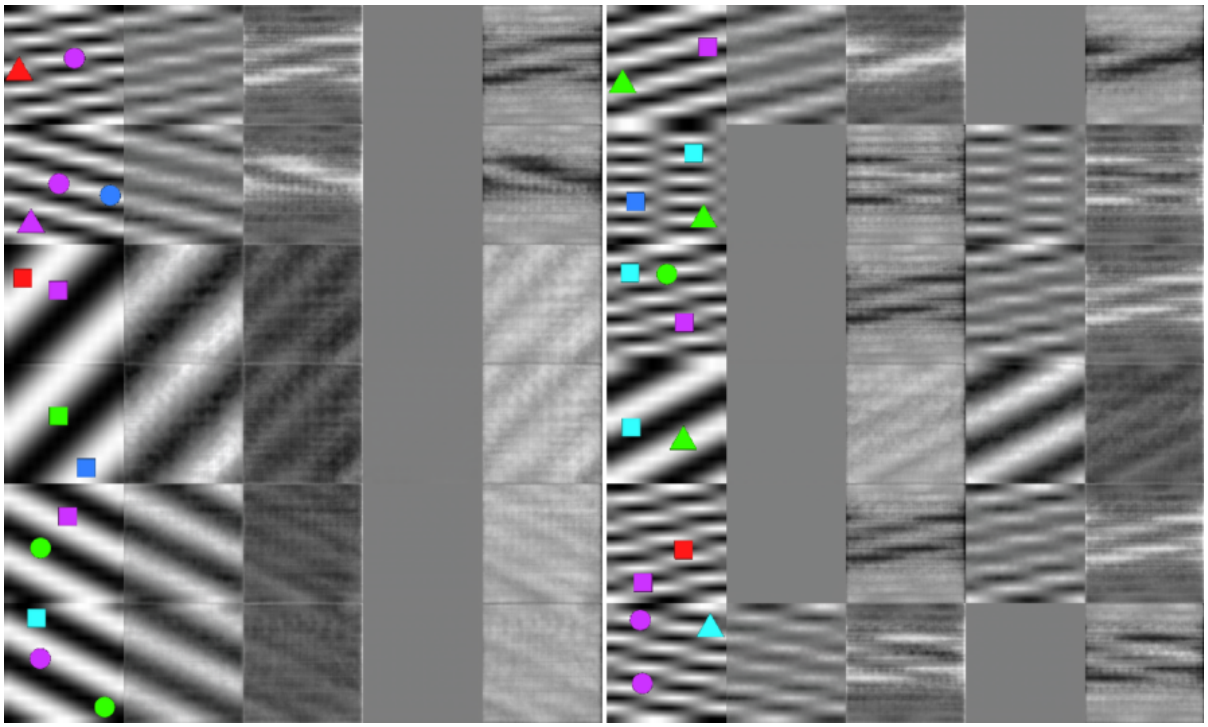


Figure 6.18: Results of Slot-Attention on TM-dSprites datasets. We provide the observed image and masks from four object slots respectively.

6.7.7 Further Discussion

6.7.7.1 Preliminary Analysis of Real-world Datasets

We provide preliminary analysis of the statistics of the three real-world datasets. To measure the similarity of colors and textures for these datasets, we calculate the image histogram for the foreground objects and background regions of each dataset; see Fig. 6.19. To probe the similarity of shape distributions, we also provide the heatmap of foreground masks, as shown in Fig. 6.20. The heatmaps are calculated by overlapping the ground-truth masks and normalizing the summarized intensities with the maximum values. Despite the apparent difference in Birds vs Dogs and Cars, we can see that the data distribution of Birds dataset is more similar to that of Dogs dataset than to that of Cars dataset. We can also observe the similarity between the distributions of Dogs and Cars datasets. This could partly explain why the proposed method shows relatively strong performance on objects from unseen categories, *i.e.*, it effectively combines the colors, textures and shapes for foreground extraction.

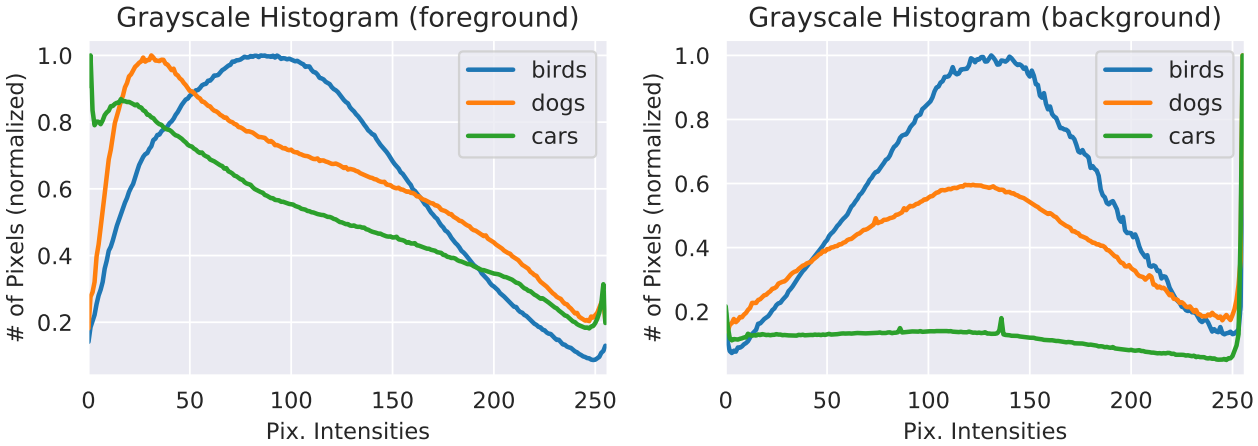


Figure 6.19: Image histograms for foreground objects and background regions from each dataset.

6.7.7.2 Possible Extension to Multi-Object Segmentation

We explore the possibility of using our model for segmenting and disentangling multiple objects. As shown in Fig. 6.21, the proposed method can disentangle the foreground objects, while providing explicit identification of the background region. However, we find that the model occasionally distributes a single object into several slots based on the difference in

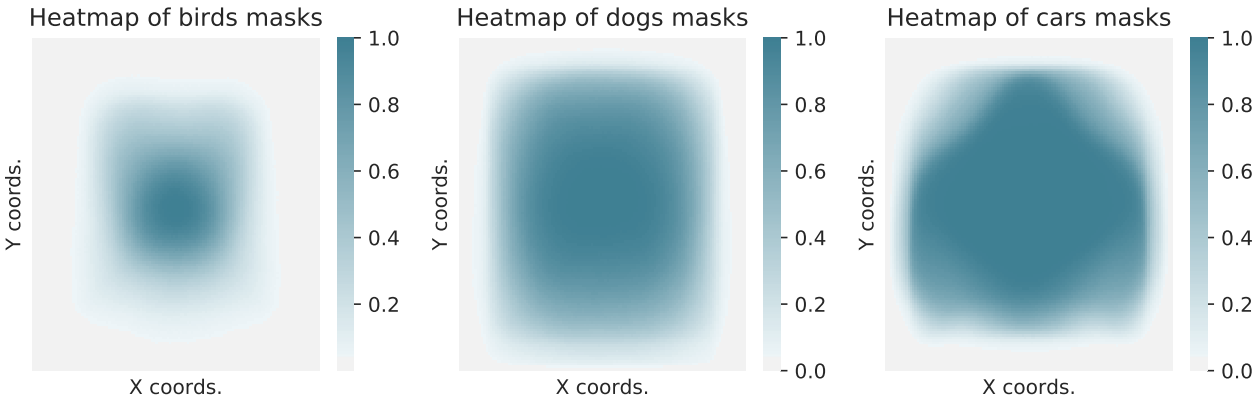
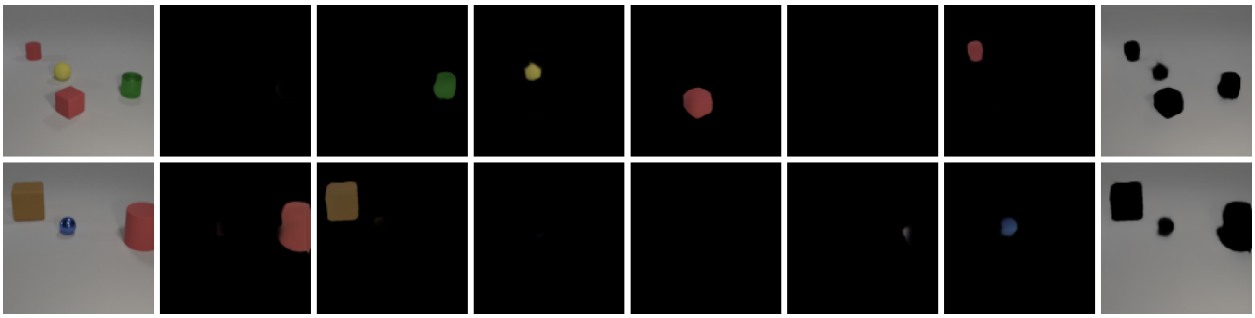
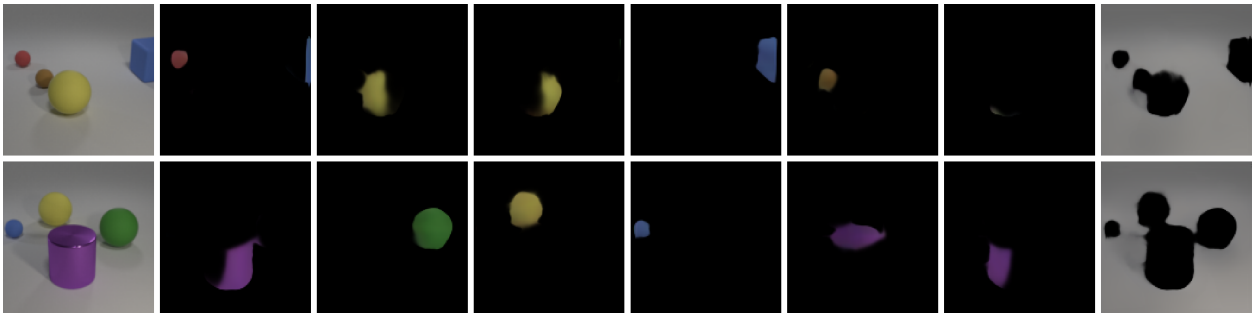


Figure 6.20: Heatmaps of ground-truth masks for each dataset.


Figure 6.21: Preliminary results on learning slot-based object representation.


Figure 6.22: Failure modes of energy-based slot representation model.

texture and shading; see Fig. 6.22. We conjecture that this is due to the lack of objectness modeling. We would like to investigate more on this direction in future works.

6.7.7.3 Prior Sampling Results on Birds Dataset



Figure 6.23: Prior sampling results on Birds dataset.

We provide preliminary results of sampling from the learned energy-based priors, as shown in Fig. 6.23. Of note, the generated prior samples are generally less realistic compared with the posterior samples, as prior sampling does not involve the region competition between foreground and background components, which may lead to worse separation and the generation of foreground and background regions. We would further explore generating foreground and background in future work.

Bibliography

[1] Song-Chun Zhu and Alan Yuille. Region competition: Unifying snakes, region growing, and bayes/mdl for multiband image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 18(9):884–900, 1996.

[2] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 22(8):888–905, 2000.

[3] Zhuowen Tu and Song-Chun Zhu. Image segmentation by data-driven markov chain monte carlo. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 24(5):657–673, 2002.

[4] Yuri Y Boykov and M-P Jolly. Interactive graph cuts for optimal boundary & region segmentation of objects in nd images. In *Proceedings of International Conference on Computer Vision (ICCV)*, 2001.

[5] Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. " grabcut" interactive foreground extraction using iterated graph cuts. *ACM Transactions on Graphics (TOG)*, 23(3):309–314, 2004.

[6] Ming-Ming Cheng, Niloy J Mitra, Xiaolei Huang, Philip HS Torr, and Shi-Min Hu. Global contrast based salient region detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 37(3):569–582, 2014.

[7] Huaizu Jiang, Jingdong Wang, Zejian Yuan, Yang Wu, Nanning Zheng, and Shipeng Li. Salient object detection: A discriminative regional feature integration approach. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013.

[8] Wangjiang Zhu, Shuang Liang, Yichen Wei, and Jian Sun. Saliency optimization from robust background detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[9] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *Proceedings of European Conference on Computer Vision (ECCV)*, 2014.

[10] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision (IJCV)*, 88(2):303–338, 2010.

[11] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[12] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[13] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 39(12):2481–2495, 2017.

[14] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous

convolution, and fully connected crfs. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 40(4):834–848, 2017.

[15] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer-assisted Intervention*, 2015.

[16] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of International Conference on Computer Vision (ICCV)*, 2017.

[17] Nick Chater, Joshua B Tenenbaum, and Alan Yuille. Probabilistic models of cognition: Conceptual foundations. *Trends in Cognitive Sciences*, 10(7):287–291, 2006.

[18] Thomas F Shipley and Philip J Kellman. *From fragments to objects: Segmentation and grouping in vision*. Elsevier, 2001.

[19] Xide Xia and Brian Kulis. W-net: A deep model for fully unsupervised image segmentation. *arXiv preprint arXiv:1711.08506*, 2017.

[20] Asako Kanezaki. Unsupervised image segmentation by backpropagation. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018.

[21] Xu Ji, João F Henriques, and Andrea Vedaldi. Invariant information clustering for unsupervised image classification and segmentation. In *Proceedings of International Conference on Computer Vision (ICCV)*, 2019.

[22] Pablo Arbelaez, Michael Maire, Charless Fowlkes, and Jitendra Malik. Contour detection and hierarchical image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 33(5):898–916, 2010.

[23] Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, and Sabine Süsstrunk. Slic superpixels compared to state-of-the-art superpixel methods.

IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), 34(11): 2274–2282, 2012.

[24] Yanchao Yang, Antonio Loquercio, Davide Scaramuzza, and Stefano Soatto. Unsupervised moving object detection via contextual information separation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[25] Yanchao Yang, Brian Lai, and Stefano Soatto. Dystab: Unsupervised object segmentation via dynamic-static bootstrapping. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[26] Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C Courville, and Yoshua Bengio. Generative adversarial nets. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2014.

[27] Jianwei Yang, Anitha Kannan, Dhruv Batra, and Devi Parikh. Lr-gan: Layered recursive generative adversarial networks for image generation. In *International Conference on Learning Representations (ICLR)*, 2017.

[28] Mickaël Chen, Thierry Artières, and Ludovic Denoyer. Unsupervised object segmentation by redrawing. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[29] Pavel Ostyakov, Roman Suvorov, Elizaveta Logacheva, Oleg Khomenko, and Sergey I Nikolenko. Seigan: Towards compositional image generation by simultaneously learning to segment, enhance, and inpaint. *arXiv preprint arXiv:1811.07630*, 2018.

[30] Krishna Kumar Singh, Utkarsh Ojha, and Yong Jae Lee. Finegan: Unsupervised hierarchical disentanglement for fine-grained object generation and discovery. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[31] Yaniv Benny and Lior Wolf. Onegan: Simultaneous unsupervised learning of conditional image generation, foreground segmentation, and fine-grained clustering. In *Proceedings of European Conference on Computer Vision (ECCV)*, 2020.

[32] Klaus Greff, Antti Rasmus, Mathias Berglund, Tele Hotloo Hao, Jürgen Schmidhuber, and Harri Valpola. Tagger: Deep unsupervised perceptual grouping. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2016.

[33] SM Ali Eslami, Nicolas Heess, Theophane Weber, Yuval Tassa, David Szepesvari, Koray Kavukcuoglu, and Geoffrey E Hinton. Attend, infer, repeat: Fast scene understanding with generative models. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2016.

[34] Klaus Greff, Sjoerd van Steenkiste, and Jürgen Schmidhuber. Neural expectation maximization. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[35] Sjoerd van Steenkiste, Michael Chang, Klaus Greff, and Jürgen Schmidhuber. Relational neural expectation maximization: Unsupervised discovery of objects and their interactions. In *International Conference on Learning Representations (ICLR)*, 2018.

[36] Christopher P Burgess, Loic Matthey, Nicholas Watters, Rishabh Kabra, Irina Higgins, Matt Botvinick, and Alexander Lerchner. Monet: Unsupervised scene decomposition and representation. *arXiv preprint arXiv:1901.11390*, 2019.

[37] Klaus Greff, Raphaël Lopez Kaufman, Rishabh Kabra, Nick Watters, Christopher Burgess, Daniel Zoran, Loic Matthey, Matthew Botvinick, and Alexander Lerchner. Multi-object representation learning with iterative variational inference. In *Proceedings of International Conference on Machine Learning (ICML)*, 2019.

[38] Francesco Locatello, Dirk Weissenborn, Thomas Unterthiner, Aravindh Mahendran, Georg Heigold, Jakob Uszkoreit, Alexey Dosovitskiy, and Thomas Kipf. Object-centric

learning with slot attention. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[39] Martin Engelcke, Adam R Kosiorek, Oiwi Parker Jones, and Ingmar Posner. Genesis: Generative scene inference and sampling with object-centric latent representations. In *International Conference on Learning Representations (ICLR)*, 2020.

[40] Zhixuan Lin, Yi-Fu Wu, Skand Vishwanath Peri, Weihao Sun, Gautam Singh, Fei Deng, Jindong Jiang, and Sungjin Ahn. Space: Unsupervised object-oriented scene representation via spatial attention and decomposition. In *International Conference on Learning Representations (ICLR)*, 2020.

[41] Michael Kass, Andrew Witkin, and Demetri Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision (IJCV)*, 1(4):321–331, 1988.

[42] Laurent D Cohen. On active contour models and balloons. *CVGIP: Image understanding*, 53(2):211–218, 1991.

[43] Rolf Adams and Leanne Bischof. Seeded region growing. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 16(6):641–647, 1994.

[44] Song-Chun Zhu, Yingnian Wu, and David Mumford. Filters, random fields and maximum entropy (frame): Towards a unified theory for texture modeling. *International Journal of Computer Vision (IJCV)*, 27(2):107–126, 1998.

[45] Cheng-en Guo, Song-Chun Zhu, and Ying Nian Wu. Primal sketch: Integrating structure and texture. *Computer Vision and Image Understanding (CVIU)*, 106(1):5–19, 2007.

[46] Yvan G Leclerc. Constructing simple stable descriptions for image partitioning. *International Journal of Computer Vision (IJCV)*, 3(1):73–102, 1989.

[47] Bo Pang, Tian Han, Erik Nijkamp, Song-Chun Zhu, and Ying Nian Wu. Learning latent

space energy-based prior model. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[48] Robert A Jacobs, Michael I Jordan, Steven J Nowlan, and Geoffrey E Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3(1):79–87, 1991.

[49] Michael I Jordan and Robert A Jacobs. Hierarchical mixtures of experts and the em algorithm. *Neural Computation*, 6(2):181–214, 1994.

[50] Karol Gregor, Ivo Danihelka, Alex Graves, Danilo Rezende, and Daan Wierstra. Draw: A recurrent neural network for image generation. In *Proceedings of International Conference on Machine Learning (ICML)*, 2015.

[51] Joe Marino, Yisong Yue, and Stephan Mandt. Iterative amortized inference. In *Proceedings of International Conference on Machine Learning (ICML)*, 2018.

[52] Trung T Pham, Thanh-Toan Do, Niko Sünderhauf, and Ian Reid. Scenecut: Joint geometric and object segmentation for indoor scenes. In *Proceedings of International Conference on Robotics and Automation (ICRA)*, 2018.

[53] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *Proceedings of European Conference on Computer Vision (ECCV)*, 2012.

[54] D Ulyanov, A Vedaldi, and V Lempitsky. Deep image prior. *International Journal of Computer Vision (IJCV)*, 128(7), 2020.

[55] Dominik Lorenz, Leonard Bereska, Timo Milbich, and Bjorn Ommer. Unsupervised part-based disentangling of object shape and appearance. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[56] George Papandreou, Liang-Chieh Chen, Kevin P Murphy, and Alan L Yuille. Weakly-and

semi-supervised learning of a deep convolutional network for semantic image segmentation. In *Proceedings of International Conference on Computer Vision (ICCV)*, 2015.

[57] Deepak Pathak, Philipp Krahenbuhl, and Trevor Darrell. Constrained convolutional neural networks for weakly supervised segmentation. In *Proceedings of International Conference on Computer Vision (ICCV)*, 2015.

[58] Zilong Huang, Xinggang Wang, Jiasi Wang, Wenyu Liu, and Jingdong Wang. Weakly-supervised semantic segmentation network with deep seeded region growing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[59] Jifeng Dai, Kaiming He, and Jian Sun. Boxsup: Exploiting bounding boxes to supervise convolutional networks for semantic segmentation. In *Proceedings of International Conference on Computer Vision (ICCV)*, 2015.

[60] Anna Khoreva, Rodrigo Benenson, Jan Hosang, Matthias Hein, and Bernt Schiele. Simple does it: Weakly supervised instance and semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[61] Seong Joon Oh, Rodrigo Benenson, Anna Khoreva, Zeynep Akata, Mario Fritz, and Bernt Schiele. Exploiting saliency for object segmentation from image level labels. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[62] Yu Zeng, Yunzhi Zhuge, Huchuan Lu, and Lihe Zhang. Joint learning of saliency detection and weakly supervised semantic segmentation. In *Proceedings of International Conference on Computer Vision (ICCV)*, 2019.

[63] Andrey Voynov, Stanislav Morozov, and Artem Babenko. Object segmentation without labels with large-scale generative models. In *Proceedings of International Conference on Machine Learning (ICML)*, 2021.

[64] Max Welling and Yee W Teh. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of International Conference on Machine Learning (ICML)*, 2011.

[65] Steve Brooks, Andrew Gelman, Galin Jones, and Xiao-Li Meng. *Handbook of markov chain monte carlo*. CRC press, 2011.

[66] Bo Pang and Ying Nian Wu. Latent space energy-based model of symbol-vector coupling for text generation and classification. In *Proceedings of International Conference on Machine Learning (ICML)*, 2021.

[67] Bela Julesz. Textons, the elements of texture perception, and their interactions. *Nature*, 290(5802):91–97, 1981.

[68] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[69] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: interpretable representation learning by information maximizing generative adversarial nets. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2016.

[70] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[71] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of International Conference on Machine Learning (ICML)*, 2015.

[72] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*, 2016.

[73] Andrew M Saxe, James L McClelland, and Surya Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. In *International Conference on Learning Representations (ICLR)*, 2014.

[74] Andrew Brock, Theodore Lim, James M Ritchie, and Nick Weston. Neural photo editing with introspective adversarial networks. *arXiv preprint arXiv:1609.07093*, 2016.

[75] Leonid I Rudin, Stanley Osher, and Emad Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D: nonlinear phenomena*, 60(1-4):259–268, 1992.

[76] P. Welinder, S. Branson, T. Mita, C. Wah, F. Schroff, S. Belongie, and P. Perona. Caltech-UCSD Birds 200. Technical Report CNS-TR-2010-001, California Institute of Technology, 2010.

[77] Aditya Khosla, Nityananda Jayadevaprakash, Bangpeng Yao, and Fei-Fei Li. Novel dataset for fine-grained image categorization: Stanford dogs. In *CVPR Workshop on Fine-Grained Visual Categorization (FGVC)*, 2011.

[78] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *ICCV workshops*, 2013.

[79] Justin Johnson, Bharath Hariharan, Laurens Van Der Maaten, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[80] Loic Matthey, Irina Higgins, Demis Hassabis, and Alexander Lerchner. dsprites: Disentanglement testing sprites dataset. https://github.com/deepmind/dsprites-dataset/, 2017.

[81] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[82] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. `https://github.com/facebookresearch/detectron2`, 2019.

[83] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[84] Erik Nijkamp, Mitch Hill, Song-Chun Zhu, and Ying Nian Wu. Learning non-convergent non-persistent short-run mcmc toward energy-based model. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[85] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Chapter 7

Latent Diffusion Energy-Based Model for Interpretable Text Modeling

7.1 Introduction

Text modeling has achieved impressive progress with the fast development of neural generative models [16, 32, 51, 74, 75]. It allows near human-level text generation quality and also leads to a wide range of real-world applications such as dialog system [70] and machine translation [5]. Although the quality of generation (*e.g.*, fluency and diversity) is the primary concern of most work, interpretability of the generation process has drawn much attention recently. Among the existing frameworks, the DLVM is especially suitable for the task, as the learned latent space could capture high-level structures with semantic meanings like topics [65] and dialog actions [76]; such latent space could further enable more interpretable text modeling, featuring unsupervised text attributes discovery [68], conditional and controllable text generation [11, 53], and semi-supervised text classification [43].

In essence, DLVM summarizes the observed sample (*e.g.*, a piece of text) into inferred latent variables. Earlier text-modeling methods with DLVM mostly follow the formulation of Variational Auto-Encoder (VAE) [4, 28, 49], which assumes a continuous latent space.

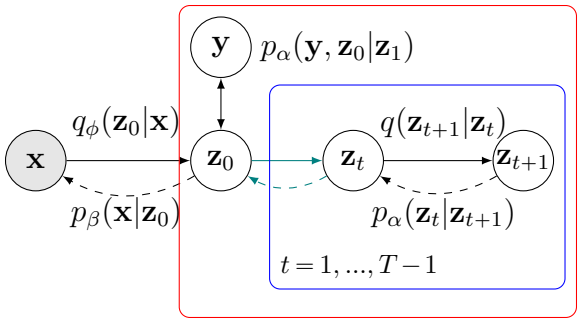


Figure 7.1: **Graphical illustration of the latent diffusion process.** We construct the forward and reverse diffusion processes in the latent space. The symbolic one-hot vector is coupled with the initial latent vector $\mathbf{z}_0$. The latent and diffused latent variables are highlighted by the red and blue plates, respectively. The cyan arrows indicate that $\mathbf{z}_0$ is connected with only $\mathbf{z}_1$. We learn a sequence of EBMs to model the reverse diffusion process $p_\alpha(\mathbf{z}_t|\mathbf{z}_{t+1})$.

More recently, Zhao et al. [76] explore the possibility of using a discrete latent space to capture dialog actions; Shi et al. [53] propose to use VAE with the mixture of Gaussians as the prior, demonstrating promising interpretability of dialog utterance generation. To further improve the expressivity of the latent space, Pang & Wu [43] leverage the flexibility of *energy-based prior* [44] and learn a structured latent space for interpretable text generation and classification. Specifically, they propose a symbol-vector coupling prior model. The continuous latent variables are coupled with discrete one-hot symbol variables, allowing better discrete structure induction without sacrificing the generation quality offered by the continuous latent space. However, similar to learning an EBM in data space, the learning of energy-based prior requires MCMC sampling, whose quality can degenerate in practice [14, 15, 41, 42], especially on data with complex latent structures; it often leads to instability during training. As we demonstrate empirically in Sec. 7.4.1, this phenomenon is particularly concerning when adopting the variational learning scheme to update model parameters.

To remedy this MCMC sampling issue, we may take a look at the endeavor of EBM learning in general. Among the recent efforts, methods drawn inspiration from the diffusion probabilistic models [22, 55, 57, 58] have demonstrated superior results. In particular, Gao et al. [14] propose a diffusion recovery likelihood method to learn and sample from a sequence of EBMs defined on increasingly noisy versions of a dataset; the models are trained by optimizing conditional likelihoods, which are more tractable than the marginal

likelihood. It greatly mitigates the burden of sampling during training. A natural question thus emerges: *Can we leverage the methodology of diffusion models to address the learning issue of energy-based prior?*

In this work, we make the first attempt to address the learning issue of energy-based prior through leveraging diffusion models in the latent space, with a focus on interpretable text modeling. We first unveil the non-trivial symbiosis between latent-space EBMs and diffusion models. Specifically, we focus on the symbol-vector coupling prior; we construct a flexible process that restores the hidden structure in text data by noise-level-aware sampling from a learned sequence of conditional EBMs in the latent space. A variational learning framework is then derived from it. We further employ a geometric clustering-based regularization that complements the symbol-inducing information bottleneck to improve the quality of learned latent space. We term the resulting model LDEBM. Compared to Gao et al. [14], which deals with EBMs in the data space, LDEBM is directly applicable to text data with or without labels; it extracts interpretable latent structures that benefit potential downstream tasks such as semi-supervised classification. Although there are methods using diffusion models in the latent space, some of which have achieved very impressive image generation results, *e.g.*, Vahdat et al. [62], few of them to our knowledge have explored (unsupervised) symbol induction in the latent space especially on text data. In addition, our method can be trained from scratch and form a well-structured latent space without pretraining, as required by concurrent works on image modeling such as Vahdat et al. [62] and Nie et al. [40]. In our experiments on generative modeling and interpretable text modeling, LDEBM improves the quality and interpretability of the learned latent space over strong baselines on most evaluated benchmarks.

Research contributions (1) We introduce a novel symbiosis of the latent space EBM and diffusion model in a variational learning framework; the model can be trained from scratch, is directly applicable to text data with or without labels, and shows superior sampling quality. (2) We develop a geometric clustering-based regularization jointly with the information

bottleneck that tackles the mode-collapse problem in variational learning of the latent space EBM. (3) Our experiments demonstrate that the proposed model learns a well-structured latent space and delivers strong results on interpretable text modeling.

7.2 Preliminaries: Symbol-Vector Coupling Energy-Based Model

We assume that for an observed high-dimensional sample $\mathbf{x} \in \mathbb{R}^D$, there exists $\mathbf{z} \in \mathbb{R}^d$ as its compact continuous latent variables. We assume that $\mathbf{y}$ is the symbolic one-hot vector indicating one of K categories that $\mathbf{z}$ belongs to. The complete-data distribution is $p_\theta(\mathbf{y}, \mathbf{z}, \mathbf{x}) = p_\alpha(\mathbf{y}, \mathbf{z}) p_\beta(\mathbf{x}|\mathbf{z})$, where $p_\alpha(\mathbf{y}, \mathbf{z})$ is the joint prior model with parameters α, and $p_\beta(\mathbf{x}|\mathbf{z})$ is the top-down generation model with parameters β; henceforth, we use $\theta = (\alpha, \beta)$ to summarize the parameters. Given $\mathbf{z}$, $\mathbf{y}$ and $\mathbf{x}$ are independent; *i.e.*, $\mathbf{z}$ is sufficient for $\mathbf{y}$ in this model.

Pang & Wu [43] propose to formulate the joint prior model, $p_\alpha(\mathbf{y}, \mathbf{z})$, as an EBM,

$$p_\alpha(\mathbf{y}, \mathbf{z}) = \frac{1}{Z_\alpha} \exp(\langle \mathbf{y}, f_\alpha(\mathbf{z}) \rangle) p_0(\mathbf{z}), \tag{7.1}$$

where $p_0(\mathbf{z})$ is a reference distribution, assumed to be the non-informative prior (*e.g.*, isotropic Gaussian or uniform) of the conventional generation model, $f_\alpha(\mathbf{z}) \in \mathbb{R}^K$ is parameterized by a small multi-layer perceptron, and Z_α is the normalizing constant or partition function. The energy term $\langle \mathbf{y}, f_\alpha(\mathbf{z}) \rangle$ in Eq. (7.1) forms an associative memory that couples the symbol $\mathbf{y}$ and the dense vector $\mathbf{z}$. Given $\mathbf{z}$,

$$p_\alpha(\mathbf{y}|\mathbf{z}) \propto \exp(\langle \mathbf{y}, f_\alpha(\mathbf{z}) \rangle) \tag{7.2}$$

becomes a softmax classifier, where $f_\alpha(\mathbf{z})$ provides the logit scores for the K categories. Marginally, we have

$$p_\alpha(\mathbf{z}) = \frac{1}{Z_\alpha} \exp(F_\alpha(\mathbf{z}))p_0(\mathbf{z}), \tag{7.3}$$

where the marginal energy term has the log-sum-exponential form

$$F_\alpha(\mathbf{z}) = \log \sum_{\mathbf{y}} \exp(\langle \mathbf{y}, f_\alpha(\mathbf{z}) \rangle).$$

It is shown that the coupling between $\mathbf{z}$ and $\mathbf{y}$ enables a symbol-aware continuous vector computation during prior and posterior sampling, which helps to induce a structural latent space [43]. Finally, the prior model $p_\alpha(\mathbf{y}, \mathbf{z})$ stands on a generation model $p_\beta(\mathbf{x}|\mathbf{z})$. In text modeling, let $\mathbf{x} = (\mathbf{x}^{(t)}, t = 1, ..., T)$ be a sentence, where $\mathbf{x}^{(t)}$ is the t-th token. $p_\beta(\mathbf{x}|\mathbf{z})$ can be defined as a conditional autoregressive model, $p_\beta(\mathbf{x}|\mathbf{z}) = \prod_{t=1}^{T} p_\beta(\mathbf{x}^{(t)}|\mathbf{x}^{(1)}, ..., \mathbf{x}^{(t-1)}, \mathbf{z})$. The complete model $p_\theta(\mathbf{y}, \mathbf{z}, \mathbf{x})$ with the energy-based prior $p_\alpha(\mathbf{y}, \mathbf{z})$ and the generation model $p_\beta(\mathbf{x}|\mathbf{z})$ is termed as SVEBM.

In principle, a SVEBM can be learned through maximizing the log-likelihood function, where the learning gradient is $\nabla_\theta \log p_\theta(\mathbf{x}) = \mathbb{E}_{p_\theta(\mathbf{z}|\mathbf{x})}[\nabla_\theta(\log p_\alpha(\mathbf{z}) + \log p_\beta(\mathbf{x}|\mathbf{z}))]$. To estimate the expectation, one may sample from the prior $p_\alpha(\mathbf{z})$ and the posterior $p_\theta(\mathbf{z}|\mathbf{x})$ with Langevin dynamics [67]. Since f_α is a small network, prior sampling is particularly affordable. In comparison, the posterior sampling can be more expensive as it requires back-propagating through the generation network. One promising solution is to follow the variational learning scheme [28] that amortizes the posterior sampling from $p_\theta(\mathbf{z}|\mathbf{x})$ by an inference network $q_\phi(\mathbf{z}|\mathbf{x})$; MCMC-based sampling can be used for prior samples.

7.3 Latent Diffusion Energy-Based Model

7.3.1 A Symbiosis between Symbol-Vector Coupling EBMs and Diffusion Models

Contrasting to the vanilla sampling process of the latent variables in SVEBM, LDEBM follows the philosophy of diffusion probabilistic models [55]; it assumes a sequence of perturbed samples, $\mathbf{z}_0, \mathbf{z}_1, ..., \mathbf{z}_T$, to construct a flexible process that restores the structure in data. First, we define the forward diffusion process that systematically and gradually destroys structure in a data distribution: $\mathbf{z}_0 \sim q_\phi(\mathbf{z}_0|\mathbf{x}); \mathbf{z}_{t+1} = \sqrt{1-\sigma_{t+1}^2}\mathbf{z}_t + \sigma_{t+1}\boldsymbol{\epsilon}_{t+1}$, where $t = 0, 1, ..., T-1$ and $\boldsymbol{\epsilon}_t$ is the zero-mean standard Gaussian noise. The scaling factor $\sqrt{1-\sigma_{t+1}^2}$ ensures that the sequence is a spherical interpolation between the posterior sample and the Gaussian white noise. The forward trajectory and the Markov transition between each perturbed samples $\mathbf{z}_1, ..., \mathbf{z}_T$ are thus

$$\begin{aligned} q_\phi(\mathbf{z}_{0:T}|\mathbf{x}) &= q_\phi(\mathbf{z}_0|\mathbf{x}) \prod_{t=0}^{T-1} q(\mathbf{z}_{t+1}|\mathbf{z}_t); \\ q(\mathbf{z}_{t+1}|\mathbf{z}_t) &= \mathcal{N}(\mathbf{z}_{t+1}; \sqrt{1-\sigma_{t+1}^2}\mathbf{z}_t, \sigma_{t+1}^2\mathbf{I}). \end{aligned} \tag{7.4}$$

Our goal is to learn the generative distribution that describes the same trajectory but in reverse. Inspired by Gao et al. [14], we start by constructing a sequence of *marginal* EBMs at each diffusion step in the latent space. The *conditional* EBMs aiming at recovering $\mathbf{z}_0$ from noisy inputs then follow as (see the derivation in Sec. 7.8.1):

$$\begin{aligned} &p_\alpha(\tilde{\mathbf{z}}_t|\mathbf{z}_{t+1}) = \\ &\frac{1}{\tilde{Z}_{\alpha,t}(\mathbf{z}_{t+1})} \exp\left(F_\alpha(\tilde{\mathbf{z}}_t, t) - \frac{1}{2\sigma_{t+1}^2}||\tilde{\mathbf{z}}_t - \mathbf{z}_{t+1}||^2\right), \end{aligned} \tag{7.5}$$

where $t = 0, 1, ..., T-2$. We denote $\tilde{\mathbf{z}}_t = \sqrt{1-\sigma_{t+1}^2}\mathbf{z}_t$ for brevity. $F_\alpha(\tilde{\mathbf{z}}_t, t)$ is the neural network that parameterizes the energy function at each diffusion step, and $\tilde{Z}_{\alpha,t}(\mathbf{z}_{t+1}) = \int \exp\left(F_\alpha(\tilde{\mathbf{z}}_t, t) - \frac{1}{2\sigma_{t+1}^2}||\tilde{\mathbf{z}}_t - \mathbf{z}_{t+1}||^2\right) d\tilde{\mathbf{z}}_t$ is the partition function of each conditional EBM. For

$t = T-1$, $p_\alpha(\tilde{\mathbf{z}}_t|\mathbf{z}_{t+1}) = \frac{1}{\tilde{Z}_{\alpha,t}} \exp\left(F_\alpha(\tilde{\mathbf{z}}_t, t) - \frac{1}{2\sigma_{t+1}^2}||\tilde{\mathbf{z}}_t||^2\right)$ since the diffused samples at time step T should be close to Gaussian white noise; the distribution of $\tilde{\mathbf{z}}_{T-1}$ can thus be exponentially tilting of a zero-mean Gaussian distribution.

Eq. (7.5) shares the idea of denoising generative modeling [3], where a denoising autoencoder is trained by maximizing the conditional probabilities of the observed samples given their noisy versions. Compared to the vanilla definition (see Eq. (7.3)), the noise-level-aware quadratic term constrains the energy landscape to be localized around the noisy sample; this makes the latent space much less multi-modal and easier to sample from. To be specific, Gao et al. [14] show that $p_\alpha(\tilde{\mathbf{z}}_t|\mathbf{z}_{t+1})$ is approximately a single-mode Gaussian distribution when σ is sufficiently small; it greatly reduces the burden of MCMC sampling. After sampling $\tilde{\mathbf{z}}_t$ from the model, we can easily obtain $\mathbf{z}_t = \tilde{\mathbf{z}}_t/\sqrt{1-\sigma_{t+1}^2}$.

Next, we show that the forward and reverse process in the latent space can be naturally integrated into the variational learning scheme to amortize the time-consuming posterior sampling. Similar to VAE, the ELBO in SVEBM is

$$\begin{aligned} \text{ELBO}_{\theta,\phi} &= \log p_\theta(\mathbf{x}) - \mathbb{D}_{\text{KL}}(q_\phi(\mathbf{z}|\mathbf{x}) \| p_\theta(\mathbf{z}|\mathbf{x})) \\ &= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\beta(\mathbf{x}|\mathbf{z})] - \mathbb{D}_{\text{KL}}(q_\phi(\mathbf{z}|\mathbf{x}) \| p_\alpha(\mathbf{z})), \end{aligned} \tag{7.6}$$

where $\mathbb{D}_{\text{KL}}$ denotes the Kullback-Leibler divergence. Since we now consider the full trajectory of the perturbed samples, in LDEBM we may optimize

$$\begin{aligned} \text{ELBO}_{\text{Diff},\theta,\phi} = \mathbb{E}_{q_\phi(\mathbf{z}_0|\mathbf{x})} \left[\log p_\beta(\mathbf{x}|\mathbf{z}_0) - \log q_\phi(\mathbf{z}_0|\mathbf{x})\right] \\ + \mathbb{E}_{\substack{q_\phi(\mathbf{z}_0|\mathbf{x}), \\ q(\mathbf{z}_{1:T}|\mathbf{z}_0)}} \left[\log \frac{p_\alpha(\mathbf{z}_{0:T})}{q(\mathbf{z}_{1:T}|\mathbf{z}_0)}\right]. \end{aligned} \tag{7.7}$$

which is a valid ELBO by applying Jensen's inequality to Eq. (7.6). The joint training of inference, prior and generation models can be largely reduced to finding the agreement of the forward and reverse Markov transitions defined by q_ϕ and p_θ, respectively. Please refer to Sec. 7.8.2 for more detailed derivations and discussions.

Finally, we show how to introduce the symbolic one-hot vector $\mathbf{y}$ into our formulation. We assume a complete data distribution that considers the full trajectory of the perturbed latent variables, $p_\theta(\mathbf{y}, \mathbf{z}_{0:T}, \mathbf{x})$. Among several possibilities for coupling the symbolic vector y with the latent variables, two major options arise: We can couple the symbol with the whole trajectory, *i.e.*, $p_\theta(\mathbf{y}, \mathbf{z}_{0:T}, \mathbf{x}) = p_\alpha(\mathbf{y}, \mathbf{z}_{0:T}) p_\beta(\mathbf{x}|\mathbf{z}_{0:T})$; or we can couple the symbol with only the clean posterior sample $\mathbf{z}_0$, *i.e.*, $p_\theta(\mathbf{y}, \mathbf{z}_{0:T}, \mathbf{x}) = p(\mathbf{z}_T) p_\alpha(\mathbf{y}, \mathbf{z}_0|\mathbf{z}_1) \prod_{t=1}^{T-1} p_\alpha(\mathbf{z}_t|\mathbf{z}_{t+1}) p_\beta(\mathbf{x}|\mathbf{z}_0)$. We prefer the latter one, since it is sufficient to model the reverse Markovian transition, while enabling a simpler and more efficient training scheme following Ho et al. [22] (see Sec. 7.3.4). Of note, coupling only $\mathbf{z}_0$ to $\mathbf{y}$ means that we condition only the final reverse diffusion step $[\mathbf{z}_0|\mathbf{z}_1]$ on $\mathbf{y}$ when performing controllable generation. This could be a bit counter-intuitive as no label information is injected in previous reverse steps. By construction, $\mathbf{y}$ and $\mathbf{z}_{1:T}$ are conditionally independent given $\mathbf{z}_0$ in our formulation; however, we empirically observe that $\mathbf{y}$ and $\mathbf{z}_t$ for $t > 0$ are nearly independent even marginally, after integrating out $\mathbf{z}_{0:t-1}$ in our model. In other words, $p_\alpha(\mathbf{y}|\mathbf{z}_t),\ t > 0$ are in general non-informative since adding noise in the latent space could be much more corrupting than adding noise in the data space. The model learns to enjoy the less multi-modal energy landscape in previous reverse steps; it then seeks the given mode only in the most informative final reverse step. Specifically, we achieve this coupling by similarly defining $p_\alpha(\mathbf{y}, \mathbf{z}_0|\mathbf{z}_1)$ as in Eq. (7.1) and using the log-sum-exponential form for learning as in Eq. (7.3). Please refer to Fig. 7.1 for a graphical illustration of our model and Secs. 7.8.3 and 7.9.3 for more details and discussions.

7.3.2 Information Bottleneck

To learn the symbolic vector $\mathbf{y}$, we may consider adopting the Information Bottleneck (IB) principle [60], an appealing approach for inducing symbolic representations. In this section, we re-interpret the above ELBO as a cooperative learning objective, defined as the divergence between two joint distributions; we then show how this formulation helps to incorporate the IB-based regularization into LDEBM in a principled manner.

As shown in Han et al. [18], the variational learning scheme can be regarded as performing alternating projection between two joint distributions, Q_ϕ and P_θ. In our modeling, we have: $Q_\phi(\mathbf{x}, \mathbf{z}_{0:T}) = q_{\text{data}}(\mathbf{x}) q_\phi(\mathbf{z}_{0:T}|x)$, and $P_\theta(\mathbf{x}, \mathbf{z}_{0:T}) = p(\mathbf{z}_T) \prod_{t=0}^{T-1} p_\alpha(\mathbf{z}_t|\mathbf{z}_{t+1}) p_\beta(\mathbf{x}|\mathbf{z}_0)$; we use $q_{\text{data}}(\mathbf{x})$ to denote the data distribution of $\mathbf{x}$ for notation consistency. Maximizing $\mathbb{E}_{q_{\text{data}}(\mathbf{x})}[\text{ELBO}_{\text{Diff},\theta,\phi}(\mathbf{x})]$ over (θ, ϕ) is equivalent to minimizing the following divergence:

$$\begin{aligned} \mathbb{D}_{\text{KL}}(Q_\phi \| P_\theta) &= \mathbb{D}_{\text{KL}}(q_{\text{data}}(\mathbf{x}) \| p_\theta(\mathbf{x})) \\ &+ \mathbb{E}_{q_{\text{data}}(\mathbf{x})}[\mathbb{D}_{\text{KL}}(q_\phi(\mathbf{z}_{0:T}|\mathbf{x}) \| p_\theta(\mathbf{z}_{0:T}|\mathbf{x}))], \end{aligned} \tag{7.8}$$

since $\mathcal{H}(\mathbf{x}) = -\mathbb{E}_{q_{\text{data}}(\mathbf{x})}[\log q_{\text{data}}(\mathbf{x})]$, *i.e.*, the entropy of data distribution is fixed. Minimizing the KL-divergence $\min_\theta \min_\phi \mathbb{D}_{\text{KL}}(Q_\phi \| P_\theta)$ defines a cooperative game, with the dynamics that q_ϕ and p_θ run towards each other.

Since the initial posterior sample $\mathbf{z}_0$ is coupled with the symbolic vector $\mathbf{y}$, it should be the most informative latent variable for inducing the discrete symbol. We can therefore plug in Eq. (7.8) with a mutual information term between $\mathbf{z}_0$ and $\mathbf{y}$: $\mathcal{I}(\mathbf{z}_0, \mathbf{y}) = \mathcal{H}(\mathbf{y}) - \mathcal{H}(\mathbf{y}|\mathbf{z}_0)$, which essentially incorporates the IB as we show below. Given the distribution $Q_\phi(\mathbf{x}, \mathbf{z}_{0:T})$, we can first define the marginal distribution of $\mathbf{z}_0$ as the aggregated posterior by integrating out $\mathbf{z}_{1:T}$: $q_\phi(\mathbf{z}_0) = \mathbb{E}_{q_{\text{data}}(\mathbf{x})}[q_\phi(\mathbf{z}_0|\mathbf{x})]$. The entropy of $\mathbf{z}_0$ and conditional entropy of $\mathbf{z}_0$ on $\mathbf{x}$ then follow as $\mathcal{H}(\mathbf{z}_0)$ and $\mathcal{H}(\mathbf{z}_0|\mathbf{x})$, respectively. Taken together, the KL-Divergence with $\lambda \mathcal{I}(\mathbf{z}_0, \mathbf{y})$ can therefore be parsed as

$$\begin{aligned} \mathcal{L} &= \mathbb{D}_{\text{KL}}(Q_\phi \| P_\theta) - \lambda \mathcal{I}(\mathbf{z}_0, \mathbf{y}) \\ &= \mathcal{C} + \mathcal{L}_{\text{RC}} + \mathcal{L}_{\text{EBM}} + \mathcal{L}_{\text{IB}}, \end{aligned} \tag{7.9}$$

where the constant term

$$\mathcal{C} = -\mathcal{H}(\mathbf{x})$$

does not involve learnable parameters, $\mathcal{L}_{\text{RC}} = -\mathbb{E}_{Q_\phi}[\log p_\beta(\mathbf{x}|\mathbf{z}_0)]$ is the reconstruction loss, $\mathcal{L}_{\text{EBM}} = \mathbb{D}_{\text{KL}}(q_\phi(\mathbf{z}_{0:T}) \| p_\alpha(\mathbf{z}_{0:T}))$ corresponds to matching the aggregated forward trajectory to

the learned reverse diffusion model, and $\mathcal{L}_{\text{IB}} = \mathcal{I}(\mathbf{x}, \mathbf{z}_0) - \lambda \mathcal{I}(\mathbf{z}_0, \mathbf{y})$ is the IB, where $\mathcal{I}(\mathbf{x}, \mathbf{z}_0) = \mathcal{H}(\mathbf{z}_0) - \mathcal{H}(\mathbf{z}_0|\mathbf{x})$ is the mutual information between $\mathbf{x}$ and $\mathbf{z}_0$ under Q_ϕ. Here $q_\phi(\mathbf{z}_{0:T}) = \mathbb{E}_{q_{\text{data}}(\mathbf{x})}[q_\phi(\mathbf{z}_{0:T}|\mathbf{x})]$ denotes the aggregated forward trajectory; $\lambda \geqslant 0$ controls the expressivity of $\mathbf{z}_0$ to $\mathbf{y}$. Please refer to Sec. 7.8.4 for more details.

7.3.3 Geometric Clustering Anchors the Modes

As shown in the previous section, IB provides an elegant solution for inducing the symbolic vector $\mathbf{y}$. In this section, we further introduce an approach that facilitates the emergence of $\mathbf{y}$ from a geometric perspective. To induce a latent space with interpretable structures, ideally, the location of data points in the latent space encodes their semantic meaning, *i.e.*, it indicates the semantic class; semantically similar points should be placed closer and produce the same symbolic vector $\mathbf{y}$. This resembles geometric clustering algorithms: Labels of data points are assigned based on their geometric (typically Euclidean) distance from each other. Below, we show how to realize this intuition in LDEBM.

Let us consider the joint distribution $p_\theta(\mathbf{x}, \mathbf{y})$. We can decompose its log-likelihood into $\log p_\theta(\mathbf{x}, \mathbf{y}) = \log p_\theta(\mathbf{x}) + \log p_\theta(\mathbf{y}|\mathbf{x})$ as in Grathwohl et al. [15], where $\log p_\theta(\mathbf{x})$ is substituted with the ELBO derived in Sec. 7.3.1. $p_\theta(\mathbf{y}|\mathbf{x})$ is the classification model in the latent space: $p_\theta(\mathbf{y}|\mathbf{x}) \approx \mathbb{E}_{q_\phi(\mathbf{z}_0|\mathbf{x})}[p_\alpha(\mathbf{y}|\mathbf{z}_0)]$. $p_\alpha(\mathbf{y}|\mathbf{z}_0)$ is the softmax classifier of $\mathbf{y}$ based on $\mathbf{z}_0$ similarly as in Eq. (7.2), detailed in Sec. 7.8.3. Therefore, we can encode the semantic information from the label $\mathbf{y}$ into $\mathbf{z}_0$ through learning the classifier $p_\alpha(\mathbf{y}|\mathbf{z}_0)$. In case there is full or partial access to the ground-truth semantic class labels, we could directly utilize these labels to supervise the classifier, jointly with the existing ELBO objective. When no label is provided, we generate pseudo label $\hat{\mathbf{y}}$ by clustering $\mathbf{z}_0$, which optimizes $\mathbb{E}_{\mathbf{y}} \log p_\theta(\mathbf{x}, \mathbf{y})$ instead; $\mathbb{E}_{\mathbf{y}}$ is defined by the clustering algorithm. It is akin to the EM algorithm, where geometric clustering serves as a hard-decision E-step to help induce $\mathbf{y}$. In practice, we employ K-means to cluster $\mathbf{z}_0$. In Sec. 7.4.1, we empirically show that this strategy learns a better latent space and significantly alleviates the mode-collapse problem.

7.3.4 Algorithms and Implementation

Algorithm 3 Learning algorithm.

input: initial parameters (α, β, ϕ), learning rate η, observed unlabeled examples $\{\mathbf{x}^{(i)}\}_{i=1}^{M}$, observed labeled examples $\{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\}_{i=M+1}^{M+N}$ (alternative, needed in controllable generation or semi-supervised learning).
repeat
posterior sampling: For each $\mathbf{x}^{(i)}$, sample $\mathbf{z}_0^{(i)} \sim q_\phi(\mathbf{z}_0|\mathbf{x}^{(i)})$ using inference network.
prior sampling: For each $\mathbf{z}_0^{(i)}$, sample diffusion step t from Unif($\{0, ..., T-1\}$), and the perturbed pair $(\tilde{\mathbf{z}}_t^{(i)}, \mathbf{z}_{t+1}^{(i)})$ following Eq. (7.4). Set $\tilde{\mathbf{z}}_t^{(i)}$ as the positive sample $\tilde{\mathbf{z}}_t^{(i)+}$. Initialize the MCMC using $\mathbf{z}_{t+1}^{(i)}$ and update by Eq. (7.12) for K steps to obtain $\tilde{\mathbf{z}}_t^{(i)-}$.
learning prior model: Update α with
$\eta(\sum_i[\nabla_\alpha F_\alpha(\tilde{\mathbf{z}}_t^{(i)+}, t) - \nabla_\alpha F_\alpha(\tilde{\mathbf{z}}_t^{(i)-}, t)] + \lambda \nabla_\alpha \mathcal{I})$.
learning inference and generation models:
Update β and ϕ with Eq. (7.11) and $\nabla_\phi \mathcal{I}$.
if labeled data $(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})$ is available **then**
update $\gamma = (\alpha, \phi)$ **using** $\mathbf{y}^{(i)}$**:**
Learning gradient $\eta \sum_i \nabla_\gamma \log p_\alpha(\mathbf{y}^{(i)}|\mathbf{z}_0^{(i)})$ is provided by ground-truth label.
else if only unlabeled data is available **then**
update $\gamma = (\alpha, \phi)$ **using pseudo-label** $\hat{\mathbf{y}}^{(i)}$**:**
Geometric clustering generates $\hat{\mathbf{y}}^{(i)}$ for each $\mathbf{x}^{(i)}$. $\eta \sum_i \nabla_\gamma \log p_\alpha(\hat{\mathbf{y}}^{(i)}|\mathbf{z}_0^{(i)})$, *i.e.*, the gradient comes from pseudo-label generated by geometric clustering.
end if
until converged.

Training and sampling algorithms For learning the prior model, we have for each $t = 0, 1, ..., T-1$:

$$\begin{aligned} \nabla_\alpha \text{ ELBO}_t = &\mathbb{E}_{q_\phi(\tilde{\mathbf{z}}_t, \mathbf{z}_0|\mathbf{x})}[\nabla_\alpha F_\alpha(\tilde{\mathbf{z}}_t, t)] \\ &- \mathbb{E}_{q_\phi(\mathbf{z}_{t+1}, \mathbf{z}_0|\mathbf{x}), p_\alpha(\tilde{\mathbf{z}}_t|\mathbf{z}_{t+1})}[\nabla_\alpha F_\alpha(\tilde{\mathbf{z}}_t, t)]. \end{aligned} \tag{7.10}$$

Let $\psi = \{\beta, \phi\}$ collect the parameters of the inference (encoder) and generation (decoder) models.

$$\begin{aligned} &\nabla_\psi \text{ ELBO} = \nabla_\psi \mathbb{E}_{q_\phi(\mathbf{z}_0|\mathbf{x})}[\log p_\beta(\mathbf{x}|\mathbf{z}_0) - \log q_\phi(\mathbf{z}_0|\mathbf{x})] \\ &+ \nabla_\phi \mathbb{E}_{q_\phi(\mathbf{z}_{0:T}|\mathbf{x})}\left[\log p(\mathbf{z}_T) + \sum_{t=0}^{T-1} \log p_\alpha(\mathbf{z}_t|\mathbf{z}_{t+1})\right]. \end{aligned} \tag{7.11}$$

Recall that we denote $\tilde{\mathbf{z}}_t = \sqrt{1-\sigma_{t+1}^2}\mathbf{z}_t$. $\mathbb{E}_{p_\alpha(\tilde{\mathbf{z}}_t|\mathbf{z}_{t+1})}$ is approximated by MCMC samples from the prior. $\mathbb{E}_{q_\phi(\mathbf{z}_0|\mathbf{x})}$ is approximated by samples from the inference network. We also add the gradient from $\mathcal{I}(\mathbf{z}_0, \mathbf{y})$, denoted as $\nabla \mathcal{I}$, to Eqs. (7.10) and (7.11) during training to

incorporate IB. Please see Sec. 7.8.5 for detailed derivations.

Note that the expectation in Eq. (7.10) requires MCMC sampling (*e.g.*, Langevin dynamics [67]) of the prior model. For a target distribution $\pi(\tilde{\mathbf{z}})$, the dynamics iterates $\tilde{\mathbf{z}}^{k+1} = \tilde{\mathbf{z}}^k + \frac{s^2}{2}\nabla_{\tilde{\mathbf{z}}} \log \pi(\tilde{\mathbf{z}}^k) + s\boldsymbol{\epsilon}^k$, where k indexes the iteration of the dynamics, s is a small step size, and $\boldsymbol{\epsilon}^k \sim \mathcal{N}(0, \mathbf{I})$ is the Gaussian noise. In this work, we follow the heuristics in Gao et al. [14] and set the step size $s_{t+1} = b\sigma_{t+1}c_{t+1}$, where $b < 1$ is a tuned hyperparameter, and $c_j = \sqrt{(\prod_{i=1}^{j} \sigma_i)/\sigma_1}$ is a scaling factor. Let t index the diffusion step; K steps of Langevin dynamics thus iterates

$$\begin{aligned}\tilde{\mathbf{z}}_t^{k+1} = \tilde{\mathbf{z}}_t^k + \frac{b^2\sigma_{t+1}^2 c_{t+1}^2}{2}\Big(\nabla_{\tilde{\mathbf{z}}} F_\alpha(\tilde{\mathbf{z}}_t^k, t) \\ - \frac{\tilde{\mathbf{z}}_t^k - \mathbf{z}_{t+1}}{\sigma_{t+1}^2}\Big) + b\sigma_{t+1}c_{t+1}\boldsymbol{\epsilon}^k.\end{aligned} \tag{7.12}$$

At the final reverse step $t = T-1$, the conditional energy-based model in this formulation does not depend on $\mathbf{z}_T$. Accordingly, the score term in parentheses in Eq. (7.12) is replaced by $\nabla_{\tilde{\mathbf{z}}} F_\alpha(\tilde{\mathbf{z}}_{T-1}^k, T-1) - \tilde{\mathbf{z}}_{T-1}^k/\sigma_T^2$.

Algorithm 4 Synthesizing algorithm.

input: $\mathbf{z}_T \sim \mathcal{N}(0, \mathbf{I})$
output: $\mathbf{z}_0$
for $t = T-1$ **to** $t = 0$ **do**
  Initialize $\tilde{\mathbf{z}}_t = \mathbf{z}_{t+1}$.
  for $k = 1$ **to** K **do**
    Update $\tilde{\mathbf{z}}_t$ using Eq. (7.12).
  end for
  $\mathbf{z}_t = \tilde{\mathbf{z}}_t/\sqrt{1-\sigma_{t+1}^2}$
end for

In principle, training the model amounts to maximizing the ELBO in Eq. (7.7), which requires a summation over all the diffusion steps; it involves sampling a full forward trajectory. To make the training simpler and more efficient, following Ho et al. [22], we randomly choose one diffusion step from the summation to optimize at each training iteration. After training, we initialize the reverse trajectory from Gaussian white noise. The synthesized sample at each step serves to initialize an MCMC that samples from the model of the previous step.

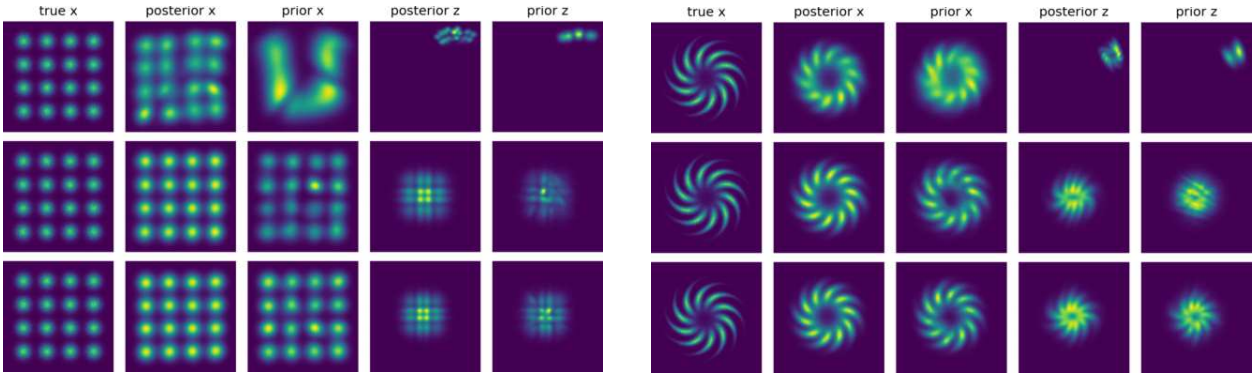
The learning and synthesizing algorithms are summarized in Algs. 3 and 4, respectively.

Implementation For the K-means algorithm, we use the implementation of Johnson et al. [26], which explicitly deals with the empty clusters and trivial parameterization problems. To emphasize that the proposed model shows better capability of modeling latent space, we use the same encoder and decoder as Pang & Wu [43] for all the experiments. We use a shared network $F_\alpha(\tilde{\mathbf{z}}_t, t)$ for each $t = 0, 1, ..., T-1$; $T = 6$; t is encoded by sinusoidal position embedding as in Ho et al. [22], and we set σ_t^2 to increase linearly. For Langevin dynamics, we use $K = 50$ and $b^2 = 0.002$ throughout the experiments. See Sec. 7.9.1 for network architecture and further training details.

7.4 Experiments

Through a series of experiments, we empirically examine the capability of our model for generative modeling and interpretability on text modeling tasks. Please refer to Sec. 7.9.2 for additional experiment settings and baselines.

7.4.1 Generative Modeling



(a) Gaussian mixture (b) Pinwheel distribution

Figure 7.2: **Evaluation on 2D synthetic data**: a mixture of 16 Gaussians (left panel) and a 10-arm pinwheel-shaped distribution (right panel). In each panel, the top, middle, and bottom rows display densities learned by SVEBM-IB, our model without geometric clustering, and our full model, respectively. From left to right, each row shows the data distribution and the KDEs of: $\mathbf{x}$ generated from amortized posterior $\mathbf{z}$ samples, $\mathbf{x}$ generated from MCMC-sampled prior $\mathbf{z}$ samples, posterior $\mathbf{z}$ samples, and prior $\mathbf{z}$ samples.

2D synthetic data We first perform experiments of our model on 2D synthetic datasets as a sanity check to validate our assumptions; results are displayed in Fig. 7.2. The gap between LDEBM and SVEBM is very clear. As mentioned in Sec. 7.1, for more complex datasets (*e.g.*, datasets with more modes or more complex data structure), SVEBM struggles to capture regularities in the data; the model is prone to collapse, which features an exploding KL-term and poor performance on generation. We provide more results that show the full evolution of these models during training with more discussions in Sec. 7.9.3. In contrast, LDEBM without geometric clustering already overcomes this problem, performing relatively well in terms of modeling both *posterior* $\mathbf{x}$ and *prior* $\mathbf{x}$. Although LDEBM without geometric clustering faithfully reconstructs the data and shows significant improvement on generation quality, the generated distribution can be slightly distorted, and some modes are missing. The problem is clearer in the latent space: Mode-collapse occurs in the *prior* $\mathbf{z}$ distribution, where the latent structure is broken. LDEBM with geometric clustering maintains the number of modes as in the data distribution and induces a highly-structural latent space, echoing our intuition in Sec. 7.3.3. Fig. 7.3 shows the structural similarity between data distribution and the learned latent distribution.

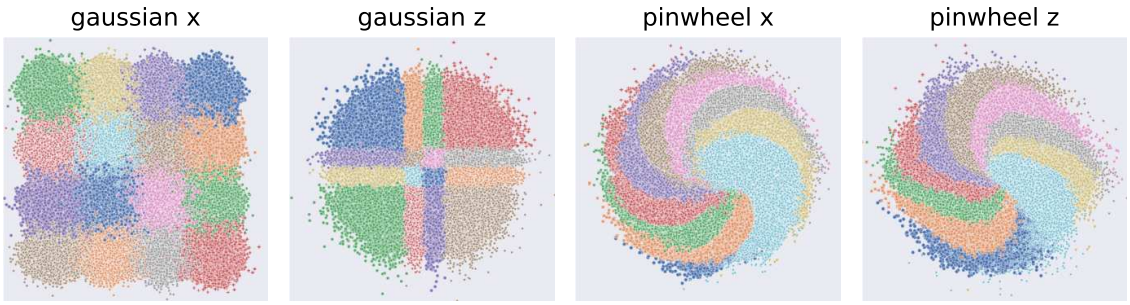


Figure 7.3: **Visualization of color-coded data points.** We visualize data points and the corresponding inferred latent variables of two 2D synthetic datasets (*gaussian* and *pinwheel*). Data points with different labels are assigned with different colors.

Language generation Following previous state-of-the-art competitors [43, 53, 76], we evaluate the quality of generation on a real-world text dataset, Penn Treebanks (PTB) [34] as pre-processed by Mikolov et al. [37]. We report four metrics of the generation performance: Reverse Perplexity (rPPL) [74], BELU [47], Word-Level KL Divergence (wKL), and Negative

Log-Likelihood (NLL); Tab. 7.1 summarizes results.

Table 7.1: **Results of language generation on PTB dataset.** We highlight our model results in gray color. The best and second-best performances are marked in bold numbers and underlines, respectively; tables henceforth follows this format.

Model	**rPPL**$^{\downarrow}$	**BLEU**$^{\uparrow}$	**wKL**$^{\downarrow}$	**NLL**$^{\downarrow}$
Test Set	-	100.0	0.14	-
RNN-LM	-	-	-	101.21
AE	730.81	10.88	0.58	-
VAE	686.18	3.12	0.50	100.85
DAE	797.17	3.93	0.58	-
DVAE	744.07	1.56	0.55	101.07
DI-VAE	310.29	4.53	0.24	108.90
semi-VAE	494.52	2.71	0.43	100.67
semi-VAE $+\mathcal{I}$	260.28	5.08	0.20	107.30
GM-VAE	983.50	2.34	0.72	99.44
GM-VAE $+\mathcal{I}$	287.07	6.26	0.25	103.16
DGM-VAE	257.68	8.17	0.19	104.26
DGM-VAE $+\mathcal{I}$	247.37	8.67	0.18	105.73
SVEBM	180.71	9.54	0.17	95.02
SVEBM-IB	177.59	9.47	0.16	94.68
Ours w/o GC	<u>168.32</u>	<u>11.12</u>	<u>0.07</u>	**79.84**
Ours	**164.57**	**11.16**	**0.06**	<u>82.38</u>

The proposed model, either w/ or w/o geometric clustering, demonstrates the best performance on reconstruction (highest BLEU) and fitting capacity (lowest NLL) than all baseline models. Moreover, the higher expressivity of our models enables the generation of high-quality sentences. The lowest rPPL indicates that our models can further improve over these strong baselines on fluency and diversity of generated text; the lowest wKL indicates that the word distribution of the generated sentences is the most consistent with that of the original data.

Sentence completion Further, we show that the trained model enables text completion on a masked JerichoWorld dataset [1]. We perform conditional sampling in the latent space to complete the masked sentences; please see more details in Sec. 7.9.3 and Tab. 7.2.

Table 7.2: **Sentence completion on JerichoWorld dataset.** The gray words in the input sentences are masked with <unk> token.

Input	... A bathroom lies to the south, while a door to the east leads to the living room. On the bed are a driver's license, some keys and a wallet On the end table is a telephone.
Pred.	... A bathroom lies to the south, while a door to the east leads to the living room. On the bed is a wallet. On the end table are a telephone and some keys.
Input	... All around you the crowd is in a state of pandemonium. The paths of least resistance are up, down and west.
Pred.	... All around you the crowd is in a state of pandemonium. The paths of least resistance are down and east.

7.4.2 Interpretable Text Modeling

In this section, we move on to evaluate our model on the interpretability of text modeling.

Unsupervised text attributes discovery First, we examine the efficacy of our model on the unsupervised text attributes discovery task. We assess the model on the DD dataset [33], a chat-oriented dataset of 13,118 daily conversations with human-annotated dialog action and emotion labels for the utterances. The interpretability is evaluated through the ability to unsupervisedly capture the utterance attributes of DD. We flatten the dialogues for text modeling and use $p_\theta(\mathbf{y}|\mathbf{x})$ to infer the utterance label. In particular, we take the argmax of the classification head as the inferred label. Following Zhao et al. [76], we use homogeneity to evaluate the consistency between ground-truth action and emotion labels and those inferred from our model. Tab. 7.3 displays the results of our model and baselines. It shows that the proposed model outperforms other baselines in reconstruction by a large margin and achieves much higher homogeneity on both the dialog action and emotion. The superior performance of LDEBM equipped with latent space geometric clustering again verifies our intuition in Sec. 7.3.3.

Table 7.3: **Results of interpretable text modeling on DD.** We use mutual information (MI), BLEU, and homogeneity with actions and emotions for evaluation.

Model	**MI**$^{\uparrow}$	**BLEU**$^{\uparrow}$	**Act.**$^{\uparrow}$	**Emo.**$^{\uparrow}$
DI-VAE	1.20	3.05	0.18	0.09
semi-VAE	0.03	4.06	0.02	0.08
semi-VAE $+\mathcal{I}$	1.21	3.69	0.21	0.14
GM-VAE	0.00	2.03	0.08	0.02
GM-VAE $+\mathcal{I}$	1.41	2.96	0.19	0.09
DGM-VAE	0.53	7.63	0.11	0.09
DGM-VAE $+\mathcal{I}$	1.32	7.39	0.23	0.16
SVEBM	0.01	11.16	0.03	0.01
SVEBM-IB	2.42	10.04	0.59	0.56
Ours w/o GC	<u>2.44</u>	<u>16.72</u>	<u>0.65</u>	<u>0.63</u>
Ours	**3.94**	**28.75**	**0.74**	**0.74**

Conditional response generation Next, we evaluate our model on dialog generation with SMD [10] and DD datasets. We evaluate the quality of generated responses using BLEU and three word-embedding-based topic similarity metrics [53]: embedding average [38], embedding extrema [12], and embedding greedy [50]. Tab. 7.4 shows that LDEBM achieves competitive performance compared with SVEBM-IB on SMD and outperforms the strong baselines on all metrics on DD; see qualitative examples in Tabs. 7.5 and 7.6.

Table 7.4: **Dialog evaluation results on SMD and DD.** Models are assessed using four metrics: BLEU, average, extrema, and greedy word embedding based similarity.

Data	Model	**BLEU**$^{\uparrow}$	**Avg.**$^{\uparrow}$	**Extr.**$^{\uparrow}$	**Grdy.**$^{\uparrow}$
SMD	DI-VAE	7.06	76.17	43.98	60.92
	DGM $+\mathcal{I}$	10.16	78.93	48.14	64.87
	SVE-IB	**12.01**	**80.88**	<u>51.35</u>	<u>67.12</u>
	w/o GC	11.44	<u>80.16</u>	51.26	66.51
	Ours	<u>11.51</u>	**80.88**	**51.57**	**67.13**
DD	DGM $+\mathcal{I}$	2.19	74.73	<u>45.85</u>	<u>64.28</u>
	SVE-IB	<u>2.23</u>	<u>77.37</u>	43.32	63.99
	Ours	**3.72**	**78.89**	**46.19**	**65.87**

Sentence sentiment control Finally, we inspect the capability of our model for controllable generation on Yelp reviews, pre-processed by [31]. The Yelp dataset is of larger scale, containing 180,000 negative reviews and 270,000 positive ones. For a controllable

Table 7.5: **Samples of unsupervisedly discovered action categories and corresponding utterances on SMD.**

Action	Request-weather
Utterance	I need to know if it is going to be foggy in Fresno today and tomorrow car.
	Manhattan, please. Will it be cloudy on Monday?
	I need current weather data about New York, specifically information about the temperature.
Action	Request-city
Utterance	In what city are you interested?
	What city would you like to know the weather about?
	Okay, what city should I look in?

Table 7.6: **Dialog cases generated by LDEBM given the context.** On SMD, we provide the same context but with different $\mathbf{y}$ values to generate each response; actions indicated by $\mathbf{y}$ are listed in parentheses. On DD, LDEBM can well capture the dialog topic; we provide the ground-truth response in each case for reference.

SMD	
Ctx.	*User:* What gas stations are here? *Sys:* There is a Chevron.
Ref.	That's good! Please pick the quickest route to get there and avoid all heavy traffic!
Pred.	(Req.-address) What is the address? (Req.-route) Please set the quickest route to go.
DD	
Ctx.	*A:* Hi. Have you got a personal computer? *B:* Certainly. What ' s the matter? *A:* I wonder if you often trade with others on the internet.
Ref.	Sure. I often buy things or do business through it without going out to the physical stores.
Pred.	Yes, but I think it is a little different way.

generation process, the symbolic vector $\mathbf{y}$ is provided to guide the sampling in latent space; see details in Sec. 7.9.3. Following Pang & Wu [43], we train the model with sentiment

supervision and use the same pre-trained classifier to determine the sentiment of the generated sentence. The pre-trained classifier has an accuracy of 98.5% on the testing data and thus can accurately evaluate the sentiment of given sentences. The quantitative and qualitative results are summarized in Tabs. 7.7 and 7.8, respectively. LDEBM generates positive and negative reviews with near-saturated accuracy, significantly outperforming all the baselines.

Table 7.7: **Accuracy of sentence attribute control on Yelp.**

Model	**Overall**$^{\uparrow}$	**Positive**$^{\uparrow}$	**Negative**$^{\uparrow}$
DGM-VAE $+\mathcal{I}$	64.7%	95.3%	34.0%
CGAN	76.8%	94.9%	58.6%
SVEBM-IB	90.1%	95.1%	85.2%
Ours	**99.0%**	**98.8%**	**99.1%**

Table 7.8: **Generated positive and negative reviews on Yelp.**

Positive	The food here was very tasty and our server was very attentive.
	I was very satisfied for my birthday party!
	Definitely the best Philly Cheesesteaks I've ever been.
	They are the best customer service ever!
Negative	Ugh the staff is so incompetent and rude.
	It just can't make it worse.
	Avoid this company at all costs.
	Just ruined the experience with a horrible attitude on it.

7.4.3 Semi-supervised Classification

In this experiment, we switch from neural sequence models used in previous experiments to neural document models [7, 36]; we show our model can be similarly extended to semi-supervised settings as in Pang & Wu [43] and benefit from the better learned latent space. Our model is evaluated on AGNews [73], a popular benchmark for text classification with 127,600 documents from 4 classes. Tab. 7.9 shows that LDEBM performs the best when

having only partial access to ground-truth data labels; it further validates the proposed formulation for learning a well-structured latent space.

Table 7.9: **Accuracy on AGNews.** We report semi-supervised classification accuracy with varied number of labeled data.

Model	200	500	2500	10000
Glove-ID	70.4	78.0	84.1	87.1
Glove-OD	68.8	78.8	85.3	88.0
VAMPIRE	82.9	84.5	85.8	87.7
Hard EM	83.9	84.6	85.1	86.9
CatVAE	84.6	85.7	86.3	87.5
SVEBM	84.5	84.7	86.0	88.1
SVEBM-IB	86.4	87.4	87.9	88.6
Ours	**87.4**	**88.1**	**89.2**	**90.1**

7.5 Discussion and Related Work

Text modeling VAE has been one of the most prominent latent variable models for generative modeling [28, 49]. It is first applied to text modeling in Bowman et al. [4], followed by a wide range of work attacking challenging text generation problems using the shared framework of VAE. These include dialog generation [11, 51, 52, 68, 75, 76], machine translation [72], text summarization [32], and paraphrase generation [16]. In parallel, extensive efforts have been made to address issues like posterior collapse [4, 13, 20, 21, 30, 74, 75] and mode-collapse [53] in training VAE to further improve the language modeling performance and text generation quality.

The interpretability of the generation process is naturally brought up as the generation quality achieves impressive progress. Recently, Zhao et al. [76], Shi et al. [53], and Pang & Wu [43] have explored interpretable text generation with deliberately designed latent spaces. Zhao et al. [76] use a discrete latent space to capture dialog actions; Shi et al. [53] adopt a mixture of Gaussians as the VAE prior. To further improve the expressivity of latent space, Pang & Wu [43] propose a symbol-vector coupling energy-based prior to learn

a structured latent space. The coupling formulation provides a natural interface to induce the symbolic representation, which eliminates the need of training extra auxiliary inference networks for symbol induction. Our formulation inherits the advantages from Pang & Wu [43] by choosing an appropriate symbol-vector coupling scheme and principally incorporating the IB. We further develop a geometric clustering-based regularization that complements the IB; it alleviates the mode-collapse problem in variational learning of the latent space model.

Energy-based model EBMs [19, 41, 42, 69] have drawn growing interest in generative modeling. As an interesting branch, Pang et al. [44] learn an EBM in the latent space as a prior model for continuous latent variables; it greatly improves the expressivity over non-informative priors and demonstrates strong performance on downstream tasks, *e.g.*, image segmentation, molecule generation, and trajectory prediction [24, 25, 45, 46, 71]. However, both EBM and latent space EBM require MCMC sampling to learn the model. The degenerate sampling quality in practice can lead to poor generation quality and instability in training [9, 15]. We leverage diffusion models as a cure for the vanilla latent space EBM in this work; the proposed model shows reliable sampling quality in practice.

Diffusion model Diffusion models [14, 22, 55], originating from Sohl-Dickstein et al. [55], learn from a sequence of noise-perturbed versions of the data. From such perturbed data, one can learn the conditional model to invert the diffusion process and generate high-quality samples given noisy inputs. On another front, Song & Ermon [56, 57], Song et al. [58] extend the denoising score matching method [63], modeling the diffusion process with continuous time step. Our formulation moves the model to the latent space in a variational framework with two benefits: (a) learning in a lower-dimensional space enables faster sampling and better convergence, and (b) learning the diffusion model in a continuous latent space avoids the discreteness of text data, which hinders the direct application of vanilla diffusion models to text modeling [2].

Similar to our work, Wehenkel & Louppe [66], Sinha et al. [54], Nie et al. [40], and Vahdat et al. [62] have proposed to learn a diffusion model in the latent space. Specifically,

Wehenkel & Louppe [66] empirically demonstrate that a diffusion prior can perform better than the non-informative Gaussian prior when jointly trained with a VAE. Sinha et al. [54] combine contrastive learning with diffusion models in the latent space of VAEs for controllable generation. Nie et al. [40] and Vahdat et al. [62] extend the idea of Song et al. [58] in the latent space: Nie et al. [40] perform controllable image generation by training a latent energy-based attribute classifier on a pre-trained generator; Vahdat et al. [62] train score-based denoising diffusion models in the latent space of a powerful VAE [61]. Both methods have achieved very impressive image generation results. However, the listed methods are generally limited to image generation with tailored or pre-trained encoders and decoders. In contrast, our method is a general improvement for the sampling quality of latent space EBM; it is not restricted to a certain data type. Moreover, the proposed model can be trained from scratch to form a well-structured latent space, in contrast to Vahdat et al. [62] and Nie et al. [40] which require a pre-learned latent space.

7.6 Conclusion

We presented LDEBM, a novel symbiosis between symbol-vector coupling EBM and diffusion model that offers the best of both worlds. The proposed model shows reliable sampling quality, learns a well-structured and meaningful latent space from scratch, and can be flexibly extended to scenarios where data labels are available. It demonstrates superior performance over strong baselines on interpretable text modeling. We hope our work inspires future research along this challenging but promising research direction. A potential follow-up research problem is to reuse powerful pre-trained language models. One could consider integrating pre-trained models with our method to realize high-quality controllable generation at low computational cost.

7.7 Acknowledgments for This Research Chapter

Chapter 7 is adapted from Peiyu Yu, Sirui Xie, Xiaojian Ma, Baoxiong Jia, Bo Pang, Ruiqi Gao, Yixin Zhu, Song-Chun Zhu, and Ying Nian Wu, "Latent Diffusion Energy-Based Model for Interpretable Text Modelling," *Proceedings of the 39th International Conference on Machine Learning*, Proceedings of Machine Learning Research, vol. 162, pp. 25702–25720, 2022.

For the research presented in this chapter, Peiyu Yu was responsible for conceiving the method, designing and conducting the experiments, and writing the manuscript. Sirui Xie, Xiaojian Ma, Baoxiong Jia, Bo Pang, Ruiqi Gao, Yixin Zhu, Song-Chun Zhu, and Ying Nian Wu participated in research discussions.

Ying Nian Wu was supported by NSF grant DMS-2015577. The authors thank the anonymous reviewers for their constructive comments.

7.8 Additional Derivations and Discussion

7.8.1 Derivation of Conditional Energy-Based Models

We first define the marginal EBMs at each diffusion step:

$$\begin{cases} p_\alpha(\mathbf{z}_t) = \dfrac{1}{Z_{\alpha,t}} \exp(F_\alpha(\mathbf{z}_t, t)) p_0(\mathbf{z}_t), \ t = T-1 \\ p_\alpha(\mathbf{z}_t) = \dfrac{1}{Z_{\alpha,t}} \exp(F_\alpha(\mathbf{z}_t, t)), \ t = 0, 1, ..., T-2 \end{cases} \tag{7.13}$$

where the marginal energy term is in a log-sum-exponential form,

$$F_\alpha(\mathbf{z}_t, t) = \log \sum_{\mathbf{y}} \exp(\langle \mathbf{y}, f_\alpha(\mathbf{z}_t, t) \rangle).$$

It serves to aggregate the energy score from each category. Of note, the marginal EBM corresponding with the last diffusion step has a slightly different definition. We set this term as exponential tilting of a non-informative Gaussian prior $p_0(\mathbf{z}_t)$ which helps to stabilize training in practice.

Recall that $\mathbf{z}_{t+1} = \sqrt{1-\sigma_{t+1}^2}\mathbf{z}_t + \sigma_{t+1}\boldsymbol{\epsilon}_{t+1}$. Let $\tilde{\mathbf{z}}_t = \sqrt{1-\sigma_{t+1}^2}\mathbf{z}_t$. For $t = 0, 1, ..., T-2$, we have

$$\begin{aligned} p_\alpha(\tilde{\mathbf{z}}_t | \mathbf{z}_{t+1}) &= \frac{p_\alpha(\tilde{\mathbf{z}}_t) p(\mathbf{z}_{t+1} | \tilde{\mathbf{z}}_t)}{p_\alpha(\mathbf{z}_{t+1})} \\ &= \frac{1}{\tilde{Z}_{\alpha,t}} \frac{\exp(F_\alpha(\tilde{\mathbf{z}}_t, t))}{p_\alpha(\mathbf{z}_{t+1})} \exp\left(-\frac{1}{2\sigma_{t+1}^2} ||\tilde{\mathbf{z}}_t - \mathbf{z}_{t+1}||^2 \right) \\ &= \frac{1}{\tilde{Z}_{\alpha,t}(\mathbf{z}_{t+1})} \exp\left(F_\alpha(\tilde{\mathbf{z}}_t, t) - \frac{1}{2\sigma_{t+1}^2} ||\tilde{\mathbf{z}}_t - \mathbf{z}_{t+1}||^2 \right), \end{aligned} \tag{7.14}$$

where $\tilde{Z}_{\alpha,t} = (2\pi\sigma_{t+1}^2)^{\frac{n}{2}} Z_{\alpha,t}$; we slightly abuse the notation and use $p(\mathbf{z}_{t+1}|\tilde{\mathbf{z}}_t)$ to represent the forward transition $q(\mathbf{z}_{t+1}|\mathbf{z}_t)$ defined in Eq. (7.4) for notation consistency.

The diffused samples at time step T are close to Gaussian white noise; $p_\alpha(\tilde{\mathbf{z}}_{T-1}|\mathbf{z}_T)$ therefore falls to its marginal distribution $p(\tilde{\mathbf{z}}_{T-1})$ defined in Eq. (7.13).

7.8.2 Derivation of the ELBO

Recall that the ELBO in SVEBM is

$$\begin{aligned}
\text{ELBO}_{\theta,\phi} &= \log p_\theta(\mathbf{x}) - \mathbb{D}_{\text{KL}}(q_\phi(\mathbf{z}|\mathbf{x}) \| p_\theta(\mathbf{z}|\mathbf{x})) \\
&= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\beta(\mathbf{x}|\mathbf{z})] - \mathbb{D}_{\text{KL}}(q_\phi(\mathbf{z}|\mathbf{x}) \| p_\alpha(\mathbf{z})) \\
&= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log p_\beta(\mathbf{x}|\mathbf{z}) - \log q_\phi(\mathbf{z}|\mathbf{x}) + \log p_\alpha(\mathbf{z})\right],
\end{aligned} \tag{7.15}$$

where $\mathbb{D}_{\text{KL}}$ denotes the Kullback-Leibler divergence. Let us consider the full trajectory of the perturbed samples $\mathbf{z}_0, \mathbf{z}_1, ..., \mathbf{z}_T$. The above equation can be written as

$$\begin{aligned}
\text{ELBO}_{\theta,\phi} = \mathbb{E}_{q_\phi(\mathbf{z}_0|\mathbf{x})}\left[\log p_\beta(\mathbf{x}|\mathbf{z}_0) - \log q_\phi(\mathbf{z}_0|\mathbf{x})\right] \\
+ \mathbb{E}_{q_\phi(\mathbf{z}_0|\mathbf{x})}\left[\log \int_{\mathbf{z}_{1:T}} p_\alpha(\mathbf{z}_{0:T}) d\mathbf{z}_{1:T}\right],
\end{aligned} \tag{7.16}$$

where the last term is further lower-bounded by introducing the forward trajectory distribution; the inequality holds by applying Jensen's Inequality:

$$\begin{aligned}
& \mathbb{E}_{q_\phi(\mathbf{z}_0|\mathbf{x})}\left[\log \int_{\mathbf{z}_{1:T}} p_\alpha(\mathbf{z}_{0:T}) d\mathbf{z}_{1:T}\right] \\
= \; & \mathbb{E}_{q_\phi(\mathbf{z}_0|\mathbf{x})}\left[\log \int_{\mathbf{z}_{1:T}} q(\mathbf{z}_{1:T}|\mathbf{z}_0) \frac{p_\alpha(\mathbf{z}_{0:T})}{q(\mathbf{z}_{1:T}|\mathbf{z}_0)} d\mathbf{z}_{1:T}\right] \\
\geqslant \; & \mathbb{E}_{q_\phi(\mathbf{z}_0|\mathbf{x})}\left[\int_{\mathbf{z}_{1:T}} q(\mathbf{z}_{1:T}|\mathbf{z}_0) \log \frac{p_\alpha(\mathbf{z}_{0:T})}{q(\mathbf{z}_{1:T}|\mathbf{z}_0)} d\mathbf{z}_{1:T}\right] \\
= \; & \mathbb{E}_{q_\phi(\mathbf{z}_0|\mathbf{x}), q(\mathbf{z}_{1:T}|\mathbf{z}_0)}\left[\log \frac{p_\alpha(\mathbf{z}_{0:T})}{q(\mathbf{z}_{1:T}|\mathbf{z}_0)}\right].
\end{aligned} \tag{7.17}$$

Further, we can decompose the joint distribution of forward and backward trajectories as

$$\begin{aligned}
&\mathbb{E}_{q_\phi(\mathbf{z}_0|\mathbf{x}),q(\mathbf{z}_{1:T}|\mathbf{z}_0)}\left[\log\frac{p_\alpha(\mathbf{z}_{0:T})}{q(\mathbf{z}_{1:T}|\mathbf{z}_0)}\right]= \\
&\mathbb{E}_{q_\phi(\mathbf{z}_0|x),q(\mathbf{z}_{1:T}|\mathbf{z}_0)}\left[\log p(\mathbf{z}_T)+\sum_{t=0}^{T-1}\log\frac{p_\alpha(\mathbf{z}_t|\mathbf{z}_{t+1})}{q(\mathbf{z}_{t+1}|\mathbf{z}_t)}\right]= \\
&\mathbb{E}\left[\log p(\mathbf{z}_T)+\sum_{t=0}^{T-1}\log p_\alpha(\mathbf{z}_t|\mathbf{z}_{t+1})\right]+\sum_{t=0}^{T-1}\mathcal{H}(\mathbf{z}_{t+1}|\mathbf{z}_t),
\end{aligned} \tag{7.18}$$

where $p(\mathbf{z}_T)$ is standard Gaussian distribution; $\mathbb{E}$ is the abbreviation of $\mathbb{E}_{q_\phi(\mathbf{z}_0|x),q(\mathbf{z}_{1:T}|\mathbf{z}_0)}$. $\mathcal{H}(\mathbf{z}_{t+1}|\mathbf{z}_t), t=0,\ldots,T-1$ is the conditional entropy under the forward trajectory distribution. We obtain $\mathbf{z}_t$ by sampling $\tilde{\mathbf{z}}_t$ from $p_\alpha(\tilde{\mathbf{z}}_t|\mathbf{z}_{t+1})$ and then applying $\mathbf{z}_t=\tilde{\mathbf{z}}_t/\sqrt{1-\sigma_{t+1}^2}$; the reverse trajectory in our model is primarily defined by $p_\alpha(\tilde{\mathbf{z}}_t|\mathbf{z}_{t+1})$ for $t>0$. We use $[\mathbf{z}_t|\mathbf{z}_{t+1}]$ to represent this process in the following sections; we may interchangeably use the notation of $\tilde{\mathbf{z}}_t$ and $\mathbf{z}_t$ for simplicity.

Note that the entropies can be analytically computed and do not involve learnable parameters. The joint training of inference, prior and generation models can be largely reduced to finding the agreement of the forward and reverse Markov transitions defined by q_ϕ and p_θ respectively.

7.8.3 Detailed Discussion of Symbol Coupling

In Sec. 7.2, we briefly describe how to introduce the symbolic one-hot vector $\mathbf{y}$. Since only $\mathbf{z}_0$ is connected with $\mathbf{y}$, we can first define the joint prior $p_\alpha(\mathbf{y},\mathbf{z}_0)$ as in Eq. (7.13) by substituting $F_\alpha(\tilde{\mathbf{z}}_0,0)$ with $\langle\mathbf{y},f_\alpha(\tilde{\mathbf{z}}_0,0)\rangle$. Then the conditional symbol-vector coupling joint distribution follows as

$$\begin{aligned}
p_\alpha(\mathbf{y},\mathbf{z}_0|\mathbf{z}_1)=\frac{1}{\tilde{Z}_{\alpha,0}(\mathbf{z}_1)}&\exp\left(\langle\mathbf{y},f_\alpha(\tilde{\mathbf{z}}_0,0)\rangle\right) \\
&\exp\left(-\frac{1}{2\sigma_1^2}||\tilde{\mathbf{z}}_0-\mathbf{z}_1||^2\right).
\end{aligned} \tag{7.19}$$

Note that $p_\alpha(\mathbf{y}, \mathbf{z}_0|\mathbf{z}_1) = p_\alpha(\mathbf{y}|\mathbf{z}_0)p_\alpha(\mathbf{z}_0|\mathbf{z}_1)$, *i.e.*, $\mathbf{z}_0$ is sufficient for inferring $\mathbf{y}$ in this formulation:

$$\begin{aligned} p_\alpha(\mathbf{y}|\mathbf{z}_0, \mathbf{z}_1) &= \frac{p_\alpha(\mathbf{y}, \mathbf{z}_0|\mathbf{z}_1)}{p_\alpha(\mathbf{z}_0|\mathbf{z}_1)} \\ &= \frac{\exp\left(\langle \mathbf{y}, f_\alpha(\tilde{\mathbf{z}}_0, 0)\rangle\right)}{\exp\left(F_\alpha(\tilde{\mathbf{z}}_0, 0)\right)}, \end{aligned} \tag{7.20}$$

so that given $\mathbf{z}_0$,

$$p_\alpha(\mathbf{y}|\mathbf{z}_0) \propto \exp(\langle \mathbf{y}, f_\alpha(\tilde{\mathbf{z}}_0, 0)\rangle). \tag{7.21}$$

It similarly becomes a softmax classifier where $f_\alpha(\tilde{\mathbf{z}}_0, 0)$ provides the logit scores for the K categories.

7.8.4 Derivation of the Information Bottleneck

We first define the mutual information term between $\mathbf{z}_0$ and $\mathbf{y}$. Consider the joint distribution of $\mathbf{x}, \mathbf{z}_0$ and $\mathbf{y}$, $\pi(\mathbf{y}, \mathbf{z}_0, \mathbf{x}) = p_\alpha(\mathbf{y}|\mathbf{z}_0)q_\phi(\mathbf{z}_0|\mathbf{x})q_{\text{data}}(\mathbf{x})$; the mutual information $\mathcal{I}(\mathbf{z}_0, \mathbf{y})$ defined under π then follows as:

$$\begin{aligned} \mathcal{I}(\mathbf{z}_0, \mathbf{y}) &= \mathcal{H}(\mathbf{y}) - \mathcal{H}(\mathbf{y}|\mathbf{z}_0) \\ &= -\sum_{\mathbf{y}} q(\mathbf{y}) \log q(\mathbf{y}) \\ &\quad + \mathbb{E}_{q_\phi(\mathbf{z}_0)} \sum_{\mathbf{y}} p_\alpha(\mathbf{y}|\mathbf{z}_0) \log p_\alpha(\mathbf{y}|\mathbf{z}_0), \end{aligned} \tag{7.22}$$

where $q(\mathbf{y}) = \mathbb{E}_{q_\phi(\mathbf{z}_0)}[p_\alpha(\mathbf{y}|\mathbf{z}_0)]$; $p_\alpha(\mathbf{y}|\mathbf{z}_0)$ is the softmax probability over K categories in Eq. (7.21).

We then show how to obtain the quantities defined in Sec. 7.3.2. For the marginal distribution of $\mathbf{z}_0$:

$$\begin{aligned} q_\phi(\mathbf{z}_0) &= \int_{\mathbf{x}, \mathbf{z}_{1:T}} Q_\phi(\mathbf{x}, \mathbf{z}_{0:T}) d\mathbf{x} d\mathbf{z}_{1:T} \\ &= \mathbb{E}_{q_{\text{data}}(\mathbf{x})}[q_\phi(\mathbf{z}_0|\mathbf{x})]. \end{aligned} \tag{7.23}$$

The entropy and conditional entropy of $\mathbf{z}_0$ are thus

$$\begin{aligned}\mathcal{H}(\mathbf{z}_0) &= -\mathbb{E}_{q_\phi(\mathbf{z}_0)}[\log q_\phi(\mathbf{z}_0)];\\ \mathcal{H}(\mathbf{z}_0|\mathbf{x}) &= -\mathbb{E}_{Q_\phi(\mathbf{x},\mathbf{z}_0)}[\log q_\phi(\mathbf{z}_0|\mathbf{x})].\end{aligned} \tag{7.24}$$

We can now decompose the joint KL divergence in Eq. (7.8). Writing $q_\phi(\mathbf{z}_{0:T}) = \mathbb{E}_{q_{\text{data}}(\mathbf{x})}[q_\phi(\mathbf{z}_{0:T}|\mathbf{x})]$ for the aggregated forward trajectory and using the Markov factorization in Eq. (7.4), we obtain

$$\begin{aligned}\mathbb{D}_{\text{KL}}(Q_\phi \| P_\theta) = &-\mathcal{H}(\mathbf{x}) - \mathcal{H}(\mathbf{z}_0|\mathbf{x}) - \sum_{t=0}^{T-1} \mathcal{H}(\mathbf{z}_{t+1}|\mathbf{z}_t)\\ &- \mathbb{E}_{Q_\phi}[\log p_\alpha(\mathbf{z}_{0:T})] - \mathbb{E}_{Q_\phi}[\log p_\beta(\mathbf{x}|\mathbf{z}_0)].\end{aligned} \tag{7.25}$$

The KL divergence between the aggregated forward trajectory and the learned reverse trajectory is

$$\begin{aligned}\mathbb{D}_{\text{KL}}(q_\phi(\mathbf{z}_{0:T}) \| p_\alpha(\mathbf{z}_{0:T})) = &-\mathcal{H}(\mathbf{z}_0) - \sum_{t=0}^{T-1} \mathcal{H}(\mathbf{z}_{t+1}|\mathbf{z}_t)\\ &- \mathbb{E}_{Q_\phi}[\log p_\alpha(\mathbf{z}_{0:T})].\end{aligned} \tag{7.26}$$

Since $\mathcal{I}(\mathbf{x}, \mathbf{z}_0) = \mathcal{H}(\mathbf{z}_0) - \mathcal{H}(\mathbf{z}_0|\mathbf{x})$, substituting this identity into Eq. (7.25) gives

$$\begin{aligned}\mathbb{D}_{\text{KL}}(Q_\phi \| P_\theta) = &-\mathcal{H}(\mathbf{x}) - \mathbb{E}_{Q_\phi}[\log p_\beta(\mathbf{x}|\mathbf{z}_0)]\\ &+ \mathbb{D}_{\text{KL}}(q_\phi(\mathbf{z}_{0:T}) \| p_\alpha(\mathbf{z}_{0:T})) + \mathcal{I}(\mathbf{x}, \mathbf{z}_0),\end{aligned} \tag{7.27}$$

which establishes the decomposition in Eq. (7.9).

7.8.5 Derivation of the Learning Gradient

Recall that we derive the extended version of Eq. (7.6) in Sec. 7.8.2. To calculate the gradient of α, we have

$$\begin{aligned}\nabla_\alpha \,\mathrm{ELBO}_{\mathrm{Diff},\theta,\phi} &= \nabla_\alpha \mathbb{E}\left[\sum_{t=0}^{T-1} \log p_\alpha(\mathbf{z}_t|\mathbf{z}_{t+1})\right] \\ &= \mathbb{E}\left[\sum_{t=0}^{T-1} \nabla_\alpha \log p_\alpha(\mathbf{z}_t|\mathbf{z}_{t+1})\right],\end{aligned} \tag{7.28}$$

where $\mathbb{E}$ is the abbreviation of $\mathbb{E}_{q_\phi(\mathbf{z}_0|x),q(\mathbf{z}_{1:T}|\mathbf{z}_0)}$; in practice, we use Monte-Carlo average to approximate the expectation. We next examine the learning gradient for each diffusion step t.

$$\nabla_\alpha \log p_\alpha(\mathbf{z}_t|\mathbf{z}_{t+1}) = \nabla_\alpha F_\alpha(\tilde{\mathbf{z}}_t, t) - \nabla_\alpha \log \tilde{Z}_{\alpha,t}(\mathbf{z}_{t+1}), \tag{7.29}$$

where the quadratic term $\frac{1}{2\sigma_{t+1}^2}||\tilde{\mathbf{z}}_t - \mathbf{z}_{t+1}||^2$ is not related to α and gets cancelled. According to the definition of the partition function in Sec. 7.2, we can similarly derive

$$\nabla_\alpha \log \tilde{Z}_{\alpha,t}(\mathbf{z}_{t+1}) = \mathbb{E}_{p_\alpha(\tilde{\mathbf{z}}_t|\mathbf{z}_{t+1})}\left[\nabla_\alpha F_\alpha(\tilde{\mathbf{z}}_t, t)\right], \tag{7.30}$$

as in Pang et al. [44]. For the prior model, we thus have

$$\begin{aligned}\nabla_\alpha \,\mathrm{ELBO}_t = &\mathbb{E}_{q_\phi(\tilde{\mathbf{z}}_t,\mathbf{z}_0|\mathbf{x})}[\nabla_\alpha F_\alpha(\tilde{\mathbf{z}}_t, t)] \\ &- \mathbb{E}_{q_\phi(\mathbf{z}_{t+1},\mathbf{z}_0|\mathbf{x}),p_\alpha(\tilde{\mathbf{z}}_t|\mathbf{z}_{t+1})}[\nabla_\alpha F_\alpha(\tilde{\mathbf{z}}_t, t)],\end{aligned} \tag{7.31}$$

where $q_\phi(\tilde{\mathbf{z}}_t, \mathbf{z}_0|x) = q(\tilde{\mathbf{z}}_t|\mathbf{z}_0)q_\phi(\mathbf{z}_0|\mathbf{x})$. Note that we can sample $\mathbf{z}_t,\ t>0$ directly from

$$q(\mathbf{z}_t|\mathbf{z}_0) = \mathcal{N}(\mathbf{z}_t; \sqrt{\bar{\gamma}_t}\mathbf{z}_0, (1-\bar{\gamma}_t)\mathbf{I}), \tag{7.32}$$

by merging the Gaussian noises during forward diffusion process; we denote $\gamma_t = 1-\sigma_t^2$ and $\bar{\gamma}_t = \prod_{i=1}^t \gamma_i$.

For the encoder and decoder, based on Eq. (7.6) and Eq. (7.18), we have

$$\begin{aligned}\nabla_\psi \,\mathrm{ELBO} &= \nabla_\psi \mathbb{E}_{q_\phi(\mathbf{z}_0|\mathbf{x})}[\log p_\beta(\mathbf{x}|\mathbf{z}_0) - \log q_\phi(\mathbf{z}_0|\mathbf{x})] \\ &+ \nabla_\phi \mathbb{E}_{q_\phi(\mathbf{z}_{0:T}|\mathbf{x})}\left[\log p(\mathbf{z}_T) + \sum_{t=0}^{T-1} \log p_\alpha(\mathbf{z}_t|\mathbf{z}_{t+1})\right],\end{aligned} \tag{7.33}$$

where the summation of energy terms provides extra guidance for the optimization of encoder.

7.9 Additional Experimental Details and Discussion

7.9.1 Network Architecture and Hyperparameters

We provide detailed network architecture for the latent space model of this work in Tab. 7.10; we adopt the same architecture throughout the experiments. Spectral normalization [39] is used to regularize parameters in linear layers. The encoder and decoder in all models are the same as in Pang & Wu [43], implemented with a single-layer GRU with a hidden size of 512. The key hyperparameters of LDEBM for each dataset are listed in Tab. 7.11. Of note, we use the same dimension of the latent space as in [43] for a fair comparison.

λ_1 is the hyperparameter that reweights the term in Eq. (7.18); it generally controls how fast q_ϕ and p_θ run towards each other. λ_2 refers to the hyperparameter in Eq. (7.9); it controls the trade-off between the compressivity of $\mathbf{z}_0$ about $\mathbf{x}$ and its expressivity to $\mathbf{y}$. λ_3 controls the weight of classification loss mentioned in Sec. 7.3.3; recall that we use pseudo-label $\hat{\mathbf{y}}$ inferred by the geometric clustering algorithm or the ground-truth label $\mathbf{y}$ to supervise $p_\alpha(\mathbf{y}|\mathbf{z}_0)$ in our modeling. For controllable generation and semi-supervised classification, we find it important to have a larger weight on the classification loss so that the model is forced to capture the major modes of the data.

For optimization, we use Adam optimizer [27] with $\beta_1 = 0.9$ and $\beta_2 = 0.999$ for all the experiments. On all datasets except the 2D synthetic datasets and AGNews dataset, we use a batch size of 128 and a constant learning rate of $1e-3$ for encoder and decoder without

Table 7.10: **Network architecture for the LDEBM prior.** N is set to 12 for all the experiments.

Layers	Output size	Note
Time Embedding		
Input: t	1	Index of diffusion step
Sin. embedding	200	
Linear, LReLU	200	negative_slope 0.2
Linear	200	
Input Embedding		
Input: $\mathbf{z}$	d_{lat}	
Linear, LReLU	200	negative_slope 0.2
Linear	200	
Context Embedding (for response generation only)		
Input: $\mathbf{z}_{\text{ctx}}$	512	ctx. embedding
Linear, LReLU	200	negative_slope 0.2
Linear	200	
LDEBM Prior		
Input: $\mathbf{z}, t$ $*\mathbf{z}_{\text{ctx}}$	1, d_{lat} 512	optional $\mathbf{z}_{\text{ctx}}$
Embedding	200	Embedding of each input
Concatenate	400 600	w/o ctx. w/ ctx.
LReLU, Linear	200	negative_slope 0.2
N ResBlocks	200	LReLU, Linear + Input
LReLU, Linear	K	K class logits
Log-Sum-Exp	1	energy score

weight decay. For LDEBM, we use a constant learning rate of $1e-4$. We use a larger batch size of 1000 on 2D synthetic datasets. On the AGNews dataset, we use the same set of hyperparameters as in Pang & Wu [43] for optimization. The batch size is set to 200; the initial learning rate is $1e-4$ for encoder and decoder, and $1e-5$ for LDEBM. Learning rates are exponentially decayed with a decay rate of 0.998 for each model. Encoder and LDEBM have a weight decay rate of $2e-3$ and $1e-3$, respectively.

Table 7.11: **Hyperparameters of LDEBM.** DD-CLS presents the set of hyperparameters used in unsupervised clustering on DD dataset. DD-GEN presents the set of hyperparameters used in conditional response generation on DD dataset.

Dataset	d_{lat}	K	λ_1	λ_2	λ_3
2D Gaussian	2	16	1	0.05	0.05
2D Pinwheel	2	10	1	0.05	0.05
PTB	40	20	0.1	0.05	0.05
Jericho	40	20	0.1	0.05	0.05
DD-CLS	32	125	0.01	0.05	0.5
DD-GEN	32	125	1	0.05	0.05
SMD	32	125	10	10	5
Yelp	40	2	50	50	200
AGNews	20	4	1e-3	5	200

7.9.2 Experiment Settings and Baselines

Experiment settings For generative modeling, following previous methods [43, 53], the NLL term is computed with importance sampling [6] using 500 importance samples. To compute rPPL, we set the generated sample size as $40,000$, which is the same size as PTB training set. We recruit ASGD Weight-Dropped LSTM [35] to compute rPPL as in previous works.

In terms of conditional response generation, for word-embedding-based evaluation on SMD and DD, we use the publicly available GloVe [48] word embeddings of 300 dimension trained on 840B tokens, and report the score from 1 response per context. We use a context window size of 5 during training and evaluation.

The maximum length of each sentence is set to 40 words for most datasets and 70 words for the JerichoWorld dataset. On JerichoWorld dataset, we extract the description of each state as the text data.

Baselines On **PTB, DD** and **SMD**, our model is compared with the following baselines: (1) RNNLM [37], the language model implemented with GRU [8]; (2) AE [64], the deterministic auto-encoder which has no regularization to the latent space; (3) DAE, the AE with a discrete latent space; (4) VAE [28], the vanilla VAE with a continuous latent space and a non-informative Gaussian prior; (5) DVAE, the VAE with a discrete latent space; (6) DI-VAE

[76], a DVAE variant with a mutual information term between the observed piece of text $\mathbf{x}$ and its inferred latent variable $\mathbf{z}$; (7) semi-VAE [29], the semi-supervised VAE model with independent discrete and continuous latent variables; (8) GM-VAE, the VAE with a Gaussian mixture prior; (9) DGM-VAE [53], the GM-VAE with a dispersion term that avoids the mode-collapse of Gaussian mixture prior; (10) semi-VAE $+\mathcal{I}(\mathbf{x}, \mathbf{y})$, GM-VAE $+\mathcal{I}(\mathbf{x}, \mathbf{y})$, DGM-VAE $+\mathcal{I}(\mathbf{x}, \mathbf{y})$, are the same models as (7), (8), and (9) respectively, but with a mutual information term between $\mathbf{x}$ and $\mathbf{y}$ computed using separate inference networks for $\mathbf{y}$ and $\mathbf{z}$. We compare with the close competitors (11) SVEBM, the symbol-vector coupling prior model and (12) SVEBM-IB, SVEBM with regularization based on information-bottleneck.

On **Yelp** dataset, we additionally include text conditional GAN [59] as a baseline for controllable generation. On **AGNews** dataset, we further compare our model to VAMPIRE [17], a VAE-based semi-supervised text learning model. Other baselines include its supervised learning variants: (1) the model trained with Glove embedding pre-trained on 840 billion words (Glove-OD); (2) the model trained with Glove embedding on in-domain unlabeled data (Glove-ID). We also include more recent baselines such as Hard EM and CatVAE [23] that improve over VAMPIRE.

7.9.3 Extra Details for Experiments

Additional ablation studies We conduct additional experiments on both PTB and DD datasets to inspect the contribution of the proposed techniques. In Sec. 7.4.1, we have reported results on PTB and DD of Ours w/o GC which represents the model with Information Bottleneck but without Geometric Clustering (GC); Ours denotes the full model.

We further conduct experiments on the proposed model without using IB or GC. We observe that the proposed model using only diffusion-based sampling scheme has a rPPL of 166.26, BLEU of 11.30, wKL of 0.07 and NLL of 80.76 on PTB; it has a MI of 0.01, BLEU of 19.28, Act. of 0.12 and Emo. of 0.06 on DD, which is better than SVEBMs (please see

Tabs. 7.1 and 7.3 in Sec. 7.4.1).

We also add GC to SVEBM (denoted as SVE-IB w/ GC). We find that SVE-IB w/ GC does perform better compared with SVE-IB, showing a rPPL of 179.95, BLEU of 10.08, wKL of 0.15 and NLL of 93.28 on PTB; it has a MI of 2.88, BLEU of 11.75, Act. of 0.61 and Emo. of 0.60 on DD. Notably, SVE-IB w/ GC is still inferior to LDEBMs.

In summary, we think these additional experiments (1) emphasize the importance of our diffusion-based modeling approach, and (2) demonstrate the effectiveness of GC as additional regularization.

2D synthetic data We provide the full evolution of SVEBM-IB and our models as visualized in Fig. 7.5. Though SVEBM-IB can capture some regularities of the data in the early stages of training, the model is prone to collapse due to the degenerated sampling quality. This features an exploding KL-term and leads to poor performance on generation. Our preliminary experiments indicate that common deep learning heuristics for improving the model capacity barely help. These include but are not limited to increasing the number of parameters in SVEBM, *i.e.*, using larger models, and adopting deliberately designed activation functions or normalization modules. LDEBM w/o geometric clustering has a better sampling quality and effectively mitigates the instability in training. However, the mode coverage is not satisfying in data space; the structure is unclear in latent space. In contrast, LDEBM w/ geometric clustering shows superior generation quality with better mode coverage. It demonstrates a better-structured latent space.

Sentence completion To perform sentence completion, we adopt a two-stage training scheme. We first train the LDEBM with inference, prior and generation models on the JerichoWorld dataset. After the first-stage training, the parameters of prior, inference and generation models are fixed. We then train a shallow MLP in the latent space to project the inferred posterior $\mathbf{z}_0$ to a disentangled space; the variables in the projected $\mathbf{z}_0$ can be grouped as: (a) the representation of observable words $\hat{\mathbf{z}}_{\text{obs}}$ in the input sentence and (b) the representation of unknown words $\hat{\mathbf{z}}_{\text{unk}}$. Conditional sampling in the latent space then refers

to updating $\hat{\mathbf{z}}_{\text{unk}}$ based on the fixed $\hat{\mathbf{z}}_{\text{obs}}$ by running Langevin dynamics guided by the latent space model.

We mask half of the words in the sentences with `<unk>` token to prepare the inputs. In the second stage of training, we supervise the MLP by minimizing the reconstruction error between only the observable words of the input the sentence and the corresponding outputs of the model.

Sentence sentiment control Recall that in our formulation only $\mathbf{z}_0$ is connected to $\mathbf{y}$. We therefore condition only the final reverse diffusion step $[\mathbf{z}_0|\mathbf{z}_1]$ on $\mathbf{y}$ when performing controllable generation, *i.e.*, using $\mathbf{y}$ to guide the generation only when $t=0$ in Alg. 4. This can be a bit counter-intuitive since no label information is injected in previous reverse steps. By construction, $\mathbf{y}$ and $\mathbf{z}_{1:T}$ are conditionally independent given $\mathbf{z}_0$ in our formulation; however, we empirically observe that $\mathbf{y}$ and $\mathbf{z}_t$ for $t>0$ are nearly independent even marginally after integrating out $\mathbf{z}_{0:t-1}$ in our model. In other words, $p_\alpha(\mathbf{y}|\mathbf{z}_t),\ t>0$ are in general non-informative since adding noise in the latent space could be much more corrupting than adding noise in the data space. The model learns to enjoy the less multi-modal energy landscape in previous reverse steps; it then seeks the given mode only in the most informative final reverse step. We examine $p_\alpha(\mathbf{y}|\mathbf{z}_t),\ t>0$ for the model trained on Yelp dataset by marginalizing out $\mathbf{z}_{t-1}$ of $p_\alpha(\mathbf{y},\mathbf{z}_{t-1}|\mathbf{z}_t),\ t>0$. For example, for $t=1$, we may calculate

$$\begin{aligned} p_\alpha(\mathbf{y}|\mathbf{z}_1) &= \int_{\mathbf{z}_0} p_\alpha(\mathbf{y}|\mathbf{z}_0) p_\alpha(\mathbf{z}_0|\mathbf{z}_1) d\mathbf{z}_0 \\ &= \mathbb{E}_{p_\alpha(\mathbf{z}_0|\mathbf{z}_1)}\left[p_\alpha(\mathbf{y}|\mathbf{z}_0)\right] \\ &\approx \frac{1}{M}\sum_{i=1}^{M} p_\alpha(\mathbf{y}|\mathbf{z}_0^{(i)}). \end{aligned} \tag{7.34}$$

See Fig. 7.4 for the visualization of $p_\alpha(\mathbf{y}|\mathbf{z}_t)$ over t.

A more intuitive method is to use the data label $\mathbf{y}$ to supervise each $[\mathbf{y},\mathbf{z}_t|\mathbf{z}_{t+1}]$, so that we can propagate the label information through the whole trajectory. Given $\mathbf{z}_0$, $\mathbf{y}$ and $\mathbf{z}_{1:T}$ are independent. But if we marginalize out $\mathbf{z}_0$, $\mathbf{y}$ will depend on $\mathbf{z}_1$. Similarly, if we continue

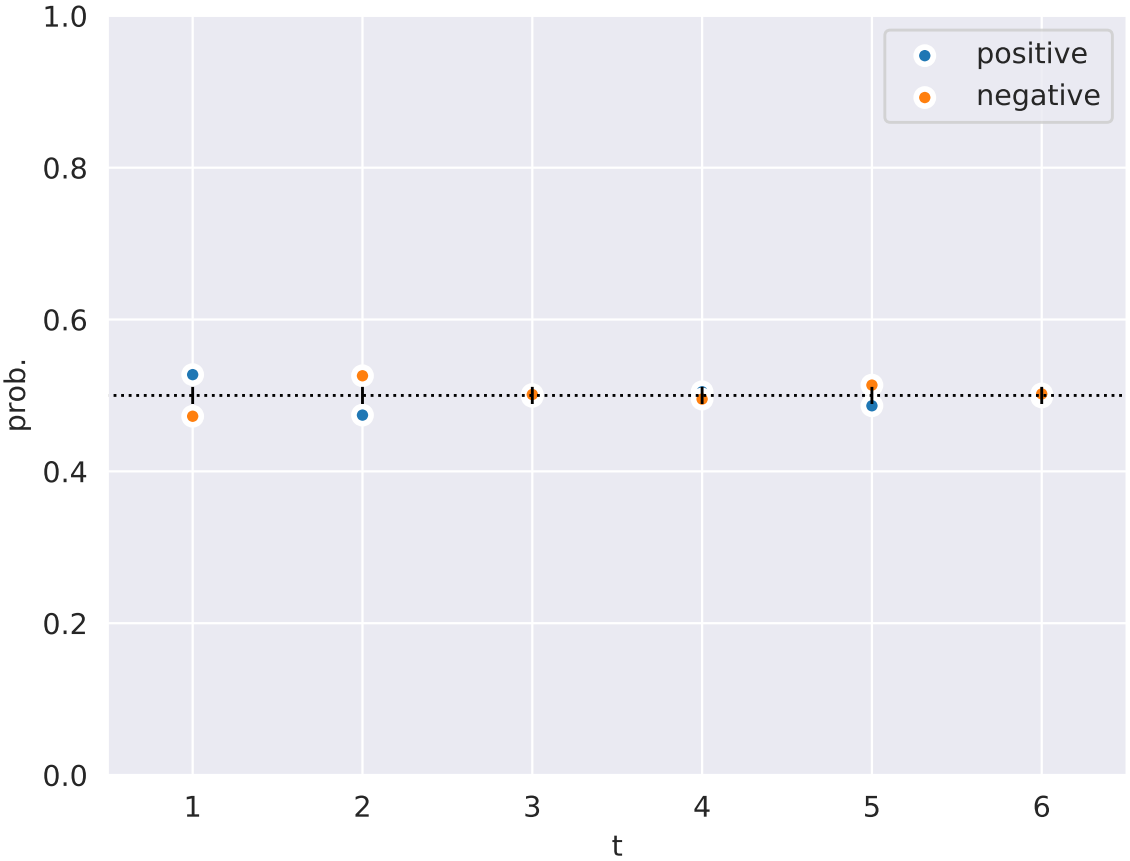
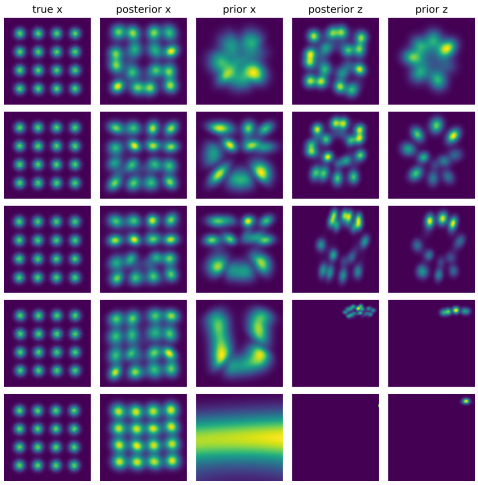


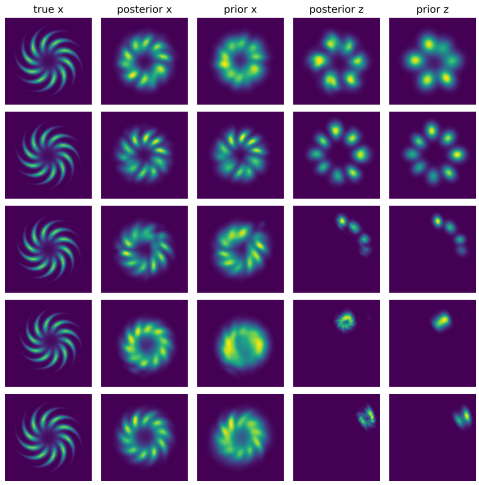
Figure 7.4: **Visualization of $p_\alpha(\mathbf{y}|\mathbf{z}_t)$ over t.** $p_\alpha(\mathbf{y}|\mathbf{z}_t)$ is constantly around the probability of 0.5 over t.

to marginalize out $\mathbf{z}_1$, $\mathbf{y}$ will depend on $\mathbf{z}_2$. Repeating this process results in $p_\alpha(\mathbf{y}|\mathbf{z}_t)$ for each t after integrating out $\mathbf{z}_{0:t-1}$. Supervising $p_\alpha(\mathbf{y}|\mathbf{z}_t),\ t>0$ using $\mathbf{y}$ therefore effectively encodes the label information into the whole trajectory.

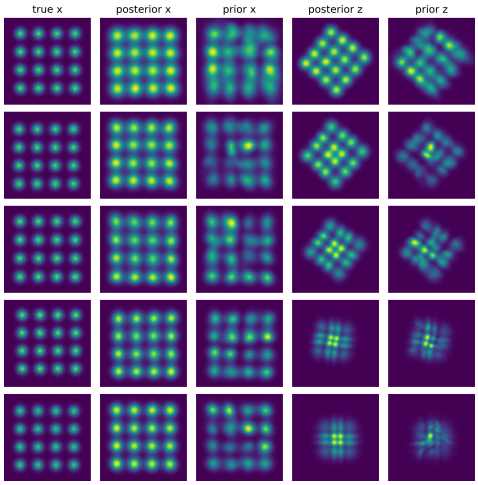
While the marginalization can be difficult, we may approximate it by learning the amortized version of $p_\alpha(\mathbf{y}|\mathbf{z}_t),\ t>0$ as $p_\alpha(\mathbf{y}, \mathbf{z}_{t-1}=\mu_{\phi,t-1}|\mathbf{z}_t),\ t>0$, where $\mu_{\phi,t}$ is the posterior mean of $\mathbf{z}_t$. We may therefore circumvent the intractable integration in practice and guide the whole trajectory for controllable generation.



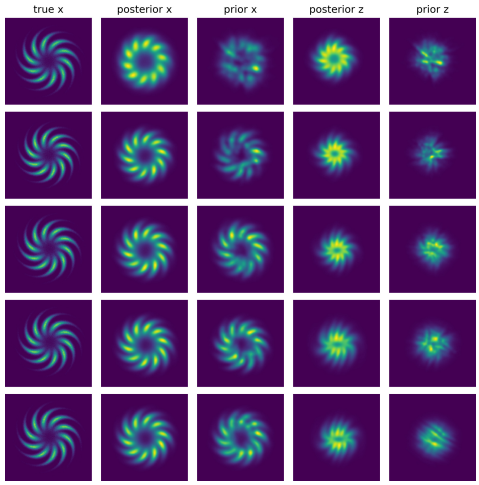
(a) SVEBM–IB, Gaussian



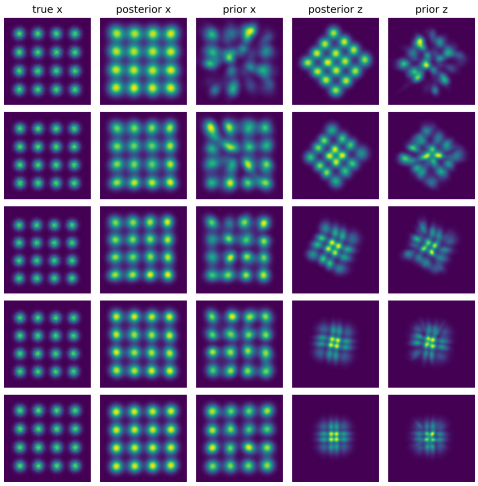
(b) SVEBM–IB, pinwheel



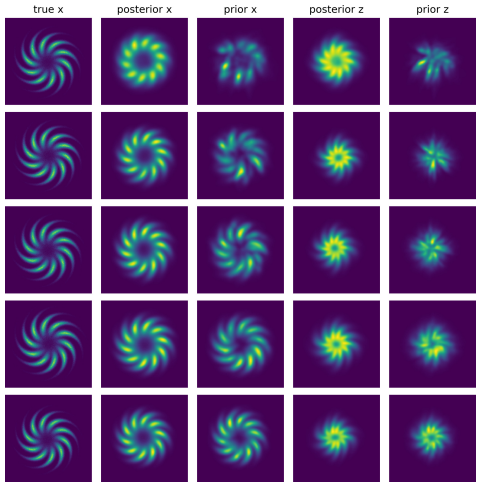
(c) Without PL, Gaussian



(d) Without PL, pinwheel



(e) LDEBM, Gaussian



(f) LDEBM, pinwheel

Figure 7.5: Full evolution of SVEBM–IB and LDEBM variants on Gaussian and pinwheel synthetic distributions. Each panel displays representative states over training.

Bibliography

[1] Ammanabrolu, P. and Riedl, M. O. Modeling worlds in text. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[2] Austin, J., Johnson, D. D., Ho, J., Tarlow, D., and van den Berg, R. Structured denoising diffusion models in discrete state-spaces. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[3] Bengio, Y., Yao, L., Alain, G., and Vincent, P. Generalized denoising auto-encoders as generative models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2013.

[4] Bowman, S. R., Vilnis, L., Vinyals, O., Dai, A. M., Jozefowicz, R., and Bengio, S. Generating sentences from a continuous space. In *Conference on Computational Natural Language Learning (CoNLL)*, 2016.

[5] Brown, P. F., Della Pietra, S. A., Della Pietra, V. J., and Mercer, R. L. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 1993.

[6] Burda, Y., Grosse, R. B., and Salakhutdinov, R. Importance weighted autoencoders. In *International Conference on Learning Representations (ICLR)*, 2016.

[7] Card, D., Tan, C., and Smith, N. A. Neural models for documents with metadata. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2018.

[8] Cho, K., van Merrienboer, B., Gülçehre, Ç., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *Annual Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014.

[9] Du, Y., Li, S., Tenenbaum, J., and Mordatch, I. Improved contrastive divergence training of energy based models. In *International Conference on Machine Learning (ICML)*, 2021.

[10] Eric, M., Krishnan, L., Charette, F., and Manning, C. D. Key-value retrieval networks for task-oriented dialogue. In *Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDial)*, 2017.

[11] Fang, L., Li, C., Gao, J., Dong, W., and Chen, C. Implicit deep latent variable models for text generation. In *Annual Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2019.

[12] Forgues, G., Pineau, J., Larchevêque, J.-M., and Tremblay, R. Bootstrapping dialog systems with word embeddings. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2014.

[13] Fu, H., Li, C., Liu, X., Gao, J., Çelikyilmaz, A., and Carin, L. Cyclical annealing schedule: A simple approach to mitigating kl vanishing. In *North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, 2019.

[14] Gao, R., Song, Y., Poole, B., Wu, Y. N., and Kingma, D. P. Learning energy-based models by diffusion recovery likelihood. In *International Conference on Learning Representations (ICLR)*, 2020.

[15] Grathwohl, W., Wang, K.-C., Jacobsen, J.-H., Duvenaud, D., Norouzi, M., and Swersky,

K. Your classifier is secretly an energy based model and you should treat it like one. In *International Conference on Learning Representations (ICLR)*, 2019.

[16] Gupta, A., Agarwal, A., Singh, P., and Rai, P. A deep generative framework for paraphrase generation. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2018.

[17] Gururangan, S., Dang, T., Card, D., and Smith, N. A. Variational pretraining for semi-supervised text classification. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2019.

[18] Han, T., Nijkamp, E., Fang, X., Hill, M., Zhu, S.-C., and Wu, Y. N. Divergence triangle for joint training of generator model, energy-based model, and inferential model. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[19] Han, T., Nijkamp, E., Zhou, L., Pang, B., Zhu, S.-C., and Wu, Y. N. Joint training of variational auto-encoder and latent energy-based model. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[20] He, J., Spokoyny, D., Neubig, G., and Berg-Kirkpatrick, T. Lagging inference networks and posterior collapse in variational autoencoders. In *International Conference on Learning Representations (ICLR)*, 2018.

[21] Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., Mohamed, S., and Lerchner, A. beta-vae: Learning basic visual concepts with a constrained variational framework. In *International Conference on Learning Representations (ICLR)*, 2016.

[22] Ho, J., Jain, A., and Abbeel, P. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[23] Jin, S., Wiseman, S., Stratos, K., and Livescu, K. Discrete latent variable representations for low-resource text classification. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2020.

[24] Jing, M., Ma, X., Sun, F., and Liu, H. Learning and inferring movement with deep generative model. *arXiv preprint arXiv:1805.07252*, 2018.

[25] Jing, M., Ma, X., Huang, W., Sun, F., and Liu, H. Task transfer by preference-based cost learning. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2019.

[26] Johnson, J., Douze, M., and Jégou, H. Billion-scale similarity search with gpus. *IEEE Transactions on Big Data*, 2019.

[27] Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[28] Kingma, D. P. and Welling, M. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[29] Kingma, D. P., Mohamed, S., Rezende, D. J., and Welling, M. Semi-supervised learning with deep generative models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2014.

[30] Li, B., He, J., Neubig, G., Berg-Kirkpatrick, T., and Yang, Y. A surprisingly effective fix for deep latent variable modeling of text. In *Annual Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2019.

[31] Li, J., Jia, R., He, H., and Liang, P. Delete, retrieve, generate: a simple approach to sentiment and style transfer. In *North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, 2018.

[32] Li, P., Lam, W., Bing, L., and Wang, Z. Deep recurrent generative decoder for abstractive text summarization. In *Annual Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2017.

[33] Li, Y., Su, H., Shen, X., Li, W., Cao, Z., and Niu, S. Dailydialog: A manually labelled

multi-turn dialogue dataset. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2017.

[34] Marcus, M. P., Marcinkiewicz, M. A., and Santorini, B. Building a large annotated corpus of english: the penn treebank. *Computational Linguistics*, 1993.

[35] Merity, S., Keskar, N. S., and Socher, R. Regularizing and optimizing lstm language models. In *International Conference on Learning Representations (ICLR)*, 2018.

[36] Miao, Y., Yu, L., and Blunsom, P. Neural variational inference for text processing. In *International Conference on Machine Learning (ICML)*, 2016.

[37] Mikolov, T., Karafiát, M., Burget, L., Cernockỳ, J., and Khudanpur, S. Recurrent neural network based language model. In *Interspeech*, 2010.

[38] Mitchell, J. and Lapata, M. Vector-based models of semantic composition. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2008.

[39] Miyato, T., Kataoka, T., Koyama, M., and Yoshida, Y. Spectral normalization for generative adversarial networks. In *International Conference on Learning Representations (ICLR)*, 2018.

[40] Nie, W., Vahdat, A., and Anandkumar, A. Controllable and compositional generation with latent-space energy-based models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[41] Nijkamp, E., Hill, M., Zhu, S.-C., and Wu, Y. N. Learning non-convergent non-persistent short-run mcmc toward energy-based model. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[42] Nijkamp, E., Hill, M., Han, T., Zhu, S.-C., and Wu, Y. N. On the anatomy of mcmc-based maximum likelihood learning of energy-based models. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2020.

[43] Pang, B. and Wu, Y. N. Latent space energy-based model of symbol-vector coupling for text generation and classification. In *International Conference on Machine Learning (ICML)*, 2021.

[44] Pang, B., Han, T., Nijkamp, E., Zhu, S.-C., and Wu, Y. N. Learning latent space energy-based prior model. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[45] Pang, B., Han, T., and Wu, Y. N. Learning latent space energy-based prior model for molecule generation. *arXiv preprint arXiv:2010.09351*, 2020.

[46] Pang, B., Zhao, T., Xie, X., and Wu, Y. N. Trajectory prediction with latent belief energy-based model. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[47] Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. Bleu: a method for automatic evaluation of machine translation. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2002.

[48] Pennington, J., Socher, R., and Manning, C. D. Glove: Global vectors for word representation. In *Annual Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014.

[49] Rezende, D. J., Mohamed, S., and Wierstra, D. Stochastic backpropagation and approximate inference in deep generative models. In *International Conference on Machine Learning (ICML)*, 2014.

[50] Rus, V. and Lintean, M. An optimal assessment of natural language student input using word-to-word similarity metrics. In *International Conference on Intelligent Tutoring Systems*, 2012.

[51] Serban, I., Sordoni, A., Bengio, Y., Courville, A., and Pineau, J. Building end-to-end dialogue systems using generative hierarchical neural network models. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2016.

[52] Serban, I., Sordoni, A., Lowe, R., Charlin, L., Pineau, J., Courville, A., and Bengio, Y. A hierarchical latent variable encoder-decoder model for generating dialogues. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2017.

[53] Shi, W., Zhou, H., Miao, N., and Li, L. Dispersed exponential family mixture vaes for interpretable text generation. In *International Conference on Machine Learning (ICML)*, 2020.

[54] Sinha, A., Song, J., Meng, C., and Ermon, S. D2c: Diffusion-denoising models for few-shot conditional generation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[55] Sohl-Dickstein, J., Weiss, E., Maheswaranathan, N., and Ganguli, S. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning (ICML)*, 2015.

[56] Song, Y. and Ermon, S. Generative modeling by estimating gradients of the data distribution. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[57] Song, Y. and Ermon, S. Improved techniques for training score-based generative models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[58] Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., and Poole, B. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations (ICLR)*, 2020.

[59] Subramanian, S., Rajeswar, S., Sordoni, A., Trischler, A., Courville, A., and Pal, C.

Towards text generation with adversarially learned neural outlines. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

[60] Tishby, N., Pereira, F. C., and Bialek, W. The information bottleneck method. *arXiv preprint physics/0004057*, 2000.

[61] Vahdat, A. and Kautz, J. NVAE: A deep hierarchical variational autoencoder. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[62] Vahdat, A., Kreis, K., and Kautz, J. Score-based generative modeling in latent space. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[63] Vincent, P. A connection between score matching and denoising autoencoders. *Neural Computation*, 2011.

[64] Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., Manzagol, P.-A., and Bottou, L. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research (JMLR)*, 2010.

[65] Wang, W., Gan, Z., Xu, H., Zhang, R., Wang, G., Shen, D., Chen, C., and Carin, L. Topic-guided variational auto-encoder for text generation. In *North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, 2019.

[66] Wehenkel, A. and Louppe, G. Diffusion priors in variational autoencoders. *arXiv preprint arXiv:2106.15671*, 2021.

[67] Welling, M. and Teh, Y. W. Bayesian learning via stochastic gradient langevin dynamics. In *International Conference on Machine Learning (ICML)*, 2011.

[68] Wen, T.-H., Miao, Y., Blunsom, P., and Young, S. Latent intention dialogue models. In *International Conference on Machine Learning (ICML)*, 2017.

[69] Xie, J., Lu, Y., Zhu, S.-C., and Wu, Y. A theory of generative convnet. In *International Conference on Machine Learning (ICML)*, 2016.

[70] Young, S., Gašić, M., Thomson, B., and Williams, J. D. Pomdp-based statistical spoken dialog systems: A review. *Proceedings of the IEEE*, 2013.

[71] Yu, P., Xie, S., Ma, X., Zhu, Y., Wu, Y. N., and Zhu, S.-C. Unsupervised foreground extraction via deep region competition. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[72] Zhang, B., Xiong, D., Su, J., Duan, H., and Zhang, M. Variational neural machine translation. In *Annual Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2016.

[73] Zhang, X., Zhao, J., and LeCun, Y. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2015.

[74] Zhao, J., Kim, Y., Zhang, K., Rush, A., and LeCun, Y. Adversarially regularized autoencoders. In *International Conference on Machine Learning (ICML)*, 2018.

[75] Zhao, T., Zhao, R., and Eskenazi, M. Learning discourse-level diversity for neural dialog models using conditional variational autoencoders. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2017.

[76] Zhao, T., Lee, K., and Eskenazi, M. Unsupervised discrete sentence representation learning for interpretable neural dialog generation. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2018.

Chapter 8

Conclusion and Future Directions

8.1 Summary of the Dissertation

This dissertation has examined learning and sampling in neural generative models from three complementary perspectives: estimation of complex probability distributions, inference-time adaptation of generative procedures, and structured latent-variable modeling. Its central theme is that a flexible probabilistic representation does not, by itself, guarantee tractable estimation or reliable generation. The statistical properties of a generative system depend jointly on its representation, learning algorithm, and sampling procedure.

Chapters 2 and 3 address difficulties in learning neural probability distributions. Chapter 2 studies how scaling the noise magnitude in noisier noise contrastive estimation affects the approximation to maximum-likelihood gradients, highlighting the tradeoff between statistical accuracy and computational practicality. Chapter 3 develops diffusion-amortized MCMC for learning latent energy-based priors. By refining an amortized sampling distribution with MCMC transitions, the method reduces reliance on repeated long-chain simulation while retaining a mechanism for improving approximate samples. These chapters approach difficult estimation problems from complementary directions: modifying the estimation objective and improving the sampling procedure used to approximate its expectations.

Chapters 4 and 5 focus on inference-time adaptation of diffusion sampling with pretrained models. Chapter 4 formulates multi-denoiser guidance, in which predictions from several pretrained models are combined using learnable, noise-dependent weights, including weights optimized through preferences. Chapter 5 learns instance-level denoising schedules, conditioned on information available before generation, using reward-based policy optimization and a James–Stein shrinkage baseline for variance reduction. The distinction is between changing the direction of denoising updates and changing the time points at which those updates are evaluated. Both approaches demonstrate that the sampling algorithm itself can be optimized, even when the underlying generative networks remain fixed.

Chapters 6 and 7 investigate how latent-variable structure supports interpretable generative modeling. Deep Region Competition decomposes images by allowing foreground and background generative explanations to compete for image regions. The latent diffusion energy-based model for text combines expressive latent priors with symbol-vector coupling, relating continuous latent representations to interpretable discrete symbols. These studies show that sampling and inference are not merely computational concerns: their design also affects the structures and controls that a generative model can learn.

8.2 Three Cross-Cutting Lessons

8.2.1 Expressiveness entails inference costs

Energy-based priors can represent complex, multimodal latent distributions, while diffusion models support flexible generative dynamics. This expressiveness carries computational costs. Learning an energy-based model may require samples from a changing and difficult target distribution; diffusion sampling generally requires sequential model evaluations. Expressive power should therefore be assessed together with estimation accuracy, sampling reliability, and computational cost.

8.2.2 Approximate computation can be optimized statistically

The approximations inside a learning or sampling algorithm need not be fixed implementation choices. Noise scaling changes the behavior of a contrastive estimator, diffusion amortization changes how MCMC computations are reused, and schedule policies change the allocation of denoising evaluations. These examples suggest treating approximation quality and computation as joint design objectives, with attention to estimation error, sampling bias, numerical error, and resource constraints.

8.2.3 Inference-time control is distinct from model fitting

A pretrained diffusion model can be deployed with different time discretizations or guidance rules without changing its learned parameters. Such choices may improve a downstream quality measure while also changing the distribution induced by the implemented sampler. In particular, a formal distribution-level interpretation of combined scores does not automatically establish that a finite-step sampling algorithm produces draws from that distribution. Claims about inference-time improvements should therefore distinguish fidelity to a probabilistic target from deliberate optimization of perceptual or preference-based objectives.

8.3 Limitations and Open Problems

The contributions in this dissertation address related questions in different statistical settings; they do not constitute a single end-to-end learning and sampling algorithm. Their guarantees and empirical findings must be interpreted within the assumptions and evaluation conditions of the corresponding chapters. For example, a closer approximation to a maximum-likelihood update does not by itself guarantee faster MCMC mixing, while a higher reward for generated samples does not establish improved likelihood calibration or distributional fidelity.

Approximation errors also remain method-dependent. Finite MCMC chains and imperfect amortized proposals can affect the estimation of latent-variable models. Learned sampling

schedules depend on the chosen reward and computational budget. The distributions induced by multi-denoiser guidance may differ from formal product-of-experts targets. The multi-denoiser chapter primarily develops a methodological formulation and optimization procedure, so systematic evaluation of its quality, robustness, and additional inference cost remains an important open task.

8.4 Future Directions

8.4.1 Jointly optimizing guidance and schedules

Chapters 4 and 5 suggest a unified inference-time decision problem: jointly select which denoisers to evaluate, how to combine their predictions, and when to place denoising steps. An adaptive policy could optimize expected generation quality subject to a budget on model evaluations or wall-clock cost. Accounting for the cost of each constituent denoiser would be essential, since a schedule with fewer steps need not be faster if each step invokes more networks.

8.4.2 Bias-controlled amortized inference

A second direction is to integrate learned proposal mechanisms more tightly with probabilistic model fitting. Amortized samplers could be paired with occasional corrective transitions and diagnostics that measure mismatch between their sampling distributions and the desired targets. A central question is how improvements in computation translate into controlled errors in parameter estimation and final sample quality.

8.4.3 Statistical characterization of controlled generation

The distributional consequences of learned guidance and reward-optimized schedules deserve further analysis. In particular, time-dependent combinations of approximate score predictors

need not produce the marginals suggested by a formal product-of-experts expression. Characterizing this discrepancy, together with the effects of finite-step discretization and reward misspecification, would clarify when inference-time control should be viewed as approximate sampling and when it is better understood as changing the generative target.

8.4.4 Structured and controllable latent representations

The structured image and text models suggest further work on representations that support both expressive generation and interpretable intervention. Discrete–continuous latent structures and region-specific generative components offer potential interfaces for constrained generation and compositional manipulation. Their usefulness should be assessed not only through sample quality, but also through the stability and identifiability of the learned structure and the reliability of the resulting controls.

8.5 Closing Perspective

A neural generative model is more than the probability family represented by its architecture. In practice, its behavior emerges from a combination of probabilistic representation, statistical estimation, approximate inference, and numerical sampling. The contributions of this dissertation illustrate how improvements at each of these levels can affect the reliability, efficiency, and usefulness of generation. Developing methods that reason about these components together is a promising direction toward generative systems that are expressive, computationally tractable, and controllable.