

UC Irvine

UC Irvine Electronic Theses and Dissertations

Title

Hierarchical Variational Inference for Deep Generative Models

Permalink

<https://escholarship.org/uc/item/3sr5732n>

Author

Agarwal, Sakshi

Publication Date

2025

Copyright Information

This work is made available under the terms of a Creative Commons Attribution License, available at <https://creativecommons.org/licenses/by/4.0/>

Peer reviewed|Thesis/dissertation

UNIVERSITY OF CALIFORNIA,
IRVINE

Hierarchical Variational Inference for Deep Generative Models

DISSERTATION

submitted in partial satisfaction of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

in Computer Science

by

Sakshi Agarwal

Dissertation Committee:
Professor Erik Sudderth, Chair
Professor Padhraic Smyth
Professor Stephan Mandt

2025

DEDICATION

To my family and friends for providing support throughout my graduate studies.

TABLE OF CONTENTS

	Page
LIST OF FIGURES	v
LIST OF TABLES	viii
LIST OF ALGORITHMS	ix
ACKNOWLEDGMENTS	x
VITA	xii
ABSTRACT OF THE DISSERTATION	xiv
1 Introduction	1
1.1 Background	2
1.1.1 Probabilistic Graphical Models	2
1.1.2 Latent Variable Models	3
1.1.3 Variational Inference	4
1.2 Deep Latent Variable Models for Images	5
1.2.1 VAEs	5
1.2.2 Hierarchical VAEs	6
1.2.3 Diffusion Models	8
1.3 Outline of Contributions	10
2 Computing Partition Function for Discrete Generative Models	14
2.1 Introduction	15
2.2 Background	17
2.2.1 Bucket Elimination	18
2.2.2 Weighted Mini-Bucket	20
2.2.3 Deep Bucket Elimination	20
2.3 NeuroBE	22
2.4 Error Analysis	28
2.5 Empirical evaluation	31
2.5.1 Experimental Setup	31
2.5.2 Results	32
2.6 Conclusion & Future Work	36

2.7	Appendix	37
2.7.1	Estimating the pseudo-dimension of a NN:	37
2.7.2	Estimating error in partition function:	37
3	Variational Inference for (Hierarchical) VAEs	42
3.1	Introduction	43
3.2	Background and Related Work	45
3.2.1	The Variational Autoencoder (VAE)	45
3.2.2	Hierarchical VAEs	46
3.2.3	Methods for Inferring Missing Data	47
3.3	Query Adaptive VI	49
3.3.1	VI via Missing Feature Posteriors	50
3.3.2	Non-amortized VI	52
3.4	Experiments & Results	56
3.4.1	Experimental Setup	56
3.4.2	Results	60
3.5	Conclusion	63
3.6	QAVI: Derivations	65
3.6.1	Feat. QAVI	65
3.6.2	Non-Amortized Inference	66
4	Variational Inference for Denoising Diffusion Models	67
4.1	Introduction	68
4.2	Background	71
4.3	Background: Inference with Diffusion Models	72
4.3.1	Sampling Methods for Inverse Problems	73
4.3.2	Red-Diff: Variational Posterior over data x	75
4.4	VIPaint	76
4.5	Experiments	81
4.5.1	Experimental Setup	81
4.5.2	Results	84
4.6	Conclusion	87
4.7	Appendix	88
4.7.1	VIPaint: VI method using Diffusion Models as priors	88
4.7.2	Derivation of $L_{(T_e, T)}(z_{T_e})$	89
5	Conclusion and Future Directions	92
5.1	Summary of Methods and Contributions	92
5.2	Suggestions for Future Research	94
5.2.1	Faster non-amortized variational inference	94
5.2.2	Extension to Text-to-Image diffusion models	95
5.2.3	Other Application Areas	96
	Bibliography	98

LIST OF FIGURES

	Page
1.1 A primal graph of a graphical model with 7 variables.	3
1.2 An example of latent variable model, also adapted by a simple VAE [97]. The shaded nodes is observed.	4
1.3 Graphical model for generative and inference procedure in a hierarchical VAE.	6
1.4 Graphical model for the generative and inference process in Diffusion Models.	9
2.1 (a) A primal graph of a GM with 7 variables. (b) Illustration of BE with an ordering A B C E D F G.	18
2.2 For a bucket of width w , we illustrate a NN architecture with $L(= 2)$ layers and $b \cdot w$ hidden-units with $b \geq 1$	24
3.1 Overview of Feat. QAVI (a , Sec. 3.3.1) and non-amortized QAVI (b , Sec. 3.3.2). <i>Top</i> : Generative model for unobserved variables x_M, z and observed data x_O (shaded). We show two possible inference models for the joint variational distribution $q_\lambda(x_M, z)$ on latent variables. In (a), missing features are modeled directly by specifying $q_\lambda(x_M)$. In (b), queries are modeled indirectly via a tunable latent code distribution $q_\lambda(z)$. QAVI is easily adapted to existing VAEs by reusing generative (decoder) networks p_θ , and possibly inference (encoder) networks q_ϕ . <i>Bottom</i> : Computational flow of QAVI for each posterior factorization. In Feat. QAVI (a), samples from $q_\lambda(x_M)$ are autoencoded by the pre-trained VAE to compute the ELBO. In non-amortized QAVI (b), samples from $q_\lambda(z)$ are only passed through the decoder; the pre-trained encoder is not needed.	50
3.3 Kernel density visualizations of the prior and approximate posteriors for five QAVI variants, fit using $S = 1000$ samples from the optimized variational distribution. We show one test sample chosen from two UCI tabular datasets, Banknote (<i>top</i>) and Concrete (<i>bottom</i>), containing two missing features (labeled on axes). We observe that the true data point (<i>red</i>) is often enclosed by the different posteriors, but sometimes missed by Feat. QAVI. We approximate the true posterior via a highly expressive QAVI mixture of 100 Gaussians, fit by extended optimization.	58

3.5	Digit completion results on MNIST (<i>left</i>) and SVHN (<i>right</i>) images for inference queries with pixels obscured by the Rotating-Half or Random-Patches distributions. We show 5 samples from each inferred posterior. Monte Carlo and amortized inference methods propose one sometimes-valid digit completion; amortized VI is typically more accurate. The performance of Posterior Match varies widely depending on the missing-feature distribution it is trained on. In contrast, QAVI automatically adapts to queries, and proposes multiple valid imputations that effectively capture posterior uncertainty.	60
4.1	Top: Sampling methods like CoPaint-TT [137] and DPS [17] anneal from Gaussian noise to noise-free images while guiding samples towards observation data and tend to produce incoherent images. Whereas, RedDiff [78] defines its variational posterior directly on the noise-free data space and produces blurry images. Direct optimization of its variational bound (RedDiff-V) leads to poor local optima, so must resort to annealing through diffusion time (RedDiff, $T \rightarrow 0$). Our VIPaint method suffers from no such difficulty and uses the variational bound for optimization, leading to more robust inference. Bottom: We visualize samples generated at intermediate time points in the diffusion latent space and compare VIPaint with the best performing baseline, CoPaint-TT. We see that intermediate samples for CoPaint-TT don't infer the underlying image correctly, and that these errors cannot be corrected later in the sampling process, leading to severe inconsistencies in final inpaintings as shown. In contrast, samples from our VIPaint posterior align better with the observations, even at higher noise levels, resulting in more coherent inpaintings.	70
4.2	<i>Top:</i> The hierarchical approximate posterior of VIPaint is defined over a coarse sequence of intermediate latent steps between T_e and T_s . During optimization, the variational parameters λ defining the posterior on a subset of latent times are fit via a prior loss on times above T_e , a hierarchical loss defined across K intermediate times, and a reconstruction loss estimated using a one-step approximation $p_\theta(x z_{T_s})$ from the posterior samples. <i>Bottom:</i> After variational inference, samples from the hierarchical posterior (now <i>aligned</i> with the observation) transition smoothly in the intermediate latent space $[0, T_s]$ via gradient updates.	73
4.3	We show VIPaint's posterior fitting progress and sample generation every 50 iterations for two test cases. We see VIPaint quickly grasps the image semantics within 50 optimization iterations.	78
4.4	An ablation showing the effect of addition the perceptual loss (PLoss) in the reconstruction term for the task of image inpainting using latent diffusion priors. We see that even though VIPaint can inpaint the image semantically without the Perceptual loss, this loss becomes important to produce sharper reconstructions.	79

4.5	Image completion results using the LDM prior for Imagenet256 (left) and LSUN (right) for large-mask inpainting (Random Masking and Rotated Window schemes). We show inpaintings from each method in the following columns. DPS, PS LD, and ReSample produce blurry inpaintings of varying quality. Despite being conditioned on class labels, baseline methods' inpaintings for ImageNet are inconsistent with the observed image. In contrast, VIPaint captures global semantics from the observed image, producing highly realistic inpaintings.	80
4.6	Image completion results using the pixel-based EDM model for ImageNet64 with for large-mask inpainting (Random Masking and Rotated Window schemes). We show inpaintings from each method in the following columns. Posterior sampling methods for pixel-based diffusion priors, like RePaint and CoPaint, are generally more accurate but still produces inconsistent samples. In contrast, VIPaint infers the underlying image correctly, and produces highly realistic inpaintings.	80
4.7	Left: Sample completions comparing VIPaint with the best performing baseline, CoPaint, for two test images. We show the true and masked images, and 5 inpainted samples for each method. VIPaint yields coherent samples while capturing uncertainty in the missing pixels in images. In comparison, CoPaint has high variance in the quality of results. Right: We compute summary statistics (minimum, mean, maximum) of the LPIPS score across 100 sampled completions per test image. We show the average value of each of these statistics across the test set. We see that VIPaint improves on baselines, both in terms of the average quality of results and in the consistency in result quality.	82
4.8	Qualitative results on other linear inverse problems : Gaussian Deblurring (left) and Super Resolution (right) for Imagenet256 using LDM prior. VIPaint leads to sharp, realistic samples without any artifacts.	83
5.1	We use the pre-trained conditional diffusion model from [98] that takes a masked observation (examples on the left) and their respective mask as input to the diffusion model, and produces an inpainting (shown on right) after the sequential denoising process. We see that for the large-mask image inpainting scenarios, this pre-trained diffusion model is unable to yield realistic images.	94
5.2	For the masked image on the left and the input prompt "Kitchen appliances and cabinets as seen through opening." we present two possible image completions generated using the Stable Diffusion Inpainting model [98] on the right . These inpaintings exhibit artifacts, such as incomplete fridge handles and an incoherent water dispenser. To address these issues, one could use an inference algorithm like VIPaint on top of the Stable Diffusion model to produce more realistic and diverse inpaintings.	95
5.3	We use the pre-trained MolMo model [28] that takes an image and produces its caption. On the left , we show the generated caption given a clean image, and on the right , we show the generated caption on its cropped version. You can see that the caption on the right incorrectly infers the caption of the book (without acknowledging uncertainty), and ignores description in the missing parts of the image.	96

LIST OF TABLES

	Page
3.1 Test missing data log-likelihoods (LL, higher is better) and normalized root mean-square error (NRMSE, lower is better) for 5 tabular datasets from the UCI repository, estimated using $S = 10,000$ samples. NRMSE per test row is the minimum across S samples. QAVI variants have superior performance (highlighted in bold) for almost all data.	59
3.2 Quantitative comparison of perceptual inpainting quality on the FFHQ-256 dataset. We compare QAVI against two state-of-the-art adaptations of HVAEs to inpainting, using the same base “very-deep” HVAE architecture.	62
4.1 Quantitative results (LPIPS, lower is better) for ImageNet64 for the task of image inpainting using pixel-based EDM prior (<i>top</i>) and Imagenet-256 and LSUN-Church using LDM priors (<i>bottom</i>). LPIPS is estimated as the mean score of 10 inpaintings with respect to the true image, averaged across the test set. VIPaint has superior performance (highlighted in bold) in nearly all cases. We <u>underline</u> the second best method.	82
4.2 Table comparing time (in mins) for three inference methods using EDM prior (top) and LDM prior (bottom): Sampling (CoPaint-TT for pixel-based EDM model and PSLD for LDM prior), RedDiff and VIPaint	86
4.3 Quantitative results (LPIPS, PSNR) for solving linear inverse problems on ImageNet256 using LDM priors and ImageNet64 using EDM priors. Best results are in bold and second best results are underlined. For nonlinear deblurring.	86

LIST OF ALGORITHMS

	Page
1 NeuroBE	23
2 generate-samples(X, F, N)	25
3 NN-train($F, X, L, b, \eta, \#epochs$)	27

ACKNOWLEDGMENTS

First of all, I would like to thank my great advisor, Professor Erik Sudderth, who has been an incredible guiding force throughout my entire PhD journey. From the very beginning, he patiently listened to my broad ideas and interests, helping me shape them into meaningful projects in the early years. As my research progressed, he not only provided insightful guidance but also became a constant source of motivation during the challenging times when I felt low. His vast knowledge in the field allowed me to gain a deep understanding of my research area, and his caring nature made me feel supported every step of the way. In addition to his academic mentorship, he also celebrated my personal milestones, including supporting my wedding celebrations at the end of my PhD. His guidance, kindness, and unwavering support have truly made this journey a remarkable and fulfilling experience.

I would like to express my sincere gratitude to my thesis committee members, Professor Padhraic Smyth and Stephan Mandt, for their invaluable support throughout my graduate studies. Both Padhraic and Stephan taught courses during my early years at UCI, and the knowledge I gained from their classes ultimately led me toward choosing the exciting topic for my dissertation. Their individual experience and thoughtful advice on professional development have been instrumental in shaping my academic journey, always eager to offer guidance and advocating for my best interests. I deeply appreciate their contributions to my academic and professional growth.

I am deeply grateful to all the labmates in our Learning, Inference, and Vision group. Gabriel Hope and Harry Bendegkey were my closest collaborators, and I spent much of my time with them. Their brilliant mathematical backgrounds, combined with their engineering skills, provided an excellent foundation for peer learning and stimulating discussions. I am fortunate to have had them as sounding boards for my ideas. I also thoroughly enjoyed collaborating with Ali Younis, Yasaman Razeghi, Bobak Pezeshki, and Nick Cohen on various projects. I am especially thankful for my labmate and friend, Federica Zoe Ricci, whose discussions have been invaluable to me both personally and professionally. A heartfelt thank you goes to my close friends Rachel Longjohn, Catalina Meri Medina, and Bianca Brusco in the Statistics Department, whose support helped me navigate the rigorous demands of conference submissions throughout my PhD. I am also grateful to my mentee, Jimin Heo, an incredibly smart incoming student in the LIV lab, whose energy and excellent learning attitude continue to inspire me. Lastly, I truly enjoy interacting with junior members like Mehrnaz Motamed and Daniele Micheletti, who give me confidence in the bright future of the LIV lab—a place I will always feel connected to.

I would also like to thank all the other residents of the DBH building at UC Irvine, who spent days and nights with me together during my graduate life. I'm also super grateful to the wonderful colleagues I met at Amazon during the summer internship.

To my family, thank you for your unwavering support from India throughout my doctoral journey. To my brother, Ayush, I'm deeply grateful for staying up late to chat with me and always adjusting your schedule to mine. To my parents, your endless guidance has enriched

my life beyond measure. Through countless WhatsApp calls, helping me recreate nostalgic dishes in the kitchen whenever I missed home, you've taken care of me in ways I will never forget. Your love and support have made this journey possible, and I am forever grateful.

To my husband, Rahul, I'm so thankful that our paths crossed during the first year of our undergraduate studies at IIT KGP. Knowing you has been the happiest part of my life. From our time as students in Kharagpur to the hustle and bustle of Bangalore, and wherever life may take us, your constant support, appreciation, and joy give me so much stability and happiness. I look forward to your humor and insight every day. I love you deeply.

Finally I want to thank the NSF grants and HPI Research Center in Machine Learning and Data Science at UC Irvine for supporting my PhD works.

VITA

Sakshi Agarwal

EDUCATION

Doctor of Philosophy in Computer Science	2021-2025
University of California, Irvine	<i>Irvine, California</i>
Masters of Science in Computer Science	2019-2021
University of California, Irvine	<i>Irvine, California</i>
Bachelor of Technology in Instrumentation Engineering	2013-2017
Indian Institute of Technology, Kharagpur	<i>Kharagpur, West Bengal</i>

INDUSTRY EXPERIENCE

Applied Scientist Intern	2007–2012
Amazon	<i>Seattle, Washington</i>

TEACHING EXPERIENCE

UCI CS 175 Project In Artificial Intelligence	Winter 2022
UCI CS 171 Introduction In Artificial Intelligence	Spring & Winter 2021
UCI CS 265 Graph Algorithms	Winter 2020

ACADEMIC REVIEWING

International Conference on Machine Learning (ICML)	2025
Conference on Neural Information Processing Systems (NeurIPS, top reviewer)	2024
Association for Uncertainty in Artificial Intelligence	2024
Society for Artificial Intelligence & Statistics	2023

REFEREED CONFERENCE PUBLICATIONS

Sakshi Agarwal, Gabriel Hope, Ali Younis, Erik B. Sudderth. A decoder suffices for query-adaptive variational inference. Spotlight in Uncertainty in Artificial Intelligence (UAI), 2023.

Sakshi Agarwal, Kalev Kask, Alexander Ihler, Rina Dechter. NeuroBE: Escalating NN Approximations of Bucket Elimination. Uncertainty in Artificial Intelligence (UAI), 2022

Sakshi Agarwal, Kalev Kask, Alexander Ihler, Rina Dechter. NeuroBE: NN Approximations to Bucket Elimination. AAAI Workshop Combining Learning and Reasoning (CLeaR), 2022

Yasaman Razeghi, Kalev Kask, Yadong Lu, Pierre Baldi, **Sakshi Agarwal** and Rina Dechter. Deep Bucket Elimination. International Joint Conference on Artificial Intelligence (IJCAI), 2021.

ABSTRACT OF THE DISSERTATION

Hierarchical Variational Inference for Deep Generative Models

By

Sakshi Agarwal

Doctor of Philosophy in Computer Science

University of California, Irvine, 2025

Professor Erik Sudderth, Chair

Probabilistic generative models offer a principled way to represent and learn complex data distributions. By defining a joint probability distribution over the data, these models enable a wide array of inference queries, such as imputation of missing data given observations of other data dimensions. State-of-the-art deep generative models, including Variational Autoencoders (VAEs) and Denoising Diffusion Models, integrate deep learning methods to better model complex data like images. Despite rapid advances in learning deep generative models, accurate estimation of inference queries remains a significant challenge, and standard methods perform poorly in many scenarios. VAEs typically employ learning-based amortized inference, which results in an inflexible model that struggles to handle queries it has not been trained on, and requires expensive retraining of the inference network for each new query task. Diffusion models often require heuristics to apply sampling-based inference, leading to slow convergence and highly inaccurate predictions for hard inference queries like large-mask image inpainting.

In cases where data sensitivity is crucial, we often seek not only accurate predictions, but estimates of posterior uncertainty. To address these challenges, we propose leveraging classical optimization-based variational inference that adapts to versatile queries, showcasing its efficacy for two example inference queries: 1) estimating partition functions in discrete graphical

models, and 2) imputing missing features with VAEs and diffusion models. Through both theoretical analysis and empirical validation, we demonstrate that our approach significantly enhances the accuracy of inference while capturing posterior uncertainty, making it suitable for real-world user interactions. This thesis thus lays the foundation for further research into scalable and accurate inference techniques, and facilitates the practical deployment of probabilistic generative models in increasingly complex and high-dimensional domains.

Chapter 1

Introduction

Probabilistic graphical models (PGMs) [66, 77] are the basis for modeling uncertainty in many areas such as computer vision, speech processing, time-series and sequential data modelling, bioinformatics, and artificial intelligence in general. *Learning* such graphical models utilizes unlabeled data, incorporates domain knowledge into the model and employs unsupervised training methods. The framework of graphical models further enables diverse *inference* queries where computing conditional probability distributions is of interest. For instance, *imputation* queries (or, inpainting in the image domain) observe arbitrary subsets of features and require the prediction of the values of missing features. While exact inference is intractable for many models, sampling approaches like Markov Chain Monte Carlo (MCMC) approximate the conditionals by randomly drawing samples from them. On the other hand, variational methods [120] approximate the distribution of interest by optimizing a simpler, tractable distribution to closely resemble the true conditional, often using gradient descent-like algorithms. Previous work has shown inference queries on graphical models could be effectively approximated through variational methods, whereas exact or sampling-based inference is often computationally infeasible [51, 48, 37, 45, 124]. In this chapter, we first do a quick review of the basics of probabilistic graphical models and a special class of latent

variable models along with their inference queries of interest. This is followed by the fundamental ideas behind variational inference and some widely known Deep Generative models like VAEs and Diffusion Models. Then we discuss some challenges around inference methods to compute queries of interest and introduce our contributions that empower variational algorithms for a wide range of inference queries.

1.1 Background

1.1.1 Probabilistic Graphical Models

A probabilistic graphical model [92, 23, 26, 25] can be defined by a 3-tuple $\mathcal{M} = (\mathbf{X}, \mathbf{D}, \mathbf{F})$, where $\mathbf{X} = \{X_i : i \in V, V = \{1, \dots, n\}\}$ is a set of n variables indexed by V , and $\mathbf{D} = \{D_i : i \in V\}$ is the set of domains for each X_i (i.e. each X_i can only assume values in D_i , and each D_i can be finite for discrete variables or infinite for continuous variables). Each function $f_\alpha \in \mathbf{F}$ is defined over a subset of the variables called its scope, X_α , where $\alpha \subseteq V$ are the indices of variables in its scope, and D_α denotes the Cartesian product of their domains so that $f_\alpha : D_\alpha \rightarrow R \geq 0$.

The **primal graph** of a graphical model associates each variable with a node. An edge between node i and node j is created if and only if there is a function containing X_i and X_j in its scope. Figure 2.1a shows a primal graph of a graphical model with variables indexed from A to G and functions over pairs of variables are connected by an edge. Graphical models can be used to represent a global function, often a probability distribution, defined by $P(X) \propto \prod_\alpha f_\alpha(X_\alpha)$. An important task is to compute the normalization constant, also known as the partition function $Z = \sum_X \prod_\alpha f_\alpha(X_\alpha)$ or compute conditionals $P(X|Y)$ over some evidence variable (Y) and other variables must be inferred.

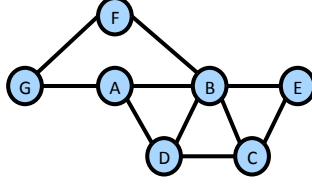


Figure 1.1: A primal graph of a graphical model with 7 variables.

1.1.2 Latent Variable Models

Latent variable models [8, 11, 12] are probabilistic models where unobserved (latent) variables z explain the dependencies between observed data x . Take $p(x, z) = p(z)p(x|z)$ as an example shown in Fig 1.2. $p(z)$ describes the prior knowledge about the distribution of values that z may take, and $p(x|z)$ is the likelihood function. Learning these models captures underlying structures or factors that influence the data, making them useful for generative modeling.

Recent advancements also leverage deep neural networks to model complex relationships between the unobserved z and observed x variables. For instance, Variational Autoencoders (VAEs, [97, 60]), use a neural network to approximate the likelihood $p(x | z)$ and the posterior $p(z | x)$ using a set of parameters θ, ϕ respectively. Other notable models, like Diffusion Models [44, 53, 62], have also emerged, creating hierarchical dependencies among the latent codes z and further increasing model expressiveness for complex data like images. Hence, these models play a particularly useful role as priors for image data and can be further used to compute the distribution over all possible images x , conditioned on a partial observation y , i.e. $P(x | y)$. Since exact computation is intractable due to the non-linear nature of neural networks, approximations are widely used. As we explore in Chapters 3 and 4, a central challenge in current inference methods is the *accurate* approximation of the latent variable z based on the observation y , i.e. $P(z | y)$.

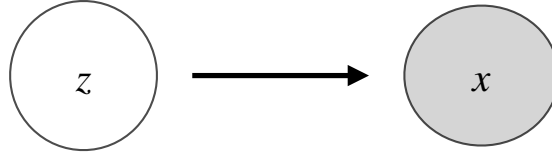


Figure 1.2: An example of latent variable model, also adapted by a simple VAE [97]. The shaded nodes is observed.

1.1.3 Variational Inference

Given fully-observed data x , we are often interested in the posterior probability $p(z|x)$, which describes how the latent patterns of the model explain the data observations. According to Bayes' rule, we have:

$$p(z | x) = \frac{p(z, x)}{p(x)} = \frac{p(z, x)}{\int_z p(x, z) dz}. \quad (1.1)$$

While the numerator $p(z, x)$ is easy to evaluate given the generative process, the data marginal $p(x)$ in the denominator, also known as the evidence, is often intractable to compute. Mean-field variational inference (VI) [120] seeks to find a variational distribution $q(z)$ that minimizes the Kullback-Leibler (KL) divergence to the true posterior

$$D_{KL}(q(z)||p(z|x)) = \mathbb{E}_{q(z)} \left[\log \frac{q(z)}{p(z | x)} \right] = \int \left[\log q(z) - \log p(z | x) \right] q(z) dz. \quad (1.2)$$

Because the KL divergence is always non-negative, minimizing Equation 1.2 is equivalent to maximizing the evidence lower bound (ELBO, [51])

$$\begin{aligned} \mathcal{L}(q(z)) &\approx \mathcal{E}_{q(z)}[\log p(z, x) - \log q(z)] \\ &= \log p(x) - D_{KL}(q(z)||p(z | x)) \leq \log p(x), \end{aligned} \quad (1.3)$$

in which equality is achieved if and only if $q(z)$ is identical to the true posterior $p(z|x)$ (exact inference). To make variational inference useful in practice, one typically restricts $q(z)$ to a family of simpler distributions Q , and looks for the best approximation within it. The simplest and most common choice is the naïve mean field, where $q(z)$ is fully factorized across each hidden variable. In this thesis, we also consider a more structured mean field, which tightens the evidence lower bound by enabling more complex hierarchical variational dependencies.

1.2 Deep Latent Variable Models for Images

In this section, we provide background on two deep latent variable models: VAEs and Diffusion Models. Both are widely adopted for generating realistic images. Test-time image inpainting with these models will be explored in the chapters 3 and 4 respectively.

1.2.1 VAEs

VAEs [64, 97] model the distribution of typically high-dimensional observed data x using continuous, lower-dimensional latent variables z , via the following generative model:

$$z \sim p(z), \quad x \sim p_\theta(x | z). \tag{1.4}$$

For standard VAEs, the latent code $z \in \mathbb{R}^d$ has a factorized Gaussian prior $p(z)$. Given z , data is generated via a *decoder* (deep) neural network with weights θ . The decoder maps z to a likelihood $p_\theta(x|z)$ such as a factorized Gaussian. The VAE log-likelihood $\log p_\theta(x) = \log \int p_\theta(x|z)p(z) dz$ is intractable. Learning thus typically employs *amortized* VI, where an *encoder* with parameters ϕ approximates the posterior over latent codes $q_\phi(z|x) \approx p_\theta(z|x)$.

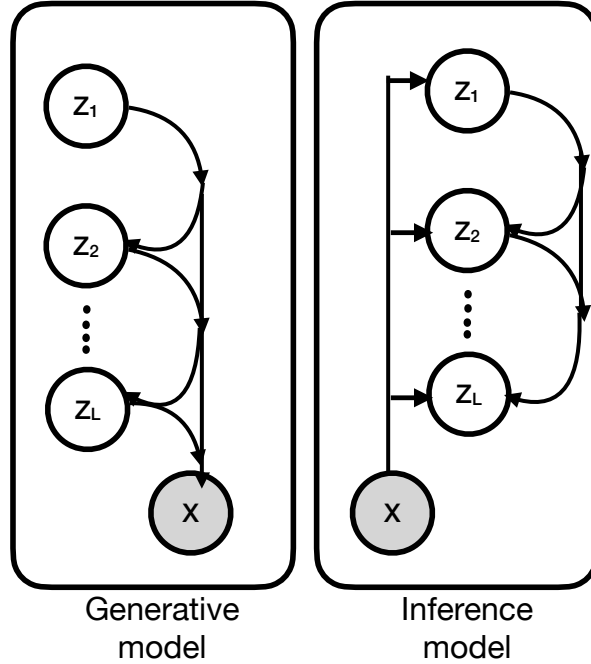


Figure 1.3: Graphical model for generative and inference procedure in a hierarchical VAE.

We jointly learn θ, ϕ by maximizing the *evidence lower-bound* (ELBO):

$$\mathcal{L}(\theta, \phi; x) = E_{q_\phi(z|x)}[\log p_\theta(x|z)] - D_{KL}(q_\phi(z|x) || p(z)). \quad (1.5)$$

Here, D_{KL} is the Kullback–Leibler divergence, calculated analytically when $q_\phi(z|x)$ and $p(z)$ are both Gaussian. The ELBO provides a lower bound on the evidence $\log p_\theta(x)$ that is tight when the variational posterior $q_\phi(z|x)$ is exact. This expectation can be approximated via Monte Carlo samples from $q_\phi(z|x)$. Gradients with respect to θ, ϕ can then be estimated by the reparameterization “trick” of sampling from $q_\phi(z|x)$ via linear transforms of standard normal variables.

1.2.2 Hierarchical VAEs

Hierarchical VAEs (HVAEs, [107, 65]) extend the VAE by partitioning the latent code into L disjoint groups $z = (z_1, z_2, \dots, z_L)$, increasing model expressiveness for complex data like

images [117, 16]. HVAEs generate these stochastic codes sequentially as

$$p_\theta(x|z) = p_\theta(z_1) \left(\prod_{\ell=2}^L p_\theta(z_\ell | z_{<\ell}) \right) p_\theta(x | z_L). \quad (1.6)$$

with a similar encoder:

$$q_\phi(z|x) = q_\phi(z_1|x) \prod_{\ell=2}^L q_\phi(z_\ell | z_{<\ell}, x). \quad (1.7)$$

A graphical model representation is shown in Fig. 1.3. Each conditional in the decoder $p_\theta(z_\ell | z_{<\ell})$, and the encoder $q_\phi(z_\ell | z_{<\ell}, x)$, is typically Gaussian with mean and variance determined by (non-linear) neural networks.

The HVAE ELBO equals

$$\begin{aligned} \mathcal{L}_H(\theta, \phi; x) = & E_{q_\phi(z|x)} [\log p_\theta(x|z)] - (q_\phi(z_1|x) || p_\theta(z_1)) \\ & - \sum_{\ell=2}^L E_{q_\phi(z_{<\ell}|x)} [(q_\phi(z_\ell | z_{<\ell}, x) || p_\theta(z_\ell | z_{<\ell}))], \end{aligned} \quad (1.8)$$

where $q_\phi(z_{<\ell}|x) = \prod_{i=1}^{\ell-1} q_\phi(z_i | z_{<i}, x)$ is the approximate posterior up to latent group $(\ell - 1)$.

Reparameterization is then used to provide Monte Carlo gradient estimates. We can rewrite the conditional prior and approximate posterior for layer ℓ to make the set of relevant networks explicit:

$$\begin{aligned} p_\theta(z_\ell | z_{<\ell}) &= \mathcal{N}(z_\ell | \mu_{\theta_\ell}(z_{<\ell}), \sigma_{\theta_\ell}(z_{<\ell})), \\ q_\phi(z_\ell | z_{<\ell}, x) &= \mathcal{N}(z_\ell | \mu_{\phi_\ell}(f_{\phi_\ell}(x), g_{\phi_\ell}(z_{<\ell})), \sigma_{\phi_\ell}(\dots)). \end{aligned} \quad (1.9)$$

Here, f_{ϕ_ℓ} and g_{ϕ_ℓ} are networks that extract feature representations of the observation x and the previous layers $z_{<\ell}$, respectively. These features determine the mean and scale of the conditional Gaussian posterior via $\mu_{\phi_\ell}, \sigma_{\phi_\ell}$. Networks $\mu_{\theta_\ell}, \sigma_{\theta_\ell}$ similarly generate the prior

parameters for layer ℓ .

1.2.3 Diffusion Models

The second class of models we consider is Diffusion Models [44, 112, 89, 110]. Different from an encoder-decoder architecture, they employ a sequential denoising technique to generate data. The diffusion process begins with clean data x , and defines a sequence of increasingly noisy versions of x , which we call the *latent variables* z_t , where t runs from $t = 0$ (low noise) to $t = T$ (substantial noise). The distribution of latent variable z_t conditioned on x , for any integer time $t \in [0, T]$, is

$$q(z_t | x) = \mathcal{N}(z_t | \alpha_t x, \sigma_t^2 I), \quad (1.10)$$

where α_t and σ_t are strictly positive scalar functions of t . This noise implicitly defines a Markov chain for which the conditionals $q(z_t | z_{t-1})$, $q(z_{t-1} | z_t, x)$ are Gaussians. The signal-to-noise ratio [62] induced by this diffusion process at time t equals $SNR(t) = \alpha_t^2 / \sigma_t^2$. The SNR monotonically decreases with time, so that $SNR(t) < SNR(s)$ for $t > s$. This DM specification includes variance-preserving diffusions [44, 106] as a special case, where $\alpha_t = \sqrt{1 - \sigma_t^2}$. Another special case, variance-exploding diffusions [110, 112], takes $\alpha_t = 1$.

The generative model reverses the diffusion process outlined in Eq. (4.1), resulting in a hierarchical generative model that samples a sequence of latent variables z_t before sampling x . A graphical representation is given in Fig. 1.4. Generation progresses backward in time from $t = T$ to $t = 0$ via a finite temporal discretization into $T \approx 1000$ steps, either uniformly spaced as in discrete diffusion models [44], or via a possibly non-uniform discretization [53] of an underlying continuous-time stochastic differential equation [112]. Denoting $t - 1$ as the timestep preceding t , for $0 < t < T$, the hierarchical generative model for data x is expressed

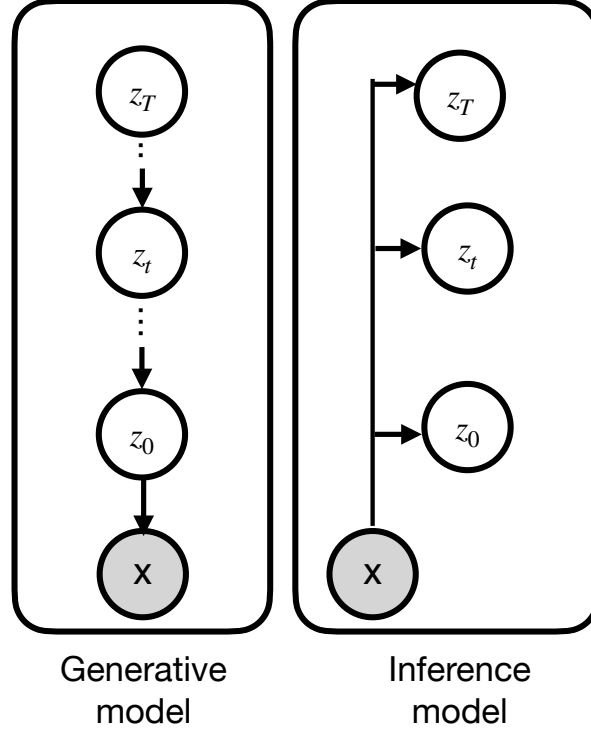


Figure 1.4: Graphical model for the generative and inference process in Diffusion Models.

as follows:

$$p_{\theta}(x) = \int_z p(z_T) p(x | z_0) \prod_{t=1}^T p_{\theta}(z_{t-1} | z_t) dz. \quad (1.11)$$

The marginal distribution of z_T is typically a spherical Gaussian $p(z_T) = \mathcal{N}(z_T | 0, \sigma_T^2 I)$. Pixel-based diffusion models take $p(x | z_0)$ to be a simple factorized likelihood for each pixel in x , while LDMs define $p(x | z_0)$ using a decoder neural network. The conditional latent distribution $p_{\theta}(z_{t-1} | z_t)$ maintains the same form as the forward conditional distribution $q(z_{t-1} | z_t, x)$, but with the original data x substituted by the output of a parameterized denoising model, z_0 as

$$p_{\theta}(z_{t-1} | z_t) = q(z_{t-1} | z_t, z_0 = \hat{z}_{\theta}(z_t, t)), \quad \text{where} \quad \hat{z}_{\theta}(z_t, t) = \frac{z_t - \sigma_t \hat{\epsilon}_{\theta}(z_t, t)}{\alpha_t}. \quad (1.12)$$

This denoising model $\hat{\epsilon}_\theta(z_t, t)$ typically uses variants of the UNet architecture [100] and is trained to optimize a re-weighted variational lower bound of the marginal likelihood of data x , simplified as [44, 112],

$$\mathcal{L}_{(0,T)}(z_0) = \frac{T}{2} \mathbb{E}_{\epsilon \sim \mathcal{N}(0,1), t \sim \mathcal{U}(1,T)} \left[\|\epsilon - \hat{\epsilon}_\theta(z_t, t)\|_2^2 \right], \quad (1.13)$$

where the expectation is taken over t , $z_t \sim q(z_t|x)$, and $x \sim p_{\text{data}}(x)$. Latent Diffusion Models [98, 118] use an encoder $q_\phi(z_0|x)$ to map x to a lower-dimensional space and train a diffusion model in this reduced encoding space for efficiency.

1.3 Outline of Contributions

Generative models aim to learn the underlying complex distribution of data. Further, deep latent variable models introduce an additional set of latent variables that capture extra information about the data, making the conditioned distribution simpler and easier to model. By learning such a probability distribution, practitioners can uncover hidden structures in the data correctly and build inference methods that address queries of interest. We explore a plethora of these inference strategies used across generative models, specifically, discrete graphical models, VAEs, and diffusion models, in their respective chapters. This thesis proposes variational inference approaches to perform accurate predictions across a wide-ranged inference queries. We also discuss the time-accuracy trade-offs across these inference schemes.

Chapter 2¹ considers the case where a fixed discrete graphical model encodes specific domain knowledge (for instance, genetic law of inheritance of a group of individuals). Querying the

¹Contents of this chapter are mainly based on our work “NeuroBE: Escalating Neural Network Approximations of Bucket Elimination” published in UAI 2022. Authors are Sakshi Agarwal, Kalev Kask, Alex Ihler and Rina Dechter.

probability of occurrence of any event (like, observing a specific gene A in person X) involves computing the *partition function*. Exact message-passing algorithms such as Bucket Elimination (BE, [24]) require exponential memory to compute the partition function; therefore, approximations are necessary. In this chapter, we build upon a recently introduced methodology called Deep Bucket Elimination (DBE, [95]) that uses classical Neural Networks to approximate the distributions over large messages generated by BE that cannot be stored in memory. These neural networks are trained to approximate these messages *sequentially* during inference. Training each network employs supervised methods and the main feature of our new scheme, renamed NeuroBE, is that it customizes the architecture of the neural networks, their learning process and in particular, adapts the loss function to the internal form or distribution of messages, disputing the one-size-fits-all policy during learning. Our experiments demonstrate significant improvements in accuracy and time compared with the earlier DBE scheme.

Chapter 3² focuses on image inpainting with pre-trained (hierarchical) variational autoencoders (VAEs) [64, 97]. These models infer latent representations via amortized inference algorithms, which require that all data dimensions are observed. Thus, VAEs inherently lack a key strength of probabilistic graphical models: the ability to infer posteriors for test queries with arbitrary structure. Many prior methods then create datasets to extensively fine-tune inference networks [20, 113, 40, 47], thus making imputation with VAEs very costly and ineffective. Moreover, the need to create a train set for fine-tuning can be expensive, and the resulting model is often biased towards mask-specific training.

Fine-tuning amortized inference networks incurs both an amortization and approximation gaps well-known in the literature [22]. We aim to reduce these inference gaps by defining the variational posterior for each inference query in the latent space, completely eliminating the use of pre-trained amortized inference network. We further enable optimizing expressive

²Contents of this chapter are mainly based on our work "A Decoder Suffices for Query-Adaptive Variational Inference" published in UAI 2023. Authors are Sakshi Agarwal, Gabe Hope, Ali Younis and Erik Sudderth.

posteriors including Gaussian mixtures and density flows [61, 96], in order to capture the uncertainty in the posterior over the missing pixels, thus capturing multiple plausible reconstructions in the observed image. We fit such posteriors using the Evidence Lower Bound (ELBO) on the masked observation. On standard image and tabular datasets, our approach substantially outperforms prior methods in the plausibility and diversity of imputations. Moreover, our non-amortized QAVI algorithms achieve excellent performance while avoiding expensive model retraining across a wide-ranged masking distribution. We also show that QAVI effectively generalizes to recent hierarchical VAE models [107] for high-dimensional images, and enables fitting a hierarchical posterior jointly over all the latent variables in the model optimizing the ELBO on the observation.

Chapter 4³ focuses on Diffusion Models [44, 53], a fairly new generative model for images. As mentioned briefly in the previous section, Denoising Diffusion Models learn to remove noise added during training, generating novel data (e.g., images) from Gaussian noise through sequential denoising. However, it is not straightforward to condition the generative process on corrupted or masked images. While various sampling based methods [112, 7, 76, 17, 137] have been proposed for inpainting images with various ways to incorporate information from the observation during the reverse diffusion process, they often fail to produce samples from the true conditional distribution, especially for *large masked regions*. These methods also suffer from slow convergence and highly inconsistent reconstructions. Additionally, many cannot be applied to latent diffusion models [99] which have been demonstrated to generate high-quality images, while offering efficiency in model training.

Different from sampling-based methods, [78] approximates the true complex posterior over image completions using a simple Gaussian variational posterior in the pixel-space. Developed only for pixel-based diffusion models, the Gaussian posterior directly optimizes for a

³Contents of this chapter are mainly based on our work “VIPaint: Image Inpainting with Pre-Trained Diffusion Models via Variational Inference“ currently under review. Authors are Sakshi Agarwal, Gabe Hope, and Erik Sudderth.

single inpainting, inherently incapable of capturing uncertainty in x . Because they optimize the variational distribution in the pixel-space, the optimization suffers from instabilities due to the denoising function’s lack of smoothness at low-noise levels [128]. The authors resort to approximations to the true objective and eventually, does not optimize the true variational bound.

Instead, we propose *VIPaint*, a non-Gaussian *hierarchical* Markov variational approximation of the true diffusion posterior that defines conditional Gaussian distributions over latent variables across handpicked time points in between $[0, T]$. We further derive a simple evidence lower bound to optimize its parameters and observe no optimization instabilities. Fitting such a posterior leads to an interpretable latent space, where samples at higher-noise levels are *globally* consistent while samples at lower-noise levels, t closer to 0, are *locally* consistent with the observed image. Moreover, samples from this latent variational posterior is able to capture uncertainty in the masked regions. Through extensive experimentation, we demonstrate that our VIPaint framework better models posterior uncertainty, enables stable optimization of an unbiased variational bound, and can be applied to both pixel-based and latent DMs. Further, our VIPaint method is easily extended to other inverse problems like deblurring and superresolution.

Chapter 5 concludes the dissertation. We summarize the entire work, highlight key contributions, and point out potential future research directions.

Chapter 2

Computing Partition Function for Discrete Generative Models

A major limiting factor in graphical model inference is the complexity of computing the partition function. Exact message-passing algorithms such as *Bucket Elimination (BE)* [24] require exponential memory to compute the partition function; therefore, approximations are necessary. In this chapter, we build upon a recently introduced methodology called *Deep Bucket Elimination (DBE)* [95] that uses classical Neural Networks to approximate messages generated by *BE* for large buckets. The main feature of our new scheme, renamed *NeuroBE*, is that it customizes the architecture of the neural networks, their learning process and in particular, adapts the loss function to the internal form or distribution of messages. Our experiments demonstrate significant improvements in accuracy and time compared with the earlier *DBE* scheme. This chapter was previously published as [5].

2.1 Introduction

Two of the critical goals of probabilistic modeling are the compact representation of probability distributions and the efficient computation of their marginals and modes. Probabilistic graphical models, such as Markov networks [92, 23, 26] provide a framework to represent distributions compactly as normalized products or factors : $P(X) = \frac{1}{Z} \prod_{\alpha} f_{\alpha}(X_{\alpha})$, where X is a set of variables, each potential f_{α} is a function over a subset X_{α} of the variables (its scope) and $Z = \sum_X \prod_{\alpha} f_{\alpha}(X_{\alpha})$ is the *partition function*. Computing the partition function is exponential in the induced width of the model’s graph even for distributions that admit a compact representation.

The partition function Z is defined by two types of operations: sums and products. It can be evaluated efficiently if $\sum_X \prod_{\alpha} f_{\alpha}(X_{\alpha})$ can be reorganized using the distributive law along a variable ordering [25]. This organization can be described using buckets as data structures, one for each variable in the ordering. When a bucket is processed, its associated variable is removed, creating a bucket output function, also called a *message*, that is passed to a subsequent bucket. The time and space complexity of computing this function is exponential in its number of arguments, called scope or the bucket’s width. Overall, Bucket Elimination (*BE*) [24] is time and memory exponential in the induced-width of the model’s graph along the ordering.

Providing good approximations to *BE* is important not only because it generates an answer to a query, but primarily because it compiles a structure and a set of messages that can be used to answer multiple queries (e.g., the probability of evidence for various evidence variables [23]). Also, the messages can be used as building blocks for generating heuristics for search to further improve performance. We therefore consider and evaluate *NeuroBE* in the context of *approximate BE*, generating approximation to its messages.

Schemes that approximate *BE* include (weighted) mini-bucket (*WMB*) [27, 73] and general-

ized belief propagation schemes [129, 80]. A recently introduced scheme, *Deep Bucket Elimination (DBE)* [95] approximates each bucket function with a neural network (NN). While this approach is inherently time consuming, requiring the independent training of many NNs to solve the partition function of a single problem, it has yielded more accurate approximations on several benchmarks when compared against competing schemes. Both *WMB* and *DBE* are restricted by memory. Yet the memory demanded by *WMB* (notwithstanding recent work [35]) increases exponentially with its i -bound not accommodating refined steps of memory increase. In contrast, NN architectures can grow more gradually and may facilitate a more flexible memory-accuracy balance.

Contributions. We present *NeuroBE*, a re-design of *DBE*, that addresses the shortcomings of its *one size fits all* policy by customizing the NN construction and training sample size to each bucket separately, in proportion to its message size. We also introduce a new loss function that is sensitive to a bucket’s message distribution, also called *local structure*. We also provide an analysis relating the local errors to an upper bound on the global error. In an extensive empirical evaluation we show that *NeuroBE* outperforms *DBE* across all benchmarks using far less resources, such as training samples and NN size, while yielding higher accuracy with less time. We provided the source code to reproduce the results of this work at <https://github.com/dechterlab/NeuroBE>.

This chapter is organized as follows. We first provide a background to *BE*, *WMB* and *DBE*; then we present *NeuroBE*; followed by error analysis; lastly, we demonstrate the efficiency of *NeuroBE* empirically.

Related work. As noted, approximating and bounding *Bucket Elimination* has been carried out extensively over the years for all probabilistic queries. Well known is the *Mini-Bucket Elimination* scheme [27] and its variants, such as *Weighted Mini-Bucket Elimination (WMB)*,

augmented with message-passing cost-shifting [74].

Neural network approximation to BE was introduced in [95]. The idea is closest in spirit to the Neuro-Dynamic Programming scheme as outlined in [10] where the cost-to-go functions (similar to messages) generated by dynamic programming can be approximated by NNs. This technique is also highly related to Deep Reinforcement Learning (DRL) [85] where, in the absence of a model, the value function is approximated by NNs learned from temporal trajectories.

Recently, *Graph Neural Networks (GNNs)* [104] are used to learn *messages* following the message-passing reasoning methods in graphical models [1, 131, 41]. However, [131, 41] is restricted to small instances (i.e., ~ 40 variables) and [1] tackles problems with a known polynomial-time approximation. GNN based methods derive a supervised end-to-end learning algorithm, generalizing across different problem instances. In contrast, we consider a different class of algorithms, where we confine learning to within a problem instance *only*.

2.2 Background

A graphical model can be defined by a 3-tuple $\mathcal{M} = (\mathbf{X}, \mathbf{D}, \mathbf{F})$, where $\mathbf{X} = \{X_i : i \in V, V = \{1, \dots, n\}\}$ is a set of n variables indexed by V , and $\mathbf{D} = \{D_i : i \in V\}$ is the set of finite domains for each X_i (i.e. each X_i can only assume values in D_i , and each D_i is finite). Each function $f_\alpha \in \mathbf{F}$ is defined over a subset of the variables called its scope, X_α , where $\alpha \subseteq V$ are the indices of variables in its scope, and D_α denotes the Cartesian product of their domains so that $f_\alpha : D_\alpha \rightarrow R \geq 0$.

The **primal graph** of a graphical model associates each variable with a node. An edge between node i and node j is created if and only if there is a function containing X_i and X_j in its scope. Figure 2.1a shows a primal graph of a graphical model with variables indexed

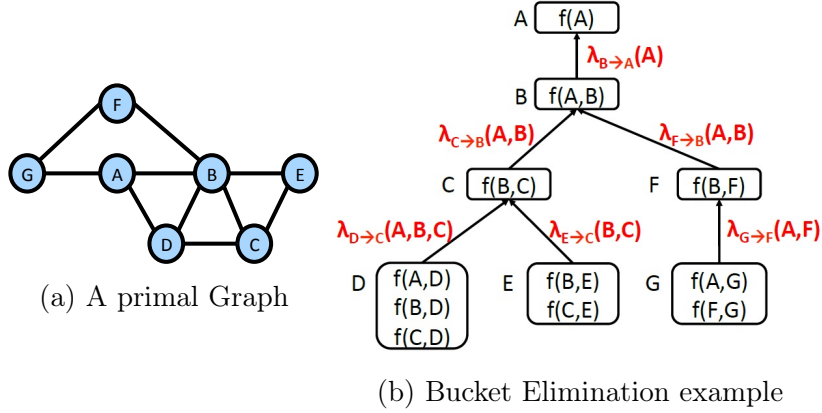


Figure 2.1: (a) A primal graph of a GM with 7 variables. (b) Illustration of *BE* with an ordering A B C E D F G.

from A to G and functions over pairs of variables are connected by an edge. Graphical models can be used to represent a global function, often a probability distribution, defined by $Pr(X) \propto \prod_{\alpha} f_{\alpha}(X_{\alpha})$. An important task is to compute the normalizing constant, also known as the partition function $Z = \sum_X \prod_{\alpha} f_{\alpha}(X_{\alpha})$.

2.2.1 Bucket Elimination

Bucket Elimination (BE) [24] is a universal exact algorithm for probabilistic inference. It is a variable elimination algorithm that can answer a wide-range of queries, including the partition function ranging from constraint satisfaction, to pure combinatorial optimization (e.g., Most Probable Explanation (MPE/MAP)), and weighted counting (Partition Function, Probability of Evidence).

Given a variable ordering d , BE (presented in Algorithm 1, omitting steps 9-12) creates a *bucket tree* where each node is a bucket representing a variable in the ordering d . Figure 2.1b shows a bucket tree for the primal graph in Figure 2.1a along an ordering. Each bucket in this tree contains a set of the model's functions depending on the given order of processing. For example, Bucket G in Figure 2.1b has functions $\{f(A, G), f(F, G)\}$, an exhaustive set

of model's functions with variable G in its scope. There is an arc from a bucket, say B_c , to a parent bucket, B_p , if X_p is the latest variable in bucket B_c 's message scope along the ordering (constants are placed in B_1). In the same example, there is an arc from Bucket G to Bucket F .

BE performs inference along the bucket tree as a 1-iteration message-passing algorithm (bottom-up). It processes each bucket from leaves to the root passing messages from child (c) to parent (p). For a child variable X_c , BE encompasses all the functions in bucket B_c . This includes the original functions in the graphical model as well as the messages received by processing previous variables. It then marginalizes X_c out from the product of functions in B_c generating a new, so called, *bucket function* or message, denoted $\lambda_{c \rightarrow p}$, or λ_c for short:

$$\lambda_c = \sum_{X_c} \prod_{f_\alpha \in B_c} f_\alpha \quad (2.1)$$

The λ_c function is placed in B_p , the bucket of X_p . Once all the variables are processed, BE outputs all the messages and the exact value of Z by taking the product of all the constants present in the bucket of the first variable. We illustrate BE message flow in our example problem in Figure 2.1b.

Complexity. Both the time and space complexity of BE are exponential in the **induced width**, which is the size of the largest number of variables in the scope of any message over all buckets [26]. Clearly, BE becomes impractical if the induced width is large.

2.2.2 Weighted Mini-Bucket

Given a variable ordering d , *Weighted Mini-Bucket* (*WMB*) [27, 74] approximates BE by partitioning each bucket B_c with high width into several disjoint “mini-buckets” B_c^j to ensure that individual B_c^j has low ($\leq i$ -bound) width. The method also assigns a weight p_{cj} to each mini-bucket B_c^j . *WMB* then eliminates the bucket’s variable X in the j^{th} mini-bucket B_c^j using the power sum following Holder’s inequality [74]:

$$\mu_c^j = \left(\sum_X \prod_{f_\alpha \in B_c^j} f_\alpha^{\frac{1}{p_{cj}}} \right)^{p_{cj}},$$

and μ_c^j is passed to a parent bucket B_p . For example, using an i -bound = 2 in Figure 2.1b, *WMB* approximates the exact message $\lambda_{D \rightarrow C}(A, B, C)$, passed from bucket D to bucket C , by three messages corresponding to partitioning bucket B_D into three mini-buckets each with a single function $f(A, D), f(B, D), f(C, D)$. Based on Holder’s inequality [74], the exact message is bounded by the product of the mini-bucket messages when the weights p_{cj} ’s are non-negative and sum to one. Thus, for any i -bound *WMB* generates an upper bound of the partition function.

Generally, time and accuracy in *WMB* increases with the i -bound. Yet, due to memory constraints it can run with a maximum i -bound of about 20 and therefore, the generated bounds can be extremely loose when a problem’s induced-width is high. Interestingly, when it is run, *WMB* terminates quickly, taking a few seconds and up to a minute.

2.2.3 Deep Bucket Elimination

Given a variable ordering d , *Deep Bucket Elimination* (*DBE*) approximates each message generated in the bucket tree whenever the scope (S) of a message is high ($> i$ -bound) using a neural network (NN). Following the previous example of i -bound = 2 in Figure

2.1b, rather than sending the exact message from bucket D to bucket C , DBE sends a NN $\mu_{\theta,D \rightarrow C}(A, B, C)$ parameterized by θ that approximates the exact message $\lambda_{D \rightarrow C}(A, B, C)$, as we elaborate next.

We use $\mu_{c \rightarrow p}^*$ to denote the *local exact message* computed using all functions in bucket c , regardless of the local functions being exact or approximate (as defined by the right side of Eq. (2.1)). However, if we execute exact BE , in which case the bucket contains exact messages only, we denote the output message as $\lambda_{c \rightarrow p}$ and refer to it as the *global exact message*.

Let B be a bucket with width $w > i$ -bound and $\mu^*(S)$ be its local exact message having scope S whose size is the bucket's width, w . DBE constructs a fully-connected feed-forward NN having w nodes in the input layer, followed by L hidden layers each having h hidden nodes with *ReLU* activation function. The output layer contains one node with a real-valued output. Subsequently, DBE generates a training set $\{(s_n, \mu^*(s_n))\}$ of size N , where s_n is the n^{th} configuration of S , sampled uniformly at random and where $\mu^*(s_n)$ is the local exact message value defined in Eq. (2.1). The NN function $\mu_{\theta}(S)$ approximating $\mu^*(S)$ is trained to minimize the mean square error loss :

$$L(\theta) = \frac{1}{N} \sum_{n=1}^N (\mu^*(s_n) - \mu_{\theta}(s_n))^2.$$

Once training is complete, DBE passes the trained NN, μ_{θ} to its parent bucket.

While DBE showed superior quality of solutions compared with WMB , its time performance was quite inferior. In particular, training each bucket message used the same fixed architecture and the same sample size (quite large), needlessly resulting in a high total time. This work is devoted to a redesign of the algorithm, aiming to improve both time and accuracy, as we elaborate in the following section.

2.3 NeuroBE

Algorithm *NeuroBE* advances *DBE*, and is described in this section, focused on the partition function task. However, extensions to other queries is straightforward, and requires altering only the message definition in Eq. (2.1) to fit the corresponding task (e.g., replacing summation by maximization in Eq. (2.1) for the MAP query.)

We rename *DBE* to *NeuroBE* since we use mostly shallow neural networks (up to 2 layers). Algorithm 1 describes *NeuroBE*. The algorithm first creates a bucket tree along a given ordering (line 2). It then processes buckets one by one along the ordering from last to first. If the current processed bucket has width $w \leq i$ -bound, then the message, $\mu_{c \rightarrow p}^*$ is computed exactly (line 7). Otherwise, the bucket’s message is approximated by a neural net (line 9). The message is placed in the appropriate parent bucket in the bucket tree. Finally, line 13 calculates the partition function using the functions in bucket B_1 . Note that if a bucket contains a NN function, then computing μ^* (line 7 or 9) requires evaluating the trained NN (see Algorithm 3, line 4).

The difference between *NeuroBE* and *DBE* is solely in the individual message approximation scheme, *NN-train*. In contrast to *DBE*, *NeuroBE* dynamically customizes the NN architecture and training set size to the bucket’s message complexity, and it modifies the loss function to depend on the message distribution. These modifications are described in the sequel.

NN Architecture selection. Clearly, the NN size should depend on both the approximated function’s complexity and, especially, its dimensionality. Since a bucket message’s scope size is the induced-width, w , we make the number of hidden units, h , a function of w while keeping the number of layers, L , constant. Specifically, we use a simple function $h = b \cdot w$, where $b \geq 1$ to fit the NN’s architecture to the message size. Figure 2.2 is an example NN model architecture with an input layer of size w and 2 hidden layers with di-

Algorithm 1 NeuroBE

Input: Graphical model $\mathcal{M} = (\mathbf{X}, \mathbf{D}, \mathbf{F})$, Ordering $d = X_1, \dots, X_n$

Parameters: i -bound i ; #layers L ; constants b, η ;

Output: the partition function constant and bucket messages

```
1: for  $c$  in  $n \dots 1$  do
2:   (Initialize buckets) put all unplaced functions mentioning  $X_c$  in  $B_c$ .
3: end for
4: for  $c$  in  $n \dots 1$  do
5:   Let  $X_p$  be the parent variable of  $X_c$  in the bucket-tree
6:   if  $\text{width}(B_c) < i$  then
7:     compute  $\mu_{c \rightarrow p}^* \leftarrow \sum_{X_c} \prod_{f_\alpha \in B_c} f_\alpha$ ,
8:   else
9:      $\mu_{\theta, c \rightarrow p} \leftarrow \text{NN-train}(\{f_\alpha | f_\alpha \in B_c\}, L, b, \eta)$ 
10:  end if
11:  Put  $\mu_{c \rightarrow p}^*$  or  $\mu_{\theta, c \rightarrow p}$  in  $B_p$ 
12: end for
13:  $\hat{Z} = \sum_{X_0} \prod_{f_\alpha \in B_1} f_\alpha$ 
14: return  $\hat{Z}$  and all messages generated
```

mension h . Next, we provide a rule to determine sample sizes to train a NN, depending on a notion of its complexity.

NN complexity. The notion of a *Pseudo-dimension* [93, 6] is often used to measure the expressive power of a set of functions that can be learned by any statistical regression algorithm. The work in [9] derived lower bounds to the pseudo-dimension for NNs with ReLU activation function (an architecture used in our work). We use the derived lower bound to estimate the pseudo-dimension (ρ) of a NN (μ_θ), having an architecture of L layers and $b \cdot w$ hidden units, yielding (see Appendix 2.7 for derivation):

$$\rho \propto (L * b * w)^2 \log(b * w). \quad (2.2)$$

Since in our experiments, the pair (L, b) are fixed for a given problem instance, our ρ estimate only varies with w and is used to determine the sample complexity.

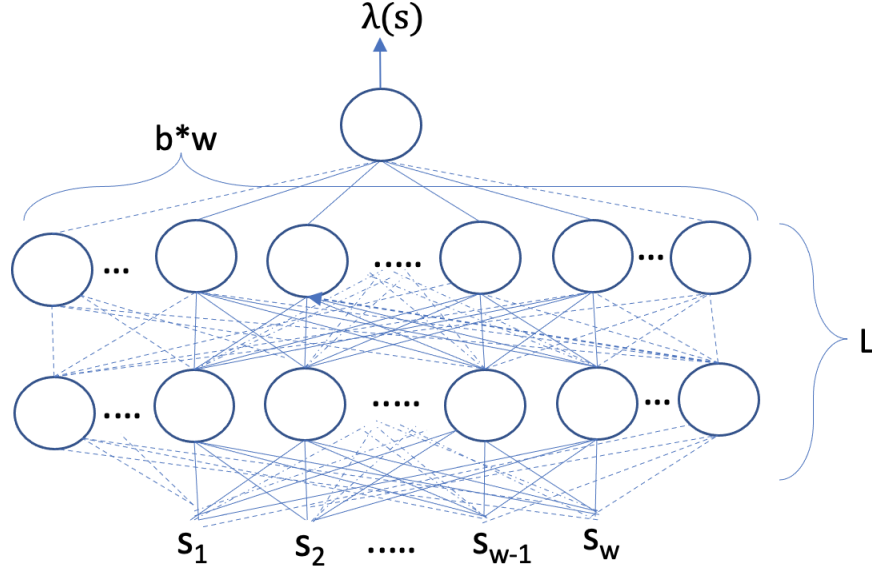


Figure 2.2: For a bucket of width w , we illustrate a NN architecture with $L(= 2)$ layers and $b \cdot w$ hidden-units with $b \geq 1$.

Sample Complexity. As suggested in [119], we choose a sample size for training a NN, μ_θ , proportional to its pseudo-dimension, Eq. (2.2). We therefore select a number of samples N satisfying the expression

$$N = \eta * (L * b * w)^2 \log(b * w), \quad (2.3)$$

where η is a constant allowing us to tweak N linearly. Since the triplet (L, b, η) of a problem is fixed, the number of samples for training, N , is a function of w only.

Sample Generation Let B be a generic bucket where variable X is eliminated; let F be the set of functions from the graphical model (initialized in line 2, Algorithm 1) as well as messages from the previous buckets (line 11, Algorithm 1) residing in B and S be the scope of the output message function μ^* . Then, Algorithm 2 generates a dataset $\mathcal{D}$ containing a given number of samples N . The algorithm iteratively and uniformly at random, samples a configuration $\{S = s\}$ from the domain of S and computes the exact local bucket function

Algorithm 2 generate-samples(X, F, N)

Input: X , a variable to be eliminated, F , a set of functions over scope $S \cup \{X\}$, N , an integer,

Output: $\mathcal{D}$, a set of N samples

```
1: initialize  $\mathcal{D} = \{\}$ 
2: for  $i = 1 \dots N$  do
3:    $s \leftarrow$  sample uniformly from domain( $S$ )
4:    $\mu^*(s) \leftarrow \sum_x \prod_{f \in F} f(s, x)$  ▷ Eq. (2.1)
5:   Add  $(s, \mu^*(s))$  to  $\mathcal{D}$ 
6:   Update  $\mu_{\min}^*, \mu_{\max}^*$ 
7: end for
8: Normalize  $\mathcal{D}$  ▷ Eq. (2.4)
9: return  $\mathcal{D}$ 
```

value for s using Eq 2.1 (lines 3,4). The pair $\langle s, \mu^*(s) \rangle$ is added to the dataset $\mathcal{D}$ (line 5). A normalization step occurs in line 10, where each sample s is shifted and scaled to the range $[-1, 1]$ and $\mu^*(s)$ is shifted and scaled to $[0, 1]$, to accelerate training of the NN [69], by:

$$\mu_{norm}^*(s) = \frac{\mu^*(s) - \mu_{min}^*}{\mu_{max}^* - \mu_{min}^*} \quad (2.4)$$

where μ_{min}^*, μ_{max}^* (line 6) are defined relative to the dataset $\mathcal{D}$ by $\mu_{min}^* = \min_{s \in \mathcal{D}} \mu^*(s)$ and $\mu_{max}^* = \max_{s \in \mathcal{D}} \mu^*(s)$.

Loss Function Algorithm *DBE* sampled each message input configuration uniformly and uses the mean square error loss function for training. However, it seems intuitive that generating the samples by taking into account the message distribution could lead to more effective training of the function. Since sampling directly from the message distribution is hard, we instead weight each sample by an *importance weight* within the loss function, described next.

Definition 1 (I.m.s.e loss). Let μ_θ be the NN for approximating the function μ_{norm}^* . Let $D = \mathcal{D}_{Train}$ be the training set. Then, the I.m.s.e loss function for a given mini-batch, $\mathcal{D}_i \in \mathcal{D}$ of size $\#\mathcal{D}_i$ is defined by:

$$L_{\mathcal{D}_i}(\mu_{norm}^*, \mu_\theta) = \frac{1}{\#\mathcal{D}_i} \sum_{s \in \mathcal{D}_i} (\mu_{norm}^*(s) - \mu_\theta(s))^2 * W(s), \quad (2.5)$$

where

$$W(s) = \frac{\mu^*(s)}{\sum_{s' \in \mathcal{D}_{Train}} \mu^*(s')}. \quad (2.6)$$

Log transformations Usually in our experiments we apply a log transformation to the input functions, for computational reasons. The algorithms presented here remain the same; however the values μ^* , μ_{min}^* and μ_{max}^* in this case refer to the log of the original function values. In cases when we use the log-space computation, the weight function $W(s)$ (Eq. 2.6) is not suitable. We instead use modified importance weights,

$$W^{\log}(s) = \frac{\log \mu^*(s) - \log \mu_{min}^*}{\sum_{s' \in \mathcal{D}_{Train}} (\log \mu^*(s') - \log \mu_{min}^*)} \quad (2.7)$$

Note that the importance weight, $W(s)$ or $W^{\log}(s)$, are computed in the original function space that is not normalized.

MaskedNet For problems with determinism, i.e., a high proportion of zero probability states, a fully connected feed-forward NN was unable to correctly predict deterministic outputs and hence [95] used a MaskedNet. The input configuration is sent to a fully connected layer with a RELU activation function to obtain a feature vector. This feature vector is then

sent to two sister layers: the first layer outputs a binary mask responsible for determining whether the final output is zero, and the second layer is responsible for predicting the target value of the Bucket’s function.

The activation functions of the two final layers are the logistic function and the softplus function, respectively. The outputs from the two sister networks are multiplied together to get the final output of the MaskedNet. The loss for the MaskedNet in *NeuroBE* is thus a sum of the binary cross-entropy loss (from the first output layer) and the proposed I.m.s.e loss (from the second output layer). Thus when a sample configuration s has $\mu^*(s) = 0$, the loss becomes the binary cross-entropy error, since $W(s) = 0$, following Eq. (2.5) and (2.6).

Algorithm 3 NN-train($F, X, L, b, \eta, \#epochs$)

Input: F , a set of functions over scope $S \cup \{X\}$ where X is to be removed, w scope size.

Parameters: L : # layers in NN, $\#epochs$, η , b : constants

Output: μ_θ : NN message approximation, $\hat{\epsilon}$: an estimated bucket error bound

```

1:  $\#h \leftarrow b * w$ 
2:  $N \leftarrow \# \text{ training samples}(w, \eta, L, b)$  ▷ Eq. 2.3
3:  $\mathcal{D} \leftarrow \text{generate-samples}(X, F, N + N/4 + 50k)$ 
4:  $D_{Train}, D_{Val}, D_{Test} \leftarrow \text{Split}(\mathcal{D})$ 
5: Initialize NN parameters  $\theta$ ,  $p=1$ , early-stopping  $\leftarrow$  False
6: while  $p \leq \#epochs$  and  $\neg$  early-stopping do
7:    $D_1, \dots, D_k \leftarrow \text{divide } D_{Train} \text{ to minibatch}$ 
8:   for  $i = 1..k$  do
9:     Let  $D_i = \{(s, \mu_{norm}^*)\}$ 
10:    Compute  $\{\mu_\theta(s) | s \in D_i\}$ 
11:     $loss_{D_i} \leftarrow L_{\mathcal{D}_i}(\mu_{norm}^*, \mu_\theta)$  ▷ Eq. 2.5
12:     $\theta \leftarrow \text{update } \theta \text{ by optimize}(\text{Adam}, loss_{D_i}, \theta)$ 
13:   end for
14:    $loss_{D_{val}} \leftarrow L_{D_{val}}(\mu_{norm}^*, \mu_\theta)$  ▷ For stop condition
15:   early-stopping  $\leftarrow \text{evaluate early-stopping}(loss_{D_{val}})$ 
16:    $p \leftarrow p + 1$ 
17: end while
18: Unnormalize  $\{\mu_\theta(s), \mu^*(s) | s \in D_{Test}\}$  ▷ Inverse of Eq. 2.4
19:  $\hat{\epsilon} \leftarrow \max_{s \in D_{Test}} (\log \mu^*(s) - \log \mu_\theta(s))$ 
20: return  $\mu_\theta, \hat{\epsilon}$ 

```

NN-Train Algorithm 3 describes the procedure *NN-Train*. Its input parameters are L, b, η where L is the number of layers, b is a constant to determine the number of hidden units, $b \cdot w$ (line 1), and η is another constant to determine the training sample size N (line 2, Eq. (2.3)). A major step occurs next where the algorithm generates a dataset $\mathcal{D}$ and splits it into the training set $\mathcal{D}_{Train}$ of size N , validation set $\mathcal{D}_{Val}$ of size $N/4$ and testing set $\mathcal{D}_{Test}$ of fixed size (50k) (lines 3-4; see also Algorithm 2).

Lines 8-12 then describe the batch training for updating the NN parameters θ using the l.m.s.e loss function (line 11, Eq. (2.5)), and the Adam optimizer [60] (line 12) with a learning rate of 0.001 and a batch-size of 256 across all benchmarks. At the end of each epoch, the current model is evaluated on a holdout validation set (line 14). We evaluate the early-stopping criteria (line 15), which is assigned *True* when either the maximum limit $\#epochs$ is reached or the validation error increases for two consecutive epochs. Once training is complete, we compute the maximum log relative error between the target and NN approximated messages over a test set (lines 18-19). In the next section, we use this maximum error to analyse the propagation of error in *NeuroBE*. The *NN-train* procedure then returns the approximated message μ_θ , along with its estimated error.

Complexity. The time and space complexity for learning a single message in *NeuroBE* is a function of the NN and sample size. In contrast to *DBE*, here the NN and sample sizes vary with the bucket’s width.

2.4 Error Analysis

We now analyse the relationship between the local errors contributed by each approximated message and the global partition function error, focusing on a simple case where the bucket tree is a chain.

Problem Description					ref logZ	WMB	#NB	DBE (#h=100, N=320K)				NeuroBE (#h=3w, N _{min} = 49K)											
i-bound=20								error	statistics on error (5 runs)				Statistics on resources			Loss : mean square error				Loss : Imp. mean square error			
									avg error	min error	stdcv	time (h)				statistics on error (5 runs)				statistics on error (5 runs)			
ld	name	k	#v	w								h _{max}	N _{avg}	N _{max}	avg error	min error	std	time (h)	avg error	min error	std	time (h)	
1	pedigree13	3	888	33	-31.18	6.4696	127	5.32	2.62	3.06	11	96	218K	706K	9.04	7.86	0.8	7.3	1.11	0.76	0.25	7.5	
2	pedigree41	5	885	32	-76.04	4.1497	92	4.27	3.25	0.73	9.9	93	190K	658K	10.2	8.4	2.16	5.6	0.47	0.153	0.21	6.2	
3	pedigree51	5	871	35	-77.27	9.7624	120	23.92	9.23	12.7	13	102	259K	809K	11.73	10.37	1.26	10.4	3.51	1.96	0.89	10.4	
4	pedigree34	5	922	33	-64.23	7.0762	106	5.91	1.57	5.98	11	96	211K	706K	6.14	4.56	4.14	6.5	0.65	0.23	0.29	6.96	
5	pedigree7	4	867	34	-64.82	6.0012	108	11.26	5.18	7.8	11	99	350K	900K	#	#	#	10.7	1.75	1.21	0.7	10.7	
7	pedigree19	5	693	28	-59.020	2.5809	43	6.054	5.41	0.92	9.4	71	149K	482K	9.14	8.7	0.6	4.6	2.61	1.91	0.6	5.2	

(a) pedigree

Problem Description					ref logZ	WMB	#NB	DBE (#h=100, N=320K)				NeuroBE (#h=w, Nmin = 19K)											
												Statistics on resources			Loss : mean square error				Loss : Imp. mean square error				
i-bound=20								error	statistics on error (5 runs)				statistics on error (5 runs)			statistics on error (5 runs)							
ld	name	k	#v	w	avg error	min error	stdv		time (h)	h _{max}	N _{avg}	N _{max}	avg error	min error	std	time (h)	avg error	min error	std	time (h)			
1	grid4040f10	2	1600	55	5490	215.45	308	97.1	11.8	65.2	12	55	120K	364K	14.57	0.15	8.94	7.69	9.71	2.1	0.91	6.4	
2	grid4040f5	2	1600	55	2800	84.92	308	39.9	6.28	35	12	55	120K	364K	8.86	2.05	8.21	7.66	3.7	0.2	3.06	6.3	
3	grid4040f2	2	1600	55	1220	25.24	308	7.34	1.2	5.4	12	55	120K	364K	3.15	1.48	1.66	7.48	2.28	1.2	0.91	6.1	
4	grid4040f15	2	1600	55	8200	338.2	308	83.46	41.8	34.2	13	55	75K	228K	24.75	6.62	18.1	5.7	17.87	7.45	9.16	5.7	
5	grid4040f10w	2	1600	114	5637	297.7	376	100.5	6.4	82.2	21	114	150K	670K	67.76	32.7	26	11	54.18	25.01	21.1	11.1	
6	grid4040f5w	2	1600	114	2819	136.99	376	78.2	72.6	5.6	21	114	150K	670K	5.37	1.65	4.77	11.6	9.62	4.62	5.27	12.1	
7	grid4040f2w	2	1600	114	1231	32	376	15.12	0.92	20.5	18	114	150K	670K	10.15	8.81	1.44	11.8	5.56	2.92	2.11	10	

(b) Grid-hard

Problem Description					ref logZ	WMB	#NB	DBE (#h=100, N=320K)				NeuroBE (#h=w, Nmin = 20K)											
i-bound=10								error	statistics on error (5 runs)				Statistics on resources			Loss : mean square error				Loss: Imp. mean square error			
ld	name	k	#v	w					avg error	min error	stdv	time (h)	h _{max}	N _{avg}	N _{max}	avg error	min error	std	time (h)	avg error	min error	std	time (h)
1	grid1010f10w	2	100	21	333.32	32	31	4.45	0.89	3.71	1.5	21	51K	87K	1.58	0.28	0.75	0.35	1.83	0.98	0.76	0.48	
2	grid1010f10	2	100	13	303.09	1.58	8	0.7	0.05	0.54	0.3	13	23K	30K	1.28	0.59	0.41	0.06	1.18	0.86	0.32	0.05	
3	grid2020f2	2	400	27	291.73	11.24	114	1.98	0.36	1.28	5.3	27	68K	177K	0.4	0.013	0.36	2.24	0.13	0.015	0.14	2.5	
4	grid2020f10	2	400	27	1312	80.86	114	10.04	1.05	9.3	5.5	27	68K	177K	2.51	1.62	1.31	2.22	2.4	0.37	1.98	2.27	
5	grid2020f5	2	400	27	665.12	39.44	114	5.75	0.24	3.1	5.5	27	68K	177K	0.84	0.37	0.53	1.95	0.8	0.081	0.78	2.3	
6	grid2020f15	2	400	27	1963	122.91	114	17.8	3.08	9.8	3	27	68K	177K	2.39	0.41	2.16	2.2	2.68	1.08	2.54	2.2	

(c) Grid-easy

Problem Description					ref logZ	WMB	#NB	DBE (#h=100, N=320K)				NeuroBE (#h=3w, Nmin= 147K)												
						error		statistics on error (5 runs)				Statistics on resources			Loss : mean square error				Loss: Imp. mean square error					
i-bound=20								avg error	min error	std	time (h)				h _{max}	N _{avg}	N _{max}	avg error	min error	std	time (h)	avg error	min error	std
Id	name	k	#v	w																				
1	rbm20	2	40	21	58.53	0.0007	20	0.22	0.03	0.17	1.1	60	147K	147K	0.37	0.08	0.29	0.45	0.37	0.06	0.25	1.73		
2	rbm21	2	42	22	63.15	6.39	22	0.48	0.27	0.19	1.3	63	163K	164K	0.9	0.59	0.2	0.46	0.57	0.15	0.26	1.75		
3	rbm22	2	40	21	66.55	8.65	24	0.47	0.14	0.35	1	66	180K	182K	0.59	0.02	0.38	0.57	0.75	0.48	0.36	1.76		
4	rbm-ferro20	2	44	23	151.16	0.005	20	1.33	0.29	1.21	1	60	147K	147K	1.11	0.38	0.75	0.44	0.75	0.37	0.32	1.13		
5	rbm-ferro21	2	42	22	152.62	1.98	22	3.43	0.83	1.89	1.2	63	163K	164K	2.3	1.27	1.08	0.48	1.82	1.06	0.87	2.5		
6	rbm-ferro22	2	44	23	166.11	0.517	24	6.52	3.86	1.5	1.3	66	180K	182K	5.32	2.515	2.69	0.59	4.17	3.8	0.32	1.29		
i-bound=10												NeuroBE (#h=3w, Nmin= 82K)												
1	rbm20	2	40	21	58.53	7.85	30	0.49	0.11	0.35	1.6	100	82K	87K	1.57	0.89	1.04	0.3	1.43	0.83	0.82	0.8		
2	rbm21	2	42	22	63.15	15.73	32	0.73	0.14	0.47	1.7	105	90K	109K	1.62	0.68	0.98	0.4	0.89	0.51	0.42	1.02		
3	rbm22	2	40	21	66.55	27.46	34	0.65	0.19	0.41	1.8	110	99.5K	121K	0.89	0.03	0.6	0.46	0.94	0.83	0.61	1.3		

(d) DBN

Figure 2.3: Results on performance of *NeuroBE* against *DBE* and *WMB*. k : domain size, $\#v$: variables, w : induced width, $\#NB$: number of buckets that are trained with NNs, $\#h$: number of hidden units per layer (reported maximum $\#h$ for *NeuroBE*), N : number of training samples (reported minimum, average and maximum $\#N$ for *NeuroBE*), *error*: L1 error for referenced and estimated log Z (reported minimum, average, and standard deviation over 5 runs for *DBE* and *NeuroBE*).

Definition 2 (local and global bucket errors). Given a bucket B , let λ be the (global) exact message generated in B , μ^* be the (local) exact message in B at the time of message computation, and $\mu = \text{NN-train}(\mu^*)$ be its NN approximation. Then, we define the local and global log relative errors as:

$$E = \log \mu^* - \log \mu,$$

and,

$$G = \log \lambda - \log \mu.$$

We use log relative error since it simplifies the analysis. We now show the following relationship:

Theorem 1. *Assume a bucket-chain along an ordering d , and let B_c be a bucket along the chain at position c having scope S of its bucket message. Let $E_c(s) = \log \mu_c^*(s) - \log \mu_c(s)$ and let $\epsilon_c = \max_{s \in D(S)} |E_c(s)|$. Then,*

$$G_c = \log \lambda_c - \log \mu_c \leq \sum_{k=0}^{n-c} \epsilon_{c+k}$$

In particular, since $\lambda_1 = Z$ and $\mu_1 = \hat{Z}$,

$$G_1 = \log Z - \log \hat{Z} \leq \sum_{k=0}^{n-1} \epsilon_{1+k} \tag{2.8}$$

2.5 Empirical evaluation

2.5.1 Experimental Setup

We conducted experiments comparing *NeuroBE* against *WMB* [27, 73] and *DBE* [95] over several benchmarks. We also compare the impact of the two loss functions, m.s.e and l.m.s.e, on the performance of *NeuroBE*. Finally, we illustrate how increasing sample and NN complexity impact performance.

i-bounds. All three algorithms, *WMB*, *DBE* and *NeuroBE*, use the i-bound parameter (i). As noted, in *WMB* higher i-bounds lead to more accurate bounds with more time and memory, up to their memory limit. Algorithms *DBE* and *NeuroBE* are also observed to improve accuracy and time with increasing i-bounds because of the reduced number of trained buckets $\#NB(i)$. Hence, for a fair comparison we use an i -bound of 10 for some (easy) benchmarks, while primarily using the highest feasible i -bound of 20 dictated by *WMB*'s memory bound for other (hard) benchmarks.

Benchmarks Following the example of *DBE*, we evaluated *NeuroBE* on instances selected from three well-known benchmarks from the UAI repository used in [55]: grids (vision domain), pedigree (genetic linkage analysis) and DBNs. We targeted diverse benchmarks (in structure and level of determinism) and aimed for different levels of hardness. Thus, in the grids benchmark, we distinguish those problems that can be solved exactly, which we call “grid-easy”, from those that cannot be solved, called “grid-hard”. We also distinguish benchmarks that possess *determinism*, namely have a high proportion of zero probabilities, since it can impact training. We randomly selected 13 instances from Grids, with easy ones (400 variables, width 20-30) and hard ones (1600 variables, width 55 or 114), 6 from pedigrees (≈ 800 variables, width ≈ 34), which possess high level of determinism and 6 from DBNs

(≈ 40 variables, width ≈ 22), totalling 25 instances. As described in section 3, we apply log transformations to Grids and DBNs since they have large message function values.

NN architectures and sample sizes. Through a process of trial and error on a selected instance from each benchmark, we selected the parameters of the architectures and sample sizes as follows. We selected $L = 1, h = 3w$, and $N_{avg} \in [149k, 350k]$ for pedigrees; $L = 2, h = 3w$, and $N_{avg} \in [80k, 180k]$ for DBN; $L = 2, h = w$, and $N_{avg} \in [23k, 68k]$, for grid-easy; and $L = 1, h = w$, and $N_{avg} \in [75k, 150k]$ for grid-hard.

Performance measures We evaluate the performance of *NeuroBE* using: $error = |\log_e Z - \log_e \hat{Z}|$ where $\hat{Z}$ estimates Z . When the exact Z is not available (i.e., for hard Grid benchmark), we use Z^* as a surrogate to Z , which is obtained using an advanced sampling scheme for a duration of $100 * 1hr$ [55].

2.5.2 Results

Figure 2.3 compares *NeuroBE* against *WMB* and *DBE* over the four benchmarks. The first few columns show the problem statistics for instances in the respective benchmarks (pedigree, grid-hard; grid-easy and DBN). We then show results on *WMB*'s error, followed by *DBE*'s and *NeuroBE*'s performance information. We omit *WMB*'s time performance in Figure 2.3 since its execution takes only a minute. For *DBE*, we report the number of buckets trained by NNs, ($\#NB$), followed by the average error, minimum error, standard deviation, and average time (in hours) over five runs, which is preferable due to stochasticity. For *NeuroBE* with both m.s.e and l.m.s.e loss functions, we also report statistics about NN architecture and sample size that varies within problem instances: the average and maximum number of training samples, N_{avg} and N_{max} , and maximum number of hidden units, h_{max} .

i-bound=20			h=3*w				h=5*w			
Id	#v	w	#N _{avg} (10 ³)	standard deviation	error	t(h)	#N _{avg} (10 ³)	standard deviation	error	t(h)
1	888	33	218	0.25	1.11	7.5	437	0.17	0.34	14.3
3	871	35	259	0.89	3.51	10.4	518	1.57	1.42	18.3
7	693	28	149	0.6	2.61	5	298	0.51	0.68	8.2

(a) pedigree

i-bound=20			h=w				h=w			
Id	#v	w	#N _{avg} (10 ³)	standard deviation	error	t(h)	#N _{avg} (10 ³)	standard deviation	error	t(h)
5	1600	114	120	10.82	55.11	12	262	16.3	17.76	17
7	1600	114	120	3.6	5.5	10	262	1.39	2.34	14.5

(b) Grid-hard

i-bound=10			h=w				h=3*w			
Id	#v	w	#N _{avg} (10 ³)	standard deviation	error	t(h)	#N _{avg} (10 ³)	standard deviation	error	t(h)
1	100	21	51	0.76	1.83	0.48	202	0.08	1.2	1.13
2	100	13	23	0.32	1.18	0.05	38	0.15	0.12	0.09

(c) Grid-easy

i-bound=20			h=3*w				h=5*w			
Id	#v	w	#N _{avg} (10 ³)	standard deviation	error	t(h)	#N _{avg} (10 ³)	standard deviation	error	t(h)
1	40	21	147	0.25	0.37	1.73	245	0.07	0.2	3.2
2	42	22	163	0.26	0.57	1.75	272	0.24	0.77	3.7
3	40	21	163	0.36	0.75	1.76	300	0.22	0.51	4.5
4	44	23	147	0.32	0.75	1.13	245	0.61	0.67	2.17
5	42	22	164	0.87	1.82	1.3	272	0.23	1.06	2.6
6	44	23	182	0.32	4.17	1.29	300	0.86	3.43	2.9

(d) DBN

Figure 2.4: Performance of NeuroBE when increasing # sample &/or NN complexity. N_{avg} : average samples, $t(h)$: average time, $Error$: global error (reported average and standard deviation over 5 runs)

Pedigrees We observe immediately that, overall, *NeuroBE* with the I.m.s.e loss function is clearly superior to *DBE* and *NeuroBE* with m.s.e loss. In particular, it is ≥ 5 times more accurate than *DBE* for almost all instances and takes less time, since it uses far less training samples. *NeuroBE* with the I.m.s.e loss function also outperforms *WMB* on most instances. *NeuroBE* with the m.s.e loss function is less accurate than both *WMB* and *DBE* for most instances. It also fails to approximate the partition function for instance 5 (errors and standard deviation denoted by $\#$ in Figure 2.3). Here, *DBE* has similar or even worse accuracy than *WMB*.

Grid-hard. The results for the grid-hard benchmark is shown in Fig. 2.4b. We used the highest possible i -bound of 20, and we observe that *NeuroBE* can achieve a far lower average error and standard deviation and takes far less time than *WMB* and *DBE*, particularly with the I.m.s.e loss. In most cases, we see a reduction in time by a factor of two. *DBE* outperforms *WMB* across all problem instances.

Grid-easy. The results for the grid-easy benchmark is shown in Figure 2.4c. We used a lower i -bound of 10 to facilitate the training of a relatively large number of buckets. As expected, when an instance has a low induced-width and only a small number of buckets are approximated (e.g., ID 2), *DBE* obtains high accuracy. As the induced-width increases and more buckets are trained, *NeuroBE* has far higher accuracy compared with both *WMB* and *DBE*. Both the loss functions in *NeuroBE* show similar performance.

DBN We report results for the DBN benchmark for two i -bounds. Overall, the results are mixed. For i -bound = 20, *NeuroBE* achieves a higher accuracy than *DBE* for half of the instances with far less $\#$ training samples (but with more training time). It is superior to *WMB* on instances 2, 3, and 5. When comparing the two loss functions in *NeuroBE*, I.m.s.e loss has better (or similar) performance for most instances. However, *WMB* performs better on instance 1, 4, and 6, as the induced-width is closer to the i -bound. For i -bound = 10, *DBE* and *NeuroBE* show better accuracy than *WMB* for those three instances. *DBE* has

better accuracy when compared with *NeuroBE*, using more #training samples (and hence, more time). *NeuroBE* with I.m.s.e loss is better performing compared with m.s.e loss on most instances. Overall, *NeuroBE* when trained with I.m.s.e loss takes more time than with m.s.e loss for the same #training samples.

In summary, *NeuroBE* using I.m.s.e compared against *DBE* is about 50% faster while also far more accurate on pedigrees, twice as fast and 5 to 10 fold more accurate on hard grids. It is also faster and more accurate on easy grids and has a mixed but still comparable performance on DBNs.

The impact of loss functions. We observe that *NeuroBE* with the I.m.s.e loss shows better performance (lower average error and standard deviation) than *NeuroBE* with the m.s.e loss for the pedigree instances and the majority of DBN, grid-easy and grid-hard instances. An F-test with a significance level of 0.05 on the two groups of partition function estimates (each consisting of five approximations) showed that the means are significantly different for pedigrees, in Figure 2.3(a). For the grids and DBN, there was no statistical difference between the two means. However, by inspection, we see a reduction in the standard deviation for almost all instances.

Impact of architecture size. Figure 2.4 shows the impact of architecture size on time and accuracy for a few problem instances. We show results for two different NN architectures and their associated sample sizes. As expected, we see that increasing the sample and NN sizes increases both time and accuracy for pedigrees. For grid-hard instances, we just increased the sample sizes and kept the same architecture having $h = w$. We observe that the average error is reduced, as expected. Instances from grid-easy and DBN (except ID 2) show a similar improvement in performance with a larger NN and training sample size. This trend illustrates that increasing the size of the NN (matched by a suitable increase in sample size), improves the accuracy of *NeuroBE*, at the cost of more time and memory. A key question for future work is how to develop a policy that can facilitate gradual control of architecture

and sample size increase to improve performance in an anytime way.

2.6 Conclusion & Future Work

In this work, we advance the earlier theme of using Neural Networks to approximate the class of bucket-elimination algorithms that is at the heart to probabilistic reasoning. *NeuroBE* can be viewed as a realization of Neuro-Dynamic Programming schemes [10], in the context of graphical models. That being said, it requires the training of numerous NNs per problem instance, and thus, the central aim of *NeuroBE*'s design (customizing NN architectures, training samples, and the loss function to the message) is to enhance *efficiency* and *scalability* of such schemes. We presented *NeuroBE* and illustrated on challenging instances over three benchmarks that it can be far more accurate and requires less time compared with *Deep Bucket Elimination (DBE)*. It is also superior to *weighted mini-bucket (WMB)* even when provided with the highest memory resources feasible.

Future Work. One could further explore how to improve *NeuroBE*'s efficiency by customizing additional features of a NN and its training per bucket (e.g., varying the number of layers). Further research could move from training buckets separately per single variable to training clusters of buckets within a tree-decomposition, thus training a single function per union of buckets [26], potentially reducing the number of trained functions at the cost of more time for sample generation. Finally, parameter sharing could be adapted by training multiple bucket functions simultaneously in a single problem instance and across a benchmark of instances.

2.7 Appendix

2.7.1 Estimating the pseudo-dimension of a NN:

In our work, we use NN architectures with ReLU activation functions. To construct a NN with L layers and a variable h , #hidden-units per layer to model a specific local bucket message μ^* , we pick the rule $h = b * w$ where w is the bucket's width and b is a constant. By doing this, the #parameters in the NN is :

$$|\theta| = (L - 1) * b^2 * w^2 + b * w^2 + (L + 1) * b * w + 1 \quad (2.9)$$

We make use of the lower bound of pseudo-dimension for NNs with ReLU activation functions from the work in [9] to get:

$$\rho = |\theta| * L \log(|\theta|/L) \quad (2.10)$$

By substituting Eq. 1 in Eq. 2 and ignoring all linear terms in w we get that ρ can be dominated by:

$$\rightarrow \rho \propto (L * b * w)^2 \log[(b * w)]$$

2.7.2 Estimating error in partition function:

Theorem. *Let B_c be a bucket in a bucket chain along an ordering d ; let B_c contain the original functions as ϕ_c and μ_{c+1} as the message passed to it from the previous bucket; let λ_c be the (global) exact message generated in B_c , μ_c^* be the local exact message in B_c and $\mu_c = APP(\mu_c^*)$ its approximation (e.g., by a trained neural network). Let $E_c = \log \mu_c^* - \log \mu_c$*

and $\epsilon_c = \max_{B_c} |E_c|$. Then,

$$\log \lambda_c - \log \mu_c \leq \sum_{k=2}^{n-c} \epsilon_{c+k}$$

In particular, since $\lambda_1 = Z$, the partition function and $\mu_1 = \hat{Z}$, the estimate to the partition function,

$$\log Z - \log \hat{Z} \leq \sum_{k=1}^{n-1} \epsilon_{1+k} \quad (2.11)$$

Proof. We will next derive the recursion, starting at the first processed bucket B_n and going down in order. Remember throughout that $\log \mu_{n-i}^* = \log \sum_{X_{n-i}} (e^{\log \phi_{n-i} + \log \mu_{n-i+1}})$

For B_n $\lambda_n = \mu_n^*$, therefore

$$\log \lambda_n - \log \mu_n = \log \mu_n^* - \log \mu_n = E_n$$

For B_{n-1} , by definition

$$\log \lambda_{n-1} - \log \mu_{n-1} = \log \sum_{X_{n-1}} e^{\log \phi_{n-1} + \log \lambda_n} - \log \mu_{n-1}$$

Substituting $\log \lambda_n$ from B_n

$$\begin{aligned} &= \log \sum_{X_{n-1}} e^{[(\log \phi_{n-1} + \log \mu_n) + E_n]} - \log \mu_{n-1} \\ &= \log \left[\sum_{X_{n-1}} e^{(\log \phi_{n-1} + \log \mu_n)} e^{E_n} \right] - \log \mu_{n-1} \end{aligned}$$

If $\max_{scope(\mu_n^*)} |E_n| = \epsilon_n$, then,

$$\begin{aligned} &\leq \log[e^{\epsilon_n} \sum_{X_{n-1}} e^{(\log \phi_{n-1} + \log \mu_n)}] - \log \mu_{n-1} \\ &\leq \epsilon_n + \log \sum_{X_{n-1}} e^{(\log \phi_{n-1} + \log \mu_n)} - \log \mu_{n-1} \end{aligned}$$

Since $\log \sum_{X_{n-1}} e^{\log \phi_{n-1} + \log \mu_n} = \log \mu_{n-1}^*$ we get

$$\log \lambda_{n-1} - \log \mu_{n-1} \leq \epsilon_n + \log \mu_{n-1}^* - \log \mu_{n-1} \quad (2.12)$$

or equivalently,

$$\log \lambda_{n-1} - \log \mu_{n-1} \leq \epsilon_n + E_{n-1} \quad (2.13)$$

Moving to B_{n-2} , by definition:

$$\log \lambda_{n-2} - \log \mu_{n-2} = \log \sum_{X_{n-2}} e^{\log \phi_{n-2} + \log \lambda_{n-1}} - \log \mu_{n-2} \quad (2.14)$$

Substituting $\log \lambda_{n-1}$ from Eq. (2.12) we get

$$\begin{aligned} &\log \lambda_{n-2} - \log \mu_{n-2} \\ &\leq \log \sum_{X_{n-2}} e^{\log \phi_{n-2} + [\log \mu_{n-1} + \epsilon_n + E_{n-1}]} - \log \mu_{n-2} \\ &\leq \log \sum_{X_{n-2}} e^{\log \phi_{n-2} + \log \mu_{n-1}} e^{\epsilon_n + E_{n-1}} - \log \mu_{n-2} \end{aligned}$$

Taking $\max_{\text{scope}(\mu_{n-1}^*)} E_{n-1} = \epsilon_{n-1}$,

$$\begin{aligned}
&\leq \log e^{\epsilon_n + \epsilon_{n-1}} \sum_{X_{n-2}} e^{\log \phi_{n-2} + \mu_{n-1}} - \log \mu_{n-2} \\
&\leq \epsilon_n + \epsilon_{n-1} + \log \sum_{X_{n-2}} e^{\log \phi_{n-2} + \mu_{n-1}} - \log \mu_{n-2} \\
&\leq \epsilon_n + \epsilon_{n-1} + \log \mu_{n-2}^* - \log \mu_{n-2}
\end{aligned}$$

yielding,

$$\log \lambda_{n-2} - \log \mu_{n-2} \leq E_{n-2} + \epsilon_{n-1} + \epsilon_n \quad (2.15)$$

Moving to bucket B_{n-3} , by definition

$$\log \lambda_{n-3} - \log \mu_{n-3} = \log \sum_{X_{n-3}} e^{\log \phi_{n-3} + \log \lambda_{n-2}} - \log \mu_{n-3}$$

Substituting for λ_{n-2} from Eq. (2.15) we get with some algebra

$$\begin{aligned}
&\log \lambda_{n-3} - \log \mu_{n-3} \\
&\leq \log \sum_{X_{n-3}} e^{\log \phi_{n-3} + [\log \mu_{n-2} + E_{n-2} + \epsilon_{n-1} + \epsilon_n]} - \log \mu_{n-3}
\end{aligned}$$

yielding

$$\log \lambda_{n-3} - \log \mu_{n-3} \leq E_{n-3} + \epsilon_{n-2} + \epsilon_{n-1} + \epsilon_n$$

and so on. Clearly the emerging expression for bucket B_c is

$$\log \lambda_c - \log \mu_c \leq E_c + \epsilon_{c+1} + \epsilon_{c+2} + \dots \quad (2.16)$$

or,

$$\log \lambda_c - \log \mu_c \leq E_c + \sum_{k=0}^{n-c-1} \epsilon_{c+1+k} \quad (2.17)$$

The general transition from $n-i$ to $n-i-1$ can be easily followed to complete the inductive proof.

Assuming that we control the derivation of μ_c for each B_c to ensure that $E_c = \log \mu_c^* - \log \mu_c \leq \epsilon_c$ and substituting in the expression we get from Eq. (2.17) that

$$\log \lambda_c - \log \mu_c \leq \epsilon_c + \sum_{k=0}^{n-c-1} \epsilon_{c+1+k} \leq (n-c+1) * \epsilon \quad (2.18)$$

□

Chapter 3

Variational Inference for (Hierarchical) VAEs

In this chapter, we propose a broadly applicable variational inference algorithm for both simple and hierarchical VAEs with continuous latent variables. Applied to the case of image inpainting, our query-adaptive variational inference (QAVI) approach converges faster and more accurately than sampling-based methods [97, 81]. At the same time, when compared with amortized inference methods [20, 113, 40], QAVI’s performance does not depend on any particular masking distribution. Our approach further leads to tighter likelihood bounds and greater robustness to local optima. By analytically marginalizing arbitrary sets of missing features, and optimizing expressive posteriors including mixtures and density flows in the latent space, our non-amortized QAVI algorithms achieve excellent performance while avoiding expensive model retraining. On standard image and tabular datasets, our approach substantially outperforms prior methods in the plausibility and diversity of imputations. We also show that QAVI effectively generalizes to recent hierarchical VAE models for high-dimensional images. This chapter was originally published in [4].

3.1 Introduction

Structured probabilistic models, like graphical models [66], are the basis for many machine learning applications. Formalizing the data generation process enables incorporation of domain knowledge and allows unsupervised learning from unlabeled data. Probabilistic models also enable diverse inference queries by users, for instance *imputation* queries (or, inpainting in the image domain) where arbitrary subsets of features are observed and the values of missing features are predicted. A number of inference algorithms [92, 66] have been developed for models with discrete or Gaussian latent variables, which efficiently compute marginals of query variables given heterogeneous observations. While exact inference is intractable for many complex models, optimization-based variational methods [120] often provide effective approximations.

Variational bounds were classically optimized via *coordinate ascent variational inference* (CAVI, [51]) algorithms that iteratively update posterior approximations for individual variables. CAVI updates are effective for many parametric models composed from conjugate priors, and can have efficient message-passing implementations [37, 124]. But, CAVI updates are based on integrals that often lack closed forms, requiring Monte Carlo approximations [90, 68] of uncertain quality.

Stochastic subsampling of data helps scale variational learning to big datasets [?], but iterative CAVI updates may still be slow for complex models. *Amortized* variational inference [84] seeks to boost training efficiency by determining variational posteriors via an inference (or recognition) network, which is shared (or amortized) across many similar inference tasks. *Variational autoencoders* (VAEs, [63, 97]) are deep generative models that utilize amortized inference to jointly train a generative “decoder” and inference “encoder”. Sophisticated generalizations to the encoder and decoder networks [64, 107, 117, 16] have produced hierarchical VAEs that realistically model complex image data via dozens of stochastic layers.

While amortized inference has enabled the learning of impressive deep generative models, it sacrifices the flexibility of CAVI to handle arbitrary inference queries. Because VAEs are typically trained from fully-observed data, the encoder assumes *complete* and *uncorrupted* observation of every data dimension (e.g., pixel). Simple heuristics (see Sec. 4.2) are sometimes used for learning VAEs with missing data [82, 86, 20], such as filling missing features with zeros. However, we show that in addition to requiring expensive encoder retraining for peak performance, these approaches are inaccurate unless the test inference queries are simple or known in advance (during model training). Amortized inference also produces sub-optimal variational bounds, and this “amortization gap” may be significant [22, 67]. While quick-and-approximate inference may be sufficient to provide a noisy gradient signal in the midst of a long training process, it is problematic when applied to test queries, especially in domains like medicine where accurate uncertainty quantification is crucial.

To address these challenges, we propose *query-adaptive variational inference* (QAVI) methods that approximate the posterior of missing data with arbitrary patterns, given only a trained generative decoder and sparse observations. Our QAVI approach has the same inferential flexibility as classic CAVI algorithms, and critically does not require a database with many examples of the missing-data pattern of interest. But unlike classic CAVI algorithms, QAVI is applicable to any (differentiable) model with continuous latent variables, including deep generative models like hierarchical VAEs. While some prior work has improved VAE training by reducing amortization gaps [58, 79], our application of non-amortized inference to missing-data queries for deep generative models is novel.

We begin in Sec. 4.2 by reviewing prior work on handling missing data with (hierarchical) VAEs. Sec. 3.3.1 then develops QAVI algorithms that optimize variational bounds for the missing feature values, rather than filling them via greedy heuristics. In Sec. 3.3.2, we develop an alternative QAVI algorithm that directly optimizes the posterior of the latent data encoding, without amortization. Doing this allows exact marginalization of missing feature

values, and enables flexible posterior approximations including mixture models [49, 36] and normalizing flows [96]. Results in Sec. 3.4 then show substantial qualitative and quantitative improvements in capturing multimodal posterior uncertainty for VAE models of tabular data, as well as state-of-the-art hierarchical VAE models of images [16].

3.2 Background and Related Work

3.2.1 The Variational Autoencoder (VAE)

VAEs model the distribution of typically high-dimensional observed data x using continuous, lower-dimensional latent variables z , via the following generative model:

$$z \sim p(z), \quad x \sim p_\theta(x | z). \quad (3.1)$$

For standard VAEs, the latent code $z \in \mathbb{R}^d$ has a factorized Gaussian prior $p(z)$. Given z , data is generated via a *decoder* (deep) neural network with weights θ . The decoder maps z to a likelihood $p_\theta(x|z)$ such as a factorized Gaussian.

The VAE log-likelihood $\log p_\theta(x) = \log \int p_\theta(x|z)p(z) dz$ is intractable. Learning thus typically employs *amortized* VI, where an *encoder* with parameters ϕ approximates the posterior over latent codes $q_\phi(z|x) \approx p_\theta(z|x)$. We jointly learn θ, ϕ by maximizing the *evidence lower-bound* (ELBO):

$$\mathcal{L}(\theta, \phi; x) = E_{q_\phi(z|x)}[\log p_\theta(x|z)] - (q_\phi(z|x) || p(z)). \quad (3.2)$$

Here, $||$ is the Kullback–Leibler divergence, calculated analytically when $q_\phi(z|x)$ and $p(z)$ are both Gaussian. The ELBO provides a lower bound on the evidence $\log p_\theta(x)$ that is tight when the variational posterior $q_\phi(z|x)$ is exact. This expectation can be approximated via

Monte Carlo samples from $q_\phi(z|x)$. Gradients with respect to θ, ϕ can then be estimated by the reparameterization “trick” of sampling from $q_\phi(z|x)$ via linear transforms of standard normal variables [63, 97].

3.2.2 Hierarchical VAEs

Hierarchical VAEs (HVAEs, [107, 65]) extend the VAE by partitioning the latent code into L disjoint groups $z = (z_1, z_2, \dots, z_L)$, increasing model expressiveness for complex data like images [117, 16]. HVAEs generate these stochastic codes sequentially as $p_\theta(x|z) = p_\theta(z_1)(\prod_{\ell=2}^L p_\theta(z_\ell|z_{<\ell}))p_\theta(x|z_L)$, with a similar encoder: $q_\phi(z|x) = q_\phi(z_1|x) \prod_{\ell=2}^L q_\phi(z_\ell|z_{<\ell}, x)$. Each conditional in the decoder $p_\theta(z_\ell|z_{<\ell})$, and the encoder $q_\phi(z_\ell|z_{<\ell}, x)$, is typically Gaussian with mean and variance determined by (non-linear) neural networks. The HVAE ELBO equals

$$\begin{aligned} \mathcal{L}_H(\theta, \phi; x) = & E_{q_\phi(z|x)}[\log p_\theta(x|z)] - (q_\phi(z_1|x) || p_\theta(z_1)) \\ & - \sum_{\ell=2}^L E_{q_\phi(z_{<\ell}|x)}[(q_\phi(z_\ell|z_{<\ell}, x) || p_\theta(z_\ell|z_{<\ell}))], \end{aligned} \quad (3.3)$$

where $q_\phi(z_{<\ell}|x) = \prod_{i=1}^{\ell-1} q_\phi(z_i|z_{<i}, x)$ is the approximate posterior up to latent group $(\ell - 1)$. Reparameterization is then used to provide Monte Carlo gradient estimates.

We can rewrite the conditional prior and approximate posterior for layer ℓ to make the set of relevant networks explicit:

$$\begin{aligned} p_\theta(z_\ell|z_{<\ell}) &= \mathcal{N}(z_\ell | \mu_{\theta_\ell}(z_{<\ell}), \sigma_{\theta_\ell}(z_{<\ell})), \\ q_\phi(z_\ell|z_{<\ell}, x) &= \mathcal{N}(z_\ell | \mu_{\phi_\ell}(f_{\phi_\ell}(x), g_{\phi_\ell}(z_{<\ell})), \sigma_{\phi_\ell}(\dots)). \end{aligned} \quad (3.4)$$

Here, f_{ϕ_ℓ} and g_{ϕ_ℓ} are networks that extract feature representations of the observation x and

the previous layers $z_{<\ell}$, respectively. These features determine the mean and scale of the conditional Gaussian posterior via μ_{ϕ_ℓ} , σ_{ϕ_ℓ} . Networks μ_{θ_ℓ} , σ_{θ_ℓ} similarly generate the prior parameters for layer ℓ .

3.2.3 Methods for Inferring Missing Data

After training, synthetic data may be easily generated from $p_\theta(x)$ by sampling z from the learned VAE or HVAE prior, and $x \sim p_\theta(x|z)$. However, the learned encoder does *not* provide a direct mechanism for conditional queries about missing features, given partial or corrupted test data.

Let $x = (x_O, x_M)$ be a test data point with observed features x_O and missing features x_M . We are particularly interested in scenarios where the specific feature dimensions that are missing and observed will vary across test instances. Given a trained (hierarchical) VAE, for which x_O and x_M are conditionally independent given z , missing data is optimally predicted via the conditional distribution:

$$p_\theta(x_M | x_O) = \int p_\theta(x_M | z) p_\theta(z | x_O) dz. \quad (3.5)$$

This approach (and QAVI) are valid as long as data is *missing-at-random* (MAR, [71, 82]); the mechanism that removes features must be independent of x_M . Like most related work, our experiments use *missing-completely-at-random* (MCAR) feature masks whose distribution is independent of x . Exactly evaluating the predictive distribution (3.5) is infeasible due to the non-linear decoder, and intractable code posterior.

Heuristic Preprocessing. Imputation heuristics, such as replacing missing features with statistical summaries like their mean or mode, are widely used. Some work on training VAEs given partially missing data ([82, 86, 20]) propose a *Fill-Zeros* heuristic, simply replacing

missing features with zeros as input to the VAE encoder. While Fill-Zeros may be effective for learning models of MNIST digits (where zeros are common) when pixels are missing uniformly at random, our experiments show that in even slightly more complex scenarios, its performance is very poor.

Monte Carlo methods. [97] first proposed a simple scheme to approximately sample from $p_\theta(x_M | x_O)$ by starting with a random imputation, which is then stochastically encoded and decoded (or autoencoded) several times. Because the encoder only approximates the true posterior distribution $p_\theta(z | x)$, this *pseudo-Gibbs* sampler will not sample from the true posterior of missing features, and encoder inaccuracies may cause its equilibrium distribution to be far from $p_\theta(x_M | x_O)$. This approach was improved by [81], who proposed a Metropolis-Hastings correction to each step of the pseudo-Gibbs sampler, inducing a *Metropolis-in-Gibbs* sampler that asymptotically samples from $p_\theta(x_M | x_O)$. While Metropolis-in-Gibbs converges to the true posterior, it does so at a rate that may be impractically slow.

Amortized Inference for Imputation. Heuristic preprocessing may be avoided by learning a new “partial” encoder approximating $p_\theta(z | x_O)$. [20] concatenates zero-filled data with a binary mask indicating missing features, and optimizes the standard VAE ELBO of Eq. (3.2), but with the log-likelihood term calculated on only the observed data x_O . While this approach was developed to *train* VAEs with missing data, it is trivially applicable to test-time missing data imputation by retraining the encoder with the masked test data, generating a *Re-tuned Encoder*.

Posterior Matching [113] instead artificially masks the complete training data x as x_O , and tunes a partial-encoder $q_\psi(z | x_O)$ to “match” the pre-trained encoder by maximizing $E_{z \sim q_\psi(z | x_O)}[\log q_\psi(z | x_O)]$. In concurrent work, [40] introduced an equivalent approach, calling it *Inference in a Pretrained Artifact (IPA)*. [47] optimize the partial-encoder as in Posterior Matching, but simultaneously retrain the encoder and decoder to produce a conditional model $p(x_M | x_O)$. Their model includes skip connections between the partial-encoder

and decoder networks, as in the HVAE feature representations f_{ϕ_i} in Eq. (3.4). We adapt their *VAEAC* (arbitrarily conditioned) training to HVAEs by fine-tuning all three networks, starting from a pre-trained HVAE.

These amortized approaches to missing data imputation with VAEs have several drawbacks compared to our QAVI method. 1) They incur a substantial initial overhead for training the partial-encoder. 2) Partial-encoder training requires access to a relatively large set of partially-observed examples, and/or continued access to the training set along with a known missing-feature distribution. 3) They are sensitive to shifts in the distribution of queries (missing-feature patterns) between training and evaluation. If evaluated on previously unseen missing-feature patterns, performance may suffer substantially (see Fig. 3.4). 4) As we will demonstrate, even without distribution shifts, performance can be sub-optimal.

3.3 Query Adaptive VI

Our *query adaptive variational inference* (QAVI) utilizes the pre-trained VAE decoder, defines variational free parameters for each inference query, and refines them. We give an overview in Fig. 3.1 and show that it is adaptive across different queries without the need for additional partial-encoder networks. QAVI defines an explicit variational posterior over the unobserved variables $q_{\lambda}(z, x_M)$ with query-specific parameters λ . Fig. 3.1 shows two possible factorizations of the latent codes z and missing features x_M , leading to a pair of QAVI algorithms that we elaborate below.

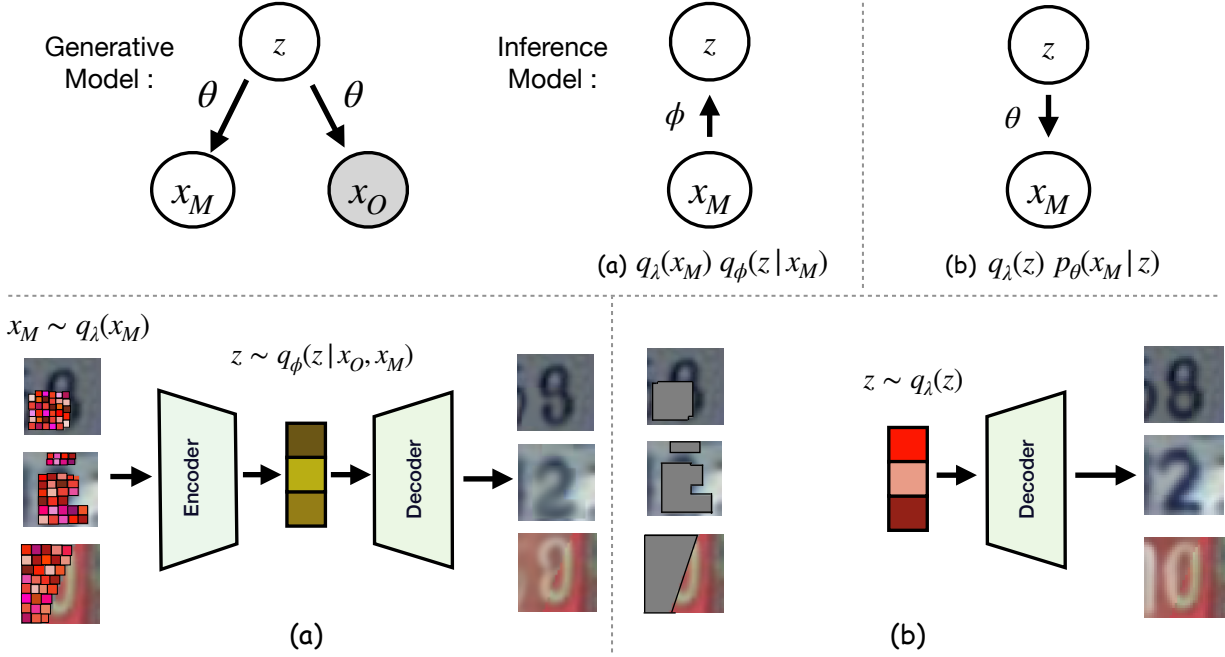


Figure 3.1: Overview of Feat. QAVI (a, Sec. 3.3.1) and non-amortized QAVI (b, Sec. 3.3.2). *Top:* Generative model for unobserved variables x_M, z and observed data x_O (shaded). We show two possible inference models for the joint variational distribution $q_\lambda(x_M, z)$ on latent variables. In (a), missing features are modeled directly by specifying $q_\lambda(x_M)$. In (b), queries are modeled indirectly via a tunable latent code distribution $q_\lambda(z)$. QAVI is easily adapted to existing VAEs by reusing generative (decoder) networks p_θ , and possibly inference (encoder) networks q_ϕ . *Bottom:* Computational flow of QAVI for each posterior factorization. In Feat. QAVI (a), samples from $q_\lambda(x_M)$ are autoencoded by the pre-trained VAE to compute the ELBO. In non-amortized QAVI (b), samples from $q_\lambda(z)$ are only passed through the decoder; the pre-trained encoder is not needed.

3.3.1 VI via Missing Feature Posteriors

Suppose first that the variational posterior factorizes as

$$q_\lambda(x_M, z | x_O) = q_\lambda(x_M)q_\lambda(z | x_M, x_O).$$

We fix $q_\lambda(z | x_M, x_O)$ to the amortized encoder $q_\phi(z|x)$, which has been pre-trained to approximate the relationship between x and z . Our *Feat. QAVI* method then defines an explicit posterior $q_\lambda(x_M)$ on missing features. This approach directly captures uncertainty

in the posterior of missing features (see Fig. 3.1) and their impact on the latent code.

In experiments, we found that fitting $q_\lambda(x_M)$ via the standard ELBO (see Section 3.6 for derivation) resulted in posteriors with unrealistically small variance. Following [43], we thus employ hyperparameters β to more strongly encourage the latent-code posterior to align with the prior, and γ to increase the entropy of the missing-feature posterior. The variational objective $\mathcal{L}_M$ is then:

$$\begin{aligned} \mathcal{L}_M(\lambda; x) = E_{q_\lambda(x_M)} [E_{q_\phi(z|x_O, x_M)} [\log p_\theta(x_O, x_M|z)] \\ - \beta(q_\phi(z|x_M, x_O) || p(z))] + \gamma H(q_\lambda(x_M)), \end{aligned} \quad (3.6)$$

where H is the entropy. We approximate $\mathcal{L}_M$ with S Monte Carlo samples as in standard VAE training:

$$\begin{aligned} \mathcal{L}_M(\lambda; x) \approx \frac{1}{S} \sum_{s=1}^S [\log p_\theta(x_O, x_M^{(s)}|z^{(s)}) \\ - \beta(q_\phi(z|x_O, x_M^{(s)}) || p(z))] + \gamma H(q_\lambda(x_M)), \end{aligned} \quad (3.7)$$

where $x_M^{(s)} \sim q_\lambda(x_M)$, $z^{(s)} \sim q_\phi(z|x_O, x_M^{(s)})$, and H are calculated in closed form. Automatic differentiation is used to compute gradients with respect to λ , tuning the missing-feature distributions to observed data indirectly.

While Feat. QAVI directly models the uncertainty in the posterior distribution of missing features, its optimization requires repeated computation of the encoder and decoder. It also inherits any suboptimality of the trained encoder. Perhaps surprisingly, we will show that in this context, a fully non-amortized inference method can have both greater computational efficiency and greater accuracy.

3.3.2 Non-amortized VI

We can construct an alternative variational posterior via the following factorization (see Fig. 3.1):

$$q_\lambda(x_M, z) = q_\lambda(z)q_\lambda(x_M|z),$$

By defining a variational posterior on z , we no longer use the encoder (except possibly for initialization), and re-use the pre-trained decoder by fixing $q_\lambda(x_M | z) = p_\theta(x_M | z)$. As derived in Section 3.6, this leads to the following *non-amortized QAVI* variational objective:

$$\mathcal{L}_N(\lambda; x) = E_{q_\lambda(z)}[\log p_\theta(x_O|z)] - \beta(q_\lambda(z)||p(z)). \quad (3.8)$$

To evaluate $\mathcal{L}_N$ we must only explicitly sample from the latent code; missing data may be *analytically* marginalized. Non-amortized QAVI seeks latent-code distributions that assign high likelihood to the observed features x_O , and are aligned with the prior via weight $\beta > 1$. In the simplest case where $q_\lambda(z)$ is a diagonal-covariance Gaussian, Eq. (3.8) is approximated via S samples from $q_\lambda(z)$, and the code mean and variance λ optimized via stochastic gradient ascent.

While VAE training implicitly encourages approximately Gaussian posteriors given *complete* observations, for queries given missing data, posteriors are often multi-modal and poorly approximated by a Gaussian $q_\lambda(z)$. To address this, we extend non-amortized QAVI to more expressive variational distributions that better capture the true posterior.

Flow Posteriors. The flow-based variational posterior aims to construct a complex distribution by transforming a simple Gaussian through a series of invertible mappings. We let $z_t = \mathcal{T}_t(z_{t-1}, \lambda_t)$ for $t = 1, \dots, T$, where z_0 is sampled from a Gaussian base distribution with

parameters λ_0 . λ_t is the set of parameters specifying flow layer $\mathcal{T}_t$, T is the total number of flow transformations, and $\lambda = \{\lambda_0, \lambda_1, \dots, \lambda_T\}$.

The idea of improving amortized variational inference in VAEs with the help of *normalizing flows* [114, 115] was first proposed by [96]. We instead employ autoregressive transformations in each layer $\mathcal{T}_t$ to capture high-dimensional dependencies in the latent space, producing *inverse autoregressive flows* (IAF, [61]). We approximate both terms in $\mathcal{L}_N(\lambda; x)$ of Eq. (3.8) via S samples from $q_\lambda(z_T)$; each Gaussian sample from the base distribution is transformed by T flow layers. Query-specific parameters λ are optimized by stochastic backpropagation.

Gaussian Mixture Posteriors. Parameterizing the latent space distribution $q_\lambda(z)$ as a *mixture of Gaussians* enables us to explicitly model different hypotheses in the latent space. Let $q_\lambda(z) = \sum_{t=1}^T w_t \mathcal{N}(z \mid \mu_t, \Lambda_t)$, where μ_t and Λ_t are the means and (diagonal) covariances of the T mixture components (posterior modes), w_t are mixture weights ($\sum_{t=1}^T w_t = 1$), and $\lambda = \{w_t, \mu_t, \Lambda_t\}_{t=1}^T$. Increasing T allows inference of more accurate posterior approximations.

Optimizing mixture parameters λ is not straightforward as discrete resampling from mixture weights cannot be continuously reparameterized. We use *implicit reparameterization gradients* (IRG, [34]) to efficiently compute gradients of the mixture component means and covariances. While in principle IRG could also be used to estimate gradients of mixture weights [39], in practice this estimator has enormous variance when posterior modes are widely separated. We instead adapt an importance-sampling gradient estimator [105] for mixture weights. Multiple samples S from the variational posterior are *necessary* to capture the impact of multiple posterior modes on the ELBO (3.8).

Hierarchical VAE Posteriors. For the hierarchical VAEs [107] introduced in Sec. 4.2, neither Gaussians nor Gaussian-mixtures are flexible enough to capture the non-linear dependencies between latent variables at different levels of the hierarchy. We therefore propose a new, more expressive (non-amortized) variational family for HVAEs that removes depen-

dency on the observation x , while retaining the expressive and non-linear dependencies of the HVAE model. Our variational posterior factorizes as:

$$q_\lambda(z) = q_\lambda(z_1) \prod_{\ell=2}^L q_\lambda(z_\ell | z_{<\ell}). \quad (3.9)$$

As in the HVAE decoder, we let $q_\lambda(z_\ell | z_{<\ell})$ be conditionally Gaussian for all ℓ . More complex conditional distributions (such as flows or mixtures) could also be used, but this conditionally Gaussian structure alone allows for expressive, multi-modal posterior approximations.

We propose a simple, generic strategy for constructing a family of non-amortized distributions given a pre-trained HVAE. Our approach applies to many recent hierarchical VAE architectures, including the “very-deep” HVAE [16] that we use in experiments.

To specify the non-amortized QAVI posterior, we begin with the amortized posterior defined in Eq. (3.4). Holding g_{ϕ_ℓ} , μ_{ϕ_ℓ} , σ_{ϕ_ℓ} fixed, we replace the features extracted from the observation with a new tunable parameter λ_ℓ . A further set of weighting parameters $\gamma_\ell, \gamma'_\ell \in [0, 1]$ interpolate these output parameters with those of the prior. Thus our hierarchical QAVI posterior for layer ℓ becomes:

$$\begin{aligned} \mu_\ell &= \gamma'_\ell \mu_{\phi_\ell}(\lambda_\ell, g_{\phi_\ell}(z_{<\ell})) + (1 - \gamma'_\ell) \mu_{\theta_\ell}(z_{<\ell}), \\ \sigma_\ell &= \gamma_\ell \sigma_{\phi_\ell}(\lambda_\ell, g_{\phi_\ell}(z_{<\ell})) + (1 - \gamma_\ell) \sigma_{\theta_\ell}(z_{<\ell}), \\ q_\lambda(z_\ell | z_{<\ell}) &= \mathcal{N}(z_\ell | \mu_\ell, \sigma_\ell). \end{aligned} \quad (3.10)$$

Re-using the pre-trained networks g_{ϕ_ℓ} , μ_{ϕ_ℓ} , σ_{ϕ_ℓ} allows the full amortized encoder to be used for initialization of λ_ℓ by simply setting $\lambda_\ell \leftarrow f_{\phi_\ell}(x_O, \tilde{x}_M)$. $\tilde{x}_M$ may be any initialization for the missing features, even Gaussian noise.

Our approach of interpolating posterior $(\mu_{\phi_\ell}, \sigma_{\phi_\ell})$ and prior $(\mu_{\theta_\ell}, \sigma_{\theta_\ell})$ network outputs is vital when reusing μ_{ϕ_ℓ} and σ_{ϕ_ℓ} from the original inference model. In the original, fully-observed

training phase, posterior variances may become extremely small for the highly overparameterized HVAE model. But with missing data, latent variables corresponding to unobserved features should have distributions close to the prior. Expressing the variational posterior as a weighted combination of prior and posterior network outputs allows our variational family to easily produce appropriate posteriors for latent variables corresponding to both observed and missing features, without needing to re-train μ_{ϕ_ℓ} and σ_{ϕ_ℓ} .

Hierarchical VAE Warmup. For hierarchical VAEs with a multi-scale architecture, we find that a warmup phase of optimization with a modified objective greatly accelerates posterior fitting. This idea is broadly proposed by [117], and refined as follows: $\mathcal{L}_H(\theta, \phi; x_O) =$

$$E_{q_\lambda(z)}[\log p_\theta(x_O|z)] - \frac{1}{d_1} D_{KL}(q_\lambda(z_1) || p_\theta(z_1)) - \sum_{\ell=2}^L \frac{1}{d_\ell} E_{q_\lambda(z_{<\ell})}[D_{KL}(q_\lambda(z_\ell|z_{<\ell}) || p_\theta(z_\ell|z_{<\ell}))], \quad (3.11)$$

where d_ℓ is proportional to the size of the latent space at layer ℓ . Intuitively, when inpainting large segments of an image, high-level structures should be determined first and details refined later. But for the unmodified ELBO, higher resolution latent layers contribute substantially more to the loss, leading to slow convergence. Fig. 3.2 illustrates the dramatic effect of this change during QAVI optimization.



Figure 3.2: Comparison of QAVI optimization for HVAEs without (*top*) and with (*bottom*) our KL-balanced warmup.

3.4 Experiments & Results

3.4.1 Experimental Setup

We evaluate QAVI¹ using six tabular datasets from the UCI Machine Learning Database [57]. For tabular data, we follow the experimental setup of [82] to train our VAEs, but use a Gaussian variational posterior instead of Student’s t .

We also consider three image datasets: real-valued MNIST ([70]), Street View House Numbers (SVHN) [87], and FFHQ-256 [54]. We use a single-stochastic-layer VAE for MNIST and SVHN, with a WideResNet architecture [136] that is known to work well with images. We train them with fully-observed images from the training set (70,000 MNIST images and 73,257 SVHN images) and maximize the ELBO of Eq. (3.2). For FFHQ-256, we adopt

¹Code available: <https://github.com/SakshiAgarwal/QAVI>

the “very-deep” hierarchical architecture of [16], but for efficient comparison we re-trained a smaller variant of their original HVAE (5.9M vs. 115M parameters).

While QAVI handles missing-at-random (MAR) data naturally, we setup our experiments to be consistent with most prior work on imputation with VAEs. For MNIST and SVHN we consider two missing-completely-at-random (MCAR) patterns: 1) two randomly placed patches (each of size 10x10 for MNIST, 15x15 for SVHN); 2) a randomly rotated mask of half of the image. We use a more challenging random mask distribution [139] for FFHQ-256. For tabular datasets, we corrupt the test set by removing half of the features in each row uniformly at random.

Baselines. We compare QAVI with several methods from the literature: *i)* The *Fill Zeros* heuristic [86]; *ii)* Monte Carlo methods: *pseudo-Gibbs* [97] and *Metropolis-in-Gibbs* [81]; *iii)* Amortized inference methods: *Re-tuned Encoder* [20] and *Posterior Match[ing]* [113]. We consider three variants of posterior matching to evaluate the importance of knowing the missing-feature pattern during training. *Posterior Match (True)* trains the posterior matching encoder with exactly the same query distribution used for test evaluation. *Posterior Match (Rand.)* assumes that only the fraction of missing features is known; features are removed at random with this probability. The generic *Posterior Match* uses the image masking distribution of [139], which assumes contiguous masked regions without specific knowledge of queries to be evaluated.

We compare also QAVI to state-of-the-art approaches to inpainting with HVAEs: *Posterior Match[ing]* and *VAEAC* [47]. Both *Posterior Match* and *VAEAC* benefitted from training with the same random mask distribution [139] used in evaluation. As our “very-deep” HVAE architecture already links the encoder and decoder, we do not include extra deterministic skip connections as in the original VAEAC architecture.

Hyperparameters. QAVI optimization for VAE models uses $S = 100$ Monte Carlo

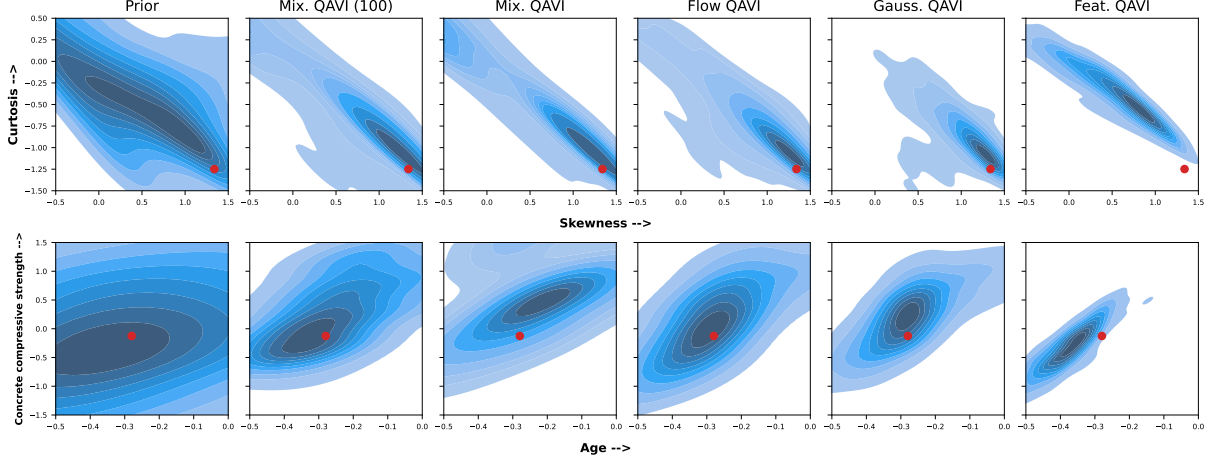


Figure 3.3: Kernel density visualizations of the prior and approximate posteriors for five QAVI variants, fit using $S = 1000$ samples from the optimized variational distribution. We show one test sample chosen from two UCI tabular datasets, Banknote (*top*) and Concrete (*bottom*), containing two missing features (labeled on axes). We observe that the true data point (*red*) is often enclosed by the different posteriors, but sometimes missed by Feat. QAVI. We approximate the true posterior via a highly expressive QAVI mixture of 100 Gaussians, fit by extended optimization.

samples to estimate our variational objective and gradients, and optimizes variational parameters λ for 300 steps using Adam [60]. We refer to our posterior on missing features (Sec. 3.3.1) as *Feat. QAVI*, and our Gaussian/Flow/Mixture non-amortized variational posteriors (Sec. 3.3.2) on latent codes z as *Gaus./Flow/Mix. QAVI*. For HVAE models, QAVI optimization uses $S = 28$ samples to estimate the ELBO, and optimizes for 1000 total steps (including 500 warmup steps as in Fig. 3.2).

Metrics. For a quantitative analysis, we estimate the marginal log-likelihood of the missing features x_M using the importance sampling estimator of [14]:

$$\begin{aligned} \log p_\theta(x_M) &\geq \mathbb{E}_{z^{(s)} \sim q(z)} \left[\log \frac{1}{S} \sum_{s=1}^S \frac{p_\theta(x_M, z^{(s)})}{q(z^{(s)})} \right] \\ &\approx \log \frac{1}{S} \sum_{s=1}^S \frac{p_\theta(x_M, z^{(s)})}{q(z^{(s)})}, \quad z^{(s)} \sim q(z). \end{aligned} \quad (3.12)$$

The true log-likelihood of missing features is constant for all inference methods, since the

Table 3.1: Test missing data log-likelihoods (LL, higher is better) and normalized root mean-square error (NRMSE, lower is better) for 5 tabular datasets from the UCI repository, estimated using $S=10,000$ samples. NRMSE per test row is the minimum across S samples. QAVI variants have superior performance (highlighted in bold) for almost all data.

	Breast Cancer		Red wine		White wine		Banknote		Concrete	
	LL	NRMSE	LL	NRMSE	LL	NRMSE	LL	NRMSE	LL	NRMSE
Mix. QAVI	-9.16	0.39	-6.35	0.26	-7.03	0.35	-2.25	0.10	-2.70	0.17
Flow QAVI	-9.14	0.39	-6.47	0.26	-7.03	0.35	-2.24	0.10	-2.65	0.17
Gaus. QAVI	-9.21	0.39	-6.56	0.30	-6.94	0.30	-2.35	0.10	-2.82	0.17
Feat. QAVI	-16.32	0.37	-13.20	0.28	-8.80	0.38	-8.27	0.15	-15.14	0.23
	-7.81	0.47								
Posterior Match	-15.14	0.33	-14.83	0.42	-9.19	0.39	-4.42	0.14	-15.05	0.40
Re-tuned Encoder	-12.82	0.40	-12.96	0.40	-9.59	0.39	-4.26	0.14	-11.48	0.35
Metropolis-in-Gibbs	-23.94	0.44	-74.22	0.68	-19.83	0.58	-49.51	0.68	-265.62	0.62
pseudo-Gibbs	-12.61	0.33	-33.53	0.36	-10.68	0.36	-44.16	0.40	-249.06	0.36
Fill Zeros	-32.56	0.40	-21.97	0.36	-10.09	0.37	-17.46	0.32	-23.33	0.32

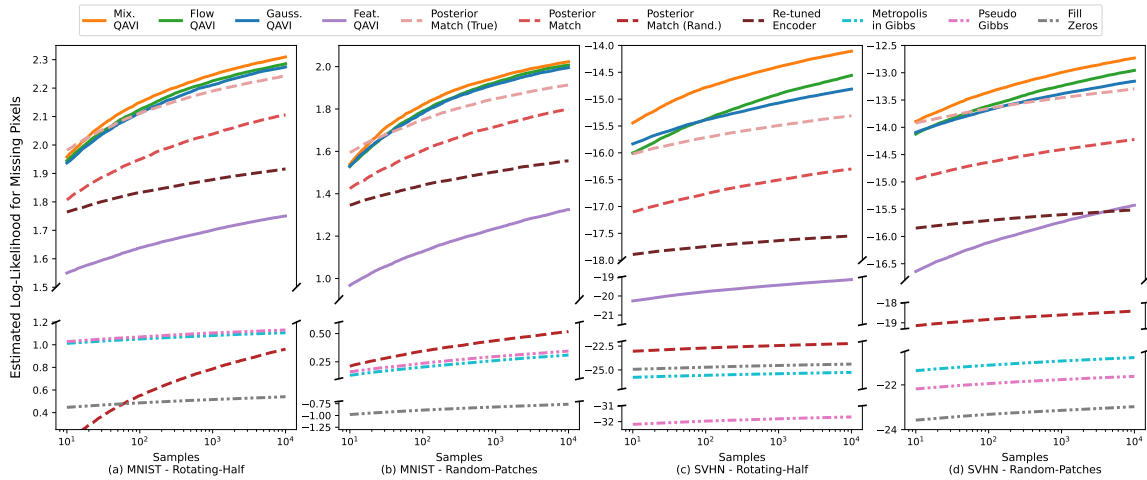


Figure 3.4: For two image mask distributions and several inference methods (*top*), we plot importance-weighted log-likelihood estimates for missing pixels and varying samples S . We average over 1000 MNIST (*left*) or SVHN (*right*) test images.

generative model $p_{\theta}(x)$ is fixed, but better posterior approximations lead to tighter lower bounds for a fixed number of samples S . Fig. 3.4 shows estimated log-likelihoods versus the number of samples.

As the importance-weighted likelihood estimator becomes expensive and unreliable to evaluate for high-dimensional models like HVAEs, we use perceptual metrics to evaluate inpainting results for HVAEs on the FFHQ dataset. Table 3.2 reports three metrics on a test set of 1000 images: FID [42] as well as P-IDS and U-IDS [139]. We modify P-IDS and U-IDS slightly to reduce sensitivity to the test set size.

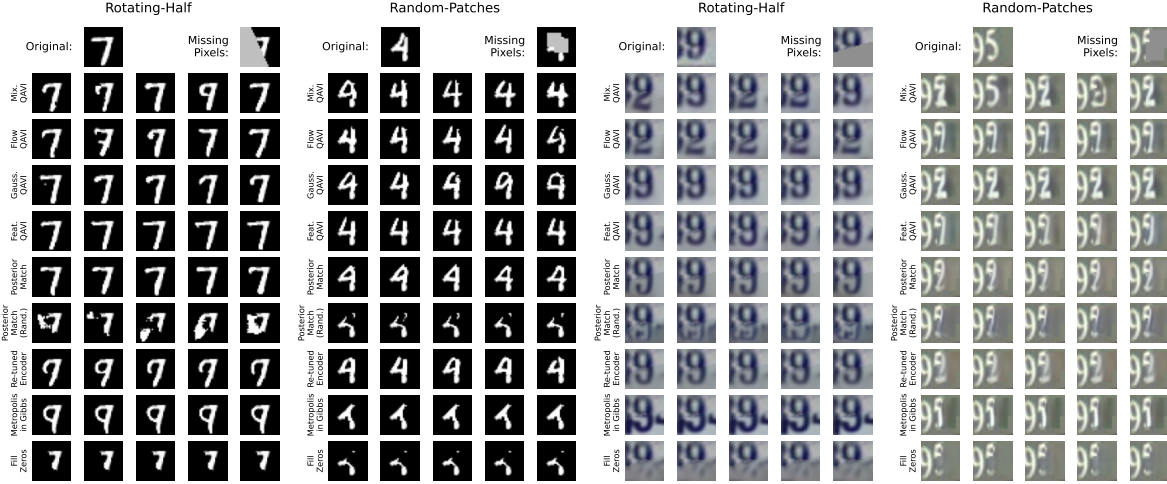


Figure 3.5: Digit completion results on MNIST (*left*) and SVHN (*right*) images for inference queries with pixels obscured by the Rotating-Half or Random-Patches distributions. We show 5 samples from each inferred posterior. Monte Carlo and amortized inference methods propose one sometimes-valid digit completion; amortized VI is typically more accurate. The performance of Posterior Match varies widely depending on the missing-feature distribution it is trained on. In contrast, QAVI automatically adapts to queries, and proposes multiple valid imputations that effectively capture posterior uncertainty.

3.4.2 Results

QAVI improves imputation quality. Fig. 3.4 and Table 3.1 compare the log-likelihood of missing features across MNIST, SVHN, and tabular datasets. We see that heuristic imputation and Monte Carlo methods perform poorly. Re-tuned Encoder and Posterior Match show relatively higher likelihoods, but do not match QAVI across any dataset or missingness pattern. Feat. QAVI is competitive for tabular data, but for high-dimensional images it is susceptible to local optima. Fig. 3.7 similarly shows that QAVI provides reliably strong performance for downstream tasks.

QAVI can capture multi-modal posterior uncertainty. We show imputations for four test examples in Fig. 3.5 to highlight differences in inference methods, and to explore the uncertainty in the inferred posterior over the missing features. We see that Gaussian and Flow QAVI defined on the latent space are capable of capturing uncertainty in the



Figure 3.6: Inpainting results on the FFHQ-256 dataset, comparing our non-amortized deep QAVI inpainting with VAEAC and Posterior Matching. We also compare QAVI results for the reduced-size models used in Table 3.1 to inpaintings from the original “very-deep” HVAE of [16]. We show the true and masked images, and 5 posterior samples for each method.

missing features. With a mixture of Gaussians variational family, QAVI produces multiple visually-plausible imputations. The classification performance in Fig. 3.7 shows high relative classification accuracy for expressive posteriors despite the increased variance in samples.

QAVI benefits extend to HVAEs. We find that QAVI also integrates well with hierarchical VAE models. The results in Table 3.2 show that our QAVI approach for HVAEs produces imputations with higher perceptual scores than prior methods for leveraging HVAEs for in-

Table 3.2: Quantitative comparison of perceptual inpainting quality on the FFHQ-256 dataset. We compare QAVI against two state-of-the-art adaptations of HVAEs to inpainting, using the same base “very-deep” HVAE architecture.

Method	FID ↓	P-IDS* ↑	U-IDS* ↑
QAVI	21.21	6.20	24.98
Posterior Match	23.68	3.36	21.55
VAEAC	26.41	2.31	18.19

painting. Figure 3.6 shows that samples produced by QAVI are qualitatively more visually plausible, and also capture substantial diversity in possible feature imputations.

Amortized imputation is sensitive to training queries. Fig. 3.4 compares the performance for Posterior Matching when trained with different masking distributions. We see that with the absence of any prior knowledge of the true masking distribution at train time, the performance of Posterior Matching can be as poor as simple heuristics like Fill Zeros. Even in the unrealistic case where the exact distribution of missingness is known at train time, posterior matching does not outperform QAVI. This sensitivity to the choice of missingness for training is significant: adaptation to new patterns of missingness require full retraining. In sensitive domains such as medicine, access to the original training set may be restricted or even impossible. QAVI is indifferent to the structure of queries, requires no retraining, and *still* outperforms the “best case” amortized imputation.

QAVI smoothly trades off time and performance. Fig. 3.8 illustrates the tradeoff between performance and optimization time for three variants of non-amortized QAVI, for 100 images from the MNIST dataset. We see that the performance of Posterior-Matching is far lower than QAVI, *and* requires substantial overhead to train the partial encoder (over 3.5 hours). Gaussian and Flow QAVI converge in about one minute. Mixture QAVI converges a bit more slowly, but ultimately reaches the best solutions of any method. Posterior-Match amortization would have computational advantages for very-large query sets (thousands of images), but would still have inferior inference accuracy.

3.5 Conclusion

We have presented a simple and a general framework that has been unexplored in prior work employing VAEs for the imputation of missing data. Previous state-of-the-art approaches make use of a restrictive inference network as an imputation strategy. We instead take an existing VAE generative model (decoder), allocate variational parameters for the latent code of each missing data point, and update the parameters stochastically to optimize the induced variational bound. The simple structure of our bounds enables efficient and accurate approximation of the posterior distribution of missing features, given any pattern of observed features.

We evaluated QAVI on a variety of VAEs, including current state-of-the-art hierarchical VAEs, and several datasets. We found that non-amortized QAVI with Gaussian, and especially Flow or Mixture, posterior approximations outperforms previous heuristic and amortized inference methods data imputation with VAEs. Importantly, we find that a Gaussian Mixture posterior is able to effectively capture the multi-modality that often arises given missing data.

In this work, we do not consider *missing-not-at-random* (MNAR) data [46], but we conjecture that QAVI will provide a foundation for future advances in MNAR inference. QAVI provides a simple, effective, and general approach for inference of missing data with arbitrary patterns, that is attractive when queries are unknown during training and uncertainty in missing data is high. QAVI can further extend to different application areas like improving Inference for Neural Image Compression [127].

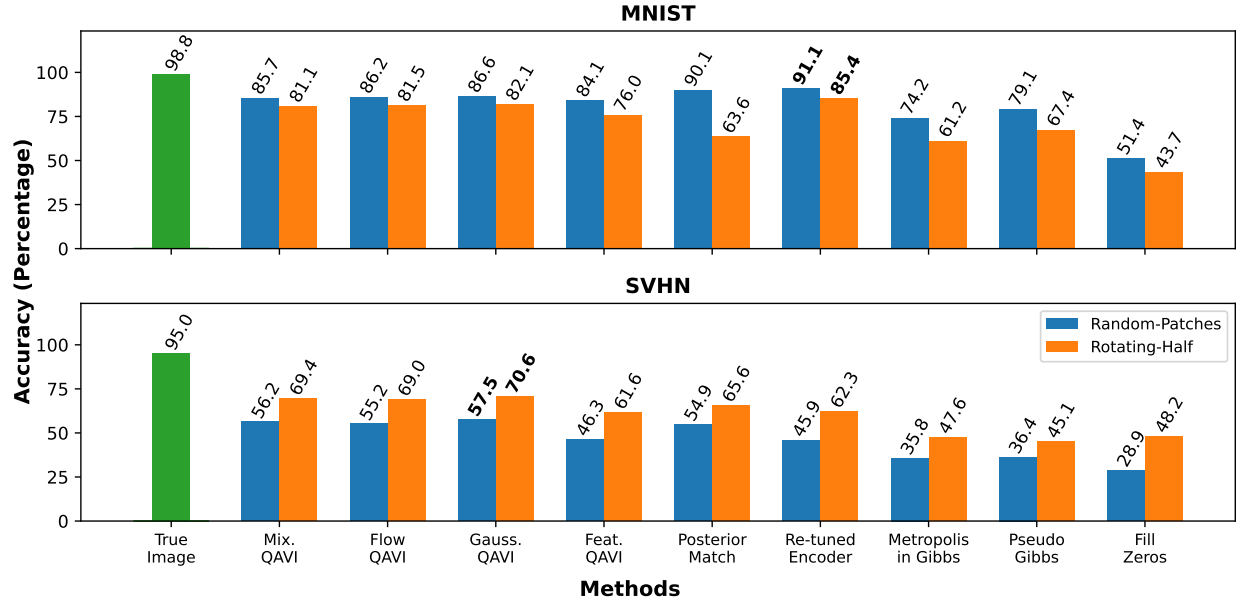


Figure 3.7: Classification accuracies using 100 samples from the inferred posterior for randomized missing queries on MNIST (*top*) and SVHN (*bottom*). We use a trained discriminative model, with WRN-28-2 architecture [136], to predict class labels.

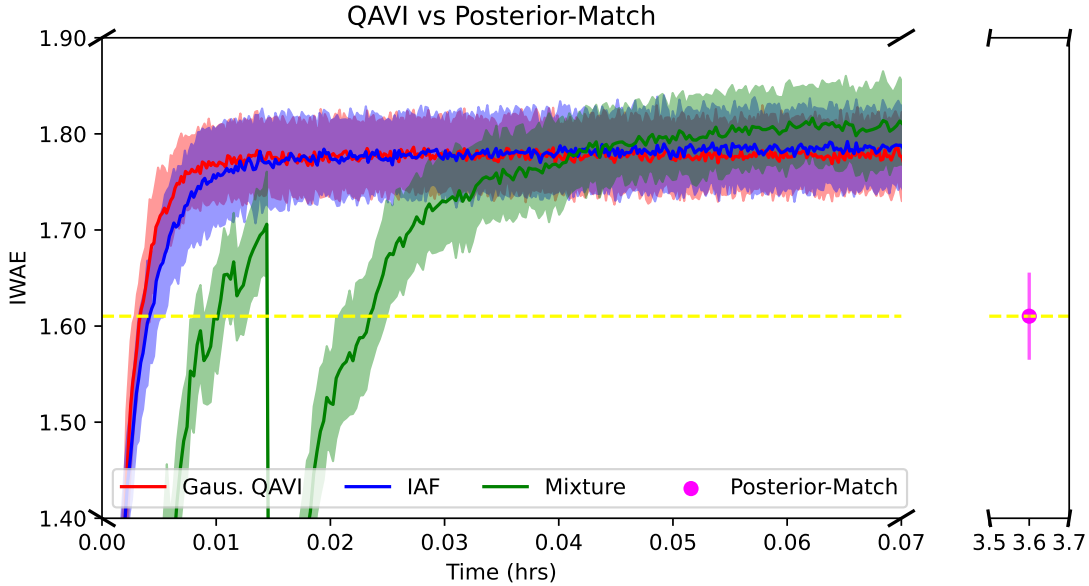


Figure 3.8: Importance-weighted log-likelihood (IWAE) estimates for missing pixels versus wall-clock training time (in hours). Likelihoods are estimated at each step of optimization for 100 MNIST images with pixels missing via Random-Patches. We plot the mean and standard deviation across 10 runs of QAVI methods. We compare to Posterior-Match, whose amortized inference network requires over 3.5 hours to train. The dip in mixture log-likelihoods occurs at random re-initialization of mixture parameters to avoid local optima.

3.6 QAVI: Derivations

3.6.1 Feat. QAVI

As specified in the main chapter, we define a variational distribution over unobserved features x_M as $q_\lambda(x_M)$ and re-use the encoder to model $q(z|x_O, x_M)$. Hence, the ELBO $\mathcal{L}_M(\lambda; x_O)$ can be written as :

$$\begin{aligned}
&= E_{q_\lambda(x_M, z|x_O)}[\log p_\theta(x_O, x_M, z) - \log q_\lambda(z, x_M|x_O)] \\
&= E_{q_\lambda(x_M, z|x_O)}[\log p_\theta(x_O, x_M|z) + \log p(z) - \log q_\lambda(x_M) - \log q_\phi(z|x_M, x_O)] \\
&= E_{q_\lambda(x_M)}[E_{q_\phi(z|x_O, x_M)}[\log p_\theta(x_O, x_M|z) - \log q_\phi(z|x_M, x_O) + \log p(z)]] - E_{q_\lambda(x_M)}[\log q_\lambda(x_M)] \\
&= E_{q_\lambda(x_M)}[E_{q_\phi(z|x_O, x_M)} \log[p_\theta(x_O, x_M|z)] - KL(q_\phi(z|x_M, x_O)||p(z))] - E_{q_\lambda(x_M)}[\log q_\lambda(x_M)] \\
&= E_{q_\lambda(x_M)}[E_{q_\phi(z|x_O, x_M)} \log[p_\theta(x_O, x_M|z)] - KL(q_\phi(z|x_M, x_O)||p(z))] + H(q_\lambda(x_M))
\end{aligned} \tag{3.13}$$

where $H(q_\lambda(x_M))$ is the entropy of $q_\lambda(x_M)$ and KL is the Kullback-Leibler divergence. Monte-Carlo approximation of Eq. 3.13 with S samples from $q_\lambda(x_M, z|x_O)$ is equivalent to sampling $x_M^{(s)} \sim q_\lambda(x_M)$ and $z^{(s)} \sim q_\phi(z|x_O, x_M^{(s)})$. We get:

$$\mathcal{L}_M(\lambda; x_O) \approx \frac{1}{S} \sum_{s=1}^S [\log p_\theta(x_O, x_M^{(s)}|z^{(s)}) - KL(q_\phi(z|x_O, x_M^{(s)})||p(z))] + H(q_\lambda(x_M)) \tag{3.14}$$

3.6.2 Non-Amortized Inference

Here, we define a variational distribution on the latent space variable z as $q_\lambda(z)$. The ELBO can thus be derived as follows:

$$\begin{aligned}
\mathcal{L}_N(\lambda; x_O) &= E_{q_\lambda(z, x_M | x_O)} [\log p_\theta(x_O, x_M, z) - \log q_\lambda(z, x_M | x_O)] \\
&= E_{q_\lambda(z, x_M | x_O)} [\log p_\theta(x_O, x_M | z) + \log p(z) - \log q_\lambda(z) - \log p_\theta(x_M | z)] \\
&= E_{q_\lambda(z)} [\log p_\theta(x_O | z) + \log p(z) - \log q_\lambda(z)] \\
&= E_{q_\lambda(z)} [\log p_\theta(x_O | z)] - (q_\lambda(z) || p(z))
\end{aligned} \tag{3.15}$$

Approximating the above with S samples, $z^{(s)} \sim q(z)$, we get :

$$\mathcal{L}_N(\lambda; x_O) \approx \frac{1}{S} \sum_{s=1}^S [\log p_\theta(x_O | z^{(s)})] - (q_\lambda(z) || p(z)) \tag{3.16}$$

Chapter 4

Variational Inference for Denoising Diffusion Models

Even though the hierarchical VAEs discussed in the previous chapter are able to cover the different modes in the data, they struggle to generate high-quality image samples [117]. However, with the groundbreaking work in [30], Denoising Diffusion Models have outperformed Generative Adversarial Networks (GANs), a previously popular deep generative model known for producing realistic images. In addition to generating high-quality images, Diffusion Models have also addressed the mode collapse issue inherent in GANs, making them a widely adopted approach for various conditional image generation tasks [98, 103, 102]. In this chapter, we present a variational inference algorithm for pre-trained diffusion models. We demonstrate that, rather than defining the posterior at every level in the latent space hierarchy, it is possible to selectively choose a few informative levels within the diffusion model. One application of such an inference algorithm is to infill partially observed images without training a specific model for this task. While previous work [17, 78, 76, 137, 7, 101, 108] utilizes pre-trained DMs and focuses on incorporating the observation in the data generating reverse diffusion process, we explicitly *infer* the posterior distribution over mid-ranged

latent variables z in the presence of an observation y , i.e. $p(z | y)$. This inference procedure captures a multi-level hierarchy of interactions from global-to-local details in the image. We show in this chapter through extensive experimentation, that this innovative inference procedure is able to result in perceptually consistent and diverse image reconstructions in the case of large masking ratios. We further show that such an approach can also be applied to other linear inverse problems like superresolution and deblurring. This chapter is based on our work [3].

4.1 Introduction

Diffusion models [44, 112, 89, 110] generate synthetic data by sequentially denoising Gaussian noise for hundreds or thousands of time steps, producing deep generative models that have advanced the state-of-the-art in natural image generation [30, 59, 53]. Diffusion models for high-dimensional data like images are computationally intensive. Hence, *Latent Diffusion Models* (LDMs) [98, 118] improve efficiency by leveraging an autoencoder [64] to map data to a lower-dimensional encoding, and then training a diffusion model for the lower-dimensional codes. This dimensionality reduction enables tractable but expressive models for images with millions of pixels. The effectiveness of LDMs has made them a new standard for natural image generation, and they are thus our focus here.

Motivated by the foundational information captured by diffusion models of images, numerous algorithms have incorporated a pre-trained diffusion model as a prior for image editing [83], inpainting [112, 123, 56, 18, 76, 15, 32, 116, 31], or other inverse problems [52, 109, 38, 78, 17]. Many of these prior methods are specialized to inpainting with pixel-based diffusion models, where every data dimension is either perfectly observed or completely missing, and are not easily adapted to state-of-the-art LDMs.

Additionally, for the task of image inpainting, prior works (DPS, [17], RedDiff [78]) only employ masks covering a small percentage of the image. More recent works (RePaint [76], CoPaint [137]) note that their methods struggle with larger masks. Recently, [72] proposed a probabilistic circuit [23] based approach and shows some improvements over some widely used methods for inpainting, but is supervised and is highly biased towards mask-specific training. Another recent paper [121] uses additional guidance (example segmentation, depth, pose etc) to solve inpaint large mask regions. Most of the widely used inpainting algorithms employ an iterative refinement procedure, like that used to generate unconditional samples, and then guide their predictions towards the partially observed image via various approximations and heuristics. In Fig. 4.1, we see that by sequentially annealing from independent Gaussian noise to noise-free images, these approaches produce myopic samples that do not adequately incorporate information from observed pixels, and fail to correct errors introduced in earlier stages of the “reverse-time” diffusion. More recent work has extended these approaches to enable image editing [7] or inpainting [101, 21, 17, 108] with LDMs, but we see in our experiments that they continue to suffer similar inaccuracies.

In this work, we present *VIPaint*, a novel application of *variational inference* (VI) [120, 13] that employs both LDMs and pixel-based DMs as priors to handle large masks for image inpainting. VI has achieved excellent image restoration results with a wide range of priors, including mixtures [33, 50] and hierarchical VAEs [4], but there is little work exploring its integration with state-of-the-art LDMs. While *Red-Diff* [78] applies VI to approximate the posterior of pixel-based DMs, its local approximation of the noise-free image posterior is difficult to optimize, requiring annealing heuristics that we demonstrate are sensitive to local optima. Instead, *VIPaint* strategically defines a hierarchical, Markovian and non-Gaussian approximation to the true (L)DM posterior that accounts for a subset of latent noise levels, enabling the inference of both high-level semantics and low-level details from observed pixels *simultaneously* (see Fig. 4.1). Further, we efficiently infer variational parameters for each inpainting query, avoiding the need to collect a training set of corrupted images [72, 21],

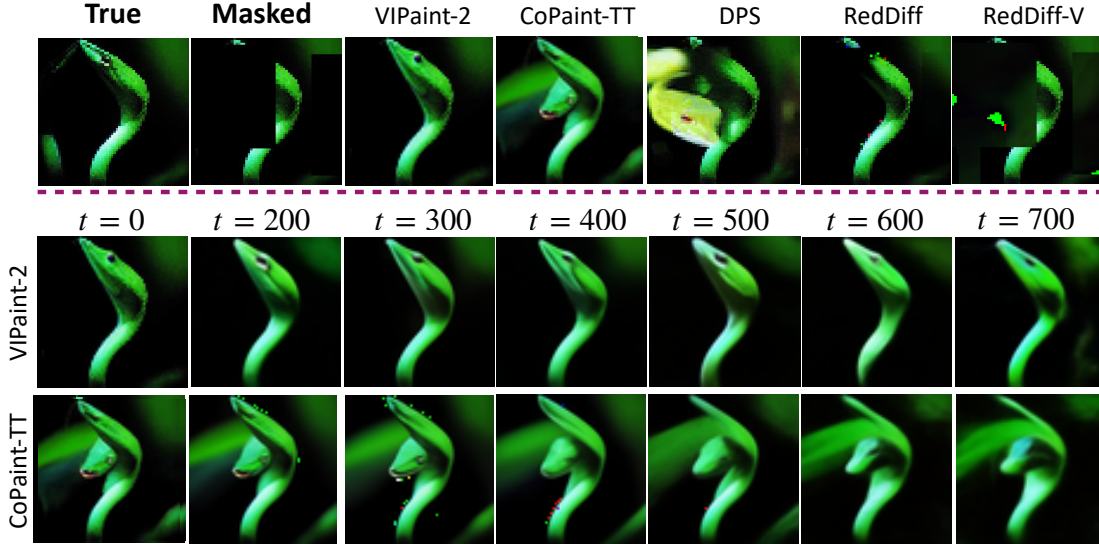


Figure 4.1: **Top:** Sampling methods like CoPaint-TT [137] and DPS [17] anneal from Gaussian noise to noise-free images while guiding samples towards observation data and tend to produce incoherent images. Whereas, RedDiff [78] defines its variational posterior directly on the noise-free data space and produces blurry images. Direct optimization of its variational bound (RedDiff-V) leads to poor local optima, so must resort to annealing through diffusion time (RedDiff, $T \rightarrow 0$). Our VIPaint method suffers from no such difficulty and uses the variational bound for optimization, leading to more robust inference. **Bottom:** We visualize samples generated at intermediate time points in the diffusion latent space and compare VIPaint with the best performing baseline, CoPaint-TT. We see that intermediate samples for CoPaint-TT don’t infer the underlying image correctly, and that these errors cannot be corrected later in the sampling process, leading to severe inconsistencies in final inpaintings as shown. In contrast, samples from our VIPaint posterior align better with the observations, even at higher noise levels, resulting in more coherent inpaintings.

expensively fine-tune generative models [7] or variational posteriors [32] for each query, or retrain large-scale conditional diffusion models [98, 102, 88, 19].

We begin by reviewing properties of (latent) diffusion models in Sec. 4.2, and prior work on inferring images via pre-trained diffusion models in Sec. 4.3. Sec. 4.4 then develops the VIPaint algorithm, which first fits a hierarchical posterior that best aligns with the observations, and then samples from this approximate posterior to produce diverse reconstruction hypotheses. Results in Sec. 4.5 on inpainting then show substantial qualitative and quantitative improvements in capturing multimodal uncertainty for both pixel-based and latent DMs.

4.2 Background

The diffusion process begins with clean data x , and defines a sequence of increasingly noisy versions of x , which we call the *latent variables* z_t , where t runs from $t = 0$ (low noise) to $t = T$ (substantial noise). The distribution of latent variable z_t conditioned on x , for any integer time $t \in [0, T]$, is

$$q(z_t | x) = \mathcal{N}(z_t | \alpha_t x, \sigma_t^2 I), \quad (4.1)$$

where α_t and σ_t are strictly positive scalar functions of t . This noise implicitly defines a Markov chain for which the conditionals $q(z_t | z_{t-1})$, $q(z_{t-1} | z_t, x)$ are Gaussians. The signal-to-noise ratio [62] induced by this diffusion process at time t equals $SNR(t) = \alpha_t^2 / \sigma_t^2$. The SNR monotonically decrease with time, so that $SNR(t) < SNR(s)$ for $t > s$. This DM specification includes variance-preserving diffusions [44, 106] as a special case, where $\alpha_t = \sqrt{1 - \sigma_t^2}$. Another special case, variance-exploding diffusions [110, 112], takes $\alpha_t = 1$.

Image Generation. The generative model reverses the diffusion process outlined in Eq. (4.1), resulting in a hierarchical generative model that samples a sequence of latent variables z_t before sampling x . Generation progresses backward in time from $t = T$ to $t = 0$ via a finite temporal discretization into $T \approx 1000$ steps, either uniformly spaced as in discrete diffusion models [44], or via a possibly non-uniform discretization [53] of an underlying continuous-time stochastic differential equation [112]. Denoting $t - 1$ as the timestep preceding t , for $0 < t < T$, the hierarchical generative model for data x is expressed as follows:

$$p_\theta(x) = \int_z p(z_T) p(x | z_0) \prod_{t=1}^T p_\theta(z_{t-1} | z_t) dz. \quad (4.2)$$

The marginal distribution of z_T is typically a spherical Gaussian $p(z_T) = \mathcal{N}(z_T | 0, \sigma_T^2 I)$. Pixel-based diffusion models take $p(x | z_0)$ to be a simple factorized likelihood for each pixel

in x , while LDMs define $p(x \mid z_0)$ using a decoder neural network. The conditional latent distribution $p_\theta(z_{t-1} \mid z_t)$ maintains the same form as the forward conditional distribution $q(z_{t-1} \mid z_t, x)$, but with the original data x substituted by the output of a parameterized denoising model, z_0 as

$$p_\theta(z_{t-1} \mid z_t) = q(z_{t-1} \mid z_t, z_0 = \hat{z}_\theta(z_t, t)), \quad \text{where} \quad \hat{z}_\theta(z_t, t) = \frac{z_t - \sigma_t \hat{\epsilon}_\theta(z_t, t)}{\alpha_t}. \quad (4.3)$$

This denoising model $\hat{\epsilon}_\theta(z_t, t)$ typically uses variants of the UNet architecture [100] and is trained to optimize a re-weighted variational lower bound of the marginal likelihood of data x , simplified as [44, 112],

$$\mathcal{L}_{(0,T)}(z_0) = \frac{T}{2} \mathbb{E}_{\epsilon \sim \mathcal{N}(0,1), t \sim \mathcal{U}(1,T)} \left[\|\epsilon - \hat{\epsilon}_\theta(z_t, t)\|_2^2 \right]. \quad (4.4)$$

where the expectation is taken over t , $z_t \sim q(z_t \mid x)$, and $x \sim p_{\text{data}}(x)$. LDMs use an encoder $q_\phi(z_0 \mid x)$ to map x to a lower-dimensional space and train a diffusion model in this reduced encoding space for efficiency.

4.3 Background: Inference with Diffusion Models

In many real-life scenarios, we encounter partial observations $y = x \odot m$, where m is a binary mask indicating missing pixels. In cases where large portions of image is masked, exactly recovering x from y is challenging, because many x could produce the same observation y . To express the resulting posterior $p_\theta(x \mid y)$ given a DM prior, we can adapt the generative process in Eq. (4.2) as:

$$p_\theta(x \mid y) = \int_z p_\theta(z_T \mid y) p_\theta(x \mid z_0, y) \prod_{t=1}^T p_\theta(z_{t-1} \mid z_t, y) dz. \quad (4.5)$$

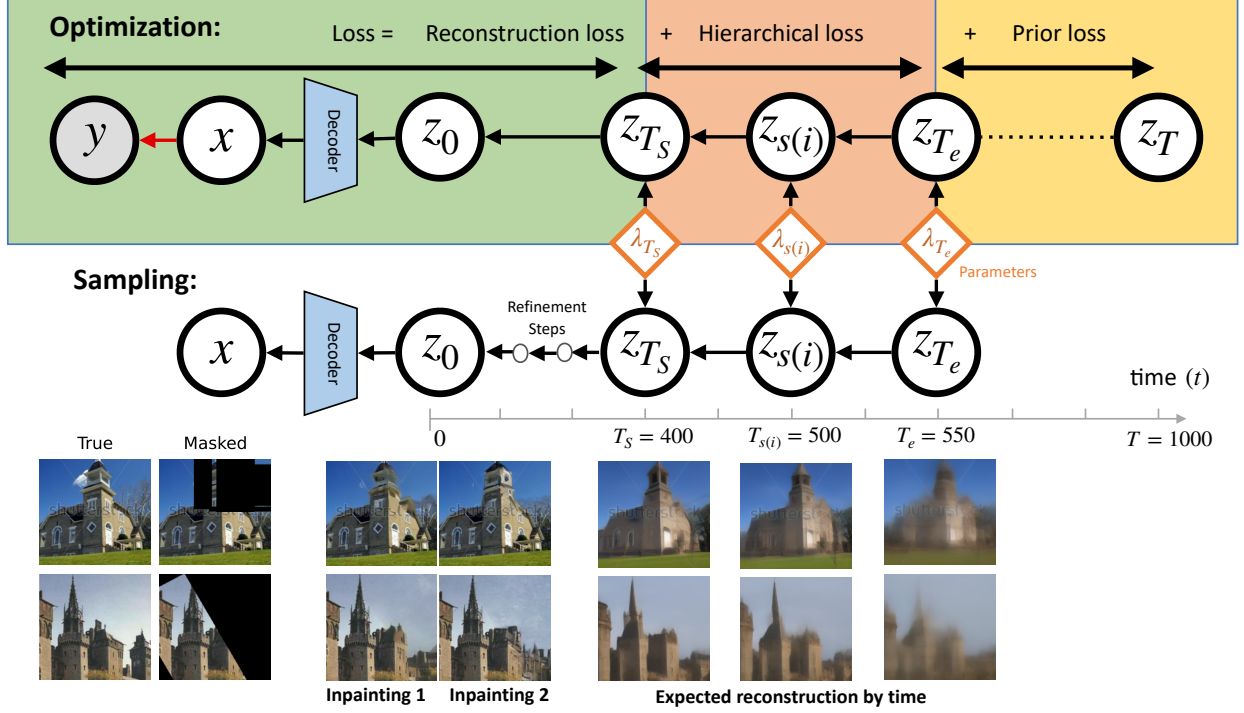


Figure 4.2: *Top*: The hierarchical approximate posterior of VIPaint is defined over a coarse sequence of intermediate latent steps between T_e and T_s . During optimization, the variational parameters λ defining the posterior on a subset of latent times are fit via a prior loss on times above T_e , a hierarchical loss defined across K intermediate times, and a reconstruction loss estimated using a one-step approximation $p_\theta(x|z_{T_s})$ from the posterior samples. *Bottom*: After variational inference, samples from the hierarchical posterior (now *aligned* with the observation) transition smoothly in the intermediate latent space $[0, T_s]$ via gradient updates.

Exactly evaluating this predictive distribution is infeasible due to the non-linear noise prediction (and decoder) network, and the intractable posteriors of latent codes $p(z_{t-1} | z_t, y)$ for all t . Various methods have been proposed to conditionally sample latent codes z_{t-1} , detailed below.

4.3.1 Sampling Methods for Inverse Problems

Blending Methods methods [111, 123] define a procedural, heuristic approximation to the posterior and are tailored for image inpainting. They first generate unconditional samples z_{t-1} from the prior using the learned noise prediction network, and then incorporate y by

replacing the corresponding dimensions with the observed measurements. RePaint [76] attempts to reduce visual inconsistencies caused by blending via a resampling strategy. A “time travel” operation is introduced, where images from the current time step z_{t-1} are first blended with the noisy version of the observed image y_{t-1} , and then used to generate images in the $(t-1)+r$, ($r \geq 1$) time step by applying a one-step forward process and following the Blended denoising process.

Gradient-Based Methods. Motivated by the goal of addressing more general inverse problems, Diffusion Posterior Sampling (*DPS*) [17] uses Bayes’ Rule to sample from $p_\theta(z_{t-1}|z_t, y) \propto p_\theta(z_{t-1}|z_t)p_\theta(y|z_{t-1})$. Instead of directly blending or replacing images with noisy versions of the observation, DPS uses the gradient of the likelihood $\log p_\theta(y|z_t)$ to guide the generative process at every denoising step t . Since computing $\nabla_{z_t} \log p(y|z_{t-1})$ is intractable due to the integral over all possible configurations of $z_{t'}$ for $t' < t-1$, DPS approximates $p(y|z_{t-1})$ using a one-step denoised prediction $\hat{x}$ using Eq. (4.3). The likelihood $p(y|x) = \mathcal{N}(f(x), \sigma_v^2)$ can then be evaluated using these approximate predictions. To obtain the gradient of the likelihood term, DPS require backpropagating gradients through the denoising network used to predict $\hat{x}$.

Specializing to image inpainting, *CoPaint* [137] augments the likelihood with another regularization term to generate samples z_{t-1} that prevent taking large update steps away from the previous sample z_t , in an attempt to produce more coherent images. Further, this work also proposes CoPaint-TT, which additionally uses the time-travel trick to reduce discontinuities in sampled images.

Originally designed for pixel-space diffusion models, it is difficult to adapt these works directly to latent diffusion models. Posterior Sampling with Latent Diffusion (*PSLD*) [101] first showed that employing *DPS* directly on latent space diffusion models produces blurry images. It proposes to add another “gluing” term to the measurement likelihood which pe-

nalizes samples z_t that do not lie in the encoder-decoder shared embedding space. However, this may produce artifacts in the presence of measurement noise (see [108]). To address this issue, recent concurrent work on the *ReSample* [108] method divides the timesteps in the latent space into 3 subspaces, and optimizes samples z_t in the mid-subspace to encourage samples that are more consistent with observations. Other work [133] highlights a 3-stage approach where data consistency can be enforced in the latter 2 stages which are closer to $t = 0$.

Below, we elaborate Red-Diff, a variational inference approach closest to our work.

4.3.2 Red-Diff: Variational Posterior over data x

RedDiff [78] uses pixel-based diffusion models as priors and defines a variational distribution to approximate $p_\theta(x | y)$. It defines a simple Gaussian variational distribution over the data space x as $q_\lambda(x) = \mathcal{N}(\mu, \sigma^2)$, where $\lambda = \{\mu, \sigma\}$ and both μ, σ are defined per pixel. It further assumes a small constant variance $\sigma^2 \approx 0$, reducing the posterior approximation to *an image* μ that minimizes the KL divergence:

$$D(q_\lambda(x) || p(x|y)) = -\log p(y|\mu)] + D(q_\lambda(x) || p_\theta(x)) \quad (4.6)$$

Thus, RedDiff seeks an image μ that reconstructs the observation y according to the given mask m , while having a high likelihood under the diffusion prior (second term).

The second term acts as a regularizer and comprises of an expectation that averages over many diffusion time steps. In practice the authors find optimization with such a loss to be difficult. Moreover, the authors observe that annealing time from $t = T$ to $t = 0$, as in standard backward diffusion samplers, yields better performance rather than directly optimizing the variational bound through random time sampling. Some visual examples are provided in Fig. 4.1 for a comparison between RedDiff-V, which uses random-time sampling as justified

by the correct variational bound, and RedDiff which employs descending time from T to 0. Furthermore, RedDiff does not propagate gradients over the denoising network $\epsilon_\theta(z_t, t)$ to prevent optimization instability, which we hypothesize may arise due to the denoising function’s lack of smoothness at low-noise levels [128].

Because Red-Diff employs a simple variational posterior that directly optimizes an image at the noise free ($t = 0$) level only, it is inherently incapable of capturing uncertainty in x , due to its mode-seeking behaviour. Additionally, its optimization process is *biased* because it relies on annealing time during the diffusion process rather than randomly sampling time points. We demonstrate that in contrast, our VIPaint framework better models posterior uncertainty, enables stable optimization of an unbiased variational bound, and can be applied to both pixel-based and latent DMs.

4.4 VIPaint

Given a pre-trained diffusion model, VIPaint infers the *latent* distribution over z_t induced by a test observation y . VIPaint constructs a hierarchical posterior over a subset of mid-ranged latents, $z_t \in [T_s, T_e]$, where $0 < T_s < T_e < T$; optimizes its parameters λ using a variational bound; and employs iterative gradient-based refinements to the posterior samples in the low-range latent space $[0, T_s]$ to produce final inpaintings. An overview is provided in Fig. 4.2.

This technique offers several advantages over RedDiff, as the latent-space hierarchical posterior: 1) infers high-to-low global semantics in the latent space, induced by a corrupted image y ; 2) accounts for uncertainty in the missing regions; and 3) strategically avoids the training instabilities observed in the low-ranged latent space $[0, T_s]$ [128] and 4) is easily extended to latent diffusion models. Below we elucidate the posterior formulation, optimization and

sampling strategies.

Posterior Formulation VIPaint defines the latent-space hierarchical posterior as follows:

$$q_\lambda(z_{T_s:T_e}) = \left(\prod_{i=1}^{K-1} q_\lambda(z_{s(i)} \mid z_{s(i+1)}) \right) q_\lambda(z_{T_e}) \quad (4.7)$$

where $K \geq 2$ and $s(i)$ denotes the timestep preceding $s(i+1)$ for all $i \in [1, K-1]$, for instance $s(1) = T_s$, $s(K) = T_e$. Experiments suggest these intermediate timesteps to achieve a SNR (α_t^2/σ_t^2) in the range $[0.2, 0.5]$ across different (latent) diffusion models. For the highest timestep T_e , we let $q_\lambda(z_{T_e})$ be a simple Gaussian $\mathcal{N}(\mu_{T_e}, \tau_{T_e})$ and the conditionals be defined as :

$$q_\lambda(z_{s(i)} \mid z_{s(i+1)}) = \mathcal{N}(z_{s(i)} \mid \gamma_{s(i)} \hat{z}_{s(i)} + (1 - \gamma_{s(i)}) \mu_{s(i)}, \tau_{s(i)}^2), \quad (4.8)$$

where $\tau_{s(i)}$ is the standard deviation and mean is a convex combination between the prior diffusion model prediction $\hat{z}_{s(i)} = \hat{z}_{s(i)}(\theta, z_t, t)$, and a contextual variational parameter, $\mu_{s(i)}$. Previous work [112, 76, 56, 108] uses linear combinations between the observed y and generated sample z_t , but use either hard constraints or fixed weights that are manually tuned. Instead, we incorporate free parameters $\lambda = \{\mu_{T_e}, \tau_{T_e}, (\gamma_{s(i)}, \mu_{s(i)}, \tau_{s(i)})_{i=1}^{K-1}\}$ across K latent levels, defined over each pixel in the image or its encoding. Such a flexible posterior is key when reusing the diffusion prior *and* align precisely with a particular observation y , without the need to re-train θ . We use y to initialize $\mu_{s(i)}$ by first encoding it using the encoder and then scaling it by the forward diffusion parameter $\alpha_{s(i)}$ and use the noise schedule to initialize our posterior variance.

Fitting the Posterior To fit our hierarchical posterior, we optimize the variational lower bound (VLB) of the marginal likelihood of the observation y . The derivation is provided in

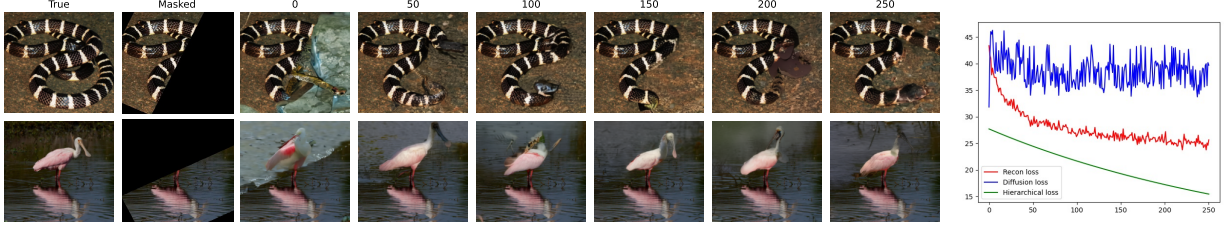


Figure 4.3: We show VIPaint’s posterior fitting progress and sample generation every 50 iterations for two test cases. We see VIPaint quickly grasps the image semantics within 50 optimization iterations.

[3], and the simplified three-term objective is expressed as follows:

$$L(\lambda) = \underbrace{-\mathbb{E}_q[\log p_\theta(y|z_{T_s})]}_{\text{reconstruction loss}} + \beta \underbrace{\left(\sum_{i=1}^{K-1} D[q_\lambda(z_{s(i)}|z_{s(i+1)})||p_\theta(z_{s(i)}|z_{s(i+1)})] \right)}_{\text{hierarchical loss}} + \underbrace{\mathcal{L}_{(T_e, T)}(z_{T_e})}_{\text{diffusion loss}}, \quad (4.9)$$

VIPaint seeks latent-posterior distributions that assign high likelihood to the observed features y (by minimizing the reconstruction loss), while simultaneously aligning with the medium-to-high noise levels encoding image semantics (hierarchical and diffusion losses) via weight $\beta > 1$ [43, 4]. We approximate $L(\lambda)$ with M Monte Carlo samples from our hierarchical posterior $q_\lambda(z_{T_s:T_e})$; we follow ancestral sampling to draw $z_{T_e}^{(m)} \sim q_\lambda(z_{T_e})$, $\{z_{s(i)}^{(m)} \sim q_\lambda(z_{s(i)}|z_{s(i+1)}^{(m)})\}_{i=1}^{K-1}$. We evaluate $L(\lambda)$ as below and use automatic differentiation to compute gradients with respect to λ .

$$\frac{1}{M} \sum_{m=1}^M \left[\underbrace{-\log p_\theta(y|z_{T_s}^{(m)})}_{\text{reconstruction loss}} + \beta \underbrace{\left(\sum_{i=1}^{K-1} D[q_\lambda(z_{s(i)}|z_{s(i+1)}^{(m)})||p_\theta(z_{s(i)}|z_{s(i+1)}^{(m)})] \right)}_{\text{hierarchical loss}} + \underbrace{\mathcal{L}_{(T_e, T)}(z_{T_e}^{(m)})}_{\text{diffusion loss}} \right] \quad (4.10)$$

Reconstruction Loss. This term guides the samples from the posterior $z_{T_s}^{(m)}$ to be closer to the observations y . We employ a one-step expected mean prediction $\mathbb{E}[\hat{z}_0^{(m)}|z_{T_s}^{(m)}]$ as in

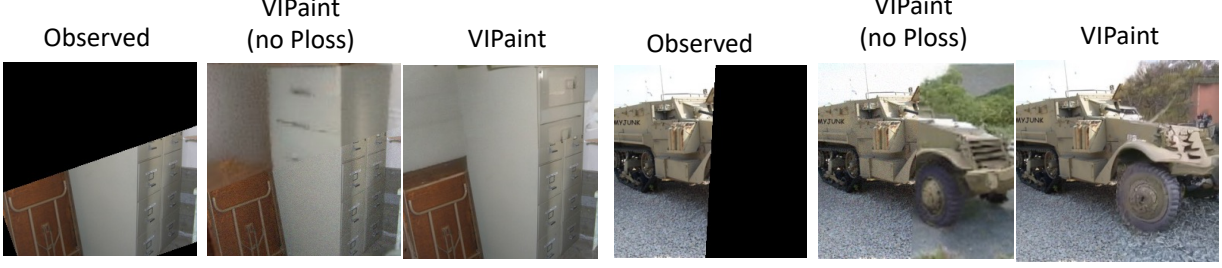


Figure 4.4: An ablation showing the effect of addition the perceptual loss (PLoss) in the reconstruction term for the task of image inpainting using latent diffusion priors. We see that even though VIPaint can inpaint the image semantically without the Perceptual loss, this loss becomes important to produce sharper reconstructions.

Eq. (4.3) to approximate z_0 . Because T_s is closer to $t = 0$, this approximation is accurate enough to guide samples z_{T_s} to be consistent with y . In case of latent diffusion models, we use decoder upsampling to produce image $\hat{x}$. We use the L1 reconstruction loss, and add a perceptual loss term [138] in case of Latent Diffusion Models. This loss was originally used to train the decoder, and Fig. 4.4 shows an ablation that adding such a term helps avoid blurry reconstructions.

Diffusion Loss. The diffusion loss is derived in [3] and write the simplified expression:

$$\mathcal{L}_{(T_e, T)}(z_{T_e}^{(m)}) = \frac{T - T_e}{2} \mathbb{E}_{t \sim \mathcal{U}(T_e, T), q(z_t | z_{T_e}^{(m)})} D[q(z_{t-1} | z_t, z_{T_e}^{(m)}) || p_\theta(z_{t-1} | z_t)]. \quad (4.11)$$

where the expectation is over t uniformly sampled in $[T_e, T]$ (instead of the entire diffusion timesteps), $z_t \sim q(z_t | z_{T_e})$ and $z_{T_e} \sim q_\lambda(z_{T_e})$. In other words, this loss term regularizes the samples $z_{T_e}^{(m)}$ in high-level image semantics encoded in the latent range of $[T_e, T]$. Following prior work, instead of summing this loss over all $t > T_e$, we sample timesteps $t \sim \mathcal{U}(T_e, T)$ defined on a non-uniform discretization [53], yielding an unbiased estimate of the loss.

Hierarchical Loss. The KL terms across the $K - 1$ intermediate times in the hierarchy are computed in closed form (an analytic function of the means and variances) between the posterior and prior conditional Gaussian distributions. Intuitively, these terms further

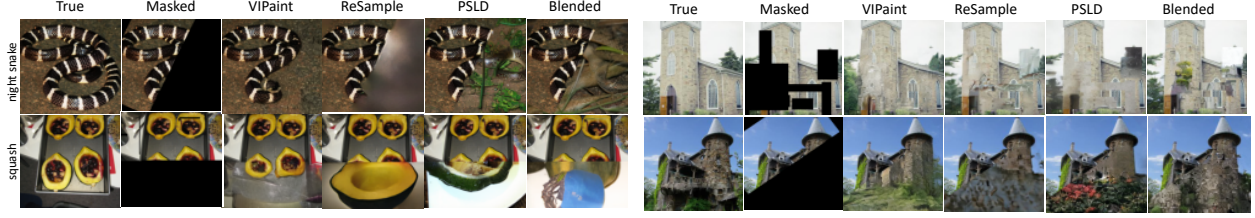


Figure 4.5: Image completion results using the LDM prior for Imagenet256 (left) and LSUN (right) for large-mask inpainting (Random Masking and Rotated Window schemes). We show inpaintings from each method in the following columns. DPS, PSLD, and ReSample produce blurry inpaintings of varying quality. Despite being conditioned on class labels, baseline methods’ inpaintings for ImageNet are inconsistent with the observed image. In contrast, VIPaint captures global semantics from the observed image, producing highly realistic inpaintings.

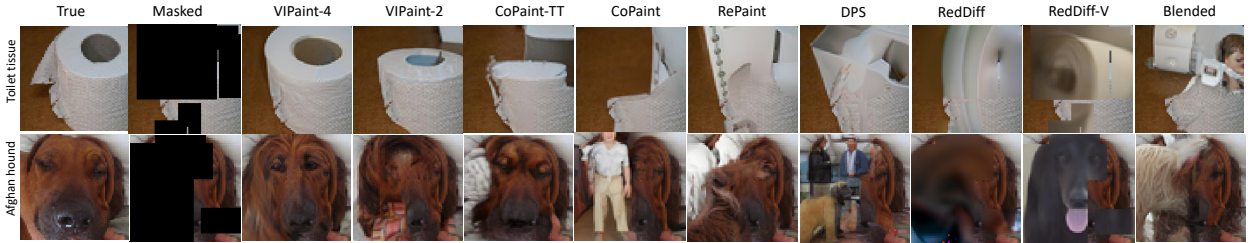


Figure 4.6: Image completion results using the pixel-based EDM model for ImageNet64 with for large-mask inpainting (Random Masking and Rotated Window schemes). We show inpaintings from each method in the following columns. Posterior sampling methods for pixel-based diffusion priors, like RePaint and CoPaint, are generally more accurate but still produces inconsistent samples. In contrast, VIPaint infers the underlying image correctly, and produces highly realistic inpaintings.

regularizes posterior samples $\{z_{s(i)}^{(m)}\}_{i=1}^{K-1}$ to capture high-to-medium level image details in the mid-ranged $[T_s, T_e]$ diffusion space.

Hence, all the loss terms in Eq. (4.9) are stochastically and differentiably estimated based on samples from the hierarchical posterior, enabling joint optimization. Progress in fitting VIPaint’s posterior is shown in Fig. 4.3; the number of optimization steps may be reduced to more quickly give approximate posteriors. If the posterior is only defined on the noise-free level z_0 as in Red-Diff [78], the VIPaint objective (Eq. (4.9)) reduces to an objective mentioned in their work. However, VIPaint strategically avoids low noise levels in its posterior, decreases training instabilities as observed by RedDiff and extends to Latent Diffusion priors.

Sampling. After optimization, the hierarchical posterior $q_\lambda(z_{T_s:T_e})$ is now *semantically* aligned with the observation. We employ ancestral sampling on our K level hierarchical posterior starting from T_e to T_s , to yield samples z_{T_s} as shown in Fig. 4.2. This step gradually adds semantic details in samples. Further, VIPaint refines z_{T_s} using the prior denoising model at every step $t < T_s$. Similar to DPS [17], we update the samples using gradient of the likelihood $\log p_\theta(y | z_t), t < T_s$. This ensures fine-grained consistency to our final inpaintings.

Time Complexity. The time taken by VIPaint- K scales primarily with the number of denoising network calls. Each optimization step for a K -step posterior ($K \ll T$) involves $O(K)$ calls to sample $z_{T_s} \sim q_\lambda(z_{T_s:T_e})$, and one to compute the diffusion prior loss, resulting in $O(K)$ function calls per step. Thus, for I optimization steps, the overall complexity is $O(KI)$. For example, our VIPaint-2 is optimized over 50 iterations, it requires only $50 * (2 + 1) = 150$ denoising network calls to infer global image semantics. Once fit, sampling requires an additional T_s ($T_s = 400$ for LDMs) denoising network calls per sample instead of T ($T = 1000$) network calls in traditional sampling methods.

4.5 Experiments

4.5.1 Experimental Setup

We conduct experiments across 3 image datasets: LSUN-Church [132], ImageNet-64 and ImageNet-256 [29]. For ImageNet-64, we use the pre-trained class-conditioned pixel-space "EDM" diffusion model [53]; for LSUN-Churches256 and ImageNet256 we use the pre-trained latent diffusion models from [98]. Then, we sample 100 non-cherry-picked test images across the three datasets. We consider three masking patterns: 1) a small mask distribution [139]

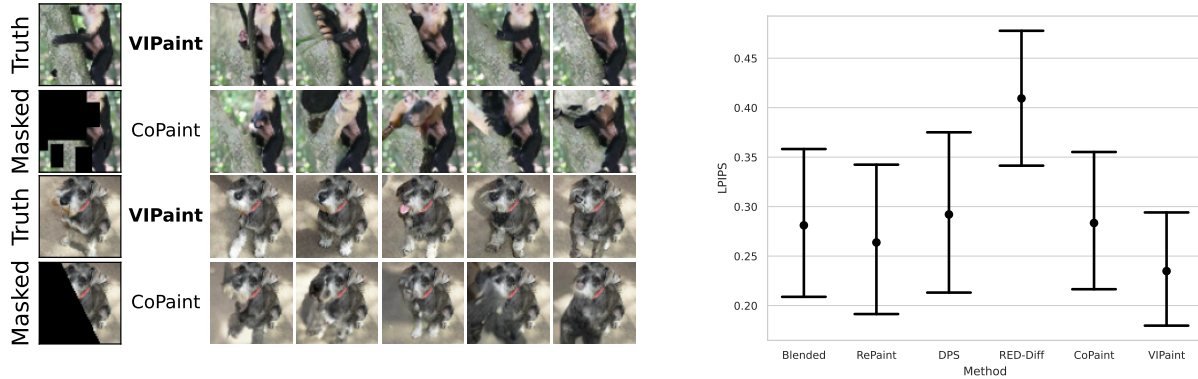


Figure 4.7: **Left:** Sample completions comparing VIPaint with the best performing baseline, CoPaint, for two test images. We show the true and masked images, and 5 in-painted samples for each method. VIPaint yields coherent samples while capturing uncertainty in the missing pixels in images. In comparison, CoPaint has high variance in the quality of results. **Right:** We compute summary statistics (minimum, mean, maximum) of the LPIPS score across 100 sampled completions per test image. We show the average value of each of these statistics across the test set. We see that VIPaint improves on baselines, both in terms of the average quality of results *and* in the consistency in result quality.

Task	VIPaint-4	VIPaint-2	CoPaint-TT	CoPaint	RePaint	DPS	Blended	RedDiff	RedDiff-V
Rotated Window	0.289	<u>0.300</u>	0.316	0.347	0.3213	0.3203	0.3409	0.463	0.407
Random Mask	<u>0.231</u>	0.227	0.245	0.278	0.2575	0.2880	0.2763	0.409	0.671

Task	Imagenet-256					LSUN-Church			
	VIPaint-4	VIPaint-2	ReSample	PSLD	DPS	VIPaint-2	ReSample	PSLD	DPS
Rotated Window	0.358	<u>0.392</u>	0.537	0.576	0.606	0.455	<u>0.510</u>	0.541	0.502
Random Mask	0.373	<u>0.409</u>	0.559	0.583	0.607	0.439	<u>0.485</u>	0.523	0.490
Small Mask	<u>0.292</u>	0.197	0.381	0.534	0.564	0.299	<u>0.374</u>	0.413	0.421

Table 4.1: Quantitative results (LPIPS, lower is better) for ImageNet64 for the task of image inpainting using pixel-based EDM prior (*top*) and Imagenet-256 and LSUN-Church using LDM priors (*bottom*). LPIPS is estimated as the mean score of 10 inpaintings with respect to the true image, averaged across the test set. VIPaint has superior performance (highlighted in **bold**) in nearly all cases. We underline the second best method.

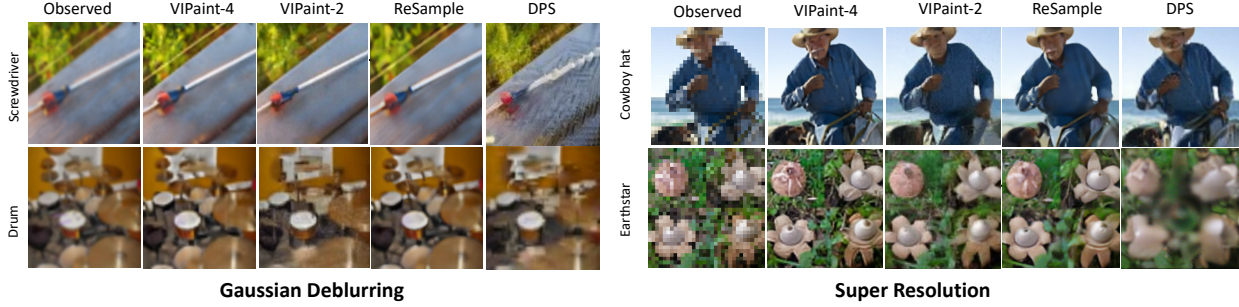


Figure 4.8: Qualitative results on other linear inverse problems : Gaussian Deblurring (**left**) and Super Resolution (**right**) for Imagenet256 using LDM prior. VIPaint leads to sharp, realistic samples without any artifacts.

that masks up to 20% of each image 1) a random mask distribution that uses a similar setup masking 40-80% of each image, and 2) a randomly rotated masking window that masks at least half of the image. By masking large portions of each image we ensure a sufficiently challenging benchmark for inpainting. Acknowledging the pluralistic outcomes in this setting, we evaluate each method across 10 reconstructions per test image, totalling 1000 inpaintings. We use the notation $VIPaint-K$ to denote the number of steps in the hierarchical posterior in our experiments. We found empirically that discretizations and hyperparameters of VIPaint translate well between models using the same noise schedule (as shown for the LSUN and ImageNet-256 latent diffusion models). Please see the appendix 4.7 for more details.

Comparison. We compare VIPaint with several recent methods that directly apply the diffusion models trained in the pixel space: *i*) blending methods: *blended* [112] and *RePaint* [76]; *ii*) Sampling methods: *DPS* [17], and *CoPaint* [137] and *iii*) variational approximations: *RED-Diff* [78]. Although not exhaustive, this set of methods summarizes recent developments in the state-of-the-art for image inpainting. For latent diffusion models, we compare VIPaint with *DPS*, *PSLD* [101] and *ReSample* [108]. Since large masks in images can induce high-uncertainty in the image, Peak-Signal-To-Noise-Ratio (PSNR) is not very well defined for this task. Hence, we report the Learned Perceptual Image Patch Similarity (LPIPS) [138] metric in Table 4.1 which computes how close the inpaintings are to the true image in an embedding space. We show qualitative images across methods and datasets in Fig. 4.5 and

4.6. For tasks like super-resolution and Gaussian Deblurring, we show qualitative results in Fig. 4.8. Additionally, we visualize multiple inpaintings in Fig. 4.7.

Selecting VIPaint’s hyperparameters. Extensive prior work [112, 89, 30, 53] explores different noise schedules for training diffusion models, and how it affects the generated image quality. Since we use these diffusion models incorporating different noise schedules, our latent hierarchical posterior needs to account for this shift, and we show that it is flexible to do so. To concentrate posterior inference on the noise levels which are most crucial to perceptual image quality, we define our posterior at intermediate time steps that induce a signal-to-noise-ratio $(\alpha_t^2/\sigma_t^2) \in [0.2, 0.5]$, [62]) approximately across our experiments. This corresponds to choosing $(T_e = 5, T_s = 2)$ for the pixel-based EDM prior with a variance-exploding noise schedule and $(T_e = 550, T_s = 400)$ (DDIM sampling coefficient, $\eta = 0.2$) for the LDM prior using the VP noise schedule for both LSUN and ImageNet256 datasets. VIPaint is not sensitive to any subset of K timesteps in between this signal-to-noise range. For instance, for VIPaint-4, we take $[T_e = 5, 4, 3.5, 2.5, T_s = 2]$ for the EDM noise schedule and $[T_e = 550, 500, 450, T_s = 400]$ for the LDM prior. Additionally, we use $\beta = 1$ for VIPaint with $K = 2$, which is optimized for 50 iterations for faster convergence. For VIPaint with $K = 4$, we use $\beta = 50$ for pixel-based EDM prior and $\beta = 10$ for LDM prior. Generally speaking, higher values of β explores the diffusion latent space more and takes more optimization steps to reach to a solution. However, lower values of β weighs the likelihood term relatively more and converges to a solution *faster*.

4.5.2 Results

VIPaint enforces consistency with large masking ratios. Table 4.1 reports LPIPS scores for the task of image inpainting with large masking ratios using pixel and latent-based diffusion models, respectively. For pixel-based diffusion models, we see that RED-Diff and DPS perform poorly. RePaint, CoPaint and CoPaint-TT show relatively better scores,

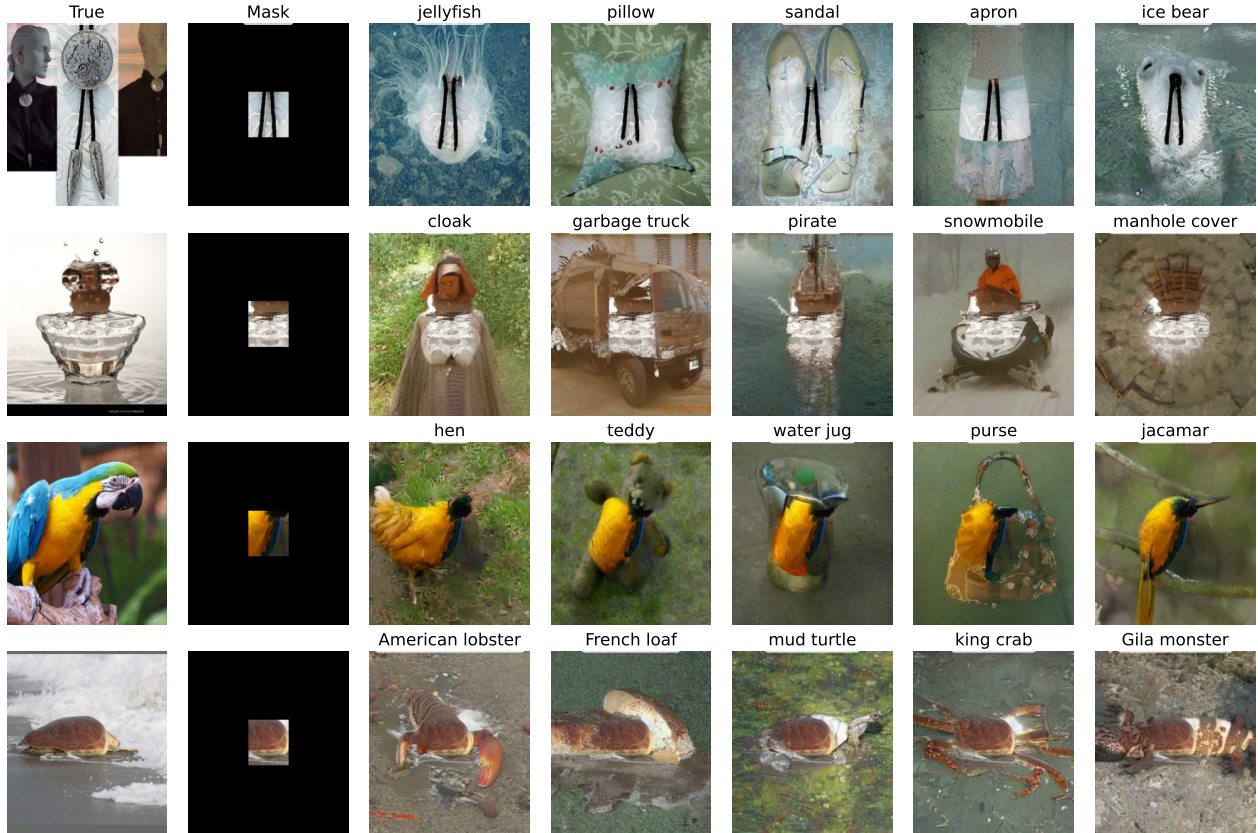


Figure 4.9: Qualitative results for VIPaint diversity for ImageNet256 with LDM prior using different class conditioning. We see that VIPaint follows the input label and ensures consistency with the observed set of pixels.

but do not match the performance of VIPaint across any dataset or masking pattern. We show imputations for multiple test examples in Fig. 4.5, 4.6 to highlight differences in inference methods. We see that VIPaint consistently produces plausible inpaintings while other methods fail to complete images for larger masking ratios meaningfully.

VIPaint yields multiple plausible reconstructions in the case of high uncertainty.

We compare VIPaint with the best performing baseline, CoPaint across multiple sample inpaintings in Fig. 4.7. We observe that VIPaint produces multiple visually-plausible imputations while not violating the consistency across observations. We show diversity in possible imputations using different class conditioning using VIPaint in Fig. 4.9.

Computational Efficiency. We report the time taken to produce 10 inpaintings for one test

Sampling	RedDiff	VIPaint-2	VIPaint-4
5.4	1.13	3.3	11.8
7.0	–	6.4	10

Table 4.2: Table comparing time (in mins) for three inference methods using EDM prior (top) and LDM prior (bottom): Sampling (CoPaint-TT for pixel-based EDM model and PSLD for LDM prior), RedDiff and VIPaint

Task	ImageNet256				ImageNet64	
	Super-resolution 4x		Gaussian Deblur		Gaussian	Deblur
Metric	LPIPS ↓	PSNR ↑	LPIPS ↓	PSNR ↑	LPIPS ↓	PSNR ↑
VIPaint-4	0.33	19.31	<u>0.44</u>	<u>17.90</u>	0.306	13.47
VIPaint-2	0.46	16.36	0.48	16.35	0.305	13.60
ReSample	<u>0.395</u>	<u>18.410</u>	0.435	18.03	–	–
PSLD	0.67	7.77	0.583	0.022	–	–
DPS	0.579	12.99	0.595	12.608	0.319	13.43

Table 4.3: Quantitative results (LPIPS, PSNR) for solving linear inverse problems on ImageNet256 using LDM priors and ImageNet64 using EDM priors. Best results are in bold and second best results are underlined. For nonlinear deblurring.

image using: the best performing sampling method: CoPaint-TT (pixel-based EDM prior) and PSLD (LDM prior), RedDiff (pixel-space variational posterior) and VIPaint (latent space variational posterior) in Table 4.2. RedDiff is fast but inconsistent and unsuitable for latent diffusion priors. Sampling methods are slower, produces better inpaintings than RedDiff but still shows inconsistencies. VIPaint-2 is faster than sampling-based methods for both pixel and latent DMs, *and* achieves better results (see Table 4.1). Moreover, VIPaint smoothly trades off time and sample quality, with VIPaint-4 converging more slowly but ultimately yielding the best solutions.

VIPaint extends to general linear inverse problems. We show some qualitative examples in Fig. 4.8 and a detailed quantitative analysis is reported in Table 4.3. In addition to the LPIPS scores, we also compute the PSNR metrics, averaged over 10 random samples

for each of 100 test images. We see that VIPaint shows strong advantages over ReSample and DPS for complex image datasets like ImageNet.

4.6 Conclusion

We present VIPaint, a simple and a general approach to adapt diffusion models for image inpainting and other inverse problems. We take widely used (latent) diffusion generative models, allocate variational parameters for the latent codes of each partial observation, and fit the parameters stochastically to optimize the induced variational bound. The simple but flexible structure of our bounds allows VIPaint to outperform previous sampling and variational methods when uncertainty is high, yielding valid and multiple plausible image completions.

VIPaint, however, is sensitive to the design choices made by the underlying diffusion model. Since the inference noise schedule directly affects the selection of T_s and T_e , the design choices of VIPaint may vary when paired with a different diffusion prior. In this work, we demonstrate that these choices remain consistent when using the same diffusion prior (e.g., LDM) across different datasets (such as LSUN256 and ImageNet256).

4.7 Appendix

4.7.1 VIPaint: VI method using Diffusion Models as priors

Derivation of VIPaint’s Optimization Objective

As specified in the main section, we define a variational distribution over the latent space variable z as $q_\lambda(z)$ and re-use the diffusion prior to generate $x \sim p_\theta(x \mid z)$. We derive the variational objective here:

$$\begin{aligned}
\mathcal{L}_N(\lambda; y) &= \mathbb{E}_{q_\lambda(z, x)}[\log p_\theta(y, x, z) - \log q_\lambda(z, x \mid y)] \\
&= \mathbb{E}_{q_\lambda(z, x)}[\log p_\theta(z) + \log p_\theta(x \mid z_{T_s}) + \log p_\theta(y \mid z_{T_s}) - \log q_\lambda(z) - \log q_\lambda(x \mid z_{T_s})] \\
&= \mathbb{E}_{q_\lambda(z)}[\log p_\theta(y \mid z_{T_s}) + \log p_\theta(z) - \log q_\lambda(z)] \\
&= \mathbb{E}_{q_\lambda(z)}[\log p_\theta(y \mid z_{T_s})] - \mathbb{E}_{q_\lambda(z)}[\log q_\lambda(z) - \log p_\theta(z)] \\
&= \mathbb{E}_{q_\lambda(z)}[\log p_\theta(y \mid z_{T_s})] - \underbrace{\mathbb{E}_{q_\lambda(z)}[\log q_\lambda(z) - \log p_\theta(z)]}_{\text{second term}} \tag{4.12}
\end{aligned}$$

The second term can be further decomposed as:

$$\begin{aligned}
&= \mathbb{E}_{q_\lambda(z)}\left[\sum_{i=0}^{K-1} \log q_\lambda(z_{s(i)} \mid z_{s(i+1)}) + \log q_\lambda(z_{T_e}) - \sum_{i=0}^{K-1} \log p_\theta(z_{s(i)} \mid z_{s(i+1)}) - \log p_\theta(z_{T_e})\right] \\
&= \sum_{i=0}^{K-1} D[q_\lambda(z_{s(i)} \mid z_{s(i+1)}) \parallel p_\theta(z_{s(i)} \mid z_{s(i+1)})] - \underbrace{D(q(z_{T_e}) \parallel p(z_{T_e}))}_{\mathcal{L}_{(T_e, T)}(z_{T_e})} \tag{4.13}
\end{aligned}$$

Finally, $\mathcal{L}_N(\lambda; y)$

$$= \mathbb{E}_{q_\lambda(z)}[\log p_\theta(y \mid z_{T_s})] - \sum_{i=0}^{K-1} D[q_\lambda(z_{s(i)} \mid z_{s(i+1)}) \parallel p_\theta(z_{s(i)} \mid z_{s(i+1)})] - \underbrace{D(q(z_{T_e}) \parallel p(z_{z_{T_e}}))}_{\mathcal{L}_{(T_e, T)}(z_{T_e})} \quad (4.14)$$

Negating the above objective, we get Eq. 4.9 in the main section. Now, let's derive the third term $L_{(T_e, T)}(z_{T_e})$ following section 4.7.2.

4.7.2 Derivation of $L_{(T_e, T)}(z_{T_e})$

For any $T_e < s < t < T$, we have :

$$\mathbb{E}_{z_{T_e} \sim q_\lambda(z_{T_e})} \left[\log \frac{q(z_{T_e+1:T} \mid z_{T_e})}{p_\theta(z_{T_e:T})} \right] \quad (4.15)$$

$$= \mathbb{E}_{z_{T_e} \sim q_\lambda(z_{T_e})} \left[-\log p(z_T) + \sum_{t \geq T_e} \log \frac{q(z_t \mid z_s)}{p_\theta(z_s \mid z_t)} \right] \quad (4.16)$$

$$= \mathbb{E}_{z_{T_e} \sim q_\lambda(z_{T_e})} \left[-\log p(z_T) + \sum_{t > T_e} \log \frac{q(z_t \mid z_s)}{p_\theta(z_s \mid z_t)} + \log \frac{q(z_{T_e+1} \mid z_{T_e})}{p_\theta(z_{T_e} \mid z_{T_e+1})} \right] \quad (4.17)$$

$$= \mathbb{E}_{z_{T_e} \sim q_\lambda(z_{T_e})} \left[-\log p(z_T) + \sum_{t > T_e} \log \frac{q(z_s \mid z_t, z_{T_e})}{p_\theta(z_s \mid z_t)} \cdot \frac{q(z_t \mid z_{T_e})}{q(z_s \mid z_{T_e})} + \log \frac{q(z_{T_e+1} \mid z_{T_e})}{p_\theta(z_{T_e} \mid z_{T_e+1})} \right] \quad (4.18)$$

$$= \mathbb{E}_{z_{T_e} \sim q_\lambda(z_{T_e})} \left[-\log \frac{p(z_T)}{q(z_T \mid z_{T_e})} + \underbrace{\sum_{t > T_e} \log \frac{q(z_s \mid z_t, z_{T_e})}{p_\theta(z_s \mid z_t)}}_{\text{diffusion loss } \mathcal{L}_{(T_e, T)}(z_{T_e})} - \log p_\theta(z_{T_e} \mid z_{T_e+1}) \right] \quad (4.19)$$

The first and third term can be stochastically and differentially estimated using standard techniques. Following [62], we derive an estimator for the diffusion loss $\mathcal{L}_{(T_e, T)}(z_{T_e})$. In the

case of finite timesteps $t > T_e$, this loss is:

$$\mathcal{L}_{(T_e, T)}(z_{T_e}) = \sum_{t > T_e} \mathbb{E}_{q(z_t | z_{T_e})} D[q(z_s | z_t, z_{T_e}) || p_\theta(z_s | z_t)] \quad (4.20)$$

Estimator of $\mathcal{L}_{(T_e, T)}(z_{T_e})$ Reparametering $z_t \sim q(z_t | z_{T_e})$ as $z_t = \alpha_{t|T_e} z_{T_e} + \sigma_{t|T_e} \epsilon$, where $\epsilon \sim \mathcal{N}(0, 1)$, and to avoid having to compute all $T - T_e$ terms when calculating the diffusion loss, we construct an unbiased estimator of $\mathcal{L}_{(T_e, T)}(z_{T_e})$ using

$$\mathcal{L}_{(T_e, T)}(z_{T_e}) = \frac{T - T_e}{2} \mathbb{E}_{\epsilon, t \sim \mathcal{U}(T_e, T)} [D(q(z_s | z_t, z_{T_e}) || p_\theta(z_s | z_t))] \quad (4.21)$$

where $\mathcal{U}(T_e, T)$ is a uniform distribution to sample $T_e < t \leq T$ from a non-uniform discretization of timesteps using [53].

Now, we elaborate on the expression $q(z_s | z_t, z_{T_e})$ and $p(z_s | z_t)$ for any $T_e < s < t < T$.

1. $q(z_s | z_t, z_{T_e})$: Our posterior at T_e is $q(z_{T_e}) = \mathcal{N}(\mu_{T_e}, \tau_{T_e}^2)$. For any $T_e < s < t < T$, we have $q(z_s | z_{T_e}) = \mathcal{N}(\alpha_{s|T_e} z_{T_e}, \tau_{s|T_e}^2)$ and $q(z_t | z_s) = \mathcal{N}(\alpha_{t|s} z_s, \sigma_{t|s}^2)$, yielding the posterior :

$$q(z_s | z_t, z_{T_e}) = \mathcal{N}(\mu_Q(z_t, z_{T_e}; s, t, T_e), \sigma_Q^2(s, t, T_e) I) \quad (4.22)$$

$$\text{where, } \sigma_Q^2(s, t, T_e) = \sigma_{t|s}^2 \frac{\tau_{s|T_e}^2}{\sigma_{t|s}^2 + \alpha_{s|T_e}^2 \tau_{s|T_e}^2} \quad (4.23)$$

$$\text{and, } \mu_Q(z_t, z_{T_e}; s, t, T_e) = \sigma_Q^2 \left(\frac{\alpha_{s|T_e}}{\tau_{s|T_e}^2} z_{T_e} + \frac{\alpha_{t|s}}{\sigma_{t|s}^2} z_t \right) \quad (4.24)$$

$$= \frac{\alpha_{s|T_e} \sigma_{t|s}^2}{(\sigma_{t|s}^2 + \alpha_{s|T_e}^2 \tau_{s|T_e}^2)} z_{T_e} + \frac{\alpha_{t|s} \tau_{s|T_e}^2}{(\sigma_{t|s}^2 + \alpha_{s|T_e}^2 \tau_{s|T_e}^2)} z_t \quad (4.25)$$

2. $p(z_s|z_t)$: The conditional model distributions can be chosen as:

$$p_\theta(z_s|z_t) = q(z_s|z_t, z_{T_e} = \hat{z}_{\theta, T_e}(z_t, t)) = \mathcal{N}(z_s; \mu_\theta(z_t, z_{T_e}; s, t, T_e), \sigma_Q^2(s, t, T_e)) \quad (4.26)$$

$$\text{where, } \mu_\theta(z_t, z_{T_e}; s, t, T_e) = \frac{\alpha_{s|T_e} \sigma_{t|s}^2}{(\sigma_{t|s}^2 + \alpha_{s|T_e}^2 \tau_{s|T_e}^2)} \hat{z}_{\theta, T_e}(z_t, t) + \frac{\alpha_{t|s} \tau_{s|T_e}^2}{(\sigma_{t|s}^2 + \alpha_{s|T_e}^2 \tau_{s|T_e}^2)} z_t \quad (4.27)$$

$$\text{and, } \sigma_Q^2(s, t, T_e) = \sigma_{t|s}^2 \frac{\sigma_{s|T_e}^2}{\sigma_{t|s}^2 + \alpha_{s|T_e}^2 \sigma_{s|T_e}^2} \quad (4.28)$$

$$\text{where } \hat{z}_{\theta, T_e}(z_t, t) = \frac{z_t - \sigma_{t|T_e} * \epsilon_\theta(z_t, t)}{\alpha_{t|T_e}}$$

Chapter 5

Conclusion and Future Directions

Previous chapters developed effective hierarchical variational inference algorithms for various deep generative models: discrete graphical models, VAEs and Diffusion Models. We now review the main contributions made in this thesis, and outline a few open areas for future research.

5.1 Summary of Methods and Contributions

Graphical models provide abundant tools to describe the relations within high-dimensional data. As the nodes in the graphical model increase and relations are expressed by deep neural networks, the need for efficient inference techniques becomes a critical issue. We propose a family of variational inference algorithms that approximate message-passing inference algorithms for discrete graphical models in Chapter 2, and approximate posteriors over latent variables in deep latent variable models of images like VAEs in Chapter 3, and Diffusion Models in Chapter 4.

For discrete graphical models, we employ message-passing inference algorithms and utilize

neural networks to approximate messages for more precise estimation. In Chapter 2, we tailor the specification and training of neural networks to the complexity of the messages they approximate, introducing a flexible function approximation framework. This chapter challenges the one-size-fits-all approach commonly used in prior work, which attempts to estimate a broad range of queries with varying complexity using a fixed neural network architecture and training scheme.

Moving on to deep latent variable models, which inherently employ deep neural networks to approximate dependencies between variables, Chapter 3 and 4 discusses different sampling, amortized and non-amortized variational inference algorithms for VAEs, and Diffusion Models respectively. While a majority of prior work trains or fine-tunes amortized inference networks to perform inpainting with (hierarchical) VAE priors (Chapter 3), a plethora of work employs sampling-based inference approaches using diffusion model priors (Chapter 4). We observe that the amortized schemes for VAEs undermodel the uncertainty in the posterior over missing pixels while the sampling schemes for diffusion models overmodels it. We instead fit a non-amortized variational posterior per inpainting query which provides sufficient flexibility to model the correct uncertainty in the missing pixels.

Our method scales well with the complexity in the masks, and presents meaningful image reconstructions even for *large masked* observations. Extensive experimentation with image datasets (MNIST, SVHN, FFHQ) reveal that amortized methods are sensitive to masking distributions that it is trained on, and sampling methods take a long time to converge and often do not lead to reasonable reconstructions. Moreover, the non-amortized variational methods that we propose outperform existing methods with respect to producing realistic image completions, and can handle a wide range of masking ratios at test-time without the need for re-training inference networks.

In summary, training/fine-tuning amortized inference networks for image inpainting is a feasible solution if the masking distribution is well-known beforehand. In other more real-world



Figure 5.1: We use the pre-trained conditional diffusion model from [98] that takes a masked observation (examples on the **left**) and their respective mask as input to the diffusion model, and produces an inpainting (shown on **right**) after the sequential denoising process. We see that for the large-mask image inpainting scenarios, this pre-trained diffusion model is unable to yield realistic images.

applications, when the test masking distribution is unknown, employing such techniques can be sub-optimal. The non-amortized approaches developed in this work are preferred in such cases. However, fitting such a flexible posterior over each inpainting query in general can be time- and memory-intensive. We leave making our non-amortized inference approach more compute-friendly to future work, which we elaborate on below.

5.2 Suggestions for Future Research

5.2.1 Faster non-amortized variational inference

In this thesis, we proposed non-amortized variational inference methods for the task of image inpainting using two widely known image generative models : VAEs and Diffusion Models that require fitting the posterior per inpainting query. While this inference is much faster than retraining an amortized inference model, it still has considerable computation requirements (in both memory and time). One way could be to employ fast samplers in

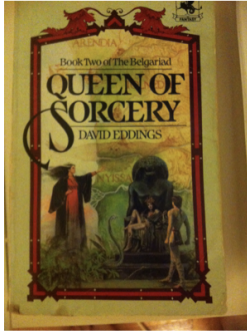


Figure 5.2: For the masked image on the **left** and the input prompt “Kitchen appliances and cabinets as seen through opening.” we present two possible image completions generated using the Stable Diffusion Inpainting model [98] on the **right**. These inpaintings exhibit artifacts, such as incomplete fridge handles and an incoherent water dispenser. To address these issues, one could use an inference algorithm like VIPaint on top of the Stable Diffusion model to produce more realistic and diverse inpaintings.

Diffusion Models [91] to accelerate the optimization and sampling stages in VIPaint using Diffusion priors. At the same time, there are many prior pieces of work that train models for the task of image inpainting [99, 72, 21, 113, 40] and predict inpaintings in one-shot, but lead to sub-optimal image reconstructions for various large-mask inpainting cases (see Fig. 5.1). We hypothesize that semi-amortized inference methods [58, 79] can prove to be a useful direction for faster, but still accurate, inference predictions.

5.2.2 Extension to Text-to-Image diffusion models

Text-to-image generation that enables personal content creation has recently gained significant attention [98, 103, 88, 94, 135]. While these generative models excel at this task, directly applying them for image inpainting requires highly specific input prompts, leaving little room for ambiguity. Typically, these models are used to inpaint a defined object with a clear mask outline [7]. To handle more complex and arbitrary masking patterns, recent advancements, including Stable Diffusion Inpainting [98], Glide [88], DALL-E [94], Smart-



The image shows a book cover for "Queen of Sorcery" by David Eddings, featuring a woman in a red dress gesturing towards a seated figure, with a man in white clothing standing nearby.



The image shows a book cover for "Queen Sorceress" by David Eddings, featuring a woman in a red dress and black cloak standing in a forest with a snake nearby.

Figure 5.3: We use the pre-trained MolMo model [28] that takes an image and produces its caption. On the **left**, we show the generated caption given a clean image, and on the **right**, we show the generated caption on its cropped version. You can see that the caption on the right incorrectly infers the caption of the book (without acknowledging uncertainty), and ignores description in the missing parts of the image.

Brush [126], and Imagen Editor [122] fine-tune these text-to-image models by incorporating additional inputs: an input mask, a masked image, and the corresponding text prompt, all fed into a conditional U-Net. This integration ensures that the masked image’s information is leveraged early in the noise prediction process, enabling more accurate semantics in the inpainting based on the observed content. However, we see from Fig. 5.2 that these fine-tuned models are unable to handle the case of large-mask image inpainting and leads to incoherent samples. Based on the success of VIPaint with unconditional and simple label-conditioned diffusion models, we hypothesize that the variational-based inference procedure would be effective in handling such large-mask inpainting queries using these text-to-image models (such as Stable Diffusion) as priors.

5.2.3 Other Application Areas

An application area for image captioning is for people with blindness or low vision (BLV) who often need human assistance to identify objects in images. This has long been done through crowd-sourcing techniques that employ people as on-demand visual interpreters, but today’s vision language models [28, 134] show promise for providing such support at any

time. However, images taken by BLV users may often have blurring artifacts or occlusions that prevent the models from generating the correct labels (see Fig. 5.3 for some examples). In addition, general reconstruction techniques may ignore consistency when reconstructing a portion of an image as seen in the previous chapter. Hence, a possible use case of VIPaint is to be able to reconstruct multiple plausible images to clean real-world degraded images, to make better use of off-the-shelf image-captioners for the BLV user community.

Bibliography

- [1] R. Abboud, I. Ceylan, and T. Lukasiewicz. Learning to reason: Leveraging neural networks for approximate dnf counting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 3097–3104, 2020.
- [2] S. Aeberhard and M. Forina. Wine. UCI Machine Learning Repository, 1991. DOI: <https://doi.org/10.24432/C5PC7J>.
- [3] S. Agarwal, G. Hope, and E. B. Sudderth. Vipaint: Image inpainting with pre-trained diffusion models via variational inference. *arXiv preprint arXiv:2411.18929*, 2024.
- [4] S. Agarwal, G. Hope, A. Younis, and E. B. Sudderth. A decoder suffices for query-adaptive variational inference. In R. J. Evans and I. Shpitser, editors, *Proceedings of the Thirty-Ninth Conference on Uncertainty in Artificial Intelligence*, volume 216 of *Proceedings of Machine Learning Research*, pages 33–44. PMLR, 31 Jul–04 Aug 2023.
- [5] S. Agarwal, K. Kask, A. Ihler, and R. Dechter. Neurobe: Escalating neural network approximations of bucket elimination. In J. Cussens and K. Zhang, editors, *Proceedings of the Thirty-Eighth Conference on Uncertainty in Artificial Intelligence*, volume 180 of *Proceedings of Machine Learning Research*, pages 11–21. PMLR, 01–05 Aug 2022.
- [6] M. Anthony and P. L. Bartlett. *Neural Network Learning - Theoretical Foundations*. Cambridge University Press, 2002.
- [7] O. Avrahami, D. Lischinski, and O. Fried. Blended diffusion for text-driven editing of natural images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 18208–18218, June 2022.
- [8] D. J. Bartholomew, M. Knott, and I. Moustaki. *Latent Variable Models and Factor Analysis: A Unified Approach*. John Wiley & Sons, 2011.
- [9] P. L. Bartlett, N. Harvey, C. Liaw, and A. Mehrabian. Nearly-tight VC-dimension and Pseudodimension Bounds for Piecewise Linear Neural Networks. *Journal of Machine Learning Research*, 20(63):1–17, 2019.
- [10] D. P. Bertsekas and J. N. Tsitsiklis. *Neuro-dynamic programming*, volume 3 of *Optimization and Neural Computation Series*. Athena Scientific, 1996.

- [11] C. M. Bishop. *Learning in Graphical Models*, chapter Latent Variable Models, pages 371–403. Springer Netherlands, Dordrecht, 1998.
- [12] C. M. Bishop and M. E. Tipping. A hierarchical latent variable model for data visualization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(3):281–293, 1998.
- [13] D. M. Blei, A. Kucukelbir, and J. D. McAuliffe. Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 112(518):859–877, 2017.
- [14] Y. Burda, R. B. Grosse, and R. Salakhutdinov. Importance weighted autoencoders. In Y. Bengio and Y. LeCun, editors, *International Conference on Learning Representations*, 2016.
- [15] G. Cardoso, Y. J. el idrissi, S. L. Corff, and E. Moulines. Monte Carlo guided denoising diffusion models for bayesian linear inverse problems. In *International Conference on Learning Representations*, 2024.
- [16] R. Child. Very deep VAEs generalize autoregressive models and can outperform them on images. In *International Conference on Learning Representations*, 2021.
- [17] H. Chung, J. Kim, M. T. Mccann, M. L. Klasky, and J. C. Ye. Diffusion posterior sampling for general noisy inverse problems. In *International Conference on Learning Representations*, 2023.
- [18] H. Chung, B. Sim, D. Ryu, and J. C. Ye. Improving diffusion models for inverse problems using manifold constraints. In A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho, editors, *Advances in Neural Information Processing Systems*, 2022.
- [19] H. Chung, B. Sim, and J. C. Ye. Come-closer-diffuse-faster: Accelerating conditional diffusion models for inverse problems through stochastic contraction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12413–12422, June 2022.
- [20] M. Collier, A. Nazabal, and C. Williams. VAEs in the presence of missing data. In *ICML Workshop on the Art of Learning with Missing Values (Artemiss)*, 2020.
- [21] C. Corneanu, R. Gadde, and A. M. Martinez. Latentpaint: Image inpainting in latent space with diffusion models. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 4334–4343, January 2024.
- [22] C. Cremer, X. Li, and D. Duvenaud. Inference suboptimality in variational autoencoders. In J. Dy and A. Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1078–1086. PMLR, 10–15 Jul 2018.
- [23] A. Darwiche. *Modeling and Reasoning with Bayesian Networks*. Cambridge University Press, 2009.

- [24] R. Dechter. Bucket elimination: A unifying framework for reasoning. *Artificial Intelligence*, 113:41–85, 1999.
- [25] R. Dechter. *Constraint Processing*. Morgan Kaufmann Publishers, 2003.
- [26] R. Dechter. Reasoning with probabilistic and deterministic graphical models: Exact algorithms. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 7(3):1–191, 2013.
- [27] R. Dechter and I. Rish. Mini-buckets: A general scheme for bounded inference. *Journal of the ACM (JACM)*, 50(2):107–153, 2003.
- [28] M. Deitke, C. Clark, S. Lee, R. Tripathi, Y. Yang, J. S. Park, M. Salehi, N. Muenighoff, K. Lo, L. Soldaini, J. Lu, T. Anderson, E. Bransom, K. Ehsani, H. Ngo, Y. Chen, A. Patel, M. Yatskar, C. Callison-Burch, A. Head, R. Hendrix, F. Bastani, E. VanderBilt, N. Lambert, Y. Chou, A. Chheda, J. Sparks, S. Skjonsberg, M. Schmitz, A. Sarnat, B. Bischoff, P. Walsh, C. Newell, P. Wolters, T. Gupta, K.-H. Zeng, J. Borchardt, D. Groeneveld, J. Dumas, C. Nam, S. Lebrecht, C. Wittlif, C. Schoenick, O. Michel, R. Krishna, L. Weihs, N. A. Smith, H. Hajishirzi, R. Girshick, A. Farhadi, and A. Kembhavi. Molmo and pixmo: Open weights and open data for state-of-the-art multimodal models. *arXiv preprint arXiv:2409.17146*, 2024.
- [29] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 248–255. Ieee, 2009.
- [30] P. Dhariwal and A. Q. Nichol. Diffusion models beat GANs on image synthesis. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021.
- [31] Z. Dou and Y. Song. Diffusion posterior sampling for linear inverse problem solving: A filtering perspective. In *International Conference on Learning Representations*, 2024.
- [32] B. T. Feng, J. Smith, M. Rubinstein, H. Chang, K. L. Bouman, and W. T. Freeman. Score-based diffusion models as principled priors for inverse imaging. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 10520–10531, October 2023.
- [33] R. Fergus, B. Singh, A. Hertzmann, S. T. Roweis, and W. T. Freeman. Removing camera shake from a single photograph. *ACM Trans. Graph.*, 25(3):787–794, July 2006.
- [34] M. Figurnov, S. Mohamed, and A. Mnih. Implicit reparameterization gradients. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.
- [35] S. Forouzan and A. T. Ihler. Incremental region selection for mini-bucket elimination bounds. In *UAI*, pages 268–277, 2015.

- [36] S. Gershman, M. D. Hoffman, and D. M. Blei. Nonparametric variational inference. In *Proceedings of the 29th International Conference on Machine Learning, ICML 2012, Edinburgh, Scotland, UK, June 26 - July 1, 2012*. icml.cc / Omnipress, 2012.
- [37] Z. Ghahramani and M. J. Beal. Propagation algorithms for variational Bayesian learning. In *NeurIPS 13*, pages 507–513. MIT Press, 2001.
- [38] A. Graikos, N. Malkin, N. Jojic, and D. Samaras. Diffusion Models as Plug-and-Play Priors. In A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho, editors, *Advances in Neural Information Processing Systems*, 2022.
- [39] A. Graves. Stochastic backpropagation through mixture density distributions. *arXiv preprint arXiv:1607.05690*, 2016.
- [40] W. Harvey, S. Naderiparizi, and F. Wood. Conditional image generation by conditioning variational auto-encoders. In *International Conference on Learning Representations*, 2022.
- [41] N. Heess, D. Tarlow, and J. Winn. Learning to Pass Expectation Propagation Messages. In *Advances in Neural Information Processing Systems*, volume 26, pages 3219–3227, 2013.
- [42] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter. GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.
- [43] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner. beta-VAE: Learning basic visual concepts with a constrained variational framework. In *International Conference on Learning Representations*, 2017.
- [44] J. Ho, A. Jain, and P. Abbeel. Denoising Diffusion Probabilistic Models. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 6840–6851. Curran Associates, Inc., 2020.
- [45] M. D. Hoffman, D. M. Blei, C. Wang, and J. Paisley. Stochastic variational inference. *Journal of Machine Learning Research*, 14(1):1303–1347, 2013.
- [46] N. B. Ipsen, P.-A. Mattei, and J. Frellsen. not-MIWAE: Deep generative modelling with missing not at random data. In *International Conference on Learning Representations*, 2021.
- [47] O. Ivanov, M. Figurnov, and D. Vetrov. Variational autoencoder with arbitrary conditioning. In *International Conference on Learning Representations*, 2019.
- [48] T. S. Jaakkola and M. I. Jordan. Recursive algorithms for approximating probabilities in graphical models. *Advances in Neural Information Processing Systems*, 9, 1996.

- [49] T. S. Jaakkola and M. I. Jordan. *Learning in Graphical Models*, chapter Improving the Mean Field Approximation via the Use of Mixture Distributions, page 163–173. MIT Press, Cambridge, MA, USA, 1999.
- [50] G. Ji, M. C. Hughes, and E. B. Sudderth. From patches to images: A nonparametric generative model. In *International Conference on Machine Learning*, pages 1675–1683. PMLR, 2017.
- [51] M. I. Jordan, Z. Ghahramani, T. S. Jaakkola, and L. K. Saul. An introduction to variational methods for graphical models. *Mach. Learn.*, 37(2):183–233, 1999.
- [52] Z. Kadkhodaie and E. Simoncelli. Stochastic solutions for linear inverse problems using the prior implicit in a denoiser. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 13242–13254. Curran Associates, Inc., 2021.
- [53] T. Karras, M. Aittala, T. Aila, and S. Laine. Elucidating the design space of diffusion-based generative models. In *Advances in Neural Information Processing Systems*, 2022.
- [54] T. Karras, S. Laine, and T. 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.
- [55] K. Kask, B. Pezeshki, F. Broka, A. T. Ihler, and R. Dechter. Scaling up AND/OR abstraction sampling. In *Proceedings of IJCAI 2020*, pages 4266–4274, 2020.
- [56] B. Kavar, M. Elad, S. Ermon, and J. Song. Denoising diffusion restoration models. In *Advances in Neural Information Processing Systems*, 2022.
- [57] M. Kelly, R. Longjohn, and K. Nottingham. UCI machine learning repository, 2017.
- [58] Y. Kim, S. Wiseman, A. Miller, D. Sontag, and A. Rush. Semi-amortized variational autoencoders. In J. Dy and A. Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 2678–2687. PMLR, 10–15 Jul 2018.
- [59] D. Kingma, T. Salimans, B. Poole, and J. Ho. Variational diffusion models. In *Advances in Neural Information Processing Systems*, 2021.
- [60] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In Y. Bengio and Y. LeCun, editors, *International Conference on Learning Representations*, 2015.
- [61] D. P. Kingma, T. Salimans, R. Jozefowicz, X. Chen, I. Sutskever, and M. Welling. Improved variational inference with inverse autoregressive flow. In *Advances in Neural Information Processing Systems*, page 4743–4751, Red Hook, NY, USA, 2016. Curran Associates Inc.
- [62] D. P. Kingma, T. Salimans, B. Poole, and J. Ho. On density estimation with diffusion models. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021.

- [63] D. P. Kingma and M. Welling. Auto-Encoding Variational Bayes. In *International Conference on Learning Representations*, 2014.
- [64] D. P. Kingma and M. Welling. An introduction to variational autoencoders. *Foundations and Trends in Machine Learning*, 12(4):307–392, 2019.
- [65] A. Klushyn, N. Chen, R. Kurle, B. Cseke, and P. van der Smagt. Learning hierarchical priors in VAEs. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.
- [66] D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. The MIT Press, 2009.
- [67] R. G. Krishnan, D. Liang, and M. D. Hoffman. On the challenges of learning with inference networks on sparse, high-dimensional data. *ArXiv*, abs/1710.06085, 2017.
- [68] A. Kucukelbir, D. Tran, R. Ranganath, A. Gelman, and D. M. Blei. Automatic differentiation variational inference. *Journal of Machine Learning Research*, 18(1):430–474, 2017.
- [69] Y. Le Cun, I. Kanter, and S. A. Solla. Eigenvalues of covariance matrices: Application to neural-network learning. *Phys. Rev. Lett.*, 66:2396–2399, May 1991.
- [70] Y. LeCun, C. Cortes, and C. Burges. Mnist handwritten digit database. *ATT Labs [Online]*. Available: <http://yann.lecun.com/exdb/mnist>, 2, 2010.
- [71] R. J. A. Little and D. B. Rubin. *Statistical analysis with missing data*, volume 793. John Wiley & Sons, 2019.
- [72] A. Liu, M. Niepert, and G. V. den Broeck. Image inpainting via tractable steering of diffusion models. In *International Conference on Learning Representations*, 2024.
- [73] Q. Liu and A. Ihler. Belief propagation for structured decision making. In *Proceedings of the 28th Conference on Uncertainty in Artificial Intelligence*, pages 523–532, 2012.
- [74] Q. Liu and A. T. Ihler. Bounding the partition function using holder’s inequality. In *International Conference on Machine Learning*, pages 849–856, 2011.
- [75] V. Lohweg. banknote authentication. UCI Machine Learning Repository, 2013. DOI: <https://doi.org/10.24432/C55P57>.
- [76] A. Lugmayr, M. Danelljan, A. Romero, F. Yu, R. Timofte, and L. V. Gool. Re-paint: Inpainting using denoising diffusion probabilistic models. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11451–11461, 2022.
- [77] T. J. M. J. Wainwright and A. S. Willskey. A new class of upper bounds on the log partition function. *IEEE Transactions on Information Theory*, pages 2313–2335, 2005.

- [78] M. Mardani, J. Song, J. Kautz, and A. Vahdat. A variational perspective on solving inverse problems with diffusion models. In *International Conference on Learning Representations*, 2024.
- [79] J. Marino, Y. Yue, and S. Mandt. Iterative amortized inference. In *Proceedings of the 35th International Conference on Machine Learning*, pages 3403–3412, 2018.
- [80] R. Mateescu, K. Kask, V. Gogate, and R. Dechter. Join-graph propagation algorithms. *J. Artif. Intell. Res. (JAIR)*, 37:279–328, 2010.
- [81] P.-A. Mattei and J. Frellsen. Leveraging the exact likelihood of deep latent variable models. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.
- [82] P.-A. Mattei and J. Frellsen. MIWAE: Deep generative modelling and imputation of incomplete data sets. In K. Chaudhuri and R. Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 4413–4423. PMLR, 09–15 Jun 2019.
- [83] C. Meng, Y. He, Y. Song, J. Song, J. Wu, J.-Y. Zhu, and S. Ermon. Sdedit: Guided image synthesis and editing with stochastic differential equations. *arXiv preprint arXiv:2108.01073*, 2021.
- [84] A. Mnih and K. Gregor. Neural variational inference and learning in belief networks. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ICML’14, page II–1791–II–1799. JMLR.org, 2014.
- [85] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.
- [86] A. Nazábal, P. M. Olmos, Z. Ghahramani, and I. Valera. Handling incomplete heterogeneous data using vaes. *Pattern Recognition*, 107:107501, 2020.
- [87] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*, 2011.
- [88] A. Nichol, P. Dhariwal, A. Ramesh, P. Shyam, P. Mishkin, B. McGrew, I. Sutskever, and M. Chen. Glide: Towards photorealistic image generation and editing with text-guided diffusion models, 2022.
- [89] A. Q. Nichol and P. Dhariwal. Improved denoising diffusion probabilistic models. In M. Meila and T. Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 8162–8171. PMLR, 18–24 Jul 2021.

- [90] J. W. Paisley, D. M. Blei, and M. I. Jordan. Variational Bayesian inference with stochastic search. In *International Conference on Machine Learning*, 2012.
- [91] K. Pandey, R. Yang, and S. Mandt. Fast samplers for inverse problems in iterative refinement models. In *Advances in Neural Information Processing Systems*, 2024.
- [92] J. Pearl. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufman, San Mateo, 1988.
- [93] D. Pollard. *Convergence of Stochastic Processes*. Springer-Verlag, 1984.
- [94] A. Ramesh, P. Dhariwal, A. Nichol, C. Chu, and M. Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 2022.
- [95] Y. Razeghi, K. Kask, Y. Lu, P. Baldi, S. Agarwal, and R. Dechter. Deep bucket elimination. In Z.-H. Zhou, editor, *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pages 4235–4242. International Joint Conferences on Artificial Intelligence Organization, 8 2021. Main Track.
- [96] D. Rezende and S. Mohamed. Variational inference with normalizing flows. In F. Bach and D. Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1530–1538, Lille, France, 07–09 Jul 2015. PMLR.
- [97] D. J. Rezende, S. Mohamed, and D. Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In E. P. Xing and T. Jebara, editors, *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 1278–1286, Beijing, China, 22–24 Jun 2014. PMLR.
- [98] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.
- [99] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10684–10695, June 2022.
- [100] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In N. Navab, J. Hornegger, W. M. Wells, and A. F. Frangi, editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, pages 234–241, Cham, 2015. Springer International Publishing.
- [101] L. Rout, N. Raoof, G. Daras, C. Caramanis, A. Dimakis, and S. Shakkottai. Solving linear inverse problems provably via posterior sampling with latent diffusion models. In *Advances in Neural Information Processing Systems*, 2023.

- [102] C. Saharia, W. Chan, H. Chang, C. Lee, J. Ho, T. Salimans, D. Fleet, and M. Norouzi. Palette: Image-to-image diffusion models. In *ACM SIGGRAPH 2022 Conference Proceedings*, SIGGRAPH '22, New York, NY, USA, 2022. Association for Computing Machinery.
- [103] C. Saharia, W. Chan, S. Saxena, L. Li, J. Whang, E. L. Denton, K. Ghasemipour, R. Gontijo Lopes, B. Karagol Ayan, T. Salimans, J. Ho, D. J. Fleet, and M. Norouzi. Photorealistic text-to-image diffusion models with deep language understanding. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 36479–36494. Curran Associates, Inc., 2022.
- [104] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini. The graph neural network model. *Trans. Neur. Netw.*, 20(1):61–80, jan 2009.
- [105] A. Ścibior, V. Masrani, and F. Wood. Differentiable particle filtering without modifying the forward pass. *International Conference on Probabilistic Programming (PROBPROG)*, 2021.
- [106] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In F. Bach and D. Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 2256–2265, Lille, France, 07–09 Jul 2015. PMLR.
- [107] C. K. Sønderby, T. Raiko, L. Maaløe, S. K. Sønderby, and O. Winther. Ladder Variational Autoencoders. In *Advances in Neural Information Processing Systems*, 2016.
- [108] B. Song, S. M. Kwon, Z. Zhang, X. Hu, Q. Qu, and L. Shen. Solving inverse problems with latent diffusion models via hard data consistency. In *International Conference on Learning Representations*, 2024.
- [109] J. Song, A. Vahdat, M. Mardani, and J. Kautz. Pseudoinverse-guided diffusion models for inverse problems. In *International Conference on Learning Representations*, 2023.
- [110] Y. Song and S. Ermon. Generative modeling by estimating gradients of the data distribution. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.
- [111] Y. Song, L. Shen, L. Xing, and S. Ermon. Solving inverse problems in medical imaging with score-based generative models. In *International Conference on Learning Representations*, 2022.
- [112] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021.

- [113] R. Strauss and J. Oliva. Posterior matching for arbitrary conditioning. In A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho, editors, *Advances in Neural Information Processing Systems*, 2022.
- [114] E. G. Tabak and C. V. Turner. A family of nonparametric density estimation algorithms. *Communications on Pure and Applied Mathematics*, 66, 2013.
- [115] E. G. Tabak and E. Vanden-Eijnden. Density estimation by dual ascent of the log-likelihood. *Communications in Mathematical Sciences*, 8:217–233, 2010.
- [116] B. L. Trippe, J. Yim, D. Tischer, D. Baker, T. Broderick, R. Barzilay, and T. S. Jaakkola. Diffusion probabilistic modeling of protein backbones in 3d for the motif-scaffolding problem. In *International Conference on Learning Representations*, 2023.
- [117] A. Vahdat and J. Kautz. NVAE: A deep hierarchical variational autoencoder. In *Advances in Neural Information Processing Systems*, 2020.
- [118] A. Vahdat, K. Kreis, and J. Kautz. Score-based generative modeling in latent space. In *Advances in Neural Information Processing Systems*, 2021.
- [119] V. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer, second edition, November 1999.
- [120] M. J. Wainwright and M. I. Jordan. Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, 1:1–305, 2008.
- [121] H. Wang, Y. Yu, T. Luo, H. Fan, and L. Zhang. MaGIC: Multi-modality guided image completion. In *International Conference on Learning Representations*, 2024.
- [122] S. Wang, C. Saharia, C. Montgomery, J. Pont-Tuset, S. Noy, S. Pellegrini, Y. Onoe, S. Laszlo, D. J. Fleet, R. Soricut, J. Baldridge, M. Norouzi, P. Anderson, and W. Chan. Imagen editor and editbench: Advancing and evaluating text-guided image inpainting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 18359–18369, June 2023.
- [123] Y. Wang, J. Yu, and J. Zhang. Zero-shot image restoration using denoising diffusion null-space model. In *International Conference on Learning Representations*, 2023.
- [124] J. Winn and C. M. Bishop. Variational message passing. *JMLR*, 6:661–694, 2005.
- [125] W. Wolberg, O. Mangasarian, N. Street, and S. W. Breast Cancer Wisconsin (Diagnostic). UCI Machine Learning Repository, 1995. DOI: <https://doi.org/10.24432/C5DW2B>.
- [126] S. Xie, Z. Zhang, Z. Lin, T. Hinz, and K. Zhang. Smartbrush: Text and shape guided object inpainting with diffusion model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 22428–22437, June 2023.

- [127] Y. Yang, R. Bamler, and S. Mandt. Improving inference for neural image compression. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 573–584. Curran Associates, Inc., 2020.
- [128] Z. Yang, R. Feng, H. Zhang, Y. Shen, K. Zhu, L. Huang, Y. Zhang, Y. Liu, D. Zhao, J. Zhou, and F. Cheng. Lipschitz singularities in diffusion models. In *International Conference on Learning Representations*, 2024.
- [129] J. S. Yedidia, W. T. Freeman, and Y. Weiss. Generalized belief propagation. In *(NIPS) 2000*, pages 689–695. MIT Press, 2000.
- [130] I.-C. Yeh. Concrete Compressive Strength. UCI Machine Learning Repository, 2007. DOI: <https://doi.org/10.24432/C5PK67>.
- [131] K. Yoon, R. Liao, Y. Xiong, L. Zhang, E. Fetaya, R. Urtasun, R. S. Zemel, and X. Pitkow. Inference in probabilistic graphical models by graph neural networks. 2018.
- [132] F. Yu, Y. Zhang, S. Song, A. Seff, and J. Xiao. LSUN: Construction of a Large-scale Image Dataset using Deep learning with Humans in the Loop. *arXiv preprint arXiv:1506.03365*, 2015.
- [133] J. Yu, Y. Wang, C. Zhao, B. Ghanem, and J. Zhang. Freedom: Training-free energy-guided conditional diffusion model. *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.
- [134] J. Yu, Z. Wang, V. Vasudevan, L. Yeung, M. Seyedhosseini, and Y. Wu. Coca: Contrastive captioners are image-text foundation models. *Transactions on Machine Learning Research*, 2022.
- [135] J. Yu, Y. Xu, J. Y. Koh, T. Luong, G. Baid, Z. Wang, V. Vasudevan, A. Ku, Y. Yang, B. K. Ayan, B. Hutchinson, W. Han, Z. Parekh, X. Li, H. Zhang, J. Baldridge, and Y. Wu. Scaling autoregressive models for content-rich text-to-image generation. *Transactions on Machine Learning Research*, 2022. Featured Certification.
- [136] S. Zagoruyko and N. Komodakis. Wide residual networks. In *BMVC*, 2016.
- [137] G. Zhang, J. Ji, Y. Zhang, M. Yu, T. S. Jaakkola, and S. Chang. Towards coherent image inpainting using denoising diffusion implicit models. *CoRR*, abs/2304.03322, 2023.
- [138] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018.
- [139] S. Zhao, J. Cui, Y. Sheng, Y. Dong, X. Liang, E. I. Chang, and Y. Xu. Large scale image completion via co-modulated generative adversarial networks. In *International Conference on Learning Representations*, 2021.