UNIVERSITY OF CALIFORNIA
RIVERSIDE

Simple Structures in Deep Networks

A Dissertation submitted in partial satisfaction
of the requirements for the degree of

Doctor of Philosophy

in

Electrical Engineering

by

Rakib Hyder

September 2022

Dissertation Committee:

Dr. M. Salman Asif, Chairperson
Dr. Amit K. Roy-Chowdhury
Dr. Samet Oymak

The Dissertation of Rakib Hyder is approved:

Committee Chairperson

University of California, Riverside

Acknowledgments

I am grateful to my advisor, without whose help, I would not have been here. He has been patient with me all these years and taught me the fundamental of research. My research thinking has been greatly influenced by him. I would also like to thank Dr. Amit K Roy-Chowdhury and Dr. Samet Oymak for their generous help as being committee members for my dissertation evaluation and defense. I would like to express my gratitude to Dr. Hassan Mansour for the wonderful opportunity to work as an intern at MERL. I would also like to thank Dr. Chenguang Liu for providing me the opportunity to work as an intern in Samsung research laboratories. Finally, I would like to thank all of my collaborators, colleagues and teachers who enriched me with their inspiring ideas and helped me learn new things along the way.

To my parents for all the love and support.

ABSTRACT OF THE DISSERTATION

Simple Structures in Deep Networks

by

Rakib Hyder

Doctor of Philosophy, Graduate Program in Electrical Engineering
University of California, Riverside, September 2022
Dr. M. Salman Asif, Chairperson

Deep networks have received considerable attention in recent years due to their applications in different problems of science and engineering. This dissertation explores the application of deep networks in continual learning and inverse problems. In this work, we enforce some simple structures on the networks to achieve better solution in terms of performance, memory and computational cost.

Continual Learning with Low-rank Increment: Continual learning is a process of training a single neural network on multiple tasks one after another, where training data for each task is often available only during the training of that task. Neural networks tend to forget older tasks when they are trained for the newer tasks; this property is often known as catastrophic forgetting. To address this issue, continual learning methods use episodic memory, parameter regularization, masking and pruning, or extensible network structures. This work proposes a continual learning framework based on low-rank factorization of the network weights. To update the network for a new task, a rank-1 (or low-rank) matrix is learned and added to the weights of every layer. An additional selector vector is also

introduced that assigns different weights to the low-rank matrices learned for the previous tasks. Our proposed approach demonstrates superior performance compared to the current state-of-the-art methods with much lower number of network parameters.

Inverse Problems with Deep Networks: Inverse problems form a family of problems where we try to recover the true signal given the modified version of the signal. Since inverse problems are often ill-posed in nature, we often need to impose some constraints on the solution set. This dissertation mainly focuses on deep generative networks as a prior for solving inverse problems. Low-rank matrix and tensor structures have been used in this work as constraints on the input latent vectors of the deep generative networks to improve quality of the reconstruction with reduced parameters. This dissertation also explores unrolled networks where classical iterative solution approaches are structured as fixed layer networks with each iteration forming a layer of the network. We use such unrolled network structures to design sensing parameters for nonlinear inverse problems that led to achieving good reconstruction quality with a fixed number of layers (or iterations).

Contents

List of Figures **xi**

List of Tables **xix**

1 Introduction **1**
1.1 Continual Learning . . . 1
1.1.1 Incremental Task Learning . . . 2
1.2 Inverse Problems . . . 3
1.2.1 Linear Inverse Problems . . . 3
1.2.2 Nonlinear Inverse Problems . . . 4
1.3 Organization . . . 5

2 Background and Related Works **7**
2.1 Continual Learning . . . 7
2.1.1 Regularization-based approaches . . . 7
2.1.2 Memory-based approaches . . . 8
2.1.3 Dynamic network architectures . . . 8
2.2 Deep Generative Models . . . 10
2.2.1 Tensor Factorization Methods . . . 11
2.3 Inverse Problems . . . 11
2.3.1 Compressive Sensing . . . 12
2.3.2 Phase retrieval . . . 13

3 Incremental Task Learning with Low-rank Increment **18**
3.1 Introduction . . . 18
3.2 Incremental Task Learning via Rank Increment . . . 21
3.3 Experiments and Results . . . 24
3.3.1 Datasets and Task Description . . . 24
3.3.2 Training Details . . . 25
3.3.3 Comparing Techniques . . . 27
3.3.4 Results with Three-Layer MLP . . . 28
3.3.5 Results with ResNet18 . . . 33

3.3.6 Effect of updating last few layers. 35
3.3.7 Relationship between the newly learned rank-1 weights and the fixed weights learned in the previous task 36
3.3.8 Effect of task similarity 37

4 Low-rank Generative Networks for Linear Inverse Problems 38
4.1 Introduction 38
4.2 Technical Approach 42
4.2.1 Latent Code Optimization 43
4.2.2 Joint Latent Codes and Generator Optimization 44
4.2.3 Low Rank Constraint 47
4.3 Experimental Setup 48
4.4 Results and Analysis 53
4.4.1 Sequence Size vs Performance 53
4.4.2 Denoising 54
4.4.3 Inpainting 57
4.4.4 Compressive Sensing 59
4.4.5 Flutter Shutter 60
4.4.6 Rank of the Latent Matrix 62
4.4.7 Computational Complexity 63
4.4.8 Comparison with Video DIP 65

5 Tensor Ring Autoencoders for Linear Inverse Problems 71
5.1 Introduction 71
5.1.1 Our Contributions 73
5.2 Technical Details 74
5.3 Experiment and Results 78
5.4 Performance as an Image Generator 83
5.4.1 Image Generation from Noise 83
5.4.2 Latent Space Interpolation 84
5.4.3 Data fitting performance 84

6 Consensus Equilibrium to Combine DIP with RED 86
6.1 Introduction 86
6.2 Related Work 90
6.3 Consensus Equilibrium for DIP and RED 91
6.3.1 DeepRED as fixed-point CE 91
6.4 Experimental Validation 94

7 Phase Retrieval with Deep Generative Network 98
7.1 Introduction 98
7.2 Algorithm 102
7.2.1 Phase update 102
7.2.2 Gradient descent update 103
7.2.3 Projection step 103

7.3 Models and Experiments . . . 104

8 Generative Network with Side Information for Phase Retrieval 108
8.1 Introduction . . . 108
8.2 Proposed Method . . . 110
8.3 Experimental Setup . . . 113
8.4 Results and Discussion . . . 116

9 Unrolling Network to Learn Reference for Phase Retrieval 121
9.1 Introduction . . . 121
9.2 Proposed Approach . . . 125
9.3 Experiments . . . 129
9.3.1 Configurations of Reference (u) . . . 131
9.3.2 Setup of Training Samples and Sample Size . . . 132
9.3.3 Generalization of Reference on Different Classes . . . 133
9.3.4 Noise Response . . . 135
9.3.5 Random Reference versus Learned Reference . . . 135
9.3.6 Comparison with Existing Phase Retrieval Methods . . . 136
9.3.7 Effects of Number of Layers (K) . . . 137
9.3.8 Localizing the Reference . . . 138

10 Unrolling Network to Learn Coded Illumination Patterns 141
10.1 Introduction . . . 141
10.2 Proposed Method . . . 145
10.3 Experiments . . . 150
10.3.1 Setup and hyper-parameter search . . . 151
10.3.2 Comparison between random and learned illumination patterns . . . 151
10.3.3 Effect of number of iterations/layers (K) . . . 154
10.3.4 Comparision with existing methods . . . 155
10.3.5 Generalization of learned patterns on different datasets . . . 158
10.3.6 Noise response . . . 160
10.3.7 Mismatch in training and test images . . . 162

11 Conclusion 163
11.1 Continual Learning with Low-Rank Increment . . . 163
11.2 Inverse Problems with Deep Networks . . . 164
11.2.1 Solving Linear Inverse Problems with Untrained Generative Prior . . . 164
11.2.2 Solving Phase Retrieval with Trained Generative Prior . . . 165
11.2.3 Learning Sensing Parameters Using Unrolling Networks . . . 165
11.3 Future Directions . . . 166
11.3.1 Continual Learning with Low-Rank Networks . . . 166
11.3.2 Inverse Problems with Structured Networks . . . 167

Bibliography 169

List of Figures

3.1 An overview of our proposed method for continual learning via low-rank network updates. We first represent (and learn) the weight matrix (or tensor) for each layer as a product of low-rank matrices. To train a network for new tasks without forgetting the earlier tasks, we reuse the factors from the earlier tasks and add a new set of factors for the new task. Our experiments suggest that a rank-1 update is often sufficient for successful continual learning. . . 19

3.2 Average test accuracy for different datasets (Permuted MNIST, Rotated MNIST, Split CIFAR100, Split miniImageNet) along different tasks using different algorithms (AGEM,EWC, Orthog. Subspace, ICARL and our approach). We use three layer MLP here. Parallel full-rank results corresponds to the case when we train every task on separate full rank networks independently (serves as an upper limit for ITL methods). We showed the average of 20 tasks. 31

3.3 Evolution of task-wise test accuracy on P-MNIST (first row) and R-MNIST (second row) datasets for EWC, Orthogonal Subspace, and Our approach. We can observe from the decrease in the test accuracy that EWC and Orthogonal Subspace forget the previous tasks as they learn new tasks. Our approach does not show any forgetting as we learn new tasks. 32

3.4 Top K singular values of weight matrices corresponding to different tasks for S-CIFAR100 with MLP experiments. 36

a Layer 1 36

b Layer 2 36

4.1 A candidate architecture we use in our experiments with one fully connected and four fractionally strided convolutional layers. Generative model: $x = G_\gamma(z)$ maps a vector $z \in \mathbb{R}^k$ into an image $x \in \mathbb{R}^n$. 40

4.2 An illustration of different generative priors discussed in the chapter: (a) Optimizing latent codes can only reconstruct images in the range of the generative network. (b) Jointly optimizing latent code and network weights enables recovery of a larger range of images. (c) Low-rank and similarity constraints on latent code further regularize the problem and potentially explain other structures in data. 41

a . . . 41
b . . . 41
c . . . 41
4.3 Joint optimization versus latent code optimization. First row is the true images of the videos sequences. The second row contains the masked samples of the sequences. In the third row, we reconstruct frames with latent code optimization using a generator trained on some other frames of the same video sequence (Generator1). In the fourth row, we use latent code optimization with a generator trained on CIFAR10 dataset (Generator2). The fourth row is the reconstruction with joint optimization of generator initialized with random weights. We can observe that latent code optimization does not perform well (row 4) when we do not have generator pretrained on similar distribution. However, joint optimization performs as good as as or better than latent code optimization without any pretrained weights. . . . 43
4.4 Sequence size vs performance for video approximation and compressive sensing tasks. Here the results corresponds to joint optimization. We can observe that increasing video length improves compressive sensing performance for joint optimization. This effect diminishes with the increased size of video sequences. . . . 55
a Handwaving . . . 55
b Handclapping . . . 55
c Apply Eye Makeup . . . 55
4.5 Reconstruction of different video sequences using different algorithms for denoising problem. Handclapping and Handwaving video sequences are 64×64 and Archery and Apply Eye Makeup video sequences are 256×256. The error bars are standard deviation intervals. The deep decoder reconstruction here correspond to overparameterized deep decoder structure. All the comparing algorithms show very good reconstruction quality. . . . 56
4.6 Reconstruction quality curves for denoising experiments with different algorithms for different levels of signal to noise ratio. The curves also show standard deviation intervals. We compare the performance for (a) Handclapping (b) Handwaving (c) Archery (d) Apply Eye Makeup video sequences. All the comparing methods other than UP deep decoder performs similarly. The curves suggest that UP deep decoder has reached its limit to generate the sequences. . . . 57
a Handclapping . . . 57
b Handwaving . . . 57
c Archery . . . 57
d Apply Eye Makeup . . . 57

4.7 Some reconstruction results on inpainting problem. Handclapping and Handwaving video sequences are 64×64 and Archery sequence is 256×256. The deep decoder reconstruction here correspond to overparameterized deep decoder structure. The boxed regions are zoomed for details. We can observe that joint optimization gives better reconstruction than the comparing algorithms in terms of details. 58
4.8 Inpainting performance for different available measurement rate for (a) Handclapping (b) Handwaving (c) Archery (d) Apply Eye Makeup video sequences. Measurement rate represents the available fraction of the total pixels. The error bars are standard deviation intervals. Other than Archery sequence, joint optimization outperforms the other comparing methods especially at lower measurement rate. 59
a Handclapping 59
b Handwaving 59
c Archery 59
d Apply Eye Makeup 59
4.9 Some reconstruction results on spatial compressive sensing problem. Handclapping and Handwaving video sequences are 64×64 and Archery and Apply Eye Makeup video sequences are 256×256. The compressive frames from Handclapping and Handwaving are 29×29 whereas the compressive frames from Archery and Apply Eye Makeup video sequences are 114×114. The deep decoder reconstruction here correspond to overparameterized deep decoder structure. We can observe that the reconstructions are similar for the comparing algorithms. 61
4.10 Compressive sensing performance for different available measurement rate for (a) Handclapping (b) Handwaving (c) Archery (d) Apply Eye Makeup video sequences. Measurement rate (or compression ratio) represents the available fraction of the total measurements. The error bars are standard deviation intervals. We can observe from the curves that joint optimization performs at par with the other comparing methods. 62
a Handclapping 62
b Handwaving 62
c Archery 62
d Apply Eye Makeup 62
4.11 Some reconstructions for flutter shutter problem. Here we have a single measurements for every 4 non overlapping frames. We can observe that TVAL3D suffers ghosting effect for the fast changing parts of the videos such as the hand or leg movement. However, they perform similarly in background details reconstruction. 63
4.12 Effect of different value of rank for low rank constraint in inpainting problem with 80% pixels randomly missing. We also show standard deviation interval for each point. 64

4.13 Pairwise cosine similarity between frames, measurements or latent codes for extended Handwaving video sequence where Handwaving action is repeated in an interval of around 45 frames. Blue indicates highest similarity whereas yellow indicates lowest similarity. We can observe that the similarity pattern in the original frames are not maintained in the compressive frames. As the Video DIP latent codes are drawn at random, we do not observe any similarity pattern in them (c). However, the corresponding latent matrix for joint optimization (d) captures the similarity structure. Low rank constraint (e) further enhances this similarity. 68
a Original Frames 68
b Compressive Frames 68
c Latent Matrix of Video DIP 68
d Latent Matrix of Joint Optimization 68
e Latent Matrix of Joint Optimization + Low Rank Constraint 68
4.14 Pairwise cosine similarity between frames, measurements or latent codes for extended mixed video sequence where 16 frames of 6 different video sequences (Handwaving, Handclapping, Walking, Archery, Apply Eye Makeup, Band Marching in order) are concatenated in the temporal dimension. Blue indicates highest similarity whereas yellow indicates lowest similarity. We observe that adding low rank constraint further bolster the similarity observed in the frames of same video sequences found by joint optimization. 69
a Original Frames 69
b Compressive Frames 69
c Latent Matrix of Video DIP 69
d Latent Matrix of Joint Optimization 69
e Latent Matrix of Joint Optimization + Low Rank Constraint 69

5.1 General overview of our proposed tensor ring factorized autoencoder. We map a set of images $\{X\}$ to latent codes $\{Z\}$ using an encoder E. We then perform tensor factorization on the latent space codes using tensor factorization (shown as $\mathbf{T}$ block). Finally, we pass the factorized representation though the decoder D to generate target images $\hat{X}$. 74
5.2 Reconstruction results for (a) denoising and (b) inpainting on Small NORB, RaFD and 3dShapes datasets. 79
5.3 Interpolation in the latent space to change object shape/size using different generators. Left and rightmost images are part of training set. The views in between are synthesized using linear interpolation in latent space. 85

6.1 Consensus Equilibrium of model mismatch, RED, and DIP. The top images result from the action of different agents that are combined to produce the CE solution. 88
6.2 Image deblurring performance of DeepRED and CE formulation under (a) the presence of high noise ($\sigma_n = 8/255$,$\sigma_k = 1.6$) and (b) the presence of high blurring ($\sigma_n = \sqrt{2}/255$,$\sigma_k = 2.4$). 94

6.3 Reconstruction quality resulting from the combination of the three different agents. 94

7.1 *Illustration of APPGD algorithm. It has two major steps: alternating minimization and projection onto the range of the generator nework. In alternating minimization step, we update phase and perform one gradient descent update using the updated phase. Starting from a random vector, we perform phase update, gradient descent update step and projection step iteratively to reach the final estimate.* 100
7.2 *Comparison of three approaches on MNIST test set.* 105
a Reconstruction results on MNIST for three different approaches with $m = 60$ measurements. 105
b Reconstruction error (per pixel) for three approaches on MNIST. . . 105
c Mean SSIM for three approaches on MNIST. 105
7.3 *Comparison of three approaches on celebA test set and some reconstruction results for our APPGD algorithm.* 106
a Reconstruction results on celebA dataset for APPGD with $m = 1000$ measurements. 106
b Reconstruction error (per pixel) for three approaches on celebA. . . 106
c Mean SSIM for three approaches on celebA. 106

8.1 Illustration of phase retrieval with side information. An image $\mathbf{x}$ is divided into a known ($\mathbf{s}$) and unknown part ($\mathbf{q}$) such that $\mathbf{x} = [\mathbf{s}\ \mathbf{q}]$. Fourier amplitude measurements of the image are observed as $|F(\mathbf{x})| \equiv |F\mathbf{q} + \mathbf{b}|$. Alternating minimization algorithm uses the knowledge of the known part to initialize the problem and enforces additional constraints on the signal estimate at every iteration. 109
8.2 Simulation results for Fourier phase retrieval with different side information. We form an image by concatenating five MNIST digit images. We have concatenated different known 32×32 image to that concatenated MNIST image as the first patch. We use a trained generative network which is trained on MNIST digits as prior. We project each 32×32 digit onto the range of that trained prior. The last column shows reconstruction without side information. We can observe that side information gives significant performance boost. . 110
8.3 Simulation results for Fourier phase retrieval with and without side information. We observed the reconstruction when we know different parts of the image. We have concatenated 4 different MNIST digits. Here 2nd to 5th columns correspond to the reconstructions where we have prior knowledge of 1st, 2nd, 3rd and 4th digit (32×32 image patch) respectively. 6th column corresponds to the reconstruction without side information. 113

8.4 Reconstruction with known name tags as side information. We have concatenated the last names of the corresponding celebrities as side information to the original image. Fourier phase retrieval performance significantly increases with such side information. We have shown reconstruction for different size and position of the name tags. In (a), (b) and (c) name tags are of size 16×64, 32×64 and 64×64 respectively. The target celebrity image is 64×64. So the ratio of side information to the unknown part is $1:4$, $1:2$ and $1:1$ respectively. In (d), (e) and (f) we placed the text on the bottom, middle and the top of the known name tag with size 64×64. 117
8.5 Reconstruction when some part of the image is known. We have shown cases when top-left, top-right, bottom-left, bottom-right 32×32 patches of 64×64 celebrity images are known. We have also shown comparison with the reconstruction without any side information. 118

9.1 Our proposed approach for learning reference signal by solving phase retrieval using an unrolled network. Unrolled network has K layers. Each layer_k gets amplitude measurements y, reference u, and estimate x^{k-1} as inputs, and updates the estimate to x^k. The operations inside layer_k are shown in the dashed box on the right, where A and B are both linear measurement operators, and A^* is the adjoint operator of A. 125
9.2 Reconstruction results using learned references. Each block **(a)-(f)** shows results for different dataset: (**left**) learned reference with a colorbar; (**middle**) sample original images and reconstruction with PSNR on top; (**right**) histogram of PSNR over the entire test dataset (vertical dashed line represents the mean PSNR). 130
a MNIST 130
b EMNIST 130
c Fashion MNIST 130
d SVHN 130
e CIFAR10 130
f CelebA 130
9.3 Phase retrieval results using learned and random references. **First Row:** Original 512×512 test images. **Second Row:** Reconstruction using random references with uniform distribution between $[0, 1]$ **best result out of 100 trials**. **Third Row:** Reconstruction using the reference learned on CelebA dataset and resized from 200×200 to 512×512. (PSNR shown on top of images). 131
9.4 Test results on shifted/flipped/rotated images using the reference learned on upright-centered (canonical) images. PSNR shown on top of images. 134
a MNIST 134
b CIFAR10 134
9.5 Reconstruction quality of the test images vs noise level of the measurements for different datasets. We learned the reference using noise-free measurements. 136
a Gaussian 136
b Poisson 136

9.6 Reconstruction PSNR vs the number of blocks (K) in the unrolled network at training and testing. (a) K is same for training and testing (shaded region shows ± 0.25 times `std` of PSNR). (b) $K = 1$ and (c) $K = 10$, but tested using different K. 138
a Training K=Testing K 138
b Training K=1 138
c Training K=50 138
9.7 Single step reconstruction with reference in range $[0, 1]$. Each of the **6 sets (a)-(f)** has the the ground truth in the first row. Second row is the reconstruction (PSNR values on top). 139
9.8 Performance of our method if the reference is an 8×8 block placed at different positions. Fixing the minimum value at 0, we increased the maximum value of the reference we learn. We observe that the small reference placed in the corners performs better than the ones placed in the center. 140
a MNIST 140
b CIFAR10 140

10.1 Pipeline of our proposed framework at inference time. Our framework mainly contains two components: (*1*) a learnable sensing system that updates the illumination patterns during training time, but at inference time the learned illumination patterns are fixed; (*2*) a fixed unrolled network that runs phase retrieval process to recover the original signal x form measurements Y. The number of layers in the network is fixed to K. Steps at every iteration are fixed and depicted as an unrolled network (details can be found in Algorithm 10). We illustrate the steps of k^{th} layer of the unrolling network. Phase retrieval algorithm uses the measurements $Y = \{y_t\}$ and illumination patterns $D = \{d_t\}$ to provide an estimate x^K after K iterations. During training time, our goal is to learn the illumination patterns D to minimize the error between the estimated x^K and the ground truth. More details can be found in section 10.2. 143
10.2 Selected ground truth (GT) images, corresponding reconstructed images using random and learned illumination patterns. PSNR is shown on top of every reconstruction. Below each dataset, we show the histograms of the PSNRs of all images with random patterns (shown in blue) and learned patterns (shown in orange). The dashed vertical line indicates the mean of all PSNRs. We used $T = 4$ illumination patterns. Random illumination patterns are selected best out of 30 trials. The learned illumination patterns are trained on 128 training images. 152
a MNIST 152
b F. MNIST 152
c CIFAR10 152
d CelebA 152
10.3 Learned illumination patterns corresponding to the reported results for MNIST, F. MNIST, CIFAR10 and CelebA in Table 10.2. 153

10.4 Comparison of the reconstruction quality with random and learned illumination patterns for different values of $K = 1, \ldots, 200$. We plot the average PSNR in bright color and the PSNR of randomly selected 100 samples in light shadows. **Learned** represents the reconstruction PSNR with learned illumination patterns (shown in red), and **Random** represents PSNR for random illumination patterns (shown in blue). The number of illumination patterns is $T = 4$. Random illumination patterns are selected best out of 30 trials. The learned illumination patterns are trained on 128 training images and number of iterations $K = 50$ during training. 153
a MNIST 153
b F. MNIST 153
c CIFAR10 153
d CelebA 153
10.5 Reconstruction quality vs number of iterations (layers) at test time (i.e., K is different for training and testing with $T = 4$). We show error bar of $\pm 0.25\sigma$ for each dataset. In (a) and (b), we fixed K (K=10, 20) and tested using different K. In (c), we trained and tested using the same number of layers. . 154
a Training K=10 154
b Training K=20 154
c Training K=Test K 154
10.6 **First Row:** Ground truth images from image processing standard test datasets. **Second Row:** Reconstruction using random illumination patterns with uniform random distribution [0, 1] (we selected $T = 4$ patterns that provided best results on celebA test images in **30 trials**). PSNR numbers are shown on the top of reconstructed images. **Third Row:** Reconstruction using the patterns trained on celebA dataset. Each image has 200×200 pixels and the number of illumination patterns is $T = 4$. 159
10.7 Reconstruction quality of the test images vs noise level of the measurements for different datasets. Here we show shaded error bar of $\pm 0.25\sigma$ for each dataset. We learn the illumination patterns ($T = 4$) on 128 noiseless training images of corresponding datasets. 159
a Gaussian 159
b Poisson 159
10.8 Test results on images shifted to bottom right by 5 pixels. From left to right: MNIST, F. MNIST, and CIFAR10. 161
10.9 Test results on images rotated by 90°. From left to right: MNIST, F. MNIST, and CIFAR10. 162

List of Tables

3.1 Average test accuracy of ITL for P-MNIST, R-MNIST, S-CIFAR100, and S-miniImageNet with three layer MLP. Standard deviation of test accuracy over five runs is shown in parenthesis for some of the experiments. [1ex] * Orthog subspace does not use replay buffer for MNIST variations. [-1.5 ex] 29

3.2 Average forgetting results corresponding to Table 3.1 for different datasets using different approaches. We report the forgetting in percentage unit (%). We also report the standard deviation over 5 experiments for some methods. 30

3.3 Number of parameters and buffer size in ITL methods with 3-layer MLP. . 33

3.4 Number of parameters used by different zero-forgetting algorithms (HAT, PackNet, and Ours) using 3-layer MLP. 33

3.5 Test accuracy for different rank choices of the proposed ITL approach and multi-task baseline networks for P-MNIST and R-MNIST. Initial rank is $r_{k,1}$ and rank increment/task is $r_{k,t}$. [-1.0ex] 34

3.6 Comparison of test accuracy and forgetting for split CIFAR-100 and split miniImageNet datasets using ResNet18 architecture. 35

4.1 Reconstruction performance measured in terms of PSNR for different compressive sensing problems. We show comparison with TVAL3D (3D extension of TVAL3 [1]) and deep decoder [2]. The results are averaged over five experiments with different random measurement matrices (or noise in the case of denoising). 50

4.2 Generator structures and corresponding number of parameters for different image sizes. $h \times w \times c$ denote height , weight, and color channels, respectively. 52

4.3 Comparison of joint optimization with DCGAN and deep decoder in terms of computational complexity and memory requirement. The memory requirement is for each frame reconstruction. The average time consumption is calculated for video sequences with 32 frames. 65

4.4 Effect of initial latent matrix for different inverse problems. We have drawn latent matrix in way that the initial latent codes form a line. The results are averaged over fifteen experiments with five different random measurement matrices and three different initializations. We use same measurement matrices and initializations for both approaches. 65

4.5 Performance analysis between Video DIP and joint optimization when all the frames in the video sequence are not close to each other. The results are averaged over twelve experiments with four different random measurement matrices and three different initializations. We use same measurement matrices and initializations for both approaches. 66

5.1 Reconstruction quality (PSNR in dB) for image denoising and inpainting with different comparing approaches. 82
5.2 Frechet Inception Distance (FID) values for images generated from noise. . 84
5.3 Training images representation performance (PSNR in dB). 84

6.1 Comparison of reconstruction PSNR among the different algorithms under low noise setting ($\sigma_n = \sqrt{2}/255$). 93
6.2 Comparison of reconstruction PSNR for different noise levels and blurring kernel strengths. 95
6.3 Reconstruction PSNR for the different agents. 96

9.1 PSNR for different training size 133
9.2 PSNR of the Same Reference Tested on Different Datasets 134
9.3 Comparison with Existing Phase Retrieval Methods 137

10.1 PSNR (`mean` $\pm$ `std`) for random and learned illumination patterns tested on different datasets. 150
10.2 Reconstruction PSNR (`mean` $\pm$ `std`) of different algorithms using random patterns and our learned patterns. The number of patterns is 4 in each case. Here we round the PSNR values to integers to fit the width of the page. We let all the algorithms to run until convergence. *For Deep Model [3] experiments, patterns are normalized to $[-1, 1]$ range. **For Deep Model, the image size for CelebA generator is 64×64. 155
10.3 Average runtime (sec) per image of different algorithms corresponding to the performance reported in Table 10.2. The reported runtime corresponds to the time required for convergence of each algorithm. **For Deep Model, the image size for CelebA is 64×64. 155
10.4 Reconstruction PSNR (`mean` $\pm$ `std`) of illumination patterns learned and tested on different datasets for $K = 50$. Every column corresponds to patterns learned on a fixed dataset and tested on all. Random column reports the performance of random illumination patterns. 158
10.5 Reconstruction PSNR (`mean` $\pm$ `std`) of different algorithms using random patterns (best out of 5 trials) and our learned patterns at different Poisson noise levels for MNIST and CIFAR10 dataset. The number of patterns is 4 in each case. We let all the algorithms to run until convergence. Here we round the PSNR values to integers to fit the width of the page. 160

Chapter 1

Introduction

Deep networks have received considerable attention in signal processing, computer vision, natural language processing etc for their ability to work as a robust generative and discriminative models. Deep networks often have many redundant parameters which makes it difficult to incorporate them in low cost devices. Applying different structures in the network can potentially reduce the computational cost for the deployment of the network and provide more control over it. In this thesis, we mainly explored different structures in deep networks to enhance performance of the networks with reduced computational cost.

1.1 Continual Learning

Continual learning [4, 5] aims to train a single model on a sequence of different tasks and perform well on all the trained tasks once the training is finished. While training on new tasks, the old data from previous tasks are not usually provided to the model. This scenario mimics the human learning process where they have the ability to acquire new

knowledge and skills throughout their lifespan. However, this setting is still challenging to neural network models as a common phenomenon called "catastrophic forgetting [6]" is observed during this learning process. Catastrophic forgetting occurs when the data from the new tasks interfere with the data seen in the previous tasks and deteriorate model performance on preceding tasks.

1.1.1 Incremental Task Learning

In Chapter 3, we focus on task-incremental continual learning (ITL) in which data for every task are provided in a sequential manner to train/update the network [7]. It has been a popular continual learning setup even in the very recent literature [8, 9, 10, 11, 12, 13]. ITL finds its application in setups where task identity is available during inference; for instance, tasks performed under different weather/light/background conditions and we know the changes, or tasks learned on different data/classes where we know the task identity.

Let us denote the network function that maps input x to output for task t as $f(x; \mathcal{W}_t)$, where $\mathcal{W}_t$ denotes the network weights for task t. We seek to update the $\mathcal{W}_t$ for all t as we sequentially receive dataset for one task at a time. Suppose the training dataset for task t is given as $(\mathcal{X}_t, \mathcal{Y}_t)$ drawn from a distribution $\mathcal{P}_t$, where $\mathcal{X}_t$ denotes the set of input samples and $\mathcal{Y}_t$ denotes the corresponding ground-truth outputs. Our goal is to update network weights from the previous task ($\mathcal{W}_{t-1}$) to $\mathcal{W}_t$ such that

$$y \approx f(x; \mathcal{W}_t), \quad \text{for all } (x, y) \sim \mathcal{P}_t. \tag{1.1}$$

ITL setup above assumes that the task identity of test samples is known at the test time and the corresponding network weights are used for inference. In Chapter 3, we propose an

approach to represent, learn, and update $\mathcal{W}_t$ using low-rank factors such that they can be stored and applied with minimal memory and computation overhead.

1.2 Inverse Problems

Inverse problems comprises a broad category of problems that arise in different aspects of science and engineering. In different problems we observe signals via different modulations/ corruptions. It is called the measurement model. In inverse problems, we want to recover the true signals given the observed corrupted/modulated signals. Based on the nature of measurement model, inverse problems could be categorized into linear and nonlinear inverse problems.

1.2.1 Linear Inverse Problems

In linear inverse problems, we deal with linear measurement models. The problem can be formulated as the task of recovering a true signal $\mathbf{x} \in \mathbb{C}^n$ from measurements $\mathbf{y} \in \mathbb{C}^m$ and given the measurement model $\mathbf{A} \in \mathbb{C}^{m \times n}$, such that,

$$\mathbf{y} = \mathbf{A}\mathbf{x} + \eta, \tag{1.2}$$

where $\eta \in \mathbb{C}^m$ is the noise introduced during measurement acquisition process. Different image and video recovery problems fall under the category of linear inverse problems such as inpainting, denoising, deblurring, superresolution etc.

Usually $m < n$ which makes the problem ill-posed i.e. we have infinitely many solutions. We need to constraint the solution set to a smaller feasible set. Deep generative prior is one such constraint where the underlying assumption is that the target signal can

be generated by a deep generative network. Often trained generative models are used as a prior since their limited range force them feasible solution set to a smaller subset. However, a trained generator cannot represent data which does not fall in the distribution it has been trained on. In practice, it is difficult to have a trained generator for every dataset. Following the work of Ulyanov *et. al.* [14], there has been different studies on using untrained convolutional network as a generative prior. In Chapters 4, 5 and 6, we explore the efficient implementation of untrained generative prior for solving different linear inverse problems. We also impose low-rank matrix and tensor factorization on the input latent space of the generator as structural constraints for improved performance with compressed representation.

1.2.2 Nonlinear Inverse Problems

Nonlinear inverse problems comprise different nonlinear measurement model based problems. One such class of problems is phase retrieval. In phase retrieval, we acquire the magnitude of the signals available to us via measurement model. In this measurement process we lose the phase/sign information which makes the problem ill posed. General phase retrieval problems can be represented as

$$\mathbf{y} = |\mathbf{A}\mathbf{x}| + \eta, \tag{1.3}$$

where $|.|$ operation computes the elementwise magnitude. We used Gaussian measurement matrix for our experiments in Chapter 7 and Fourier measurement matrix in Chapter 8, 9 and 10. Fourier phase retrieval arises in different imaging applications where we observe the power spectral density of the signal which is essentially the Fourier magnitude squared

of the true signal. The flipped and shifted versions of an image produces the same Fourier magnitude. This phenomena is called the trivial ambiguities of Fourier phase retrieval which makes Fourier phase retrieval even more challenging.

In Chapter 7, we develop an efficient procedure to apply trained generative prior for phase retrieval problem. In Chapter 8, we demonstrate that having some side information about the signal helps us avoid trivial ambiguities and achieve better convergence with generative prior.

In Chapters 9 and 10, we discuss about two special applications of Fourier phase retrieval- Holographic Imaging and Coded Diffraction Imaging. We use classical iterative Alternating Minimization (AltMin) approach instead of generative prior for solving these two problems. We fix the number of iterations for AltMin and formulated it as an unrolling network structure with each layer of the network representing an iteration. We use such unrolling network to design sensing parameters for the imaging problems which show robustly superior performance with a very small number of iterations.

1.3 Organization

We discuss the background and related literature on continual learning and different inverse problems as well as different factorization approaches in Chapter 2. In Chapter 3, we present our work on incremental task learning with low-rank matrix factorization in the network. In Chapters 4, 5 and 6, we explore untrained generative models for solving different linear inverse problems and demonstrate the advantage of different low-rank matrix and tensor structures in the network and formulation. Chapters 7, 8, 9 and 10 are focused

on phase retrieval, especially on Fourier phase retrieval which is a special class of nonlinear inverse problems. In Chapters 7 and 8, we formulated an approach to use trained generative priors for solving phase retrieval problems and demonstrated how side information from our target signal helps us converge to a better solution. In Chapters 9 and 10, we formulate a network structure by unrolling a fixed iteration alternating minimization process to design sensing parameters for holographic and coded illumination imaging. We provide conclusion of this thesis in Chapter 11 along with a discussion on future directions.

Chapter 2

Background and Related Works

2.1 Continual Learning

Continual learning or lifelong learning approaches aim to address the problem of catastrophic forgetting by adapting the network or training process to learn new tasks without forgetting the previously learned ones [15, 16, 17, 18, 19, 20, 21, 22, 23]. To overcome this issue, different approaches have been proposed so far which can be divided into three main categories: regularization-based approaches, memory and replay-based approaches, and dynamic network architecture-based approaches.

2.1.1 Regularization-based approaches

Approaches in [24, 15, 16] update the whole model in each task but a regularization term ℓ_{reg} is added to the total loss $\mathcal{L} = \ell_{current} + \lambda \ell_{reg}$ to penalize changes in the parameters important to preceding tasks thus preserving the performance on previous learned tasks. For example, Elastic Weight Consolidation (EWC) [24] estimates the importance of parameters

using Fisher Information matrix; Variational Continual Learning (VCL) [15] approximates the posterior distribution of the parameters using variational inference; Learning without Forgetting (LwF) [16] regularizes the current loss with soft targets taken from previous tasks using knowledge distillation [25]. GCL [26] mixes rehearsal with knowledge distillation and regularization to mitigate catastrophic forgetting. A number of recently proposed methods force weight updates to belong to the null space of the feature covariance [27, 28].

2.1.2 Memory-based approaches

Approaches in [29, 21, 7, 30, 28] usually use memory and replay/rehearsal mechanism to recall a small episodic memory of previous tasks while training new tasks thus reduce the loss in the previous tasks. For example, iCaRL [29] is the first replay method, which learns in a class-incremental way by selecting and storing exemplars closest to the feature mean of each class; Meta-Experience Replay (MER) [21] combines experience replay with optimization-based meta-learning to optimize the symmetric trade-off between transfer and interference by enforcing gradient alignment across examples; AGEM [7] projects the gradient on the current minibatch by using an external episodic memory of patterns from previous experiences as an optimization constraint; ER-Ring [30] jointly trains new task data with that of the previous tasks.

2.1.3 Dynamic network architectures

Approaches in [31, 32, 33, 34, 35, 8, 36] try to add new neurons to the model at additional new tasks, thus the performances on previous tasks are preserved by freezing the old parameters and only updating the newly added parameters. For example, Progressive

neural networks (PNNs) [31] leverage prior knowledge via lateral connections to previously learned features; PackNet [32] iteratively assigns parameter subsets to consecutive tasks by constituting binary masks. SupSup [33] also finds masks in order to assign different subsets of the weights for different tasks. BatchEnsemble [34] learns on separate rank-1 scaling matrices for each task which are then used to scale weights of the shared network. HAT [35] incorporates task-specific embeddings for attention masking. [37] also proposes task-conditioned hypernetworks for continual learning. [38] proposes nonoverlapping sets of units being active for each task. Piggyback [39] learns binary masks on an existing network to provide good performance on new tasks. [40] proposes task specific convolutional filter selection for continual learning. The mask-based methods listed above provide excellent results for continual learning, but they require a significantly large number of parameters to represent the masks for each task. A factorization-based approach was proposed in [41] that performs automatic rank selection per task for variational inference using Indian Buffet process. The method requires significantly large rank increments per task to achieve high accuracy; in contrast, our method uses a learning-based approach to find rank-1 increments and reuse old factors with the learned selector weights. ORTHOG-SUBSPACE [8] learns tasks in different (low-rank) vector sub-spaces that are kept orthogonal to each other in order to minimize interference.

Dynamic architecture approaches have the potential to achieve *zero forgetting*, using task specific weights for testing data; however, this also requires storing the task specific weights for all the tasks. Our proposed low-rank matrix factorization method falls under the category of dynamic network architecture approaches where we store task specific factors

for every task. Since we are saving a low-rank factor per task, the parameter overhead for our approach is very small compared to other dynamic architecture approaches.

2.2 Deep Generative Models

Deep generative models offer a new framework for compact representation of images and videos. Generative adversarial networks (GANs) and variational autoencoders (VAEs) are two popular classes of deep generative networks that learn a function that maps vectors drawn from a certain distribution in a low-dimensional space into images in a high-dimensional space [42, 43, 44, 45]. For compact representation of images, we seek a generative model that can generate a variety of images with high fidelity using a very low-dimensional latent code. Recently, a number of generative models have been proposed to learn latent representation of an image with respect to a generator [46, 47, 48]. The learning process usually involves gradient decent to estimate the best representation of the latent code, where the gradients with respect to the latent code representation are backpropagated to the pixel space [49]. In recent year, generative networks have been extensively used for learning good representations for images and videos. Generative adversarial networks (GANs) and variational autoencoders (VAEs) [42, 43, 44, 45] learn a function that maps vectors drawn from a certain distribution in a low-dimensional space into images in a high-dimensional space. An attractive feature of VAEs [43] and GANs [42] is their ability to transform feature vectors to generate a variety of images from a different set of desired distributions.

2.2.1 Tensor Factorization Methods

Tensor factorization has been a very popular method for multidimensional data and complex network compression [50, 51, 52]. Because of the favorable formulation of tensor factorization, different components of factorization framework can be used as a knob to generate novel/missing data. Recently different fields [53, 54] are utilizing the advantage of such formulation of tensor factorization. In [55], authors show that tensor ring factorization for GAN and VAE can lead to better convergence. They showed simple interpolation on CelebA as well. However, they did not use tensor factorization for attribute mapping. [56] used PARAFAC decomposition in the latent space of VAE to model audience reaction to movies. They also used matrix completion to impute missing expressions. None of those methods used tensor factorization on deterministic autoencoder because deterministic autoencoders are not good at generation tasks due to holes in the latent space [57]. However, [57] showed that using additional constraint such as implicit low-rank constraint, even deterministic autoencoders can perform comparable generation as VAEs. Based on their findings, we are using the strength of tensor factorization to compress the latent space of the deterministic autoencoder using explicit low-rank constraints in order to use it as a generative prior. We are also using it to factorize the representation of structured set of images in the latent space to have better recovery performance.

2.3 Inverse Problems

Inverse problems comprise a wide variety of problems which can be broadly divided into two categories- linear and nonlinear inverse problems. Most of the linear inverse

problems can be termed as compressive sensing since the information in the observed signal is usually less than the true signal. Among different nonlinear inverse problems, we limit our scope of study to phase retrieval problem which is very common in different sensing applications.

2.3.1 Compressive Sensing

Compressive sensing refers to a broad class of problems in which we aim to recover a signal from a small number of measurements [58, 59, 60]. The canonical compressive sensing problem in (4.2) is inherently underdetermined, and we need to use some prior knowledge about the signal structure. Classical signal priors exploit sparse and low-rank structures in images and videos for their reconstruction [61, 62, 63, 64, 65]. However, the natural images exhibits far richer nonlinear structure than sparsity alone. So, we focus on a newly emerging family of generative priors that are usually learned from massive amount of training data.

Deep Generative Models for Compressive Sensing

The generative model and optimization problems we use are inspired by recent work on using generative models for compressive sensing in [66, 67, 68, 2, 69, 14, 49]. Compressive sensing using generative models was first introduced in [66], which used a trained deep generative network as a prior for image reconstruction from compressive measurements; the reconstruction problem involves optimization over the latent code of the generator. Since the generator is fixed, this approach works well only if the unknown image/video belongs to the range of the generator used. Deep image prior (DIP) is a related method in which an

untrained convolutional generative model is used as a prior for solving inverse problems such as inpainting and denoising because of their tendency to generate natural images [14]; the reconstruction problem involves optimization of generator network parameters. Inspired by these observations, a number of methods have been proposed for solving compressive sensing problem by optimizing generator network weights while keeping the latent code fixed at a random value [2, 68]. Both DIP [14] and deep decoder [2] update the network parameters to generate a given image; therefore, the generator can reconstruct wide range of images. One key difference between the two approaches is that the network used in DIP is highly overparameterized, while the one used in deep decoder is underparameterized.

2.3.2 Phase retrieval

The classical problem of phase retrieval arises in numerous imaging applications [70, 71], where only the magnitude of the light rays can be measured but not the phase. As each linear observation loses its phase, the highly non-linear forward model makes it challenging to recover the underlying signal.

Phase retrieval problems seek to recover real- or complex valued signals/images from their quadratic or amplitude measurements. Fourier phase retrieval is a particular instance of this problem that arises in optical coherent imaging, where we seek to recover an image from its Fourier modulus [72, 73, 74, 71]. This problem has been extensively studied over last five decades in optics, signal processing, and optimization [75, 70, 76, 77].

In general, infinitely many possible solutions exist for phase retrieval problems. Classical methods for phase retrieval rely on prior knowledge about the support and positivity of the images. In recent years, sparse, low-rank, and generative models with spectral

initialization have been proposed for various phase retrieval problems. Despite the progress, phase retrieval with structured measurements, still remains a challenging problem.

Existing algorithms for solving phase retrieval can be broadly classified into convex and non-convex approaches [78]. Convex approaches usually solve a constrained optimization problem after lifting the problem. The PhaseLift algorithm [79] and its variations [80], [81] belong to this class. On the other hand, non-convex approaches usually depend on Amplitude flow [82, 83] and Wirtinger flow [84, 85, 86, 87]. We can also incorporate prior knowledge about the signal structure (e.g., sparsity, support, or positivity) in the recovery process constraints [88, 89, 90, 91, 92, 87, 82].

Furthermore, [92, 93, 78] used minimization (AltMin)-based approach and [94] used total variation regularization to solve phase retrieval. Recently, various researchers have explored the idea of replacing the sparsity priors with generative priors for solving inverse problems. Some of the generative prior-based approaches can be found in [78, 95, 76, 96].

Holography

Digital holography is an interferometric imaging technique that does not require the use of any imaging lens. Utilizing the theory of diffraction of light, a hologram can be used to reconstruct three-dimensional (3D) images [97]. With this advantage, holography can be used to perform simultaneous imaging of multidimensional information, such as 3D structure, dynamics, quantitative phase, multiple wavelengths, and polarization state of light [98]. In the computational imaging community, many attempts have been made in solving holographic phase retrieval using references, among which [99] has been very successful. Motivated by the reference design for holographic phase retrieval, we are trying to explore a

way to design references for general phase retrieval.

In recent papers [99, 100, 101, 102], authors tried to use side information with sparsity prior to mitigate trivial ambiguities of Fourier phase retrieval. However, in those studies, the reference and target signal are separated by some margin. If the separation between target and reference is large enough, then the nonlinear PR problem simplifies to a linear inverse problem [103, 99]. In our proposed approaches, we do not enforce any such separation.

Coded Diffraction Imaging

Coded diffraction imaging is a physically realistic setup in which we can first modulate the signal of interest and then collect the intensity measurements [81, 104]. The modulation can be performed using a spatial light modulator or custom transparencies [105, 73, 106]. The recovery problem involves solving a phase retrieval problem; the presence of modulation patterns makes this a more tractable problem compared to classical Fourier phase retrieval [81].

Data-Driven Approaches for Phase Retrieval

The use of deep learning-based methods to solve computational imaging problems such as phase retrieval is becoming popular. Deep learning methods leverage the power of huge amount of data and tend to provide superior performance compared to traditional methods while also run significantly faster with the acceleration of GPU devices. A few examples demonstrating the benefit of the data-driven approaches include [107] for robust phase retrieval, [108] for Fourier ptychographic microscopy, and [109] for holographic image

reconstruction.

Unrolled Network for Inverse Problem

Unrolled networks, which are constructed by unrolled iterations of a generic non-linear reconstruction algorithm, have also been gaining popularity for solving inverse problems in recent years [110, 111, 112, 113, 114, 115, 116, 117, 118]. Iterative methods usually terminate the iteration when the condition satisfies theoretical convergence properties, thus rendering the number of iterations uncertain. An unrolled network has a fixed number of iterations (and cost) by construction and they produce good results in a small number of steps while enabling efficient usage of training data.

Learn to Sense

Deep learning methods have also been recently used to design the sensing system; especially in the context of compressive sensing and computational imaging [119, 120, 121, 122, 123, 124, 125, 126, 127]. The main objective in these methods is similar to ours, which is to select sensor parameters to recover best possible signal/image from the sensor measurements. The sensor parameters may involve selection of samples/frames, design of sampling waveforms, or illumination patters as we discuss in this work. In contrast to most of the existing methods that learn a deep network to solve the inverse problem, our method uses a predefined iterative method as an unrolled network while learning the illumination patterns using a small number of training images. Unrolled network for solving non-linear inverse problems has been used in [108, 111]. [108] proposes learning sensors for Fourier ptychographic microscopy. [111] designs sensing patterns for coded illumination imaging.

One might find similarity between [111] and our problem formulation. In principle, the sensor can be treated as the first layer of the network with some physical constraints on the parameters [111]. However, the method in [111] uses an unrolled network to learn the sensing parameters for quantitative phase imaging problem under the "weak object approximation". This approximation turns the original nonlinear problem into a linear inverse problem. This assumption is only applicable where the target objects have a small scatter term (e.g., biological samples in closely index-matched fluid). In our setup, we do not make any such assumptions on target object and solve the original nonlinear coded diffraction imaging problem. This potentially makes our algorithm suitable for more general applications than [111].

Chapter 3

Incremental Task Learning with Low-rank Increment

3.1 Introduction

Deep neural networks have been extremely successful for a variety of learning and representation tasks (e.g., image classification, object detection/segmentation, reinforcement learning, generative models). A typical network is trained to learn a function that maps input to the desired output. The input-output relation is assumed to be fixed and input-output data samples are drawn from a stationary distribution [129]. If the input-output relations or data distributions change, the network can be retrained using a new set of input-output data samples. Since the storage, computing, and network capacity are limited, we may need to replace old data samples with new samples. Furthermore, privacy concerns may also force data samples to be available for a limited time [4, 129]. In such a training process, a

This work has been accepted to European Conference on Computer Vision 2022 [128]

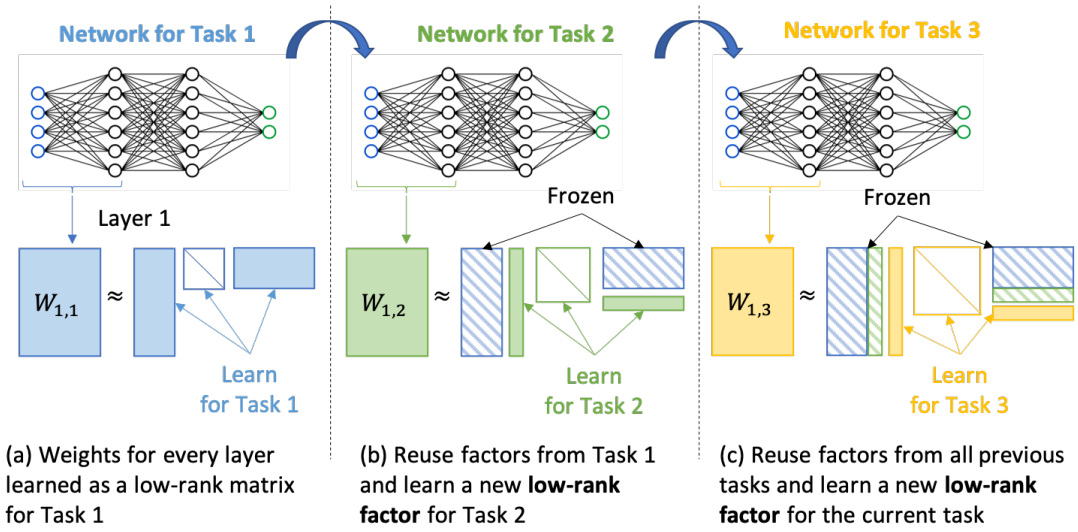


Figure 3.1: An overview of our proposed method for continual learning via low-rank network updates. We first represent (and learn) the weight matrix (or tensor) for each layer as a product of low-rank matrices. To train a network for new tasks without forgetting the earlier tasks, we reuse the factors from the earlier tasks and add a new set of factors for the new task. Our experiments suggest that a rank-1 update is often sufficient for successful continual learning.

network often forgets the previously learned tasks; this effect is termed *catastrophic forgetting* [6, 130].

We propose a new method for Incremental Task Learning (ITL) that updates network weights using rank-1 (or low-rank) increments for every new task. Figure 3.1 provides an illustration of our proposed method. We represent the network weights for each layer as a linear combination of several low-rank factors (which can be represented as a product of two low-rank matrices and a diagonal matrix). To update the network for task t without forgetting the earlier tasks, we freeze the low-rank factors learned from the previous tasks, add a new trainable rank-1 (or low-rank) factor for every layer, and combine that

with the older factors using learnable *selector weights* (shown as a diagonal matrix). We use a multi-head configuration that has an independent output layer for each task. As we are learning separate diagonal matrices for every task, we can achieve zero forgetting during inference. We present an extensive set of experiments to demonstrate the performance of our proposed method for different benchmark datasets. We observe that our proposed method outperforms the current state-of-the-art methods in terms of accuracy with small memory overhead.

The main contributions of this work [128] are as follows.

(a) **Represent layers as low-rank matrices:** We represent and learn network weights for each layer as a low-rank structure. We show that low-rank structure is sufficient to represent all the tasks in continual learning setup.

(b) **Reuse old factors for better performance with a small memory overhead:** We limit the number of parameters required for network update by reusing the factors learned from previous tasks. We demonstrate that a rank-1 increment suffices to outperform the existing techniques.

(c) **Zero forgetting without replay buffer:** Our method has zero forgetting that is achieved using incremental rank update or network weights. In contrast, most of the existing continual learning techniques require replay buffer or large memory overhead to achieve zero forgetting.

Limitations. Our method shares same inherent limitation of ITL (i.e. the requirement of task-id during inference). In addition, since we use all the previously learned factors for inference, the later tasks require more memory and computation for inference. Nevertheless,

we show that using low-rank structure, our total memory requirement is significantly lower than a single network. Furthermore, as we learn separate diagonal matrices for each task, we can maintain high performance even if the network reaches full rank with a large number of tasks.

3.2 Incremental Task Learning via Rank Increment

We focus on the incremental task learning setup in which we seek to train a network for T tasks. The main difference between incremental task learning and regular learning is that the training data for every task is only available while training the network for that task. The main challenge in incremental task learning is to not forget the previous tasks as we learn new tasks. Learning each task entails training weights for the network to learn the task-specific input-output relationship using the task-specific training data.

We seek to develop an ITL framework in which we represent the weights of any layer using a small number of low-rank factors. We initialize the network with a base architecture in which weights for each layer can be represented using a low-rank matrix. We then add new low-rank factors to each layer as we learn new tasks.

Let us assume the network has K layers and the weights for the kth layer and task t can be represented as $W_{k,t}$. Let us further assume that the weights for the kth layer and task $t = 1$ can be represented as a low-rank matrix

$$W_{k,1} = U_{k,1} S_{k,1,1} V_{k,1}^{\top}, \tag{3.1}$$

where $U_{k,1}, V_{k,1}$ represent two low-rank matrices and $S_{k,1,1}$ represents a diagonal matrix. To learn the network for task 1, we learn $U_{k,1}, V_{k,1}, S_{k,1,1}$ for all k. For task 2, we represent the

weights for kth layer as

$$W_{k,2} = U_{k,1} S_{k,1,2} V_{k,1}^\top + U_{k,2} S_{k,2,2} V_{k,2}^\top.$$

$U_{k,1}, V_{k,1}$ represent the two low-rank matrices learned for task 1 and frozen afterwards. $U_{k,2}, V_{k,2}$ represent two low-rank matrices that are added to update the weights, and these will be learned for task 2. $S_{k,1,2}, S_{k,2,2}$ represent the diagonal matrices, which will be learned for task 2. We learn $S_{k,1,2}$, which is a diagonal matrix that assigns weights to factors corresponding to task 1, to include/exclude or favor/suppress frozen factors from previous tasks for the new tasks. We can represent the weights for the kth layer and task t as

$$\begin{aligned} W_{layer,task} = W_{k,t} &= \sum_{i \le t} U_{k,i} S_{k,i,t} V_{k,i}^\top \\ &= \sum_{i<t} \underbrace{U_{k,i}}_{\text{frozen}} S_{k,i,t} \underbrace{V_{k,i}^\top}_{\text{frozen}} + U_{k,t} S_{k,t,t} V_{k,t}^\top, \end{aligned} \tag{3.2}$$

where $U_{k,i}, V_{k,i}$ are frozen for all $i < t$ and $U_{k,t}, V_{k,t}$ and all the $S_{k,i,t}$ are learned for task t. The entire network for task t can be represented as $\mathcal{W}_t = \{U_{k,i}, S_{k,i,t}, V_{k,i}\}_{i \le t}$. To update the trainable network parameters for task t, we solve the following optimization problem:

$$\begin{aligned} &\min_{U_{k,t}, S_{k,i,t}, V_{k,t}} \sum_{(x,y) \in (\mathcal{X}_t, \mathcal{Y}_t)} \text{loss}(f(x; \mathcal{W}_t[U_{k,t}, S_{k,i,t}, V_{k,t}]), y) \\ &\qquad \text{for all } k \le K \text{ and } i \le t, \end{aligned} \tag{3.3}$$

where we use $\text{loss}(\cdot, \cdot)$ to denote the loss function and $\mathcal{W}_t[U_{k,t}, S_{k,i,t}, V_{k,t}]$ to indicate the trainable parameters in $\mathcal{W}_t$, while the rest are frozen. We sometimes call $S_{k,i,t}$ a *selector weight matrix/vector* to indicate that its diagonal entries determine the contribution of each factor toward each task/layer weights.

Our proposed ITL algorithm works as follows. We train the low-rank factors for the given task using the respective training samples. Then we freeze the factors corresponding to the older tasks and only update the new factors and the diagonal matrices. In this manner, the total number of parameters we add in our model is linearly proportional to the rank of the new factors. To keep the network complexity small, we seek to achieve good accuracy using small rank for each task update and layer. We summarize our approach in Algorithms 1 and 2.

Note that we do not need to create the weight matrix $W_{k,t}$ for any layer explicitly since we can compute all the steps in forward and backward propagation efficiently using the factorized form of each layer. The size of each layer is determined by the choice of the network architecture. The rank of each layer for every task is a hyper-parameter that we can select according to the tasks at hand. To keep the memory overhead small, we need to use small values for rank increment. Let us denote the rank for $U_{k,t}$ as $r_{k,t}$, which represents the increment rank for kth layer and task t. At the time of test, we can use an appropriate number of factors depending on the task. For instance, if we want to predict output for task 1 then we use first $r_{k,1}$ factors and for task 2 we use $r_{k,1} + r_{k,2}$ factors. We can add new factors in an incremental manner as we add new tasks in the ITL setup. In the extreme case of rank-1 increments, $r_{k,t} = 1$. In our experiments, we observed that rank-1 updates compete or exceed the performance of existing ITL methods (see Table 3.1) and the performance of our method improves further as we increase the rank (see Table 3.5). Any increase in the rank comes at the expense of an increased memory overhead.

Algorithm 1 ITL with rank-1 increments (Training)

Input: Data ($\mathcal{X}_1$ and $\mathcal{Y}_1$) for the 1^{st} task.

Set initial rank, r_1.

Initialize weight factors $U_{k,1}, V_{k,1}$ at random and $S_{k,1,1}$ as an identity marix.

Learn $U_{k,1}, V_{k,1}$ and $S_{k,1,1}$. ▷ Optimization in (3.3)

for $t = 2, 3, ..., T$ **do**

Input: Training data ($\mathcal{X}_t$ and $\mathcal{Y}_t$) for t^{th} task.

Initialize low-rank update factors $U_{k,t}, V_{k,t}$.

Freeze the previous factors $\{U_{k,i}, V_{k,i}\}_{i<t}$.

Initialize the diagonal entries of $\{S_{k,i,t}\}$ as 1

for $i = t$ and 0 for $i < t$.

Learn $U_{k,t}, V_{k,t}$ and $S_{k,i,t}$

for $i < t$. ▷ Optimization in (3.3)

end for

3.3 Experiments and Results

We used different classification tasks on well known continual learning benchmarks to show the significance of our proposed approach.

3.3.1 Datasets and Task Description

Experiments are conducted on four datasets: Split CIFAR100, Permuted MNIST, Rotated MNIST, and Split MiniImageNet.

P-MNIST creates new tasks by applying a certain random permutation on the pixels of all

Algorithm 2 ITL with rank-1 increments (Inference)

Input: Test data x with task identity t.

Retrieve trained weights: $\mathcal{W}_t = \{U_{k,i}, V_{k,i}, S_{k,i,t}\}$ for all k and $i \leq t$.

Output: Calculate the network output as $f(x, \mathcal{W}_t)$.

images in the original dataset. In our experiment, we generate 20 different tasks, each of which corresponds to a certain but different permutation.

R-MNIST is similar to Permuted MNIST, but instead of applying a certain random permutation on the pixels, it applies a certain random rotation to the images in the same tasks. We create 20 different tasks, each corresponds to a certain but different version of rotation from [0, 180] degree interval.

S-CIFAR100 splits the original CIFAR-100 dataset into 20 disjoint sets, each of which, containing 5 classes, is considered as a separate task. The 5 classes in each task is randomly chosen without replacement from the total 100 classes.

S-miniImageNet splits a subset of Imagenet dataset into 20 disjoint sets, each of which, containing 5 classes, is considered as a separate task. The 5 classes in each task is randomly chosen without replacement from the total 100 classes.

3.3.2 Training Details

Network. In the first set of experiments, we used a three layer (fully-connected) multilayer perceptron (MLP) with 256-node hidden layers, similar to the network in [8]. We flattened multi-dimensional input image to a 1D vector input. We used ReLU activation for all the layers except the last one. We used Softmax for the muticlass classification tasks. We used the same network for all the tasks with necessary modifications for input and output sizes.

Our approach can be used in convolutional networks as well. We report the results using ResNet18 with our approach on S-CIFAR100 and S-miniImageNet dataset in Table 3.6.

Factorization and rank selection. We used the matrix factorization defined in (3.2) in all our experiments. We empirically selected the rank for the first task,$r_{k,1}$ as 11 based on the experiments on a sample Rotated MNIST task and kept the same value for all the experiments. We then performed rank-1 increment ($r_{k,t}$) for each additional task. We would like to point that AGEM and Orthog Subspace use first 3 tasks for hyperparameter tuning. We did not tune our hyperparameters on the test data, rather we choose the parameters which provides better convergence during training. We increment the weight matrices by rank-1 per task; therefore, learning rate and the number of epochs are the only hyperparameters in our experiments.

Optimization. We used orthogonal initialization for the low-rank factors, as described in [131]. We used all one initialization for the additional factors of the selector matrices $S_{k,t,t}$. We used Adam optimization to update the factors. We used the batch size of 128 for each task.

Performance metrics. We use *accuracy* and *forgetting* per task, which are two commonly used metrics in the continual learning literature [20, 8], to evaluate the performance of the described methods. Let $a_{t,j}$ be the test accuracy of task $j < t$ after the model has finished learning task $t \in \{1, ..., T\}$ in a incremental manner. The average accuracy A_t after the model has learned task t is defined as $A_t = \frac{1}{t}\sum_{j=1}^{t} a_{t,j}$. On the other hand, *forgetting* is the decrease in the accuracy of a task after its training, and after one or several tasks are learned incrementally. We define the average forgetting F_t as $F_t = \frac{1}{t-1}\sum_{j=1}^{t-1}(a_{j,j} - a_{t,j})$.

In Figure 3.2, we show the evolution of average accuracy A_t as t increases. We also show the evolution of task-wise accuracy $a_{t,j}$ in Figure 3.3, where (t, j) pixel intensity reflects $a_{t,j}$. We report the average accuracy A_T, the average accuracy after the model has learnt every tasks incrementally, in Table 3.1. We report the forgetting F_T after the model has learnt all the tasks incrementally in Table 3.2. Note that our method performs incremental task learning without forgetting.

3.3.3 Comparing Techniques

We compare our method against different state-of-the-art ITL methods. **EWC** [24] is a regularization-based method that uses the Fisher Information matrix to estimate posterior of previous tasks to preserve important parameters. **ICARL** [29] is a memory-based method that uses exemplars and knowledge distillation [25] to retain previous knowledge. **AGEM** [7] is a memory-based method built upon [132] that uses episodic memory to solve an constrained optimization problem. **ER-Ring** [30] is another memory-based method that jointly trains on new task data with that of the previous tasks. **Orth. sub.** [8] learn tasks in different (low-rank) vector subspaces that are kept orthogonal to each other in order to minimize interference. Other than the above mentioned approaches, we compared with masked based approaches which, like our approach, also fall under dynamic architecture category. **HAT** [35] that incorporates task-specific embeddings for attention masking. **PackNet** [32] that iteratively assigns subsets of a single binary mask to each task. The mask-based approaches utilize the redundancy of the network parameters to represent different tasks with different masked versions of the same network weights. We also present comparisons with some recent methods: **IBP-WF** [41], **DER** [26] and **Adam-NSCL** [27],

in terms of average accuracy for one experiment on two datasets.

In addition, we report results for two *non-continual* baseline methods: **Parallel learning** and **Multitask learning**. **Parallel learning** trains independent (smaller) low-rank networks of same size for each task. We report results for three such networks. **Parallel 2** uses rank-2 layers, **Parallel 4** uses rank-4 layers, and **Parallel full** uses a full-rank MLP. Parallel 2 requires approximately the same number of parameters as the rank-1 ITL network that we use in our experiments; Parallel 4 provides higher network capacity, while requiring fewer parameters than the full-rank network. We can treat the performance of the **Parallel full** approach as the upper limit that we can achieve using ITL methods. Finally, **Multitask learning** has been used as a baseline in [8, 7]. In multitask learning, we have access to all data to optimize a single network.

3.3.4 Results with Three-Layer MLP

Classification performance and comparison. We report classification results for P-MNIST, R-MNIST, S-CIFAR100, and S-miniImageNet tasks in Table 3.1. We also show the results for the comparing techniques. We observe that our method with rank-1 update perform better than all the comparing methods (EWC, ICARL, AGEM, HAT, PackNet, Orthog Subspace) on R-MNIST, S-CIFAR100 and S-miniImageNet tasks using significantly fewer number of parameters. Our method performs close to Orthog Subspace on P-MNIST tasks.

We also observe that the proposed rank-1 update outperforms non-continual Parallel

Table 3.1: Average test accuracy of ITL for P-MNIST, R-MNIST, S-CIFAR100, and S-miniImageNet with three layer MLP. Standard deviation of test accuracy over five runs is shown in parenthesis for some of the experiments.

* Orthog subspace does not use replay buffer for MNIST variations.

Method	Replay Buffer	P-MNIST	R-MNIST	S-CIFAR100	S-miniImageNet
EWC [24]	No	67.9 (±0.68)	44.5 (±1.09)	52.7 (±0.81)	29.3 (±1.08)
ICARL [29]	Yes	85.4 (±0.01)	51.2(± 2.41)	56.9(±0.31)	39.9(±0.27)
AGEM [7]	Yes	73.9 (±0.52)	53.4 (±1.80)	51.3(±1.54)	31.3(±0.89)
HAT [35]	No	**90.1**(±1.60)	89.1(±2.51)	64.8 (±0.32)	47.0 (±0.88)
PackNet [32]	No	90.0(±0.24)	88.4(±0.37)	63.7(±0.41)	45.1(±1.05)
Orth sub [8]	Yes*	86.6 (±0.79)	80.2 (±0.41)	57.8 (±1.03)	38.1 (±0.67)
DER [26]	Yes	–	–	48.21	33.19
Adam-NSCL[27]	No	–	–	64.26	47.32
IBP-WF [41]	No	–	–	53.22	40.52
Ours	No	85.6 (±0.15)	**91.1** (±0.12)	**65.9** (±2.16)	**54.7** (±2.87)
Parallel 2 (r=2)	-	65.3	65.5	62.8	55.4
Parallel 4 (r=4)	-	86.3	87.4	65.6	58.6
Parallel fullrank	-	95.9	97.3	73.1	63.1
Multitask	-	96.8	97.7	16.4	4.21

2 baseline that has similar number of parameters compared to our approach. We perform similar to Parallel 4 baseline that uses nearly twice the number of parameters as our approach. Parallel full acts as an upper limit with the network structure of our choice as it trains independent full rank networks for every task. Multitask learning is another non-continual baseline that uses all the data from all the tasks simultaneously. Table 3.1 suggests that our ITL method can learn complex tasks such as CIFAR100 and miniImageNet classification with a three layer MLP, whereas multitask learning (which is solving 100-class classification problem) fails with such a simple network. We also tested Resnet18 network, which has significantly more parameters than the network used in Table 3.1. The results for Resnet18

Table 3.2: Average forgetting results corresponding to Table 3.1 for different datasets using different approaches. We report the forgetting in percentage unit (%). We also report the standard deviation over 5 experiments for some methods.

	EWC	AGEM	Orthog subspace	DER	Adam-NSCL	Parallel fullrank, HAT, PackNet **Ours**, IBP-WF
P-MNIST	25.8 (±0.70)	19.6 (±0.64)	4.49 (±0.93)	-	-	0
R-MNIST	52.9 (±1.17)	44.2 (±1.85)	14.7 (±0.39)	-	-	0
S-CIFAR100	6.96 (±0.80)	21.5 (±2.89)	6.30 (±0.38)	10.6	8.5	0
S-miniImageNet	17.3 (±1.81)	18.8 (±1.40)	9.98 (±0.31)	20.11	11.23	0

are presented in Table 3.6.

We present the task-wise test performance for some of the comparing approaches on P-MNIST, R-MNIST, S-CIFAR100 and S-miniImageNet datasets in Figure 3.2. We observe that as we train new tasks, task-wise performance drops for the comparing approaches, especially for P-MNIST and R-MNIST.

ICARL and AGEM require replay buffer (episodic memory) for each task. Although Orthog Subspace did not use replay buffer for MNIST experiments, it requires replay buffer in their algorithm and used it for S-CIFAR100 and S-miniImageNet experiments. EWC does not require any replay buffer, but it suffers from high forgetting as shown in Figure 3.3. Our proposed approach does not require a replay buffer, and it outperforms other approaches in Table 3.1.

Accuracy vs forgetting. We report the average forgetting of different comparing approaches in Table 3.2. Our method, mask-based approaches (HAT and PackNet) and parallel baselines have zero forgetting, whereas all other comparing methods exhibit some level of forgetting. To better demonstrate the forgetting, in Figure 3.3, we show the accuracy for the tasks along the entire training procedure. i^{th} row (top-bottom) of the diagram denotes

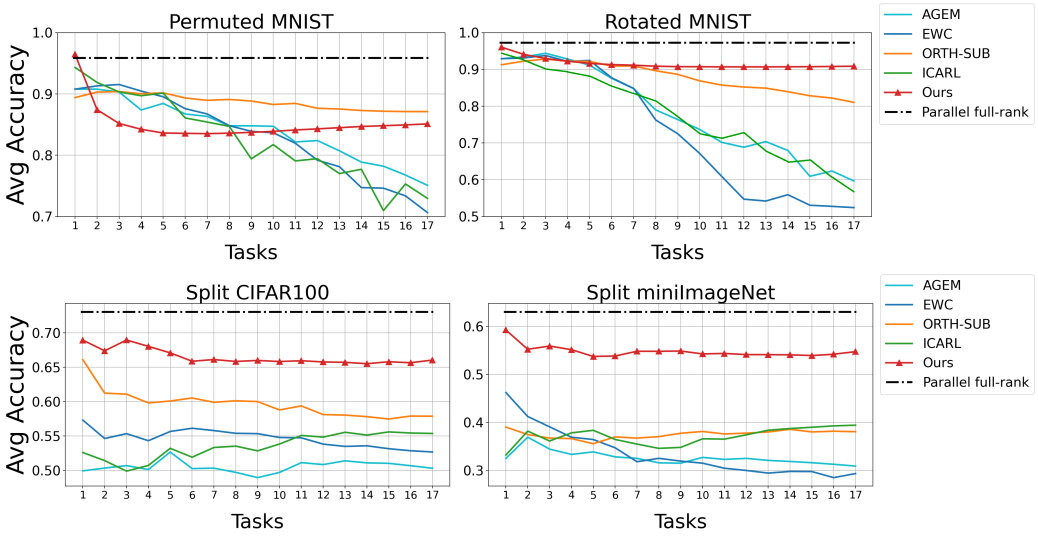


Figure 3.2: Average test accuracy for different datasets (Permuted MNIST, Rotated MNIST, Split CIFAR100, Split miniImageNet) along different tasks using different algorithms (AGEM,EWC, Orthog. Subspace, ICARL and our approach). We use three layer MLP here. Parallel full-rank results corresponds to the case when we train every task on separate full rank networks independently (serves as an upper limit for ITL methods). We showed the average of 20 tasks.

the performance of i tasks on the test sets when we train the i^{th} task. As expected, we can observe that the training performance for the previously learned tasks usually drops with the gradual training of the subsequent tasks specially for the regularization based approach, EWC. However, our algorithm maintains the same performance for the past tasks as we do not change any previously learned factors. Even orthogonal subspace approach observes such forgetting over some tasks.

Memory complexity. Our method increments the rank of each layer for each task; therefore, we compare the total number of parameters in the incrementally trained network and the Parallel baselines. Note that if the number of parameters in two approaches is same, we can train one small network per task independently. We report total number of

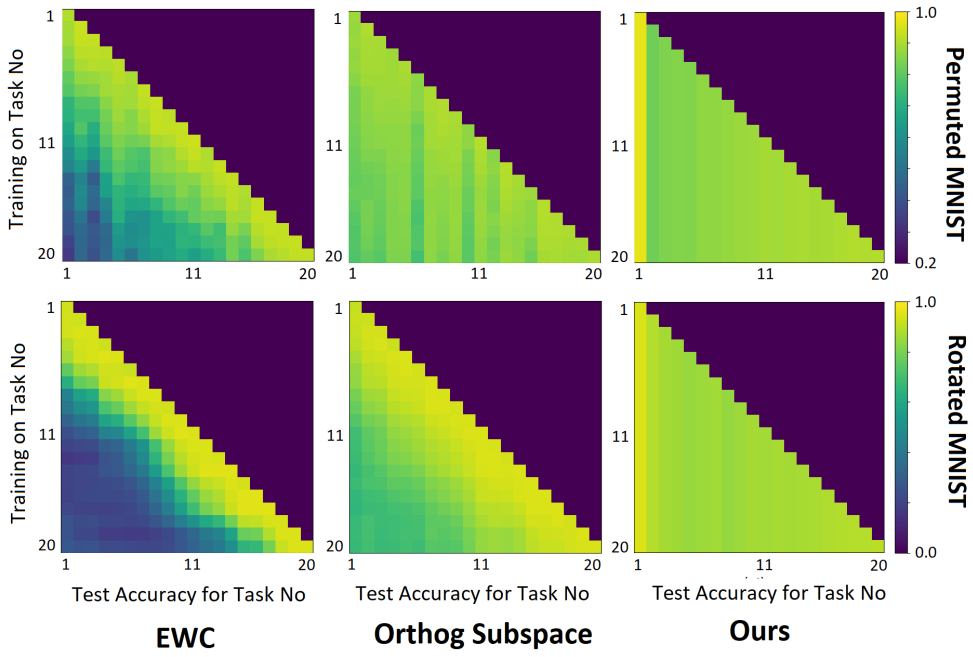


Figure 3.3: Evolution of task-wise test accuracy on P-MNIST (first row) and R-MNIST (second row) datasets for EWC, Orthogonal Subspace, and Our approach. We can observe from the decrease in the test accuracy that EWC and Orthogonal Subspace forget the previous tasks as they learn new tasks. Our approach does not show any forgetting as we learn new tasks.

parameters and replay buffer size for different methods in Table 3.3. Since we used similar fully connected network structure for all the tasks, we report results for Split CIFAR100 experiments. Although we increase the rank for every task, the increment is small enough that even after 20 tasks our total parameter count remains smaller than all other methods.

We also report the number of parameters used by mask-based zero forgetting algorithms (HAT and PackNet) to learn 20 different tasks on different datasets in Table 3.4. We can observe that our approach outperforms HAT and PackNet for R-MNIST, S-CIAR100 and S-miniImageNet with a significantly smaller number of parameters. Even though all

Table 3.3: Number of parameters and buffer size in ITL methods with 3-layer MLP.

	Ours	IBP-WF	EWC	AGEM	Ortho Sub	DER	Adam-NSCL	Para. full.
# params.	**0.17M**	0.23M	0.93M	1.76M	2.82M	0.88M	0.88M	19.7M
buffer size	0	0	1.71M	7.90M	9.01M	6.14M	0	0

Table 3.4: Number of parameters used by different zero-forgetting algorithms (HAT, PackNet, and Ours) using 3-layer MLP.

Method	P/R-MNIST	S-CIFAR100	S-miniImageNet
HAT	0.33M	0.89M	5.51M
PackNet	0.26M	0.83M	5.50M
Ours	0.11M	0.17M	0.72M

the approaches use the same network, our approach uses rank-1 factors that require a significantly smaller number of parameters for incremental learning of tasks. Note that P-MNIST and R-MNIST experiments require the same number of parameters.

Effect of rank. In Table 3.5, we evaluate the effect of different rank selection for different MNIST datasets using our ITL approach. We tested the initial rank (rank for the first task) of 1, 6, and 11, keeping the rank increment to 1. We observed that the accuracy increase as the initial rank increases, and we achieve nearly 90% accuracy with initial rank of 11. We also tested different values of rank increment per task and observe that the accuracy increases with larger rank increment. Nevertheless, rank-1 increment provides us comparable or better performance than the comparing techniques as shown in Table 3.1.

3.3.5 Results with ResNet18

The proposed low-rank increments approach can be generalized to other type of networks and layers as well. For example, convolutional kernels have four-dimensional weight

Table 3.5: Test accuracy for different rank choices of the proposed ITL approach and multi-task baseline networks for P-MNIST and R-MNIST. Initial rank is $r_{k,1}$ and rank increment/task is $r_{k,t}$.

Setup	1	2	3	4	5
$(r_{k,1}, r_{k,t})$	(1,1)	(6,1)	(11,1)	(11,2)	(11,4)
P-MNIST	74.23	82.21	85.61	90.51	93.84
R-MNIST	81.57	89.39	91.09	92.76	94.12
# parameters	0.09M	0.1M	0.11M	0.14M	0.2M

tensors as opposed to the two-dimensional weight matrices of fully connected layers. They are usually formulated as a tensor of output and input channel (C_{out}, C_{in}), and the two dimensions of the convolutional filters (H, W). We reshape the convolutional weight tensors into matrices of size $C_{out} \times C_{in}HW$ and perform similar low-rank updates per task as we described for the MLP. We report the results for S-CIFAR-100 and S-miniImageNet datasets with Resnet18 architecture. For each convolutional layers, we reshaped and decomposed the convolution weight tensors into the same low-rank factors described in (3.2) and performed low-rank updates per tasks. We report the results in Table 3.6. For most of the comparing techniques, results from [8] are reported since we use the same architecture and dataset. For missing comparisons, we trained the models using same procedure as outlined in [8].

Instead of using a fixed value for rank at each layer as we did in the MLP setup, we used rank size that is proportional to the size of $C_{out,i}$ at i^{th} convolutional layer because the weights for different layers of ResNet18 are different in size. We select initial rank = $0.1\, C_{out,i}$ for the first task and incremental rank = $0.02\, C_{out,i}$ for the subsequent incremental tasks.

The results in Table 3.6 show that the performance of every method improves

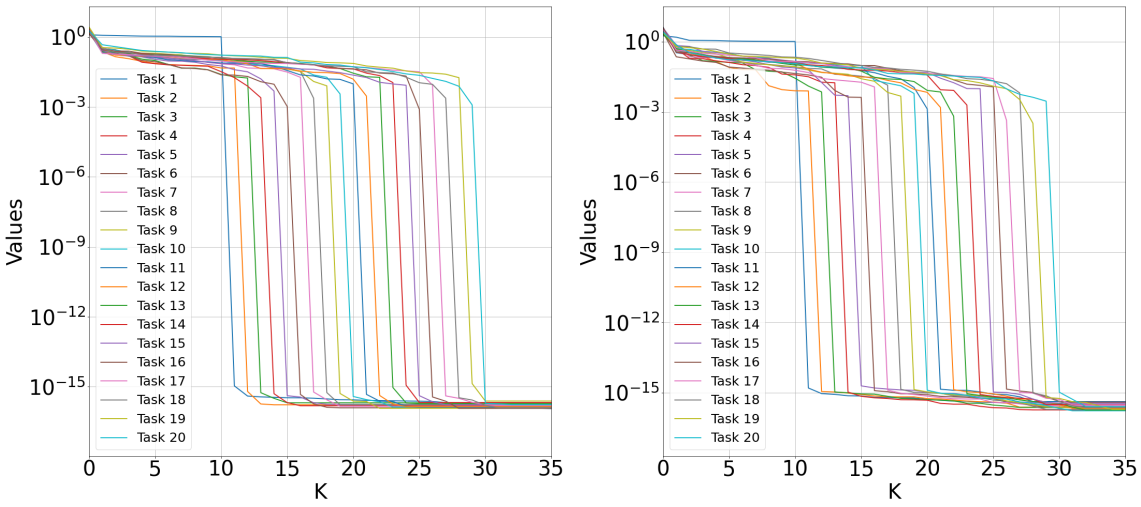
with the convolutional ResNet18 structure over the 3-layer MLP. Nevertheless, our method outperforms the comparing approaches for both datasets. Adam-NSCL [27] gets better results on CIFAR100, but it requires 11.21M parameters (compared to 1.33M parameters required by our method).

Table 3.6: Comparison of test accuracy and forgetting for split CIFAR-100 and split miniImageNet datasets using ResNet18 architecture.

Method	S-CIFAR-100		S-miniImageNet	
	Accuracy	Forgetting	Accuracy	Forgetting
EWC [24]	43.2 (±2.77)	26 (±2)	34.8 (±2.34)	24 (±4)
ICARL [29]	46.4 (±1.21)	16 (±1)	44.2	24.64
AGEM [7]	60.34 (±2.05)	11.0 (±2.88)	42.3 (±1.42)	17 (±1)
ER-Ring [30]	59.6 (±11.9)	14 (±1)	49.8 (±2.92)	12 (±1)
Ortho sub [8]	63.42 (±1.82)	8.37 (±0.71)	51.4 (±1.44)	10 (±1)
DER [26]	67.16	8.95	57.81	14.70
Adam-NSCL [27]	**74.31**	9.47	57.92	13.42
IBP-WF [41]	68.25	0	55.84	0
Ours	68.46 (±2.52)	**0**	**59.26** (±1.15)	**0**
Parallel full-rank	92.7	0	94.5	0
Multitask learning	70.2	0	65.1	0

3.3.6 Effect of updating last few layers.

We performed an experiment on S-CIFAR-100 where we factorize last L layers of the ResNet18 architecture keeping the rest of the network fixed at trained weights on Task 1. Updating last $L = \{1, 2, 3, 4, 5\}$ layers provide average accuracy of $\{34.38, 34.99, 53.41, 57.08, 65.03\}$, respectively. This result suggests that updating last few layers may suffice since the initial layers merely work as a feature extractor.



(a) Layer 1 **(b)** Layer 2

Figure 3.4: Top K singular values of weight matrices corresponding to different tasks for S-CIFAR100 with MLP experiments.

3.3.7 Relationship between the newly learned rank-1 weights and the fixed weights learned in the previous task

The newly learned rank-1 factors are not orthogonal to previous factors, but they are linearly independent. To demonstrate this relation, we plot the top singular values for weight matrices in two layers corresponding to 20 tasks in S-CIFAR100 in Fig. 3.4. The rank of the weight matrices starts at 11 and increases by one for every task. We observed similar trend in other tasks and layers. This suggests that the learned factors are linearly independent of frozen factors.

3.3.8 Effect of task similarity

Our experiments suggest that a positive knowledge transfer allows low-capacity models to perform well. We performed an experiment by selecting superclasses of CIFAR100 as separate tasks. If all classes in a task become similar (harder classification), the cross-task similarity reduces. We observe ∼60% accuracy for rank-1 ITL and Parallel rank-2. If we select tasks by sampling classes in each task at random, then cross-task similarity increases. We observe ∼65% accuracy for rank-1 ITL.

Chapter 4

Low-rank Generative Networks for Linear Inverse Problems

4.1 Introduction

Deep generative networks, such as autoencoders, generative adversarial networks (GANs), and variational autoencoders (VAEs), are now commonly used in almost every machine learning and computer vision task [42, 134, 135, 136]. One key idea in these generative networks is that they can learn to transform a low-dimensional feature vector (or latent code) into realistic images and videos. The *range* of the generated images is expected to be close to the true underlying distribution of training images. Once these networks are properly trained (which remains a nontrivial task), they can generate remarkable images in the trained categories of natural scenes.

In this work [137, 138], we propose to use a deep generative model for compact

This work has been published in IEEE Transactions on Signal Processing [133].

representation and reconstruction of videos from a small number of linear measurements. We assume that a generative network structure is available, which we represent as

$$x = G_\gamma(z) \equiv g_{\gamma_L} \circ g_{\gamma_{L-1}} \circ \cdots \circ g_{\gamma_1}(z). \tag{4.1}$$

$G_\gamma(z)$ denotes the overall function for the deep network with L layers that maps a low-dimensional (latent) code $z \in \mathbb{R}^k$ into an image $x \in \mathbb{R}^n$ and $\gamma = \{\gamma_1, \ldots, \gamma_L\}$ represents all the trainable parameters of the deep network. $G_\gamma(\cdot)$ as given in (4.1) can be viewed as a cascade of L functions g_{γ_l} for $l = 1, \ldots, L$, each of which represents a mapping between input and output of the respective layer. An illustration of such a generator with $L = 5$ is shown in Figure 4.1.

We consider a general problem of recovering a video sequence from its linear measurements. Suppose we are given a sequence of measurements for $t = 1, \ldots, T$ as

$$y_t = A_t x_t + e_t, \tag{4.2}$$

where x_t denotes the t^{th} frame in the unknown video sequence, y_t denotes its observed measurements, A_t denotes the respective measurement operator, and e_t denotes noise or error in the measurements. Our goal is to recover the video sequence (x_t) from the available measurements (y_t). The recovery problem becomes especially challenging as the number of measurements (in y_t) becomes very small compared to the number of unknowns (in x_t). To ensure quality reconstruction in such settings, we need a compact (low-dimensional) representation of the unknown signal. Thus, we use the given generative model to represent the video sequence as $x_t = G_\gamma(z_t)$ and seek to recover the unknown sequence x_t by optimizing over x_t, z_t, and γ.

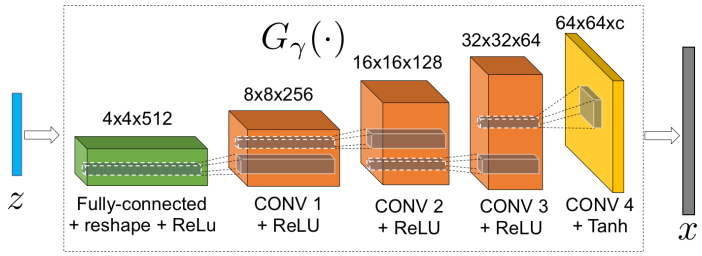


Figure 4.1: A candidate architecture we use in our experiments with one fully connected and four fractionally strided convolutional layers. Generative model: $x = G_\gamma(z)$ maps a vector $z \in \mathbb{R}^k$ into an image $x \in \mathbb{R}^n$.

We demonstrate that even if we do not have a pretrained generative network, we can still reconstruct video frames from their corrupted measurements. We use a generative model, as described in (4.1), to find compact representation of videos in the form of z_t. To reconstruct a video sequence from the compressive measurements in (4.2), we jointly optimize over the latent codes z_t and the network parameters γ. Since the frames in a video sequence exhibit rich redundancies in their representation, we impose a low-rank constraint on the latent codes to represent the video sequence with a more compact representation of the latent codes. We observe that when we optimize over latent code alongside network weights, the temporal similarity in the video frames is reflected in the latent code representation. To exploit similarities among the frames in a video sequence, we also include low-rank constraints on the latent codes. An illustration of different representations we use in this chapter are shown in Figure 4.2.

Untrained generative priors have been studied for image reconstruction in Deep Image Prior (DIP) [14] and Deep Decoder[2]. We observe two main limitations in the DIP and deep decoder-based video recovery that we seek to address in this chapter. (1) The

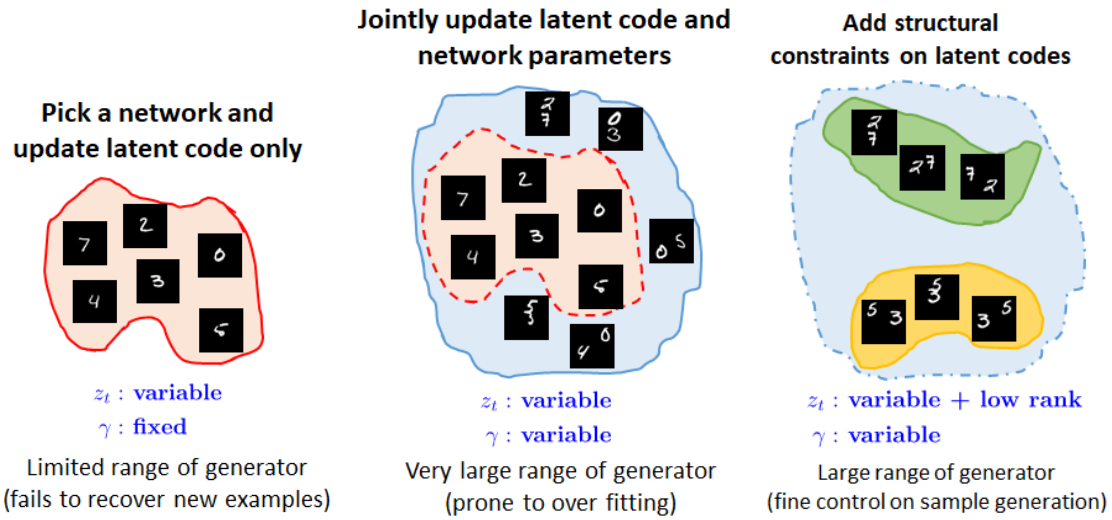


Figure 4.2: An illustration of different generative priors discussed in the chapter: (a) Optimizing latent codes can only reconstruct images in the range of the generative network. (b) Jointly optimizing latent code and network weights enables recovery of a larger range of images. (c) Low-rank and similarity constraints on latent code further regularize the problem and potentially explain other structures in data.

latent codes in DIP and deep decoder methods are initialized at random and stay fixed throughout the recovery process. Therefore, we cannot infer the structural similarities in the images from the structural similarities in the latent codes. (2) Both of these methods train one network per image. A naïve approach to train one network per frame in a video will be computationally prohibitive, and if we train a single network to generate the entire video sequence, then their performance degrades.

The key contributions of this work are as follows.

- Latent code optimization can only reconstruct a video sequence that belong to its range.

We demonstrate that by jointly optimizing the latent codes with the network weights, we can expand the range of the generator and reconstruct images that the given initial generator fails on. We show that even though the network has a very large number of parameters, the joint optimization still converges to a good solution.

- Consecutive frames in a video sequence share lot of similarities. To encode similarities among the reconstructed frames, we introduce low-rank constraints on the generator latent codes. This enables us to represent a video sequence with a very small number of parameters in the latent codes and reconstruct them from a very small number of measurements.

4.2 Technical Approach

Let us assume that $x_t \in \mathbb{R}^n$ for $t = 1, \ldots, T$ is a sequence of video frames that we want to reconstruct from the measurements $y_t = A_t x_t + e_t$ as given in (4.2). The generative model as given in (4.1) maps a low-dimensional representation vector, $z_t \in \mathbb{R}^k$, to a high-dimensional image as $x_t = G_\gamma(z_t)$. Thus, our goal of video recovery is equivalent to solving the following optimization problem over z_t:

$$y_t = A_t G_\gamma(z_t) + e_t, \tag{4.3}$$

which can be viewed as a nonlinear inverse problem. Below we discuss three different methods for solving this inverse problem.

(a) *Latent code optimization:* fixed γ, update z_t.

(b) *Joint latent code and generator optimization:* update γ, z_t

(c) *Joint optimization with lowrank constraints:* update both γ, z_t with additional low-rank constraints on z_t.

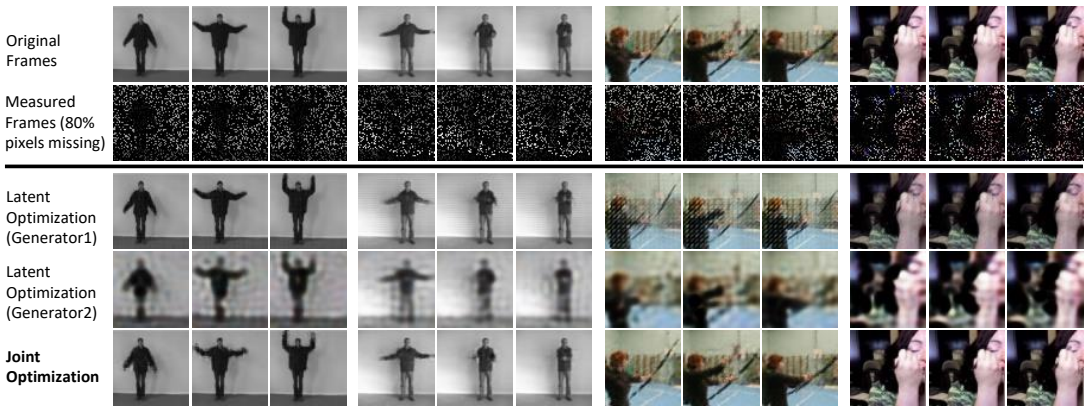


Figure 4.3: Joint optimization versus latent code optimization. First row is the true images of the videos sequences. The second row contains the masked samples of the sequences. In the third row, we reconstruct frames with latent code optimization using a generator trained on some other frames of the same video sequence (Generator1). In the fourth row, we use latent code optimization with a generator trained on CIFAR10 dataset (Generator2). The fourth row is the reconstruction with joint optimization of generator initialized with random weights. We can observe that latent code optimization does not perform well (row 4) when we do not have generator pretrained on similar distribution. However, joint optimization performs as good as as or better than latent code optimization without any pretrained weights.

4.2.1 Latent Code Optimization

In latent code optimization, we assume that the function $G_\gamma(\cdot)$ approximates the distribution of the set of natural images that contains our desired image. Thus, we can restrict our search for the underlying video sequence, x_t, within the range of the generator.

In other words, we fix the network parameters, γ, and update only the latent codes, z_t. This is the same problem studied in [66] for image compressive sensing using generative models.

Given a pretrained generator, G_γ, measurement sequence, y_t, and the measurement matrices, A_t, we solve the following optimization problem to recover the low-dimensional latent codes:

$$\underset{z_1,\ldots,z_T}{\text{minimize}} \sum_{t=1}^{T} \|y_t - A_t G_\gamma(z_t)\|_2^2. \tag{4.4}$$

The reconstructed video sequence can be computed as $\hat{x}_t = G_\gamma(\hat{z}_t)$, where $\hat{z}_1, \ldots \hat{z}_T$ denote the solution of the problem in (4.4).

To solve the problem in (4.4), we use a gradient descent approach by foward- and back-propagating the gradient w.r.t. z_t through the fixed generator network.

The latent code optimization in (4.4) can solve the compressive sensing problem with high probability if the solution belongs to the range of the generator [66]. Otherwise, its solution is a poor estimate of the original image. Since the range of natural images is very large, and it is difficult to represent all of them with a single or a few generators, the latent code optimization application is limited to the case when a pretrained generator is available.

4.2.2 Joint Latent Codes and Generator Optimization

To jointly optimize the latent codes and generator parameters, we use the same formulation as in (4.4) but optimize it over the z_t and γ. The resulting optimization problem can be written as

$$\underset{z_1,\ldots,z_T;\gamma}{\text{minimize}} \sum_{t=1}^{T} \|y_t - A_t G_\gamma(z_t)\|_2^2. \tag{4.5}$$

The reconstructed video sequence can be generated using the estimated latent codes $(\hat{z}_1, \ldots, \hat{z}_T)$ and generator weights $(\hat{\gamma})$ as $\hat{x}_t = G_{\hat{\gamma}}(\hat{z}_t)$.

The joint optimization of latent code and network parameters offers the optimization problem a lot of flexibility to generate a wide range of images. We initialize latent codes with samples drawn from a Gaussian distribution and normalize them to have unit norm. We initialize γ with random weights using the initialization scheme in [139]. Initilizing the generator with a pretrained set of weights can potentially serve as a good initialization and lead to good and faster convergence. We test both variants, but observe little difference in performance; therefore, we use random initialization of parameters in this work. Each iteration of joint optimization consists of two steps: 1) latent code optimization and 2) network parameter optimization. After every gradient descent update of the latent codes, z_t, we update the model parameters with stochastic gradient descent. In all of our experiments with joint optimization, we learn a single set of network weights for the entire sequence. We note that it is possible to divide a longer video sequences into small segments and learn different sets of network weights for each of them. At the end of our reconstruction process, we have a single set of trained weights $\hat{\gamma}$, reconstructed frames $\hat{x}_t$ and their corresponding optimal latent codes $\hat{z}_t$.

The range of any generator is quite limited and presumably depends on the types of images used during training. To highlight this limitation, we perform an experiment to reconstruct a video sequence from its masked version where 80% of the pixels are randomly missing under three different scenarios. The results are summarized in Figure 4.3 using four video sequences: '*Handwaving*' and '*Handclapping*' sequences from KTH video dataset and

'*Archery*' and '*Apply Eye Makeup*' sequence from UCF101 video dataset. We center, crop, and resize all the frames to 64×64 pixels. We only select the first 32 frames of the entire video sequence for testing reconstruction performance. We show the reconstruction under three different scenarios:

(a) In the first experiment, we train a generator using all but the first 32 frames of the corresponding video sequences that we call *Generator1*. Then we used *Generator1* as a prior for the reconstruction of the 32 test frames from their masked measurements. Since the training and test frames belong to the same video sequence and share lot of similarities, we can recover the test frames using *Generator1* in (4.4).

(b) In the second experiment, we use a generator pretrained on CIFAR10 dataset that we call *Generator2*. We reconstruct the test frames using latent code optimization with *Generator2* as a prior. As CIFAR10 contains images from diverse categories, the pretrained generator should have some generalization but it cannot reconstruct the test frames with good quality.

(c) In the third experiment, we initialize the generator with a random set of weights using the initialization technique in [139] and jointly optimize the latent codes and network parameters. As we can observe from Figure 4.3, joint optimization with random initialization provides similar or better reconstruction quality than the latent code optimization with network pretrained on the target class of images.

The latent code optimization results presented in Figure 4.3 should not be surprising for the following reasons: We are providing a measurements y_t of a video sequence to the generator $G_\gamma(z_t)$ that has k degrees of freedom for each z_t; therefore, the range of sequences

that can be generated by changing the z_t is quite limited for a fixed γ. The surprising thing, however, is that we can also recover quality images by jointly optimizing the latent codes z_t and network weights γ while solving the compressive sensing problem. If we let γ change while we learn the z_t, then the network can potentially generate any image in $\mathbb{R}^n$ because the network has very large degrees of freedom. Note that in our generator, the number of parameters in γ is significantly larger than the size of x_t, y_t or z_t. In other words, we can overcome the range limitation of the generator by optimizing network parameters alongside latent code to get a good reconstruction from compressive measurements as well as good representative latent codes for the video sequence even though the network is highly overparameterized.

4.2.3 Low Rank Constraint

As we optimize over the latent codes and the network weights in joint optimization, the latent codes capture the temporal similarity of the video frames. To further exploit the redundancies in a video sequence, we assume that the variation in the sequence of images are localized and the latent codes sequence can be represented in a low-dimensional space compared to their ambient dimension. Let us define a matrix Z with all the latent codes as

$$Z = [z_1 \ z_2 \ \dots \ z_T],$$

where z_t is the latent code corresponding to t^{th} image of the sequence. To impose a low-rank constraint, we solve the following constrained optimization:

$$\underset{z_1,\dots,z_T;\gamma}{\text{minimize}} \ \sum_{t=1}^{T} \|y_t - A_t G_\gamma(z_t)\|_2^2 \ \text{ s.t. } \ \text{rank}(Z) = r. \tag{4.6}$$

We solve (4.6) using a projected gradient descent method in which we project the latent code estimates after every iteration to a manifold of rank-r matrices. To do that, we compute Z matrix and its rank-r approximation using principal component analysis (PCA) or singular value decomposition (SVD).

In this manner, we can express each of the latent codes in terms of r orthogonal basis vectors $u_1, \ldots, u_r$ as

$$z_i = \sum_{j=1}^{r} \alpha_{ij} u_j \tag{4.7}$$

where α_{ij} is the weight of the corresponding basis vector. We can represent a video sequence with T frames with r orthogonal codes, and the lowrank representation of latent codes requires $r \times k + r \times T$ parameters compared to $T \times k$. This offers $r(\frac{1}{T} + \frac{1}{k})$ times compression to our latent code representation. As we observe later, we use $r = 4$ for $k = 256$ and $T = 32$ which gives us compression of 0.14 in latent code representation.

4.3 Experimental Setup

In this section, we describe our experimental setup and empirical results. We focus our experiments on three different compressive sensing problems: denoising, inpainting, and spatial compression by random projection. We also show some empirical results for coded flutter shutter problem where our algorithm is especially suitable. For a video sequence of T frames, we generate T independent measurement matrices. For color images, we use the same measurement matrix for each color channels. The total number of frames in each video sequence is 32, unless stated otherwise. For the low-rank constraint, we select the mean of

Algorithm 3 Generative Models for Low Rank Representation and Recovery of Videos

Input: Measurements y_t, measurement matrices A_t, A generator structure $G_\gamma(\cdot)$

Initialize the latent codes z_t and generator weights γ randomly and normalize z_t with its 2-norm.

repeat

Compute gradients w.r.t. z_t via backpropagation.

Update latent code matrix $Z = [z_1 \ \cdots \ z_T]$.

Truncate Z to a rank-r matrix via SVD or PCA.

Compute gradients w.r.t. γ via backpropagation.

Update network weights γ.

until convergence or maximum epochs

Output: Latent codes: $z_1, \ldots, z_T$ and network weights: γ

Table 4.1: Reconstruction performance measured in terms of PSNR for different compressive sensing problems. We show comparison with TVAL3D (3D extension of TVAL3 [1]) and deep decoder [2]. The results are averaged over five experiments with different random measurement matrices (or noise in the case of denoising).

Video Sequence	Rotating MNIST	Handclapping	Handwaving	Walking	Apply Eye Makeup	Archery	Band Marching
Denoising for additive Gaussian noise of 20dB SNR							
TVAL3D	35.8	32.2	30.4	30.5	34.5	31.5	30.6
UP Deep Decoder	28.9	28.4	25.6	28.3	28.1	29.6	28.1
OP Deep Decoder	36.6	31.1	30	31	34.4	**33**	31.6
Joint Optimization	**36.9**	**32.7**	30.7	**31.2**	36.1	32.1	31.3
Joint Opt + Low Rank	36.8	32.3	**30.8**	30.7	**36.4**	32	**31.7**
Inpainting with 80% pixels randomly missing							
TVAL3D	21.1	29.2	23.4	24.5	28.2	27.1	24.8
UP Deep Decoder	25.5	26.5	23.3	26.3	27.2	29	23.3
OP Deep Decoder	30.1	30.2	26.7	27.9	32.4	**32.5**	26.2
Joint Optimization	29.3	**34.9**	**28.1**	**28.9**	35.8	32	26.8
Joint Opt + Low Rank	**29.5**	34.3	27.3	27.8	**36.6**	30.4	**27.6**
Spatial compressive sensing with compression rate = 0.2							
TVAL3D	29.8	32.1	28.9	28	33.9	28.4	27.8
UP Deep Decoder	30	27	24.9	26.7	26.2	27.6	22.5
OP Deep Decoder	35.2	32.9	**30.6**	29	33.1	**31.2**	27.4
Joint Optimization	35.3	**35.6**	29.7	28.9	**36**	29.3	27.8
Joint Opt + Low Rank	**35.4**	34.7	29	**29.1**	35.9	28.8	**29.1**

the latent matrix Z and top 3 principal components (i.e., we need 4 vectors to represent the entire video sequence instead of 32.)

Choice of generator. We use the well-known DCGAN architecture [140] for our generators, except that we do not use any batch-normalization layer because gradient through the batch-normalization layer is dependent on the batch size and the distribution of the batch. As shown in Figure 4.1, in DCGAN generator framework, we project the latent code, z, to a larger vector using a fully connected network and then reshape it so that it can work as an input for the subsequent deconvolutional layers. Instead of using any pooling layers, the DCGAN architecture uses strided convolution [140]. All the intermediate strided convolution layers are followed by ReLU activation. The last strided convolution layer is

followed by Tanh activation function to generate the reconstructed image $x = G(z)$. In our experiment, we use videos of different resolutions. To generate those videos, we use different generators following the DCGAN framework. In Table 4.2, we report the detailed structure of the generators we use in the experiments.

The latent code dimension for grayscale 64×64 video sequence is 64. The latent code dimension for color 64×64 video sequence is 256. The latent code dimension for 256×256 video sequence is 512. We use Adam optimizer for generator weights optimization and SGD for latent code optimization. The learning rate for latent code optimization was 10. We use ADAM optimizer with $\beta_1 = 0.9$ and $\beta_2 = 0.999$ for network parameters optimization. The initial learning rate for network parameter optimization was 0.0025. We decay the learning rate for network parameter by 25% every 500 iterations.

Comparison with existing methods. We show comparison with classical total variation minimization based TVAL3D (3D extension of TVAL3 [1]) algorithm and generative prior based deep decoder [2] algorithm. As we mentioned earlier, deep decoder does not optimize latent code, rather it uses fixed latent codes which are drawn from Gaussian distribution.

We use two different deep decoder settings: underparameterized deep decoder (UP deep decoder) and overparameterized deep decoder (OP deepdecoder). The UP deep decoder was proposed in the original deep decoder paper [2], but we also report the results for OP deep decoder because it shows better performance. We use default 6 layer architecture of deep decoder. In the UP deep decoder, the number of parameters in UP deep decoder is 11,304 and 11,288 for RGB and grayscale images, respectively. The number of parameters

in OP deep decoder is 397,056 and 396,544 for RGB and grayscale images, respectively. We need separate generator for every frame which increases the effective number of parameters for the video sequence by a factor T for T frames. As T=32 for most of the experiments reported in the work, the effective number of parameters for OP deep decoder is 12,705,792 and 12,689,408 for RGB and grayscale images, respectively. We report the qualitative reconstruction results for OP deep decoder only because quantitative reconstruction results for UP deep decoder are significantly worse. This effect is also recently observed in [141].

We also show some comparison with video extension of deep image prior [14] algorithm. We discuss details of this approach later in the chapter.

Table 4.2: Generator structures and corresponding number of parameters for different image sizes. $h \times w \times c$ denote height , weight, and color channels, respectively.

	Network Parameters		
Output size	$\mathbf{64 \times 64}$	$\mathbf{64 \times 64 \times 3}$	$\mathbf{256 \times 256 \times 3}$
FC + ReLU	524,288	2,097,152	4,194,304
Conv 1+ ReLU	2,097,152	2,097,152	2,097,152
Conv 2+ ReLU	524,288	524,288	524,288
Conv 3+ ReLU	131,072	131,072	131,072
Conv 4+ Tanh/ReLU or ReLU	1,024	3,072	32,768
Conv 5+ ReLU	-	-	8,192
Conv 5+ Tanh	-	-	768
Total # params	**3,277,824**	**4,852,736**	**6,988,544**

Video datasets. We test all the methods on different synthetic and real video sequences. In this work we report the results for one synthetic sequence which we refer to as '*Rotating MNIST*'. In this sequence, we resize one MNIST digit to 64×64 and rotate by 2° per frame for a total of 32 frames. We experiment on different real video sequences from publicly available KTH human action video dataset [142] and UCF101 dataset [143].

In Table 4.1, we report our results for '*Handclapping*', '*Handwaving*' and '*Walking*' video sequences from KTH dataset; '*Archery*', '*Apply Eye Makeup*' and '*Band Marching*' video sequences from UCF101 dataset. We center and resize every frame in KTH videos to 64×64 and UCF101 videos to 256×256 pixels.

Performance metric. We measure the performance of our recovery algorithms in terms of the reconstruction error PSNR. For a given image x and its reconstruction $\hat{x}$, PSNR is defined as

$$\text{PSNR}(x, \hat{x}) = 20 \log_{10} \frac{\max(x) - \min(x)}{\sqrt{\text{MSE}(x, \hat{x})}}$$

where $\max(x)$ and $\min(x)$ are the maximal and minimal values in x, respectively, and MSE is the mean squared error. Unless otherwise stated, all the results are averaged over 5 experiments using different measurement matrices or noise.

4.4 Results and Analysis

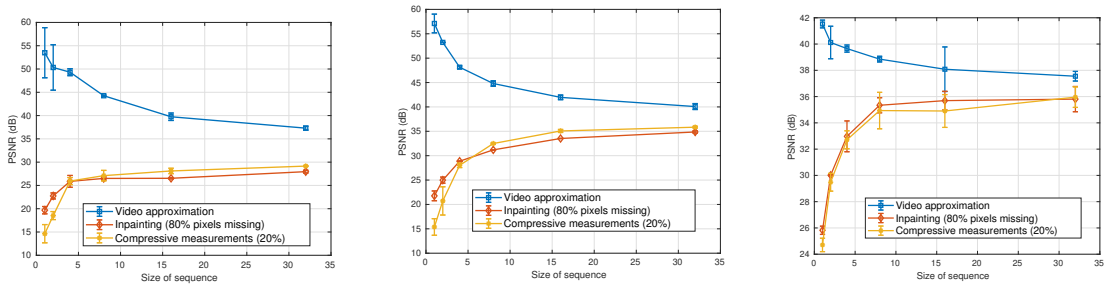
4.4.1 Sequence Size vs Performance

To evaluate the effect of sequence size on the performance of our method, we perform joint optimization experiments with video sequences of different sizes. We report our results for three different video sequences in Figure 4.4. We consider three different tasks. The first task is video approximation, where we approximate the original video sequences using a generator (i.e., A_t is an identity matrix for all t). We observe that as we increase the size of the video sequence, the quality of approximated video sequences degrades. This intuitively makes sense because a network with sufficient complexity should be able to approximate a single image perfectly. However, as we increase the size of the video sequence

while keeping the same network structure, our algorithm has to find a new set of optimal weights that can generate the entire sequence. The reduction in reconstruction performance is more pronounced when every frame of the video is different. Our second task is image inpainting with 80% randomly missing pixels and the third task is compressive sensing with 20% available measurements. In both cases, we have far fewer number of measurements available than that in the approximation task. As the consecutive frames of a video sequence are close to each other, the increased size of the video sequence actually helps by providing more effective measurements to the generator. However, the generator capacity still remains a barrier. On one hand, we have more (diverse) measurements available while optimizing over a larger video sequence, which can provide a gain in the performance with a longer sequence. On the other hand, we have to find a set of network parameters that can generate all the (diverse) frames at once, which can cause a loss in performance with longer sequence. We observe in our experiments that the recovery performance increases with the length of the video sequence up to a certain point and then it saturates. We select the size of the video sequences as 32 for our next experiments based on these results.

4.4.2 Denoising

We first explore the potential of joint optimization on the denoising problem. In our denoising setup, the measurement matrix is identity and the noise, e_t, is drawn from zero mean Gaussian distribution. We report the reconstruction results of different video sequences for different algorithms in Table 4.1. We observe that joint optimization performs significantly better than UP deep decoder and provides similar results as TVAL3D and OP



(a) Handwaving **(b)** Handclapping **(c)** Apply Eye Makeup

Figure 4.4: Sequence size vs performance for video approximation and compressive sensing tasks. Here the results corresponds to joint optimization. We can observe that increasing video length improves compressive sensing performance for joint optimization. This effect diminishes with the increased size of video sequences.

deep decoder. Note that we do not optimize over latent code for deep decoder. We also need to train a separate network for each frame, which requires huge computational power and memory. We report the memory and computational complexity comparison in Table 4.3. We also observe that joint optimization with low-rank constraint provides similar performance.

We present some denoising results for different techniques with additive Gaussian noise at 20dB SNR on different sequences in Figure 4.5. The performance curves in terms of average PSNR over a range of SNR levels are presented in Figure 4.6. We observe that reconstruction performance of joint optimization is better than classical TVAL3D. For large noise, joint optimization shows better or comparable performance with the deep decoder. However, for small noise, deep decoder seems to outperform joint optimization. In the case of deep decoder, we learn a separate network for every frame, and as we observe in the previous section that for a fixed generator, image approximation performance is better for

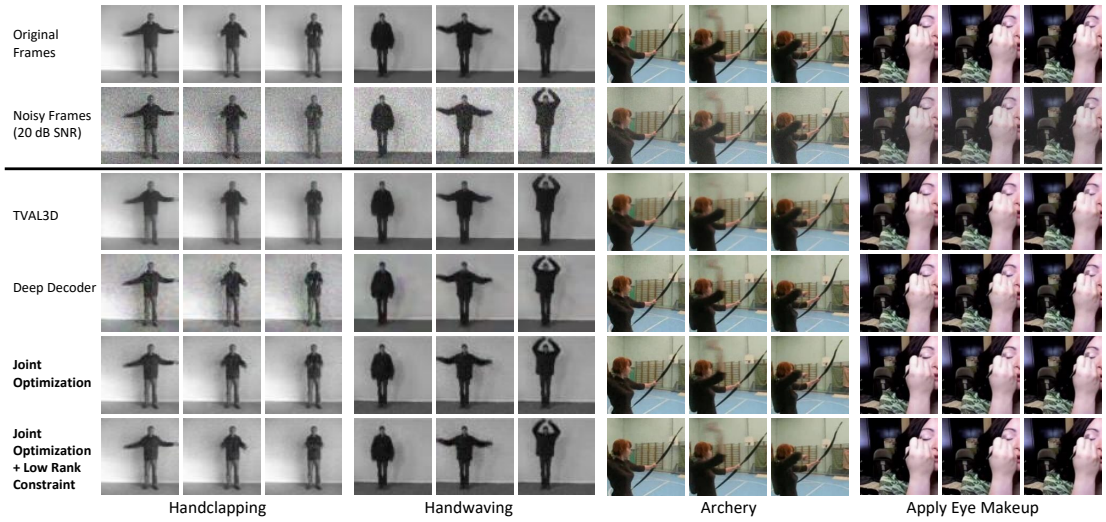
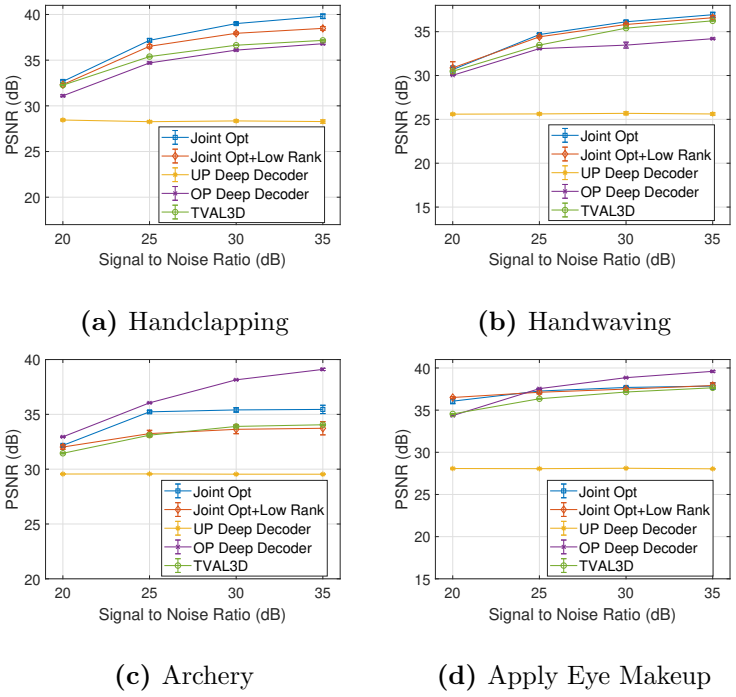


Figure 4.5: Reconstruction of different video sequences using different algorithms for denoising problem. Handclapping and Handwaving video sequences are 64×64 and Archery and Apply Eye Makeup video sequences are 256×256. The error bars are standard deviation intervals. The deep decoder reconstruction here correspond to overparameterized deep decoder structure. All the comparing algorithms show very good reconstruction quality.

single image. In the low noise regime, denoising problem is almost same as an approximation problem. In the case of joint optimization, we are learning a single set of network parameters for the entire video sequence; therefore, joint optimization has a limitation due to the representation capacity of the generator network. We can also observe in Figure 4.6 that the curves corresponding to UP deep decoder is flat, which is because of the fact that the UP deep decoder has a certain representation capacity, and once that capacity is reached for a single frame the results do not improve even if the noise level decreases.



(a) Handclapping **(b)** Handwaving

(c) Archery **(d)** Apply Eye Makeup

Figure 4.6: Reconstruction quality curves for denoising experiments with different algorithms for different levels of signal to noise ratio. The curves also show standard deviation intervals. We compare the performance for (a) Handclapping (b) Handwaving (c) Archery (d) Apply Eye Makeup video sequences. All the comparing methods other than UP deep decoder performs similarly. The curves suggest that UP deep decoder has reached its limit to generate the sequences.

4.4.3 Inpainting

Our second experiment is on inpainting problem where we randomly drop a fraction of the pixels from each frame and reconstruct the original video sequence from available pixels. We report the results for 80% missing pixels in Table 4.1. We observe that reconstruction performance of joint optimization is significantly better than classical TVAL3D and deep decoder. We also show some reconstructions of different video sequences from 20% available pixels (80% pixels are randomly missing) in Figure 4.7. From this figure, we can observe that

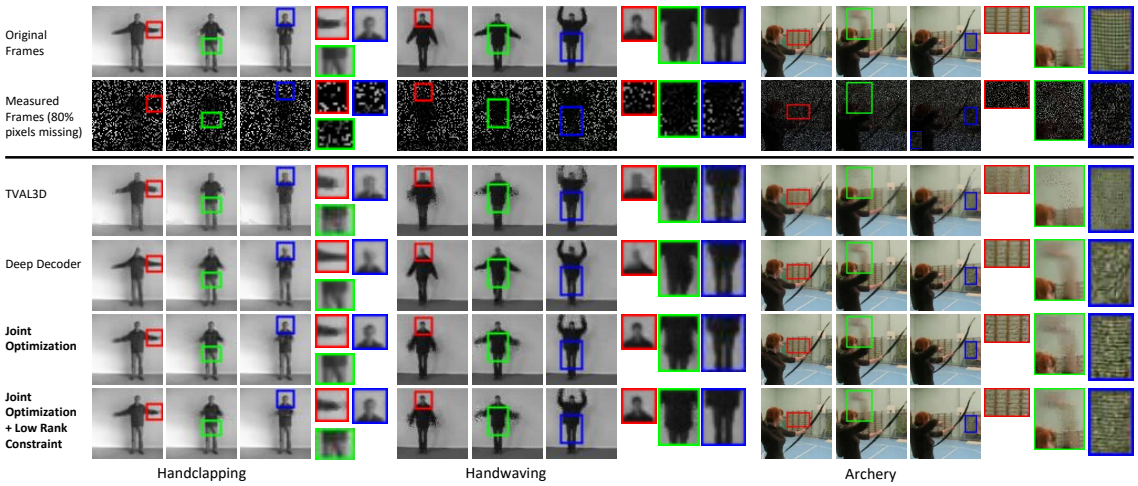
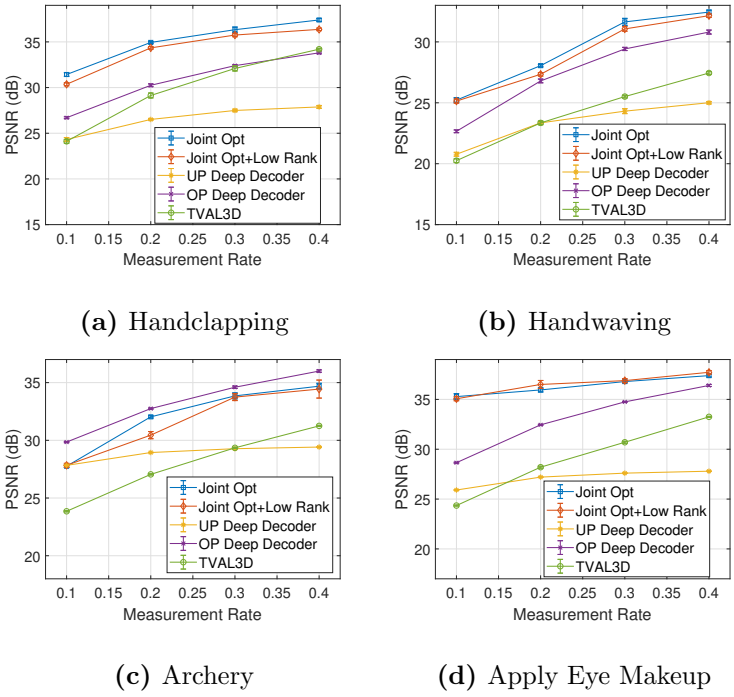


Figure 4.7: Some reconstruction results on inpainting problem. Handclapping and Handwaving video sequences are 64×64 and Archery sequence is 256×256. The deep decoder reconstruction here correspond to overparameterized deep decoder structure. The boxed regions are zoomed for details. We can observe that joint optimization gives better reconstruction than the comparing algorithms in terms of details.

even though the reconstruction results of deep decoder is similar to joint optimization in terms of PSNR, deep decoder fails to reconstruct high frequency details reliably (see *Archery* results). Since we optimize the generator using a number of frames in joint optimization, the missing information in one frame may be available in a neighboring frame, and that can potentially help the joint optimization perform better in reconstructing the high frequency details as shown in Figure 4.7.

We also show inpainting performance for different fractions of missing pixels in Figure 4.8. From the comparison with the other algorithms shown in Figure 4.8, we can observe that joint optimization with/without low rank constraint outperforms other comparing algorithms especially when we have very few number of measurements available.



(a) Handclapping **(b)** Handwaving

(c) Archery **(d)** Apply Eye Makeup

Figure 4.8: Inpainting performance for different available measurement rate for (a) Handclapping (b) Handwaving (c) Archery (d) Apply Eye Makeup video sequences. Measurement rate represents the available fraction of the total pixels. The error bars are standard deviation intervals. Other than Archery sequence, joint optimization outperforms the other comparing methods especially at lower measurement rate.

4.4.4 Compressive Sensing

In this section, we discuss our experiments on recovery of frames from their compressive random projections. In these experiments, we use separable measurements, $Y = PXQ$, where X, Y are reshaped versions of x, y as 2D matrices, P and Q are left and right random projection matrix. In our experiment, we use $P = Q^T$, and select their size so that the total number of measurements in Y is m. We draw each sample of P from $N(0, \frac{1}{\sqrt{m}})$ distribution.

We summarize the results for this experiment in Table 4.1. We select $m = 29 \times 29$ for 64×64 images and $m = 114 \times 114$ for 256×256 images, which gives us a compression of factor of approximately 20%. We observe from Table 4.1 that joint optimization with and without low-rank constraint slightly outperforms TVAL3D. It performs similarly as deep decoder with much lower memory requirement and computational complexity.

We show some reconstructions for compressive sensing with 20% compressive measurements in Figure 4.9. We can observe that he reconstructions are comparable with other algorithms. We also show reconstruction performance for different compression ratio in Figure 4.10. We can observe from Figure 4.10 that joint optimization with or without low rank constraint outperforms TVAL3D and UP deep decoder. However, it performs at par with if not better than OP deep decoder.

4.4.5 Flutter Shutter

We also perform an experiment with a computational photography problem known as coded flutter shutter [144, 145, 146, 147, 148] in which a low-speed camera is used to capture a modulated high-speed video. A single observed frame can be modeled as coded and multiplexed version of a number of frames in the sequence. Our goal is to recover the individual frames from the multiplexed frame. Mathematically, we can formulate the problem as

$$y_p = \sum_{t=Mp}^{Mp+M-1} A_t x_t + e_p, \quad \text{for } p = 1, \ldots, T/M, \tag{4.8}$$

where we observe a single measurement frame for every M consecutive frames. Thus, we have $\frac{T}{M}$ measurement frames for the entire video sequence. We choose A_i in a similar manner

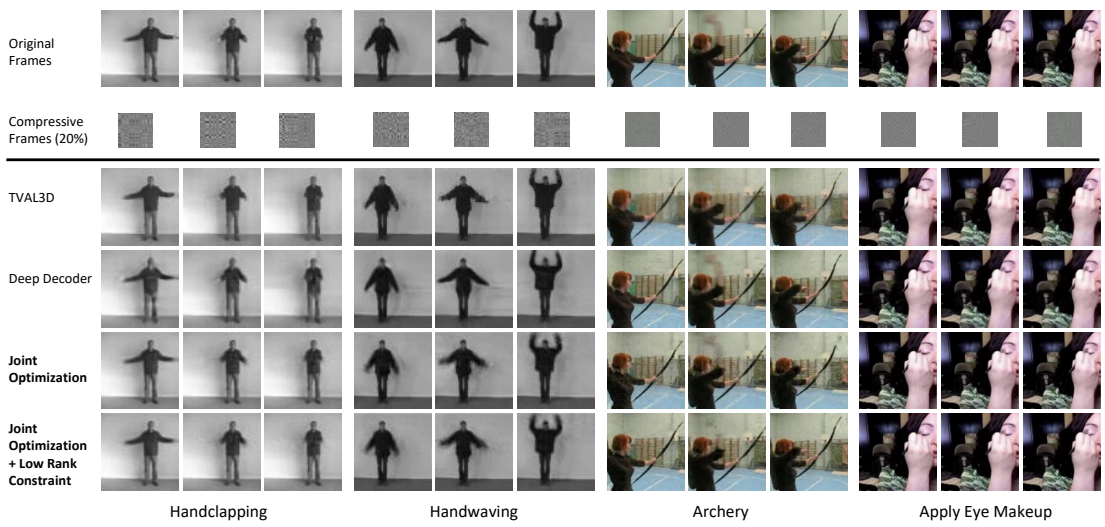
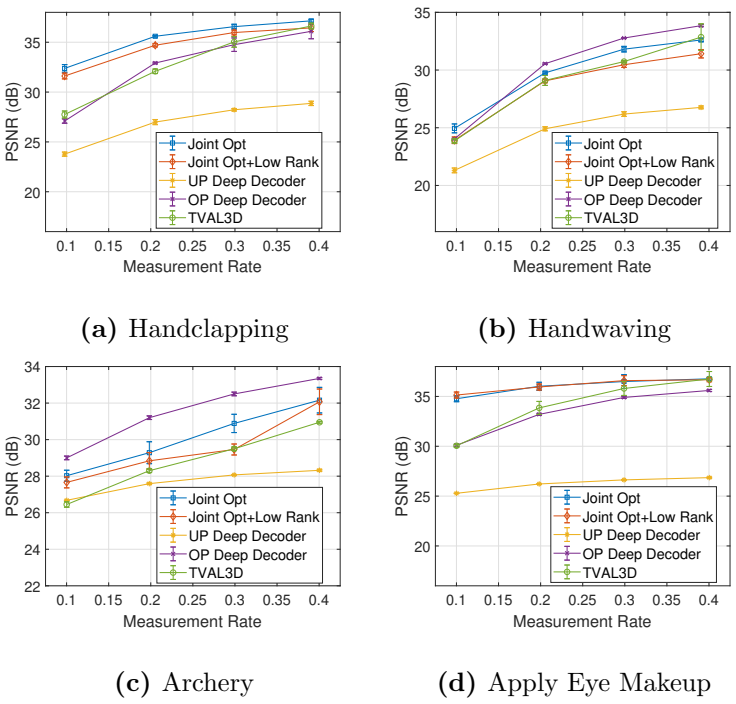


Figure 4.9: Some reconstruction results on spatial compressive sensing problem. Handclapping and Handwaving video sequences are 64×64 and Archery and Apply Eye Makeup video sequences are 256×256. The compressive frames from Handclapping and Handwaving are 29×29 whereas the compressive frames from Archery and Apply Eye Makeup video sequences are 114×114. The deep decoder reconstruction here correspond to overparameterized deep decoder structure. We can observe that the reconstructions are similar for the comparing algorithms.

as the inpainting mask (i.e., 50% pixels are randomly missing). Our joint optimization can solve this problem because we can jointly estimate multiple frames while solving a single optimization problem. In contrast, if we train a separate network for every single frame (as done in DIP and deep decoder), the recovery problem will not be as straightforward. To estimate the video sequence from coded, multiplexed measurements, we solve the following recovery problem:

$$\underset{z_1,\ldots,z_T;\gamma}{\text{minimize}} \sum_{p=1}^{T/M} \|y_p - \sum_{t=Mp}^{Mp+M-1} A_t G_\gamma(z_t)\|_.^2 \tag{4.9}$$

We present some reconstructed images for coded flutter shutter in Figure 4.11. Because of the high memory and computational requirements, deep decoder is not suitable



(a) Handclapping **(b)** Handwaving

(c) Archery **(d)** Apply Eye Makeup

Figure 4.10: Compressive sensing performance for different available measurement rate for (a) Handclapping (b) Handwaving (c) Archery (d) Apply Eye Makeup video sequences. Measurement rate (or compression ratio) represents the available fraction of the total measurements. The error bars are standard deviation intervals. We can observe from the curves that joint optimization performs at par with the other comparing methods.

for this problem. We present a comparison with TVAL3D reconstruction. Joint optimization with and without low-rank constraints successfully recover fast motion that TVAL3D fails to recover.

4.4.6 Rank of the Latent Matrix

In this section, we evaluate the performance of joint optimization with low-rank constraints for different choice of the rank. We perform inpainting experiment with 80%

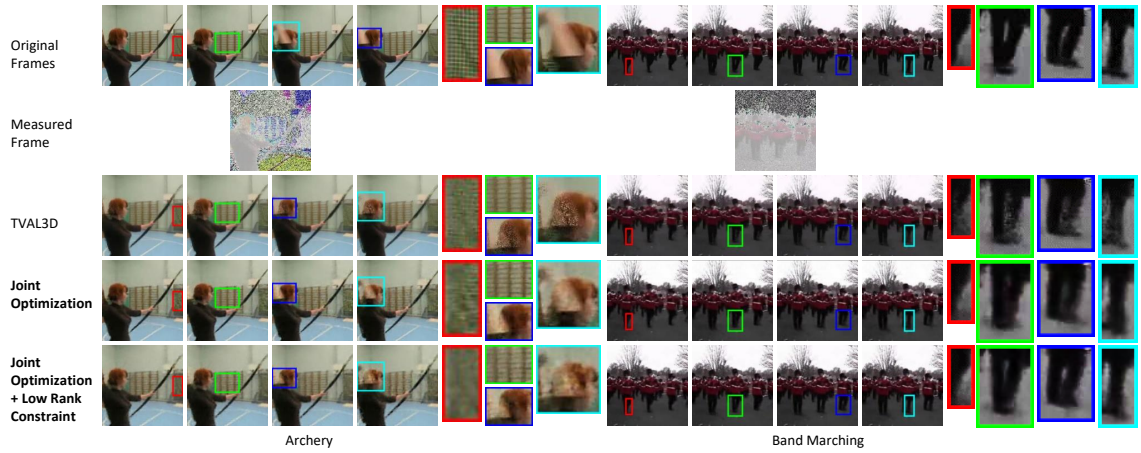


Figure 4.11: Some reconstructions for flutter shutter problem. Here we have a single measurements for every 4 non overlapping frames. We can observe that TVAL3D suffers ghosting effect for the fast changing parts of the videos such as the hand or leg movement. However, they perform similarly in background details reconstruction.

missing pixels using different values of rank. We plot the reconstruction PSNR performance curves for different video sequences in Figure 4.12. Rank-1 corresponds to using a fixed (mean) vector as the latent code for all the frames, which would reconstruct the same frame for the entire sequence. As we increase the rank of the latent code matrix, we observe that reconstruction quality improves and rank-4 reconstruction gives us a good performance for all the sequences. Note that for rank-4, we select the mean vector and top-3 principal components to represent the entire sequence with 32 frames.

4.4.7 Computational Complexity

The computational complexity of our proposed methods vary with the choice of the generator structure. We have chosen DCGAN generator structure for our experiments.

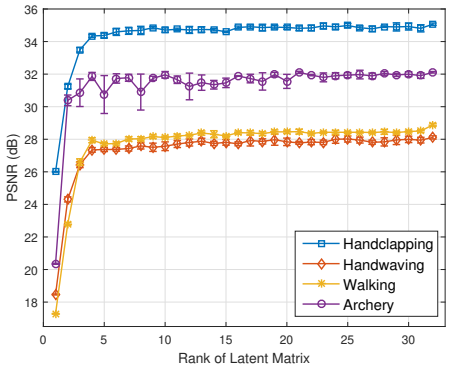


Figure 4.12: Effect of different value of rank for low rank constraint in inpainting problem with 80% pixels randomly missing. We also show standard deviation interval for each point.

We compare the computational complexity of our algorithm with UP and OP deep decoder [2]. The memory requirement mentioned here is for a single frame. The time consumption is recorded for the inpainting of RGB video sequences with 32 frames from 80% missing pixels. We report average time consumption over 5 experiments. The number of iterations, measurement matrix and the videos sequences of the corresponding size were kept the same. The experiments for this comparison were run on the same CPU equipped with Nvidia Titan Xp GPU.

From the memory requirement and time consumption, it is evident that joint optimization is much less complex and consumes much less memory compared to OP deep decoder. Although the memory requirement and complexity of UP deep decoder comes close to that of joint optimization, UP deep decoder reconstruction performance is poor (Table 4.1, Figure 4.6,4.8,4.10).

Table 4.3: Comparison of joint optimization with DCGAN and deep decoder in terms of computational complexity and memory requirement. The memory requirement is for each frame reconstruction. The average time consumption is calculated for video sequences with 32 frames.

Size	**64 × 64**	**256 × 256**
Memory Requirement (Forward and Backpropagation)		
UP Deep decoder	2.75 MB	44.03 MB
OP Deep decoder	66.48 MB	1239.75 MB
Joint Opt with DCGAN	2.06 MB	10.88 MB
Average time consumed (Forward and Backpropagation)		
UP Deep decoder	120 sec	710 sec
OP Deep decoder	180 sec	3042 sec
Joint Opt with DCGAN	14.2 sec	203 sec

Table 4.4: Effect of initial latent matrix for different inverse problems. We have drawn latent matrix in way that the initial latent codes form a line. The results are averaged over fifteen experiments with five different random measurement matrices and three different initializations. We use same measurement matrices and initializations for both approaches.

	Rotating MNIST		**Handclapping**		**Handwaving**		**Walking**		**Apply Eye Makeup**		**Archery**		**Band Marching**	
	Video DIP	**Joint Opt**	**Video DIP**	**Joint Opt**	**Video DIP**	**Joint Opt**	**Video DIP**	**Joint Opt**	**Video DIP**	**Joint Opt**	**Video DIP**	**Joint Opt**	**Video DIP**	**Joint Opt**
Inpainting (80% missing)	28.6	30.1	31	34.1	23	26.8	23.8	26.5	34.8	37	30.5	31.9	28.8	29
Compressive Measurements (20%)	33.8	35.5	33.1	35.5	24.6	30.1	23.9	28.6	32.9	36.1	29.2	29.9	28.3	29.3

4.4.8 Comparison with Video DIP

In section 4.4.1, we have demonstrated that optimizing over a video sequence improves reconstruction performance. However, as we have mentioned before, DIP [14] trains one network per image which puts it at a disadvantage while comparing with joint optimization. So, we made an extension of DIP for video sequence and refer to it as "Video DIP". In this approach, we draw entries in the latent matrix Z from a Gaussian distribution

Table 4.5: Performance analysis between Video DIP and joint optimization when all the frames in the video sequence are not close to each other. The results are averaged over twelve experiments with four different random measurement matrices and three different initializations. We use same measurement matrices and initializations for both approaches.

	Handclapping + Handwaving		**Handclapping + Walking**		**Handwaving + Walking**	
	Video DIP	**Joint Opt**	**Video DIP**	**Joint Opt**	**Video DIP**	**Joint Opt**
Inpainting (80% missing)	31.5	33.3	32	33	27.9	29.4
Compressive Measurements (20%)	29	32.7	29.4	32.4	26.5	29.6

and keep it fixed as we solve the following optimization:

$$\hat{\gamma} = \arg\min_{\gamma} \sum_{t=1}^{T} \|y_t - A_t G_\gamma(z_t)\|_2^2 \tag{4.10}$$

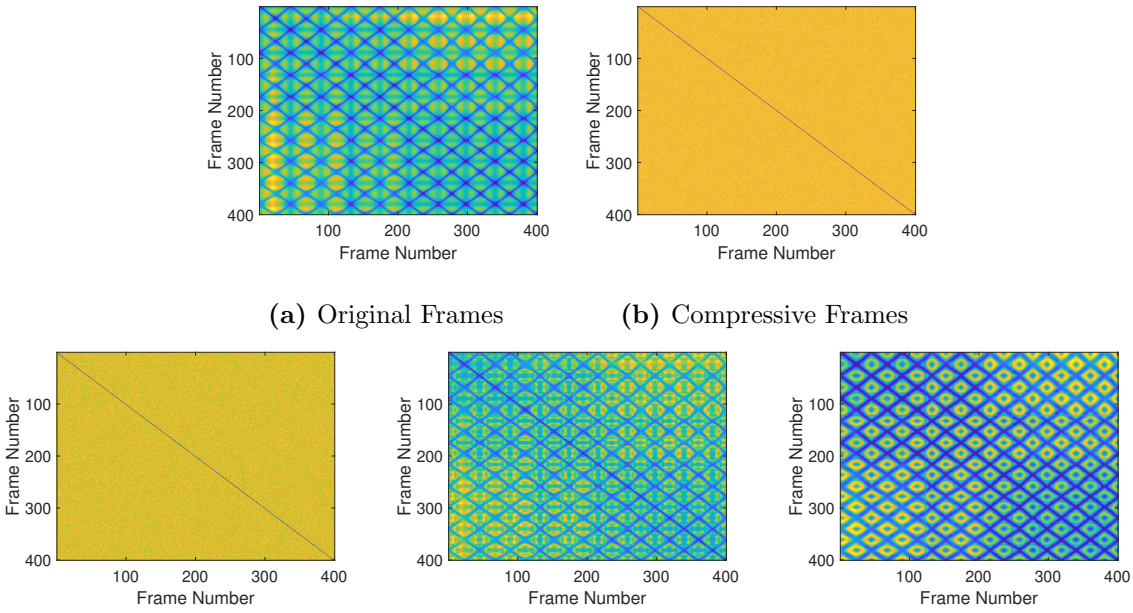
We use the same architecture for Video DIP as we use for joint optimization.

We observe that even if we extend DIP for video sequence, it still suffers from two main drawbacks that we mentioned earlier. First drawback is the dependence on the initialization of latent matrix as it remains fixed. If the initialization is *bad*, Video DIP will fail to provide good reconstruction. We demonstrate this effect with an example. We force the latent codes to fall on a line by fixing two of the latent codes z_1 and z_T, each drawn from $N(0, 1)$ and initialize the other latent codes by linear interpolating between z_1, z_T. We expect Video DIP to perform worse in this case because we are forcing the network to map frames to a line in a 2D plane whereas the videos contain much complex motions. We report the reconstruction results for both Video DIP and joint optimization in Table 4.4. We experiment on inpainting and compressive sensing with 20% available measurements.

Joint optimization performs better than Video DIP in all the cases.

The second drawback is that Video DIP does not assume or retain any similarity structure in the latent code representation. This will in turn affect the reconstruction quality if the frames in the videos are very different from one another. To demonstrate this effect, we create a video sequence with 64 frames temporally concatenating 32 frames from two different sequences one after another. In Table 4.5, we report average results of four experiments with different measurement matrices. We perform experiments for inpainting and compressive sensing problems. For both Video DIP and joint optimization, latent matrices are initialized with elements drawn from $N(0, 1)$ distribution. Since we initialize the latent matrix at random, it is possible that similar latent codes are assigned to frames that are quite different. As the latent codes are fixed in Video DIP, finding the network parameters that map very different frames to latent codes that are very similar can be a challenging task. In contrast, joint optimization method updates both the latent codes and network parameters; therefore, it can adjust the latent codes so that similar latent codes are mapped to similar frames and vice versa. From Table 4.5, it is evident that joint optimization is better suited for such videos as it outperforms Video DIP in each case.

The latent codes obtained from joint optimization also reserve the similarities among video frames. We demonstrate this property using a compressive sensing experiment, where we capture 20% measurements of each frame with an independent random matrix. We use 400 frames of '*Handwaving*' sequence for this experiment instead of 32 frames. In the sequence, the handwaving action is repeated multiple times, each of which takes around 45 frames. We compute a cosine similarity matrix between all the image pairs, which is

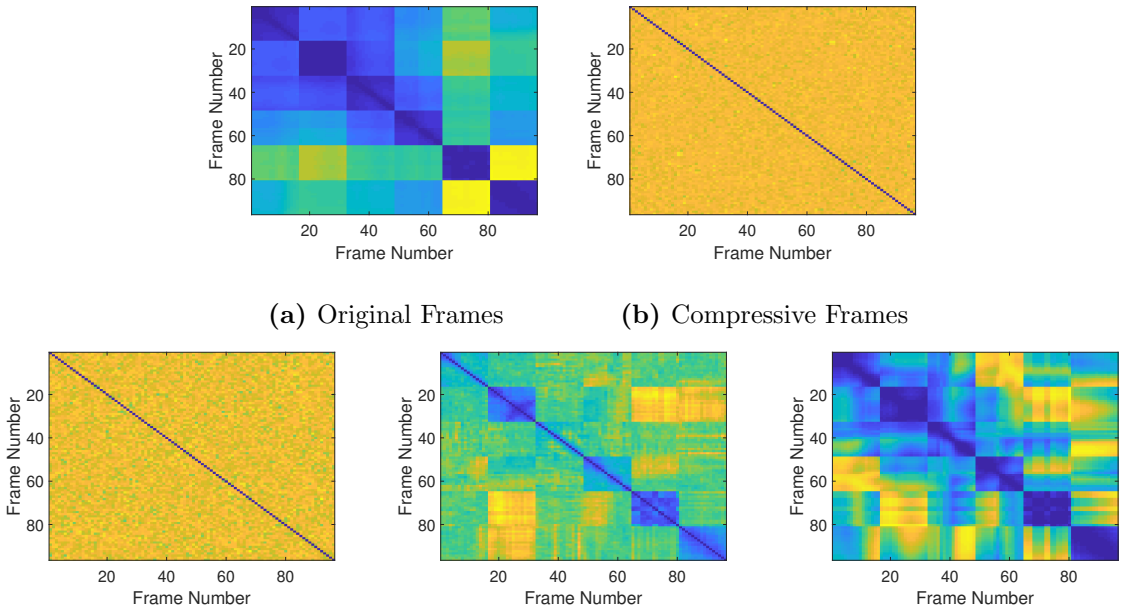


(a) Original Frames **(b)** Compressive Frames

(c) Latent Matrix of Video DIP **(d)** Latent Matrix of Joint Optimization **(e)** Latent Matrix of Joint Optimization + Low Rank Constraint

Figure 4.13: Pairwise cosine similarity between frames, measurements or latent codes for extended Handwaving video sequence where Handwaving action is repeated in an interval of around 45 frames. Blue indicates highest similarity whereas yellow indicates lowest similarity. We can observe that the similarity pattern in the original frames are not maintained in the compressive frames. As the Video DIP latent codes are drawn at random, we do not observe any similarity pattern in them (c). However, the corresponding latent matrix for joint optimization (d) captures the similarity structure. Low rank constraint (e) further enhances this similarity.

plotted in Figure 4.13(a). Since compressive measurements are independent of one another, we do not expect any similarity between them, as seen in Figure 4.13(b). Latent codes in Video DIP are also independent and randomly selected, and they do not reflect the similarity structure of the video frames, as shown in Figure 4.13(c). We optimize the latent



(a) Original Frames **(b)** Compressive Frames

(c) Latent Matrix of Video DIP **(d)** Latent Matrix of Joint Optimization **(e)** Latent Matrix of Joint Optimization + Low Rank Constraint

Figure 4.14: Pairwise cosine similarity between frames, measurements or latent codes for extended mixed video sequence where 16 frames of 6 different video sequences (Handwaving, Handclapping, Walking, Archery, Apply Eye Makeup, Band Marching in order) are concatenated in the temporal dimension. Blue indicates highest similarity whereas yellow indicates lowest similarity. We observe that adding low rank constraint further bolster the similarity observed in the frames of same video sequences found by joint optimization.

codes in joint optimization, and they preserve the similarity structure as we can observe from Figure 4.13(d). From Figure 4.13(e), we can observe that low rank constraint further enhances the similarity structure in latent code matrix.

To further investigate the similarity structure in the latent codes obtained by joint optimization, we perform another experiment in which we concatenate 16 frames from each

of the six different video sequences ('*Handwaving*', '*Handclapping*', '*Walking*', '*Archery*', '*Apply Eye Makeup*', and '*Band Marching*', in the same order) to create a new sequence with 96 frames. We perform compressive sensing experiment on this video sequence with 20% measurements. The reconstruction PSNR for joint optimization is 29.12 dB, joint optimization with low-rank is 27.9 dB, and Video DIP is 26.4 dB. The cosine similarity matrices for the video frames, compressive measurements, latent codes for Video DIP, latent codes for joint optimization, and latent codes for joint optimization with low-rank are presented in Figure 4.14(a)–(e). We can distinguish the video sequences from the pairwise similarity matrices of the latent codes we estimate with joint optimization. We observe that the low-rank constraint improves the similarity matrix.

Chapter 5

Tensor Ring Autoencoders for Linear Inverse Problems

5.1 Introduction

Low-rank tensor factorization is a powerful tool to represent multi-dimensional and multi-modal data using a small number of low-dimensional factors (cores). Different fields in science and engineering use low-rank tensor factorization to understand multidimensional correlation structures in data [149]. Tensor factorization has also been recently used for compressing data and neural network parameters [150, 151]. In contrast to deep generative models, tensor factorization usually provides a linear low-dimensional representation of data [152]. Tensor factors also provide a natural way to separate (or disentangle) different dimensions or modes of data.

In various applications, such as face recognition, social network analysis, image and

This work has been submitted to British Machine Vision Conference 2022.

video completion, and brain signal processing, we often encounter structured datasets that can be represented as tensors by aligning different modes of data along different dimensions of tensors [153, 56]. Such structured datasets are often corrupted due to imperfect acquisition [154, 155]. As a result, we do not have reliable entries for all the points in the tensor structure. Images with different imperfections can be modeled as the following system of measurements:

$$y_i = \mathbf{A}_i x_i + \eta_i, \tag{5.1}$$

where y_i is the i^{th} observed measurement, $\mathbf{A}_i$ is the corresponding measurement matrix (corruption model) and η_i is the corresponding measurement noise. As the problem in (5.1) is ill-posed, we use different prior constraints on the solution set. One such constraint is generative prior where the solution set is restricted to the range of trained/untrained generative network.

In recent years, low-rank based tensor completion, which is a higher-order extension of matrix completion, has received considerable attention for recovering data from imperfect tensor structure. However, the low-rank assumption is not sufficient for the recovery of visual data where the ratio of missing data is extremely high. We propose to learn an optimal embedding space for tensor factorization to represent and recover the entire data tensor given the available imperfect measurements.

Deep generative models provide an excellent mechanism to generate high-dimensional data from low-dimensional codes or learn low-dimensional representation of high-dimensional data. Examples of such deep generative models include generative adversarial networks (GAN)[42], variational autoencoders (VAE) [43], and generative latent optimization (GLO)

[49]. We use an autoencoder structure to learn the embedding space for tensor factorization (encoder) and generate estimated data from the tensor factorization (decoder). We map different visual attributes to different factors (cores/ matrices) of tensor factorization. Such mapping provides controlling knob for different features. Although it is an inherent advantage of using tensor factorization, different attributes of visual data are so complex that we cannot map them directly to different factors of multilinear tensor factorizations. Therefore, we learn an encoder which provides a latent embedding space where tensor factorization can be used to map different articulations using different tensor factors.

We can consider that we constrained the latent space of the autoencoder with low-rank tensor factorization. Similarity in visual data often reflect in the similarity in the latent space. Applying the low-rank tensor factorization in the latent space, we utilize this inherent similarity in the structured datasets in our advantage to achieve better reconstruction. From the reconstruction performance with and without tensor factorized latent space for highly corrupted data has supports our claim.

5.1.1 Our Contributions

Given a corrupted (noisy, missing pixels etc) multi-attribute structured image dataset, we use the structure in the latent space to recover the corrupted data in the self supervised setup. In order to achieve that, we learned an ambient embedding space for tensor factorization using an encoder. We recovered the corrupted data reliably given the available information (i.e. the corrupted dataset, attribute labels of each data, measurement matrices for the corruption). We can utilize the similarity in the structured image set with

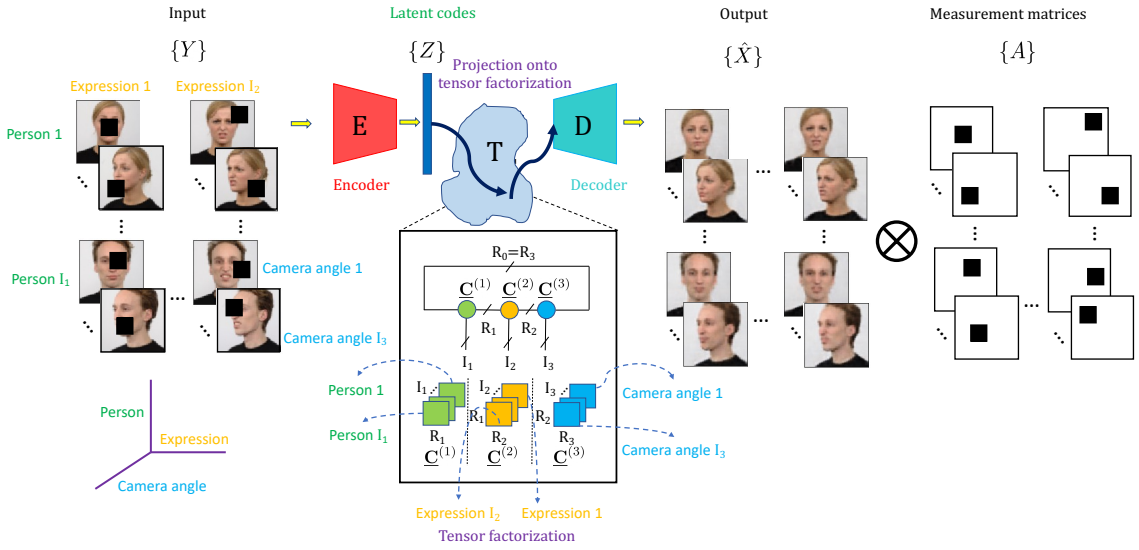


Figure 5.1: General overview of our proposed tensor ring factorized autoencoder. We map a set of images $\{X\}$ to latent codes $\{Z\}$ using an encoder E. We then perform tensor factorization on the latent space codes using tensor factorization (shown as **T** block). Finally, we pass the factorized representation though the decoder D to generate target images $\hat{X}$.

tensor ring factorization in the latent space in order to achieve better recovery. In terms of reconstruction quality, we outperform the other self supervised generative model based recovery techniques as well as least square solution with tensor ring prior on the image space which also use structural information.

5.2 Technical Details

Deterministic Autoencoder: Deterministic autoencoder [156] is a network that learns self mapping using two different networks: one is called the encoder $E(.)$ and the other one is called the decoder $D(.)$. Encoder maps the signal $X \in \mathbb{R}^n$ to the latent code $z \in \mathbb{R}^d$ and the decoder maps the latent code z back to the input signal X. We can represent the

encoder and decoder mappings as

$$z = E_{\gamma_E}(X) \text{ and } X = D_{\gamma_D}(z), \tag{5.2}$$

where γ_E and γ_D are the encoder and decoder network parameters, respectively.

Tensor Factorization: Tensor factorization can represent multi-dimensional and multi-modal data using a small number of low-dimensional factors. Instead of applying the tensor factorization directly on the image/signal space, we factorize the low-dimensional latent space. We seek two main goals with such factorization: 1) limit the degrees of freedom for the latent space and 2) generate images by changing different attributes in a controllable manner.

Suppose we have a generative model, $G_\gamma(\cdot)$, that maps a low-dimensional latent code $z_i \in \mathbb{R}^d$ to its respective X_i. We denote our entire target set of N latent codes as $\{z_1, z_2, \ldots, z_N\}$, where each $z_i \in \mathbb{R}^d$. These latent codes can be factored into K different attributes each of which has I_k variants for $k \in \{1, 2, \ldots, K\}$; therefore, we can write $N = I_1 \times I_2 \times \ldots \times I_K$. We will denote the latent code tensor with all the z_i as $\mathscr{T}_{\mathscr{Z}}$, which is an $I_1 \times I_2 \times \ldots \times I_K \times d$ tensor, and $\mathscr{T}_{\mathscr{Z}}(i_1, \ldots, i_K, :)$ denotes one of the d-dimensional latent codes.

We assume that the latent space have rank, $r \leq min\{N, d\}$. Assuming the similarities among attributes in the signal domain is reflected in the latent space, we can represent the entire latent space using different types of tensor factorization. We use tensor ring factorization in this work. Visual representations of these decompositions are depicted in Figure 5.1. Brief description for tensor ring factorization is presented below. For detailed discussion, we refer the readers to [149, 157].

Tensor Ring Factorization A *tensor ring* (TR) decomposition can represent a data tensor $\mathscr{T}_{\mathscr{Z}}$ using K different 3-order tensor cores: $\underline{\mathbf{C}}^{(1)}, \ldots, \underline{\mathbf{C}}^{(K)}$, where $\underline{\mathbf{C}}^{(k)} \in \mathbb{R}^{R_{k-1} \times I_k \times R_k}$ and $(R_1, \ldots, R_K)$ denotes the multilinear rank with $R_0 = R_K$. All the entries in $\mathscr{T}_{\mathscr{Z}}$ can be represented as

$$\mathscr{T}_{\mathscr{Z}}(i_1, \ldots, i_K) = \sum_{R_1, \ldots, R_K} \prod_{j=1}^{K} \underline{\mathbf{C}}^{(j)}(R_{j-1}, i_j, R_j), \tag{5.3}$$

which can also be represented as

$$\mathscr{T}_{\mathscr{Z}}(i_1, \ldots, i_K) = \text{Trace}[\underline{\mathbf{C}}^{(1)}(:, i_1, :) \ldots \underline{\mathbf{C}}^{(K)}(:, i_K, :)], \tag{5.4}$$

where $\underline{\mathbf{C}}^{(k)}(:, i_k, :)$ denotes i_kth slice of $\underline{\mathbf{C}}^{(k)}$ that is an $R_{k-1} \times R_k$ matrix and the trace operation sums up all the diagonal entries.

The total number of elements in $\mathscr{T}_{\mathscr{Z}}$ is $d \prod_{k=1}^{K} I_k$. The total number of parameters in TR factorization reduces to $\sum_{k=1}^{K} I_k R_{k-1} R_k$ with $R_0 = R_K$. If we set all the $R_k = R$, then the total number of parameters in TR factorization becomes $R^2 \sum_k I_k$, which is significantly less than $\prod_k I_k$ in $\mathscr{T}_{\mathscr{Z}}$. In the above formulation, we ignored latent code dimension, d, because we can consider one or multiple factors in TR factorization to represent the latent space in implementation.

Image Recovery Using Tensor Factorized Autoencoder: With our proposed tensor-based autoencoder, we can recover samples that were corrupted at the training time (e.g., missing blocks/views/frames in an image/video). To learn the tensor factors from corrupted data, we formulate the loss function in (5.5) incorporating the measurement matrices for the observed samples. Then we can minimize the loss function in (5.5) to learn $\mathscr{T}_{\mathscr{E}(\mathscr{Y})}$,$\gamma_E$ and γ_D.

$$\text{loss}_{AE,missing} = \sum_{i=1}^{N} \|E_{\gamma_E}(Y_i) - \mathscr{T}_{\mathscr{E}_{\gamma_{\mathscr{E}}}(\mathscr{Y})}\|^2 + \lambda_1 \|A_i D_{\gamma_D}(E(Y_i)) - Y_i\|^2 + \lambda_2 \|A_i D_{\gamma_D}(\mathscr{T}_{\mathscr{E}(\mathscr{Y})}) - Y_i\|^2. \tag{5.5}$$

Dual Input Loss: The three terms of the loss function in (5.5) are targetted to minimize the mismatch between encoder output and factorization, encoder-decoder measurement loss and factorization-decoder measurement loss respectively. λ_1 and λ_2 are weights for different loss terms. The first term of the total loss in (5.5) measures the mismatch between encoder output and factorization. It shows how well the learned weights of the encoder can generate the factorized latent space. The second term of the total loss in (5.5) evaluates how well the encoder output performs in terms of reconstruction. The first term can only measure how well the encoder can factorize, but it is possible to find a factorized representation that may not give a meaningful reconstruction. The third term in (5.5) measures how well the decoder performs when given the latent codes formed by the tensor factors. One might wonder about the necessity of the third term. If the first term were perfectly zero, the third term would not be necessary. However, we cannot perfectly learn a mapping where the missmatch between encoder output and factorization is zero. That's where the third term comes into play. Even though the output from the encoder and the output from the tensor factors are very close, they may provide very different realizations when passed through decoder depending on the direction of mismatch. So, we kept an extra term to make sure that the latent represent from the tensor factorization also generates as good images as encoder output does.

We present a pseudocode for the missing view recovery algorithm in Algorithm 4.

Algorithm 4 Learning Tensor Factorization using Autoencoder from Corrupted Data

Input: Available data X_i, attribute label of the data, measurement model A.

Initialize encoder and decoder weights and tensor cores randomly

for $k = 1, 2, ..., K$ **do** ▷ K steps or until convergence

Calculate the loss function in (5.5).

Calculate gradients of loss w.r.t. training parameters in $\mathscr{T}_{\mathscr{E}(\mathscr{X})}$, $E(.)$ and $D(.)$ via backpropagation.

Update them using gradient descent.

end for

Intermediate Output: Optimized $\mathscr{T}_{\mathscr{E}_{\gamma_{\mathscr{E}}}(\mathscr{X})}$, $E_{\gamma_E}(.)$, and $D_{\gamma_D}(.)$.

Use optimized $\mathscr{T}_{\mathscr{E}(\mathscr{X})}$ and $D_{\gamma_D}(.)$ to estimate the missing views by $\hat{X}_i = D_{\gamma_D}(\mathscr{T}_{\mathscr{E}(\mathscr{X})})$.

Output: X_i

We illustrate the visual representation of our proposed algorithm in Figure 5.1.

5.3 Experiment and Results

Dataset: We used Small NORB [158], RaFD [159] and 3dShapes [160] datasets in our experiments. In these datasets, images of different attribute variation is available. We select 25 toys with 3 lighting conditions, 3 elevations and 9 azimuth angles (2025 images) from Small NORB dataset. We select 15 persons with 5 camera angles, 8 expressions and 3 eye gazing (1800 images) from RaFD. We selected 4 object shapes, 5 floor colors and 5 floor colors at 8 object scales (800 images) from 3dShapes dataset.

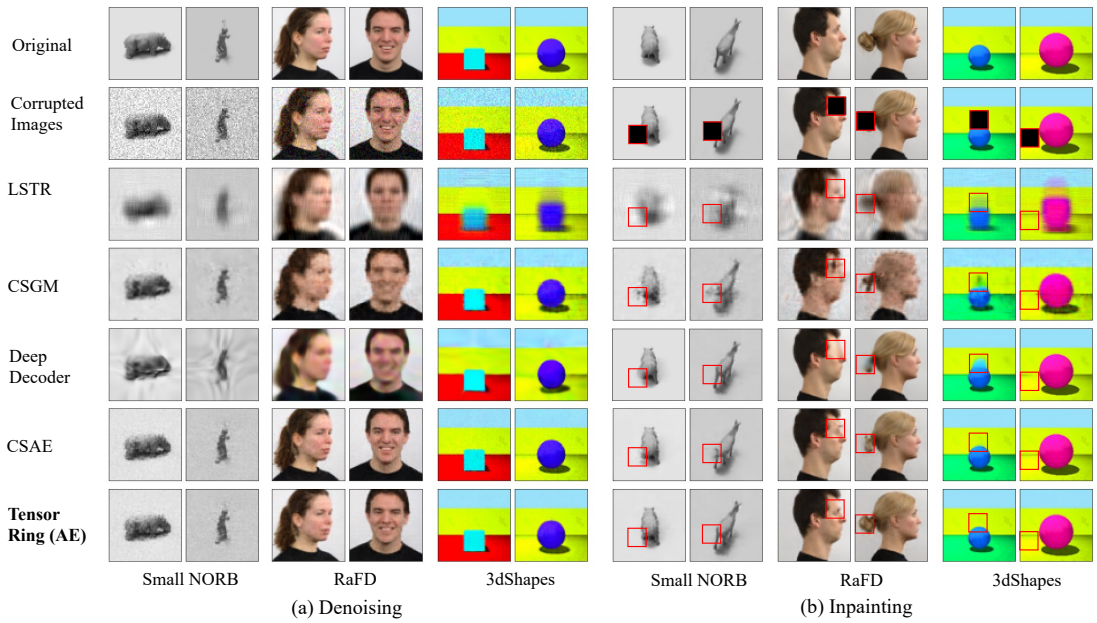


Figure 5.2: Reconstruction results for (a) denoising and (b) inpainting on Small NORB, RaFD and 3dShapes datasets.

Setup: In our experiments, we used a fully convolutional autoencoder that maps each image to a latent code $z \in \mathbb{R}^{16\times4\times4}$, which sets $d = 256(16 \times 4 \times 4)$ in our experiments. Our encoder comprises four convolutional layers (32,64,128,16 filters) and decoder comprises of four transpose convolutional layers (256, 128,64,3 (rgb) /1 (grayscale) filters) each with 3×3 filters with stride=2 followed ReLU activation except for the last layer (Sigmoid instead of ReLU). We use low-rank tensor ring factorization in the latent space. We use the same rank for all the cores of tensor ring. We empirically selected the lowest ranks for each dataset that provide good performance. We reported results with rank=25 for Small NORB, 30 for RaFD and 15 for 3dShapes. We have initialized the cores and bases for different tensor factorization using samples drawn from $N(0, 0.1)$ distribution. We kept fixed seed for every setup in order to achieve fair comparison. As we consider a batch of images correspond to

a slice of a tensor core, we could perform minibatch optimization which reduced memory requirement during training. We have used Adam [161] optimization for network parameters optimization and Stochastic Gradient Descent (SGD) for optimizing tensor factorization parameters. The learning rate for Adam was selected to be 0.001 and SGD to be 1 (or 0.1). We let the optimization run for enough iterations to converge. For autoencoder setup, we set weight terms, λ_1 and λ_2 to be 1.

Comparison: We tried to solve the inverse/compressive sensing problems (e.g. denoising and inpainting) without ground truth images. We show comparison with 4 different baselines which also perform the same task.

CSAE: We use the encoder of a deterministic autoencoder to learn the latent space from the corrupted measurements and pass the learned latent codes through decoder and measurement matrices to match with the observed measurements. One can refer to the second term of Eqn (5.5) as the objective this approach minimizes. Eventually we learn the original image given the corrupted measurements without having the ground truth. We term it **CSAE** (Compressive Sensing with Auto Encoder).

CSGM: We tried to solve the compressive sensing problems given that we have a trained generative model with the learned distribution of the target data. It is similar to the work of Bora et. al. [66]. We term it **CSGM** following [66].

LSTR: We utilize the attribute information of the structured dataset and use it as a prior to minimize the least square measurement loss with SGD. We term it **LSTR** (Least Square minimization with Tensor Ring).

Deep Decoder: Finally we use one of the self supervised generative prior based approaches

to solve the compressive sensing problems. We use **Deep Decoder** [2] for comparison.

We empirically demonstrate that our proposed Tensor Ring factorized Autoencoder outperforms all the four baselines in terms of reconstruction quality since we are using the advantages of both the structural information and generative priors.

Denoising: In this experiment, we added Gaussian noise of 20dB to all the images. We report the average reconstruction quality (dB PSNR) for different comparing techniques in Table 5.1. We also demonstrate some reconstructed images in Figure 5.2. We can observe that utilizing the structure in latent space helps us outperform the other approaches. Deep deocoder uses a single network per image recovery. So it cannot use information from the other measurements of the structure. Although CSGM uses all the training data to train the generator, it does not explicitly use the structural information. We also observe that LSTR does not provide good reconstruction performance even though it is also using the structural information because images usually do not have the tensor structure in their representation. CSAE approach performs well as it learns the optimal embedding space for solving the inverse problem using an encoder. However, it falls behind our proposed approach since it does not use any structural information. By learning an embedding space to apply tensor structure, we are utilizing the structural information to our advantage.

Image Inpainting: It is often observed in real scenario that some of the images of the structured image set are corrupted instead of being completely unavailable. We perform a set of experiments on different datasets where we missed a 16×16 block from all the images at

random locations. We feed the structured image set to the AE based tensor factorized scheme. Our tensor factorized autoencoder utilize the strength of the structured organization of the dataset to better reconstruct the images with missing blocks. We report the reconstruction results in Table 5.1. We also demonstrate some reconstructions in Figure 5.2. We can observe that we outperform the other approaches especially in recovering the original details of missing blocks as shown in Figure 5.2. Although Deep Decoder and CSAE perform very close to our approach, they fail to recover reliable details in the missing blocks. Furthermore, deep decoder is using a separate network for every image recovery. So, the memory and parameter requirement for deep decoder is significantly higher than ours.

Table 5.1: Reconstruction quality (PSNR in dB) for image denoising and inpainting with different comparing approaches.

	LSTR	Deep Decoder	CSGM	CSAE	Tensor Ring AE
Denoising					
Small NORB	23.11	27.4	28.4	27.56	**31.71**
RaFD	20.3	25.14	26.67	29.66	**32.1**
3dShapes	20.13	28.06	28.63	33.52	**35.97**
Inpainting					
Small NORB	24.7	35.1	28	33.49	**35.29**
RaFD	21.31	31.87	24.83	31.91	**33.55**
3dShapes	21.92	35.22	27.26	36.8	**39.43**

5.4 Performance as an Image Generator

5.4.1 Image Generation from Noise

Deterministic autoencoders are not usually considered to be good generative models. It is necessary to have constraints on the latent space to derive such ability [45]. Recently, IRMAE [57] also demonstrated that deterministic autoencoder can be used as a generative models with some simple modification (e.g. adding fully connected layers without nonlinearity) in the bottleneck. We demonstrate that the decoder of the tensor factorized autoencoders can generate high-quality images from random Gaussian noise. We sample random latent codes from a multivariate Gaussian distribution using a covariance matrix calculated from the latent codes of the training data. We show that we can generate class specific images from noise distribution when we sample latent codes with class-specific respective mean and covariance. We also report the noise generation results of baseline autoencoder (vanilla autoencoder), IRMAE [57] and FactorVAE [162] in and Table 5.2. FactorVAE also uses the disentanglement of different attributes/articulations for the image generation. We can observe that despite deterministic autoencoders' well-known inability to generate images from noise [57], our tensor factorized deterministic autoencoders perform significantly better than baseline autoencoder and at par with IRMAE and FactorVAE for most of the cases.

Table 5.2: Frechet Inception Distance (FID) values for images generated from noise.

	Small NORB	RaFD	3dShape
Baseline AE	202.44	227.47	123.16
IRMAE	246.19	243.79	**76.11**
Factor VAE	201.7	**78.16**	94.05
Tensor Ring AE	**173.62**	114.10	97.78

5.4.2 Latent Space Interpolation

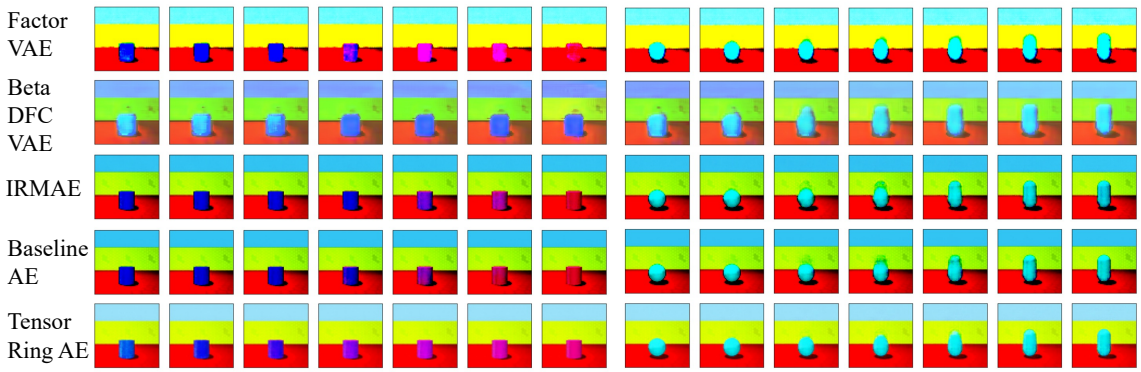
In this experiment, we analyze if we can interpolate between two image attributes by linearly interpolating between the respective tensor slices in the latent space. We report such interpolation results for 3dShapes dataset in Figure 5.3. We can observe that the baseline autoencoder fails to generate smooth transition between shapes, but tensor factorization-based models generate smooth transition with realistic images in the interpolation path. We also show latent space interpolation results for FactorVAE[162], β VAE [163], and IRMAE[57] in Figure 5.3.

Table 5.3: Training images representation performance (PSNR in dB).

	Tensor Ring AE	Image Space Tensor Ring	Baseline AE
Small NORB	**38.85**	28.56	34.52
RaFD	**33.7**	21.27	32.49
3dShapes	**40.3**	33.37	37.05

5.4.3 Data fitting performance

We learn an ambient space for tensor factorization. We observed how tensor factorized autoencoder performs in terms of clean data representation. We show comparison



(a) Interpolating object color. (b) Interpolating object shape.

Figure 5.3: Interpolation in the latent space to change object shape/size using different generators. Left and rightmost images are part of training set. The views in between are synthesized using linear interpolation in latent space.

with image space tensor factorization and baseline autoencoder in terms of representation performance (Table 5.3). We can observe that learning an ambient space significantly improves tensor factorization performance. It also surpasses the unconstrainted baseline autoencoder performance.

Chapter 6

Consensus Equilibrium to Combine DIP with RED

6.1 Introduction

Image blurring is a common artifact that arises due to a variety of problems during the image capturing stage, such as, motion/shaking of the camera, out-of-focus acquisition, atmospheric aberrations, and low-light conditions. Image deblurring is the task of resolving the blurring artifacts when the source of the blur is known. The problem can be formalized as the task of recovering a true signal $\mathbf{x}$ from blurred measurements $\mathbf{y}$ and given the blurring operator $\mathbf{A}$, such that,

$$\mathbf{y} = \mathbf{A}\mathbf{x} + \eta, \tag{6.1}$$

This work has been published in IEEE International Conference on Acoustics, Speech and Signal Processing 2021[164].

where η is the noise introduced during measurement acquisition process, which is assumed to be additive white Gaussian noise (AWGN). The blurring operator $\mathbf{A}$ convolves a blurring kernel with the true signal in the measurement model.

Image deblurring is an ill-posed problem when the operator $\mathbf{A}$ is rank deficient and in the presence of noise. Therefore, there can be infinitely many solutions $\mathbf{x}$ that satisfy equation (6.1). A general approach to solving ill-posed problems is to add a regularizer that constrains the solution set to a small subset of the feasible space. We can write a general regularized inverse problem as

$$\min_{\mathbf{x}} \|\mathbf{y} - \mathbf{A}\mathbf{x}\|_2^2 + \lambda\rho(\mathbf{x}), \tag{6.2}$$

where $\rho(\mathbf{x})$ is the regularizing penalty function, and λ is a regularization parameter. The choice of regularizer is specific to the type of signals $\mathbf{x}$ that are desired. In the case of images, a wide variety of regularizers have been proposed to regularize reconstruction in the denoising setting, i.e., when $\mathbf{A}$ is the identity operator. These include "classical" prior models [165, 166, 167, 168, 169, 170, 171, 172], and deep learning models [173, 174]. Moreover, sophisticated denoiser models that leverage nonlocal self-similarity in images are popular in state-of-the-art methods, such as, block-matching with 3D transform denoising (BM3D) [166], nonlocal means (NLM) [165], and NCSR [175].

In the context of general inverse problems with rank deficient $\mathbf{A}$, a relatively new line of work has opted to replace explicit regularizing penalty functions $\rho(\mathbf{x})$ with the deep network-based models or the sophisticated denoisers. Of note, are the plug-and-play (PnP) framework [176], and regularization by denoising (RED) [177]. The PnP framework tackles problem (6.2) using proximal gradient descent and replaces the proximal mapping with

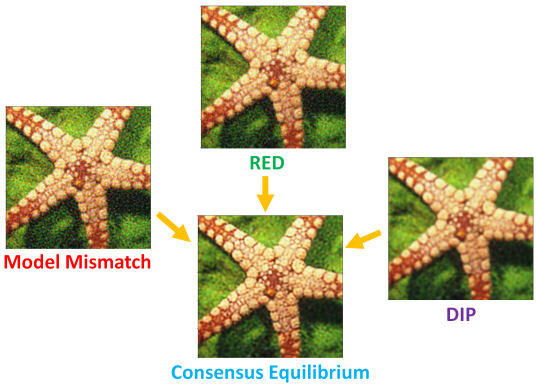


Figure 6.1: Consensus Equilibrium of model mismatch, RED, and DIP. The top images result from the action of different agents that are combined to produce the CE solution.

respect to $\rho(\cdot)$ with a signal denoiser $\mathcal{D}(\mathbf{x})$. In a related manner, the RED technique assumes that the regularizing penalty function is given by

$$\rho(\mathbf{x}) = \mathbf{x}^T(\mathbf{x} - \mathcal{D}(\mathbf{x})). \tag{6.3}$$

Note that the penalty function above is only valid when the denoiser is locally homogeneous with a symmetric Jacobian [178]. Otherwise, RED assumes that $\nabla\rho(\mathbf{x}) = \mathbf{x} - \mathcal{D}(\mathbf{x})$. We will focus in this work on the RED approach although our formulation also extends to the PnP framework.

In another line of work, untrained convolutional network architectures have been used as image prior. Deep image prior (DIP) [179] and its variants [2, 133] utilize the structural bias of convolutional networks towards producing natural images [180] in fewer update iterations compared to modeling noise. Using $\mathbf{x} = \mathcal{G}(\mathbf{z}, \boldsymbol{\theta})$ where $\mathcal{G}(\mathbf{z}, \boldsymbol{\theta})$ is a generator network using latent code, $\mathbf{z}$ and network weights $\boldsymbol{\theta}$, we can write the DIP prior as

$$\min_{\boldsymbol{\theta}} \|\mathbf{y} - \mathbf{A}\mathcal{G}(\boldsymbol{\theta})\|_2^2. \tag{6.4}$$

These untrained models, however, are susceptible to modeling measurement noise as well [179, 180] given enough optimization iterations.

In this work [164], we focus on an ensemble regularization framework, called DeepRED [181], that combines the RED approach with a nonlocal means denoiser and the deep image prior architecture. We recast the DeepRED problem in the context of Consensus Equilibrium (CE) [182] and specify the set of equilibrium equations for each of the model mismatch function, the RED denoiser, and the DIP denoiser that need to be satisfied by the target reconstructed image, as illustrated in Fig. 6.1. Contrary to the DeepRED solution that relies on an alternating direction method of multipliers (ADMM) algorithm, we use a fixed-point algorithm to solve the set of equilibrium equations. We demonstrate that the versatility provided by the CE framework leads to improved deblurring image quality compared to DeepRED, especially under high noise and high blurring situations. We also derive sufficient conditions for the generative prior network of DIP to guarantee convergence of the fixed-point iteration.

In the next section, we provide further details on the two related works, namely, DeepRED and consensus equilibrium. We then develop the CE formulation of the DeepRED problem in Section 6.3 and set up the corresponding fixed-point problem that is solved using the Mann iterations. A sufficient condition that guarantees convergence of the Mann iterations is that the fixed-point operator be nonexpansive. To that end, we derive sufficient conditions on the generative prior that guarantee that the corresponding fixed-point operator is nonexpansive. Finally, we validate the performance of our proposed approach in Section 6.4 and demonstrate improved reconstruction quality over DeepRED, DIP, RED, and NCSR.

6.2 Related Work

The deep image prior powered by the RED framework, or DeepRED [181], combines the representation power of the deep image priors with the superior denoising capabilities of a nonlocal means denoiser. [181] proposed an ADMM algorithm for solving the augmented Lagrangian of the DeepRED problem given by:

$$\begin{aligned}\min_{\boldsymbol{\theta},\mathbf{x}} \tfrac{1}{2}\|\mathbf{A}\mathcal{G}(\boldsymbol{\theta}) - \mathbf{y}\|_2^2 \quad + \quad & \tfrac{\lambda}{2}\mathbf{x}^T(\mathbf{x} - \mathcal{D}(\mathbf{x})) \\ & + \tfrac{\mu}{2}||\mathbf{x} - \mathcal{G}(\boldsymbol{\theta}) - \mathrm{u}||_2^2 - \tfrac{\mu}{2}\|\mathrm{u}\|_2^2,\end{aligned} \tag{6.5}$$

where $\mathbf{u}$ denotes the scaled dual multiplier, and μ is a step size parameter. The above formulation ties the inaccuracies of the measurement process to the rich parameterization of the generative prior. Although this formulation works well for low noise and low blurring scenarios, it suffers in the high noise setting as it tends to overfit the noisy measurements, a behavior that has also been seen in DIP.

Consensus equilibrium [182] presents a multi-agent satisfaction framework that generalizes consensus optimization to cover models and operators that are not associated with explicit optimization problems. The CE framework extends the consensus optimization objective

$$\min_{\mathbf{x}_i,\mathbf{z}} \sum_{i=1}^{N} \mu_i f_i(\mathbf{x}_i) \text{ s.t. } \quad \mathbf{x}_i = \mathbf{z}, \ \sum_{i=1}^{N} \mu_i = 1, \tag{6.6}$$

to defining a set of N vector-valued maps $F_i : \mathbb{R}^n \to \mathbb{R}^n$. The CE of these maps is then defined as any solution $(\mathbf{x}^*, \mathbf{u}^*) \in \mathbb{R}^{n \times nN}$ that satisfies the equations

$$\begin{aligned} & F_i(\mathbf{x}^* + \mathbf{u}_i^*) = \mathbf{x}^*, \ i = 1, \dots N \\ & \sum_{i=1}^{N} \mu_i \mathbf{u}_i^* = 0, \end{aligned} \tag{6.7}$$

where $\mathbf{u}^* := \left[\mathbf{u}_1^{*T}, \ldots \mathbf{u}_N^{*T}\right]^T$. In the following sections, we derive the CE equations for the DeepRED problem, setting up the corresponding fixed-point equations, and demonstrating the improved performance over DeepRED's ADMM implementation.

Other recent efforts on the image deblurring have been based on variants of the fast iterative shrinkage/thresholding algorithm (FISTA) [183, 184], and on trained generative prior [66]. While these methods demonstrate state-of-the-art performance on image deblurring, they could be combined with our framework as additional mapping functions in the CE formulation.

6.3 Consensus Equilibrium for DIP and RED

In this section, we describe a consensus equilibrium perspective for combining DIP and RED to regularize linear inverse problems.

6.3.1 DeepRED as fixed-point CE

We begin with a reformulation of the DeepRED objective, which disentangles the parameterization of the generative prior from the measurement process:

$$\min_{\mathbf{x},\boldsymbol{\theta}} \mu_1 \|\mathbf{y} - \mathbf{A}\mathbf{x}\|_2^2 + \mu_2 \mathbf{x}^T(\mathbf{x} - \mathcal{D}(\mathbf{x})) + \mu_3 ||\mathbf{x} - \mathcal{G}(\boldsymbol{\theta})||_2^2. \tag{6.8}$$

It can be seen from the above formulation that consensus is sought for three objectives:

$$f_1(\mathbf{x}, \boldsymbol{\theta}) = \|\mathbf{y} - \mathbf{A}\mathbf{x}\|_2^2$$

$$f_2(\mathbf{x}, \boldsymbol{\theta}) = \mathbf{x}^T(\mathbf{x} - \mathcal{D}(\mathbf{x}))$$

$$f_3(\mathbf{x}, \boldsymbol{\theta}) = \|\mathbf{x} - \mathcal{G}(\boldsymbol{\theta})\|_2^2,$$

where the measurement mismatch objective, $f_1(\mathbf{x}, \boldsymbol{\theta})$, and the RED objective, $f_2(\mathbf{x}, \boldsymbol{\theta})$, are in fact independent of $\boldsymbol{\theta}$. Following the CE framework, we can now define three *agents* in the form of proximal mappings with respect to each of the objectives f_i as follows:

$$\begin{aligned} F_1(\mathbf{v}) &= \arg\min_{\mathbf{x}} \|\mathbf{y} - \mathbf{A}\mathbf{x}\|_2^2 + \tfrac{1}{2\sigma^2}\|\mathbf{v} - \mathbf{x}\|_2^2 \\ F_2(\mathbf{v}) &= \arg\min_{\mathbf{x}} \|\mathbf{x}^T(\mathbf{x} - \mathcal{D}(\mathbf{x}))\|_2^2 + \tfrac{1}{2\sigma^2}\|\mathbf{v} - \mathbf{x}\|_2^2 \\ F_3(\mathbf{v}, \boldsymbol{\theta}) &= \arg\min_{\mathbf{x}, \boldsymbol{\phi}} \|\mathbf{x} - \mathcal{G}(\boldsymbol{\phi})\|_2^2 \\ &\qquad + \tfrac{1}{2\sigma^2}(\|\mathbf{v} - \mathbf{x}\|_2^2 + \|\boldsymbol{\phi} - \boldsymbol{\theta}\|_2^2) \end{aligned} \tag{6.9}$$

Consensus equilibrium for these agents is defined as in (6.7), where the variables $\mathbf{u}_i$ are slack variables that relate to the dual multipliers in the consensus optimization setting.

The evaluation of F_1 and F_2 is relatively straightforward and is given by the following equations

$$\begin{aligned} F_1(\mathbf{v}_1) &= \left(\mathbf{A}^T\mathbf{A} + \frac{1}{2\sigma^2}\mathbf{I}\right)^{-1}\left(\mathbf{A}^T\mathbf{y} + \frac{\mathbf{v}_1}{2\sigma^2}\right) \\ F_2(\mathbf{v}_2) &= \frac{2\sigma^2\mathcal{D}(\mathbf{v}_2) + \mathbf{v}_2}{2\sigma^2 + 1} \end{aligned} \tag{6.10}$$

The DIP agent, on the other hand, is a function of both the signal $\mathbf{x}$ and the generative prior parameters $\boldsymbol{\phi}$. Its evaluation can be performed sequentially by first computing $\boldsymbol{\phi}^*$, followed by the update for $\mathbf{v}_3$ using the following equations:.

$$\begin{aligned} \boldsymbol{\phi}^* &= \arg\min_{\boldsymbol{\phi}} \|\mathbf{v}_3 - \mathcal{G}(\boldsymbol{\phi})\|_2^2 + \frac{2\sigma^2 + 1}{2\sigma^2}\|\boldsymbol{\phi} - \boldsymbol{\theta}\|_2^2 \\ F_3(\mathbf{v}_3) &= \frac{\mathbf{v}_3}{2\sigma^2 + 1} + \frac{2\sigma^2}{2\sigma^2 + 1}\mathcal{G}(\boldsymbol{\phi}^*) \end{aligned} \tag{6.11}$$

Notice that with the evaluation of $\boldsymbol{\phi}^*$ in (6.11), we solve for a regularized set of generative prior parameters. This additional regularization helps in limiting the noise overfitting behavior that is generally observed with DIP. The regularization also relaxes the

Algorithm	**Gaussian Kernel** ($\sigma_k = 1.6$)					**Uniform Kernel**				
	Butterfly	Leaves	Parrots	Starfish	Average	Butterfly	Leaves	Parrots	Starfish	Average
CE-DIP+RED	**32.27**	**32.76**	**33.34**	**33.09**	**32.87**	31.31	**31.32**	**32.32**	**31.1**	**31.55**
DeepRED	32.19	32.27	32.84	32.74	32.51	**31.44**	31.21	32.03	31.06	31.43
DIP	31.21	31.51	31.91	31.83	31.62	30.26	30.38	31.00	30.42	30.51
RED	31.66	31.93	33.33	32.49	32.35	30.41	30.13	31.83	30.57	30.74
NCSR Deblur	30.84	31.57	33.39	32.27	32.02	29.68	29.98	31.95	30.28	30.47
Blurred	22.81	22.12	26.96	25.83	24.43	19.07	18.28	23.87	22.56	20.94

Table 6.1: Comparison of reconstruction PSNR among the different algorithms under low noise setting ($\sigma_n = \sqrt{2}/255$).

dependence on the heuristic of assigning an arbitrary number of iterations while updating $\boldsymbol{\phi}$ to limit the noise overfitting.

As in [182], we reformulate CE as a fixed-point problem. Denoting $\mathbf{v}_i = \mathbf{x} + \mathbf{u}_i$, we have $\mathbf{v}_\mu := \sum_i \mathbf{v}_i = \mathbf{x}$. Further with $F(\mathbf{v}) = [F_1(\mathbf{v}_1)^T, F_2(\mathbf{v}_2)^T, F_3(\mathbf{v}_3)^T]^T$ and $H_\mu(\mathbf{v}) = [\mathbf{v}_\mu^T, \mathbf{v}_\mu^T, \mathbf{v}_\mu^T]^T$, where $\mathbf{v} = [\mathbf{v}_1^T, \mathbf{v}_2^T, \mathbf{v}_3^T]^T$, the CE equations are rewritten as:

$$F(\mathbf{v}) = H_\mu(\mathbf{v}). \tag{6.12}$$

Due to the linearity of H_μ, we can use Corollary 3 of [182] to define the following equivalent fixed-point problem:

$$(2H_\mu - \mathbf{I})(2F - \mathbf{I})(\mathbf{v}) = \mathbf{v}, \tag{6.13}$$

where $\mathbf{I}$ is the identity operator.

Next, define the operator $T := (2H_\mu - \mathbf{I})(2F - \mathbf{I})$. When T is nonexpansive and has a fixed-point, the Mann iteration can be used to solve for the fixed-point of (6.13) as follows:

$$\mathbf{v}^{k+1} = (1 - \rho^k)\mathbf{v}^k + \rho^k T(\mathbf{v}^k), \tag{6.14}$$

where $\rho^k \in (0, 1)$ is a step size parameter. Theorem 5.15 in [185] shows that when T is nonexpansive, a step size sequence that obeys $\sum_{k \in K} \rho_k(1 - \rho_k) = +\infty$ allows (6.14) to

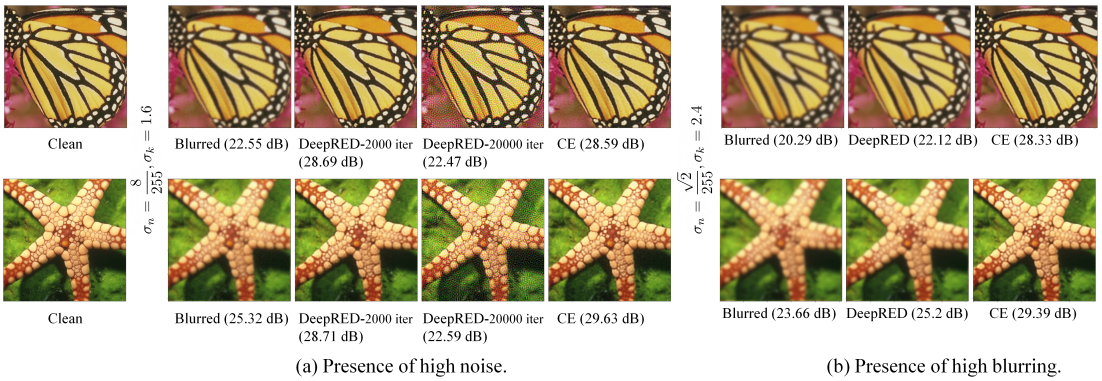


Figure 6.2: Image deblurring performance of DeepRED and CE formulation under (a) the presence of high noise ($\sigma_n = 8/255$,$\sigma_k = 1.6$) and (b) the presence of high blurring ($\sigma_n = \sqrt{2}/255$,$\sigma_k = 2.4$).

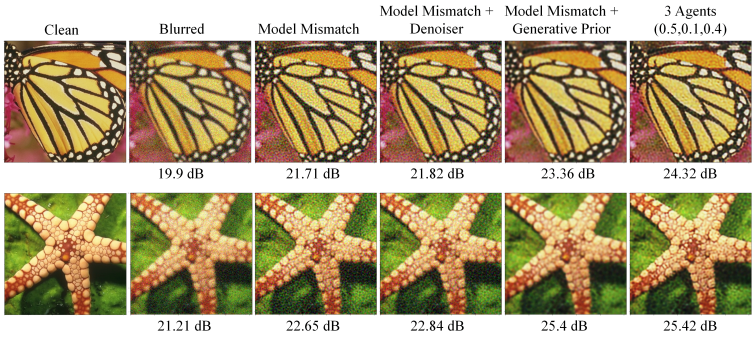


Figure 6.3: Reconstruction quality resulting from the combination of the three different agents.

converge weakly to a point in the fixed-point set of T. Examples of such sequence are the constant step size $\rho_k = \rho \ \forall k$, and the p-series, $\rho_{k+1} = \rho_k k^{-c}$ for $0 < c < 1$, which enjoys faster convergence.

6.4 Experimental Validation

We follow an experimental setup similar to the that developed in [181], [177] and [186]. Given a blurred and noisy image with a known degradation operator, the goal is to

(σ_k, σ_n)	**Algorithms**	**Butterfly**	**Leaves**	**Parrots**	**Starfish**	**Average**
$(1.6, \frac{8}{255})$	CE	**28.59**	**28.79**	**30.66**	**29.63**	**29.42**
	DeepRED (2000 iters)	**28.69**	28.23	30.1	28.71	28.93
	DeepRED (20000 iters)	22.47	24.1	22.81	22.59	23
	Blurred	22.55	21.89	26.34	25.32	24.02
$(1.6, \frac{32}{255})$	CE	**24.32**	**24.34**	**27.19**	**25.42**	**25.32**
	DeepRED (250 iters)	**25.18**	23.83	27.02	25.1	25.28
	DeepRED (1000 iters)	22	22.12	23.64	21.99	22.44
	Blurred	19.9	19.64	21.59	21.21	20.58
$(2.4, \frac{\sqrt{2}}{255})$	CE	**28.33**	**27.93**	**30.43**	**29.39**	**29.02**
	DeepRED	22.12	21.37	26.41	25.2	23.78
	Blurred	20.29	19.54	24.94	23.66	22.11
$(3.2, \frac{\sqrt{2}}{255})$	CE	**25.62**	**24.4**	**28.1**	**27.21**	**26.33**
	DeepRED	19.47	18.76	24.4	23	21.41
	Blurred	18.69	18.03	23.85	22.36	20.73

Table 6.2: Comparison of reconstruction PSNR for different noise levels and blurring kernel strengths.

recover the sharp and noise-free original image. We consider blurring kernels with varying blurring effect controlled by the parameter σ_k and add varying levels of i.i.d. Gaussian noise with variance σ_n.

To evaluate the reconstruction, we used four images from the NCSR dataset [175] (Butterfly, Leaves, Parrot and Starfish) similar to the selection in [181]. Each of these images has 256×256 pixels with RGB color channels. For fair comparison with [181], we also use the same nonlocal means (NLM) denoiser. We tuned the parameters for a single image (Butterfly) at low noise ($\sigma_n = \sqrt{2}/255$) and low-blur ($\sigma_k = 1.6$) setup. The step size parameters were set to $\rho_0 = 0.9$ and $c = 0.1$. We used $\sigma = 1, 4, 20$ for high, medium and low noise setup, respectively. We used ADAM optimizer with learning rate 0.001,

$\beta_1 = 0.9, \beta_2 = 0.999$ to compute $\boldsymbol{\phi}^*$ in equation (6.11).

In the first set of experiments, we replicate the evaluation setup from [181]. Two blurring kernels are used; one kernel is a 9×9 pixel uniform blur, and the other is a 25×25 pixel Gaussian blur of variance $\sigma_k = 1.6$. For both blurring cases, the measurement noise variance is set equal $\sigma_n = \sqrt{2}/255$. The deblurring results are shown in Table 6.1 which lists the reconstructed peak signal to noise ratio (PSNR) values for the different deblurring algorithms, namely, DeepRED [181], DIP [179], RED [177] and NCSR Deblur [175]. Notice that our proposed CE-based solution (CE-DIP+RED) achieves the best performance in almost all cases, resulting in an average improvement over DeepRED of 0.36dB in PSNR for the Gaussian blurring case and 0.12dB in the uniform blurring case. We can also observe that for this experimental regime, DeepRED performs nearly as well as CE-DIP+RED and outperforms the other three methods.

Weights on Agents (Mismatch, RED,DIP)	**Butterfly**	**Leaves**	**Parrots**	**Starfish**	**Average**
Mismatch (1,0,0)	21.71	21.85	22.98	22.65	22.3
Mismatch + RED (0.5,0.5,0)	21.82	21.92	23.19	22.84	22.44
Mismatch + DIP (0.5,0,0.5)	23.36	22.54	26.56	25.4	24.46
Mismatch +RED+DIP (0.5,0.1,0.4)	**24.32**	**24.34**	**27.19**	**25.42**	**25.32**
Blurred	19.9	19.64	21.59	21.21	20.58

Table 6.3: Reconstruction PSNR for the different agents.

The above experiment is considered a low-noise and low-blur regime. For a more extensive evaluation, we test the performance of CE-DIP+RED and compare it to DeepRED under higher noise and blurring regimes. The results are reported in Table 6.2. We can observe that under the higher noise condition, the performance of DeepRED is less stable in that it achieve a high PSNR in early iterations but converges to a solution with much lower PSNR. This behavior is consistent with the analysis from [180] and [179] and is most likely a result of the deep image prior overfitting the noise in the measurements. On the other hand, our CE-DIP+RED does achieves a higher reconstruction PSNR at convergence and does not succumb to the noise overfitting problem. A qualitative evaluation is also shown for *medium* noisy measurements regime ($\sigma_n = 8/255$) in Figure 6.2. In the case of large blurring artifacts, Table 6.2, also shows that our CE-DIP+RED significantly outperforms DeepRED. Qualitative results are also shown for *medium* blurred images ($\sigma_k = 2.4$) in Figure 6.2. Finally, we conduct an ablation study to realize the effect of each of the different agents on the reconstruction quality. For these experiments, we used a Gaussian kernel with $\sigma_k = 1.6$ and measurement noise variance $\sigma_n = 32/255$. Table 6.3 shows the reconstruction PSNR and demonstrates that the combination of all three agents results in the best reconstruction performance. Moreover, the table shows that benefit of the generative prior over the NLM denoiser. A qualitative comparison is also shown in Figure 6.3.

Chapter 7

Phase Retrieval with Deep Generative Network

7.1 Introduction

The phase retrieval problem seeks to recover a real- or complex-valued unknown signal $\mathrm{x}^* \in \mathbb{R}^n$ from its (possibly noisy) amplitude-only observations $\mathbf{y} \in \mathbb{R}^m$ of the form:

$$y_i = |\langle \mathbf{a}_i, \mathrm{x}^* \rangle| + e_i, \quad i = 1, ..., m, \tag{7.1}$$

We construct $\mathbf{A} = [\mathbf{a_1}\ \mathbf{a_2}\ ...\ \mathbf{a_m}]^T$ with i.i.d. Gaussian entries. For simplicity, we ignore the noise e_i. We consider a setting with $m < n$, thus in general, the inverse problem in (7.1) is highly ill-posed.

A conventional approach for solving such a problem is by constraining the solution to a set $\mathcal{M} \subseteq \mathbb{R}^n$ that captures some sort of known structure that x^* is expected to obey.

This work has been published in IEEE International Conference on Acoustics, Speech and Signal Processing 2019[78].

The resulting optimization can be written as

$$\widehat{\mathbf{x}} = \arg\min \; f(\mathrm{y}; |\mathbf{A}\,\mathrm{x}|) \tag{7.2}$$

$$\text{s.t.} \quad \mathrm{x} \in \mathcal{M}.$$

A common modeling assumption on x^* is *sparsity*, which alleviates the ill-posed nature of the inverse problem, and in fact, makes the accurate recovery of x^* possible.

However, while being powerful from a computational standpoint, the sparsity prior has somewhat limited discriminatory capability, and it is certainly true that nature exhibits far richer nonlinear structure than sparsity alone. Thus, we focus on a newly emerging family of priors that are *learned* from massive amounts of training data using generative networks such as GAN [42]. A well-trained generator closely captures the notion of a signal (or image) being 'natural' [187]. While such generative priors have been used successfully in solving compressive sensing and other inverse problems [66], including phase retrieval [76, 96], the optimal way to search for the solution within the range of generative prior has not yet been understood well. Most of these methods rely on loss minimization through gradient descent that often fails to search the entire solution space resulting in sub-optimal results. In this work, we provide two algorithms that enable an improved way of searching the solution space. Our work improves on the results of [76, 96] empirically.

In this work [78], we propose and analyze two phase retrieval algorithms: alternating phase gradient descent (APGD), and alternating phase projected gradient descent (APPGD) to leverage generative priors. We improve over the approaches of [76, 96] by combining the gradient descent and projected gradient descent methods for generative priors [66, 69] with AltMin-based non-convex optimization techniques used in sparse phase retrieval [92, 93].

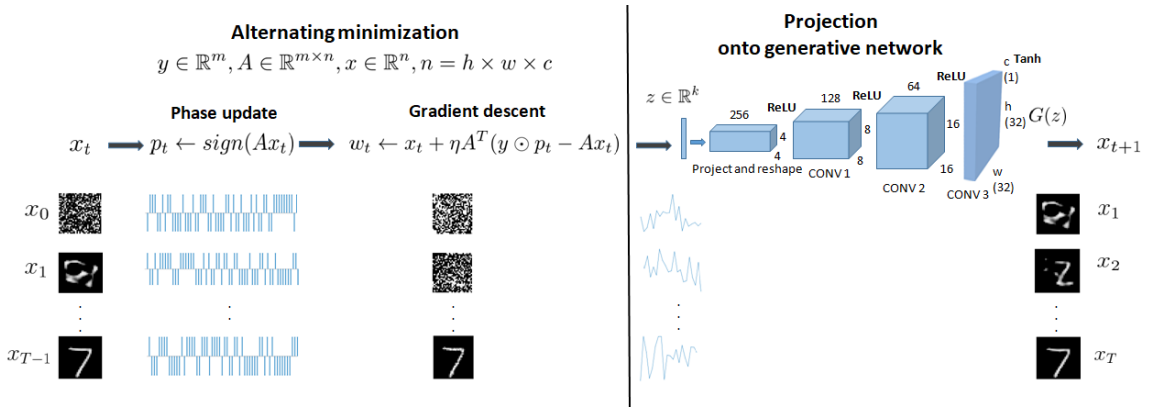


Figure 7.1: *Illustration of APPGD algorithm. It has two major steps: alternating minimization and projection onto the range of the generator nework. In alternating minimization step, we update phase and perform one gradient descent update using the updated phase. Starting from a random vector, we perform phase update, gradient descent update step and projection step iteratively to reach the final estimate.*

We adopt a setting similar to [76, 96], and assume that the generator network (say, G) well approximates the high-dimensional probability distribution of the set $\mathcal{M}$, *i.e.*, we expect that for each vector x^* in $\mathcal{M}$, there exists a vector $\mathrm{x} = G(\mathrm{z})$ that is very close to x^* in the support of the distribution defined by G.

$$\mathcal{M} = \{\mathrm{x} \in \mathbb{R}^n | \, \mathrm{x} = G(\mathrm{z}) \text{ for some } \mathrm{z} \in \mathbb{R}^k\},$$

With this assumption, the solution to (7.2) can be obtained by solving the following optimization problem:

$$\widehat{\mathrm{x}} = \arg\min_{\mathrm{x}} \| \mathrm{y} - |\mathbf{A}\,\mathrm{x}| \|^2 \tag{7.3}$$

$$\text{s.t.} \quad \mathrm{x} = G(\mathrm{z}),$$

where z is the latent code corresponding to image x. Unless otherwise stated, all norms represented by $\|\cdot\|$ in this chapter are Euclidean norms.

Recent work in [76, 96] minimizes the objective in (7.3) directly over the latent variable z using gradient descent, and sets $\widehat{\mathrm{x}}$ as:

$$\widehat{\mathrm{x}} = G(\arg\min_{\mathrm{z}} \| \mathrm{y} - |\mathbf{A}G(\mathrm{z})|\|^2). \tag{7.4}$$

We refer to this approach as the "gradient descent approach". Given that the generative models usually exhibit highly non-linear behavior, the above objective is highly non-convex. Moreover, direct application of gradient descent over z limits the explorable solution space, as at any stage it is not possible to explore the region outside the range of the generator. If initialized incorrectly, gradient descent can get stuck in local minima. In practice such algorithms require several restarts in order to provide good performance.

In phase retrieval problems, knowledge of phase and the signal is interdependent, as given the phaseless measurements just knowing the phase often enables us to estimate the signal. Thus, as alternative for solving (7.3), we can convert the phase retrieval problem to a linear inverse problem by initializing with a random phase $\mathbf{p}$ and update the phase with the solution of the linear inverse problem. Equation (7.5) describes this approach for the t^{th} iteration.

$$\widehat{\mathrm{x}}_{t+1} = G(\arg\min_{\mathrm{z}} \|\mathbf{p}_t \odot \mathrm{y} - \mathbf{A}G(\mathrm{z})\|^2) \tag{7.5}$$

We refer this approach as the alternating phase gradient descent (APGD) approach.

We also propose a third approach, in which we use projected gradient descent (PGD) to solve (7.5) directly in the ambient space based on [69]. Through iterative projections, we are able to mitigate the effects of local minima and are able to explore the space outside the range of the generator (G). In PGD, we update our estimate of x with the standard gradient descent update rule, followed by projection of the output onto the span of generator, G. We

refer this approach as the alternating phase projected gradient descent (APPGD) approach. We provide theoretical analysis of our methods, along with extensive experimental results.

7.2 Algorithm

In this section we describe the APPGD approach in details. At first, we train a generator $G : \mathbb{R}^k \to \mathbb{R}^n$ that maps a latent vector $\mathrm{z} \in \mathbb{R}^k$ to a high dimensional sample space $G(\mathrm{z}) \in \mathbb{R}^n$. We assume that our generator network can closely approximate the probability distribution of the set of natural images, $\mathcal{M}$ to which our original images x belong. With this assumption, we can limit our search for $\widehat{\mathrm{x}}$ only to the range of the generator function, $\mathcal{M}$. The generator G is assumed to be differentiable, and hence we use back-propagation for calculating the gradients of the loss functions involving G for gradient descent updates.

In each iteration of the APPGD algorithm (Alg. 5), three steps are performed: a phase update step, a gradient descent update step, and a projection step.

7.2.1 Phase update

The first step is to calculate the phase of $\mathbf{A}\,\mathrm{x}$. For real $\mathbf{A}$ and x, at the t^{th} iteration, we update the phase estimate:

$$\mathbf{p}_t = \text{phase}(\mathbf{A}\,\mathrm{x}_t) \coloneqq \text{sign}\left(\mathbf{A}\,\mathrm{x}_t\right).$$

After calculating the phase vector $\mathbf{p}$, we can use an element-wise product between $\mathbf{p}$ and y as an estimate of linear measurements and convert the phase retrieval problem into a linear inverse problem.

7.2.2 Gradient descent update

The second step is simply an application of a gradient descent update rule on the loss function $f(\cdot)$ which is given as:

$$f(\mathrm{x}) \coloneqq \|\,\mathrm{y} \odot \mathbf{p} - \mathbf{A}\,\mathrm{x}\|^2.$$

Thus, the gradient descent update at the t^{th} iteration is given by:

$$\mathbf{w}_t \leftarrow \mathrm{x}_t + \eta \mathbf{A}^T(\,\mathrm{y} \odot \mathbf{p}_t - \mathbf{A}\,\mathrm{x}_t),$$

where η is the learning rate.

7.2.3 Projection step

In projection step, we aim to find an image from the span of the generator, $\mathcal{M}$ which is closest to our current estimate $\mathbf{w}_t$. We define the projection operator $\mathcal{P}_G$ as follows:

$$\mathcal{P}_G\left(\mathbf{w}_t\right) \coloneqq G\left(\arg\min_{\mathrm{z}} L_{in}(\,\mathrm{z})\right),$$

where L_{in} is the inner loss function defined as,

$$L_{in}(\,\mathrm{z}) \coloneqq \|\mathbf{w}_t - G(\,\mathrm{z})\|^2.$$

We solve the inner optimization problem by running gradient descent with T_{in} number of updates on $L_{in}(\,\mathrm{z})$. The learning rate η_{in} is chosen empirically for this inner optimization.

In each of the T iterations, we run T_{in} updates for calculating the projection. Therefore, $T \times T_{in}$ is the total number of gradient descent updates required in our approach.

Algorithm 5 APPGD

Inputs: y, $\mathbf{A}$, G, T, **Output:** $\widehat{\mathrm{x}}$

Choose an initial point $\mathrm{x}_0 \in \mathbb{R}^n$

for $\mathrm{t} = 1,\ldots \mathrm{T}$ **do**

$\quad \mathbf{p}_{t-1} \leftarrow \mathrm{sign}\left(\mathbf{A}\,\mathrm{x}_{t-1}\right)$

$\quad \mathbf{w}_{t-1} \leftarrow \mathrm{x}_{t-1} + \eta \mathbf{A}^T(\,\mathrm{y} \odot \mathbf{p}_{t-1} - \mathbf{A}\,\mathrm{x}_{t-1})$

$\quad \mathrm{x}_t \leftarrow \mathcal{P}_G(\mathbf{w}_{t-1}) = G\left(\arg\min_{\mathrm{z}} \|\mathbf{w}_{t-1} - G(\,\mathrm{z})\|\right)$

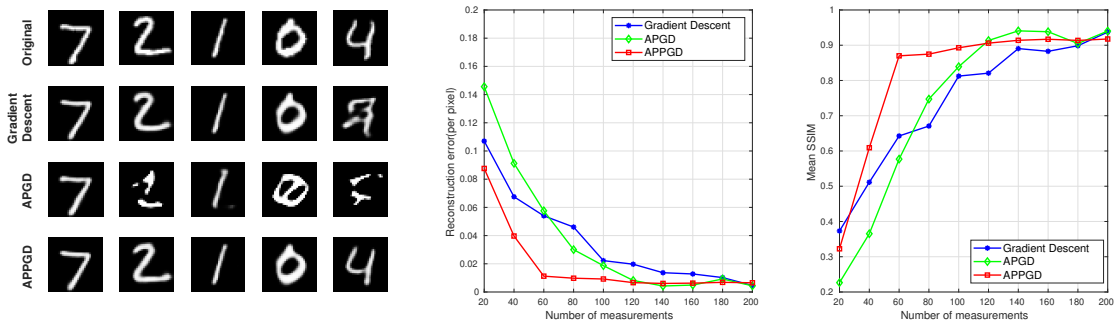
end for

$\widehat{\mathrm{x}} \leftarrow \mathrm{x}_T$

7.3 Models and Experiments

In this section, we describe our experimental setup and report the performance comparisons of the three approaches. We use two different generative models for the MNIST and CelebA datasets. The generative model for CelebA follows the DCGAN framework [140] except that we do not use any batchnorm layer since the gradient for this layer is dependent on batch size and the distribution of the batch. The generator architecture for MNIST experiments is shown in Fig. 7.1. We train our generators by jointly optimizing generator parameters, γ and the latent code, z using SGD optimization by following the procedure from [49]. We use the squared-loss function, $l_2(\,\mathrm{x}, \widehat{\mathrm{x}}) = \|\,\mathrm{x} - \widehat{\mathrm{x}}\|^2$ to train the generators. We choose z from the standard normal distribution on $\mathbb{R}^k$ and then rescale it by its Euclidean norm. We project z back to the unit norm ball after each gradient update.

In our experiments, we choose the entries of the matrix A independently from the $\mathcal{N}(0, \frac{1}{m})$ distribution. Although we ignore the presence of noise, it is possible to replicate



(a) Reconstruction results on MNIST for three different approaches with $m = 60$ measurements.

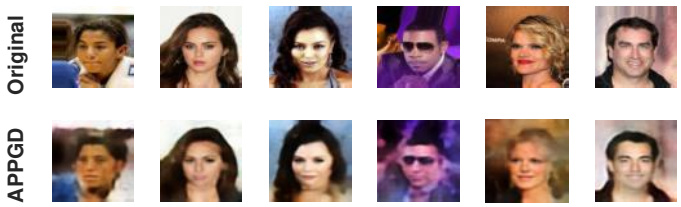
(b) Reconstruction error (per pixel) for three approaches on MNIST.

(c) Mean SSIM for three approaches on MNIST.

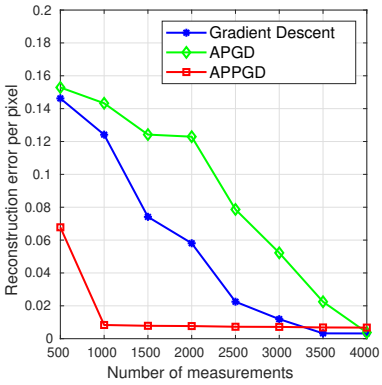
Figure 7.2: *Comparison of three approaches on MNIST test set.*

our experiments with additive Gaussian noise. For all the approaches we kept the number of update steps fixed. We do not allow random restarts. For fair comparison, we initialize x with the same random vector for all approaches and perform the same sign correction as in [76] on them.

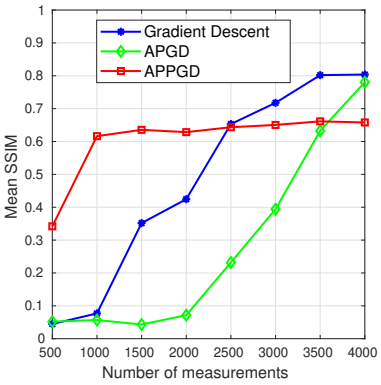
We have our first set of experiments with three different approaches on a generator trained over the MNIST training dataset resized to 32×32 pixel. Considering that the representation error is very small, we test three approaches on 10 images from the test set of MNIST dataset and provide both quantitative and qualitative results. For APPGD, at the gradient descent step we choose $\eta = 0.9$ because we need a meaningful output before passing it to the projection step [69]. We can also perform gradient descent multiple times at the first iteration before projecting it onto the range of generator so that we can start from a good initial point. For all three approaches, we use learning rate $\eta_{in} = 0.01$. We use $T = 50$ and $T_{in} = 500$ for APPGD and APGD approaches. For fair comparison, we use 2500



(a) Reconstruction results on celebA dataset for APPGD with $m = 1000$ measurements.



(b) Reconstruction error (per pixel) for three approaches on celebA.



(c) Mean SSIM for three approaches on celebA.

Figure 7.3: *Comparison of three approaches on celebA test set and some reconstruction results for our APPGD algorithm.*

iterations for Gradient descent approach. We measure reconstruction error, $\|\widehat{\mathrm{x}} - \mathrm{x}^*\|^2$, and SSIM for comparison. In Fig. 7.2b, we show the reconstruction error comparisons and in Fig. 7.2c we show SSIM comparisons for increasing values of number of measurements. As the input images are not chosen from the span of the generator itself, it is not possible to reach zero error. However, we observe from 7.2b that APPGD can reach near zero error with only 60 measurements which is significantly less than the other two approaches. Fig. 7.2a depicts reconstruction results for some of the selected MNIST images for three approaches.

For our second set of experiments, we train a generator for the CelebA dataset. For training, we resize the celebA dataset composed of 202,599 colored images of celebrity faces to $64 \times 64 \times 3$ and kept $\frac{1}{32}$ of the images apart. We do not use the aligned and cropped version which includes only the faces in the images.

We experiment on a subset of 10 images from the held out test dataset and report reconstruction results. We set the total number of updates to 1500, with $T = 50$ and $T_{in} = 300$ for APGD and APPGD approaches. Learning rates for APPGD are set as $\eta = 0.9$ and $\eta_{in} = 0.3$. Learning rates for APGD and Gradient Descent approaches are set as $\eta_{in} = 0.003$ (tuned to their best performance) for a fixed total number of updates. Image reconstruction results from $m = 1000$ measurements with APPGD algorithm are displayed in Fig. 7.3a. We show comparison of three approaches in terms of reconstruction error in 7.3b and in terms of SSIM in 7.3c. We observe that APPGD can achieve good reconstruction with far fewer measurements than the other competing approaches.

Chapter 8

Generative Network with Side Information for Phase Retrieval

8.1 Introduction

Fourier phase retrieval faces different trivial ambiguities because of the structure of Fourier transformation. As a phase shift in the Fourier domain results in a circular shift in the spatial domain, we will get the same Fourier amplitude measurements for any circular shift of the original signal.

In particular, Fourier phase retrieval is highly sensitive to the initialization and comes with inherent ambiguities about shift and flip on images. We can demonstrate shift ambiguity using the following equation

$$\mathrm{y} = |F\,\mathrm{x}| = |(F\,\mathrm{x})e^{j\theta_1}| = |(F\,\mathrm{x})e^{j\theta_2}| \tag{8.1}$$

This work has been published in Asilomar Conference on Signals, Systems, and Computers 2019[102].

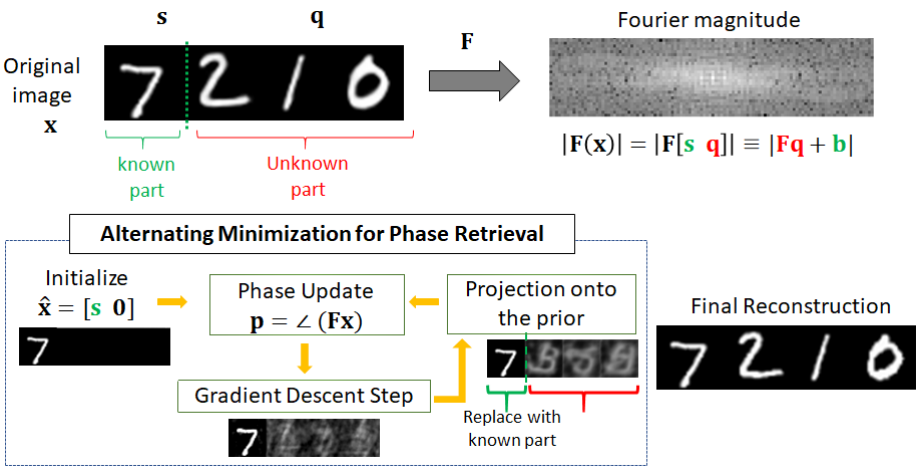


Figure 8.1: Illustration of phase retrieval with side information. An image $\mathbf{x}$ is divided into a known ($\mathbf{s}$) and unknown part ($\mathbf{q}$) such that $\mathbf{x} = [\mathbf{s}\ \mathbf{q}]$. Fourier amplitude measurements of the image are observed as $|F(\mathbf{x})| \equiv |F\mathbf{q} + \mathbf{b}|$. Alternating minimization algorithm uses the knowledge of the known part to initialize the problem and enforces additional constraints on the signal estimate at every iteration.

We can notice that the Fourier magnitude for different circular shifts (θ_1 and θ_2) has the same Fourier magnitude measurements.

In recent papers [99, 100, 101], authors tried to use side information with sparsity prior to mitigate these ambiguities. We use side information in combination with generative prior to solve Fourier phase retrieval. We show that using side information provides significant performance improvement in Fourier phase retrieval with generative prior.

In this work [102], we propose to use additional side information about the signal to initialize and regularize the phase retrieval with generative prior. In particular, we assume that a part of the signal is known a priori. An example is illustrated in Fig. 8.1. We use the known part of the signal as an initial solution and incorporate the support knowledge as an

additional constraint while solving the phase retrieval problem.

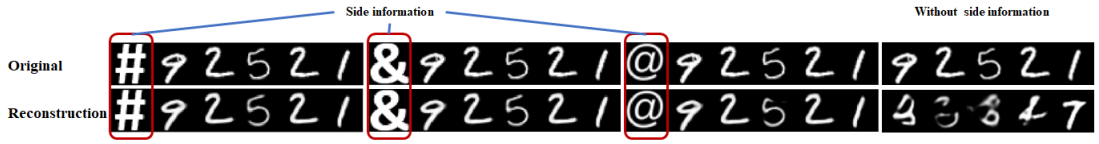


Figure 8.2: Simulation results for Fourier phase retrieval with different side information. We form an image by concatenating five MNIST digit images. We have concatenated different known 32×32 image to that concatenated MNIST image as the first patch. We use a trained generative network which is trained on MNIST digits as prior. We project each 32×32 digit onto the range of that trained prior. The last column shows reconstruction without side information. We can observe that side information gives significant performance boost.

8.2 Proposed Method

To solve the phase retrieval problem, we propose using the available side information within the alternating minimization framework. Let us denote the signal of interest as $\mathrm{x}^* \in \mathbb{C}^{N+N_s}$, where N_s values in x^* on a support set σ are known a priori. If we assume that set σ is the first N_s entries, we can write $\mathrm{x}^* = [\mathbf{s}^* \ \mathbf{q}^*]$, where $\mathbf{s}^* \in \mathbb{C}^{N_s}$ is the known part and $\mathbf{q}^* \in \mathbb{C}^N$ is the unknown part in x^*.

Suppose we are given M (possibly noisy) amplitude-only observations of x^* as

$$\mathrm{y} = |F\,\mathrm{x}^*| + \mathbf{e}, \tag{8.2}$$

where $\mathrm{y} \in \mathbb{R}_+^M$, F denotes a measurement operator, and $\mathbf{e}$ denotes noise in the measurements. Our signal can be written as the following

$$\mathrm{x}^* = \mathbf{s}^* + \mathbf{q}^*, \tag{8.3}$$

where

$$\mathbf{s}_i^* = \begin{cases} \mathrm{x}_i^*, & \text{if } i \in \sigma. \\ 0, & \text{otherwise.} \end{cases} \tag{8.4}$$

and

$$\mathbf{q}_i^* = \begin{cases} 0, & \text{if } i \in \sigma. \\ \mathrm{x}_i^*, & \text{otherwise.} \end{cases} \tag{8.5}$$

By separating the known and unknown parts, we can write equation 8.2 as

$$\mathrm{y} = |F\mathbf{s}^* + F\mathbf{q}^*| + \mathbf{e} \tag{8.6}$$

$$= |F\mathbf{q}^* + \mathbf{b}| + \mathbf{e} \tag{8.7}$$

where $\mathbf{b} = F\mathbf{s}^*$ is known a priori.

Our goal is to recover x^* (or $\mathbf{q}^*$) from the measurements in equation 8.7. So, we need to solve the following optimization

$$\underset{\mathrm{x}}{\text{minimize}} \; ||\,\mathrm{y} - |F\,\mathrm{x}|||_2^2 \quad s.t. \; \mathrm{x}_\sigma = s^*, \; \mathrm{x} \in \mathcal{M}, \; \sigma \in \mathcal{S} \tag{8.8}$$

where $\mathcal{M}$ denotes the set of constraints that we want to enforce on our estimate. We can define these constraints to ensure that the nonzero values in the estimated signal are positive and lie inside a known support [72]. We can also use some additional constraints such as the signal belongs to a union of subspaces or the range of a neural network-based generative model [188, 78].

Generator $G : \mathbb{R}^k \to \mathbb{R}^n$ maps a latent vector $\mathrm{z} \in \mathbb{R}^k$ to a high dimensional sample space $\mathrm{x} \in \mathbb{R}^n$ such that $\mathrm{x} = G_\gamma(\mathrm{z})$ where γ represents all the trainable parameters of the

deep network. To solve the phase retrieval problem in equation 8.8, we can use either trained [78] or untrained[95] generative network. In trained generative prior, we have γ fixed at some trained value and optimize the latent codes to obtain the optimal latent representation through backpropagation and gradient descent. For untrained generative prior, we fix the latent code to some random value and optimize over the generator parameters to find the best representation. One limitation with trained generative network is the unavailability of pretrained network for a target image. Whether the main problem with untrained generative prior is that finding a *good* convergence is more difficult. For trained generative prior, the optimization is as follows

$$\underset{\mathrm{z},\,\mathrm{x}}{\text{minimize}} \ \|\,\mathrm{y} - |F\,\mathrm{x}|\|_2^2 \quad s.t. \ \ \mathrm{x}_\sigma = s^*,\, \mathrm{x} = G(\,\mathrm{z}) \tag{8.9}$$

To solve the phase retrieval problem in equation 8.9 using generative prior, we use an alternating minimization framework that has three main steps: 1) update the phase of $(F\,\mathrm{x})$, 2) update the estimate via gradient descent, and 3) project estimate onto G using patchwise projection. We add one additional step to use side information. We replace the estimated points in the known part using known pixels. This approach is similar to [78] and elaborated in Algorithm 6.

The trained generators have certain range beyond which they fail to generate. It is often observed that different parts of images can be approximated using the generator well. So, we divide the signal (image) into some non overlapping patches such that most of the patches can be well approximated using the trained generator. We define this operation using $\mathcal{D}_G(.)$.

$$\mathcal{D}_G(\,\mathrm{x}) = [\,\mathrm{x}_1,\, \mathrm{x}_2, \ldots \mathrm{x}_i, \ldots] \text{ s.t. } \ \ \mathrm{x}_i = G(\,\mathrm{z}_i) \ \ \forall i,\ i \notin \sigma \tag{8.10}$$

We then project each patch onto the range of the generator using patchwise projection $\mathcal{P}_G(.)$.

$$\mathcal{P}_G(\mathrm{x}_i) = G(\mathrm{z}_i) \tag{8.11}$$

We then concatenate the the projections accordingly to form the final estimate of the projection step.

As the Fourier measurement matrix is complex valued, the estimation via gradient descent may give us complex valued estimate. However, the signal x lies in the real space. So, we impose realness constraint on the estimate before projection.

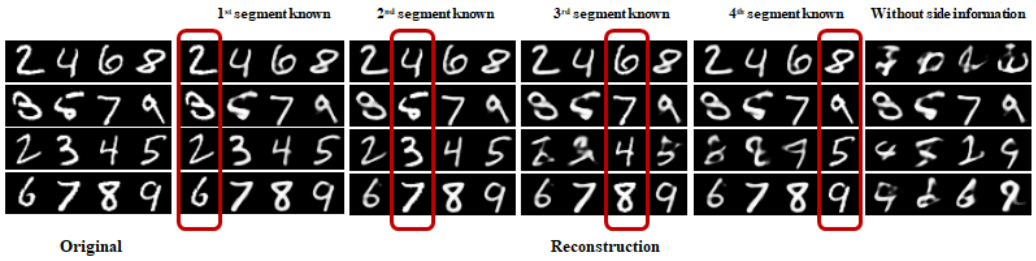


Figure 8.3: Simulation results for Fourier phase retrieval with and without side information. We observed the reconstruction when we know different parts of the image. We have concatenated 4 different MNIST digits. Here 2nd to 5th columns correspond to the reconstructions where we have prior knowledge of 1st, 2nd, 3rd and 4th digit (32×32 image patch) respectively. 6th column corresponds to the reconstruction without side information.

8.3 Experimental Setup

In this section, we discuss in details about our experimental setup including the dataset, measurements acquisition, choice of generator, optimization parameters etc.

Dataset: We have used MNIST digits and CelebA dataset for our experiments.

Algorithm 6 Generative Fourier Phase Retrieval with Side Information

Inputs: y, $\mathbf{F}$, G, T, $\mathbf{s}^*$, σ, **Output:** $\widehat{\mathrm{x}}$

Choose an initial point $\mathrm{x}_0 \in \mathbb{R}^n$

for $\mathrm{t} = 1,\dots \tau$ **do**

  Phase update: $\mathbf{p}^{t-1} \leftarrow \mathrm{sign}\left(F\,\mathrm{x}_{t-1}\right)$

  Gradient update:

  $\mathrm{u}^{t-1} \leftarrow \mathrm{x}^{t-1} + \eta F^T(\,\mathrm{y} \odot \mathbf{p}^{t-1} - F\,\mathrm{x}^{t-1})$

  Imposing realness: $\mathrm{v}^{t-1} \leftarrow \mathrm{Re}(\,\mathrm{u}^{t-1})$

  Side information: $\mathrm{v}_\sigma^{t-1} \leftarrow \mathbf{s}^*$

  Divide into patches:

  $D_G(\,\mathrm{v}^{t-1}) = [\,\mathrm{v}_1^{t-1},\, \mathrm{v}_2^{t-1}, ...,\, \mathrm{v}_i^{t-1}, ...]$

  Patchwise projection: $\mathbf{w}_i^{t-1} \leftarrow \mathcal{P}_G(\,\mathrm{v}_i^{t-1})$

  Concatenation of the projections:

  $\mathrm{x}^t \leftarrow [\mathbf{w}_1^{t-1}, \mathbf{w}_2^{t-1}, ..., \mathbf{w}_i^{t-1}, ...]$

  Side information: $\mathrm{x}_\sigma^t \leftarrow \mathbf{s}^*$

end for

$\widehat{\mathrm{x}} \leftarrow \mathrm{x}^\tau$

We have resized MNIST digits to 32×32 shape. In our experiments, we have concatenated different MNIST digits to form a larger image which demonstrates the advantage of side information more visibly. We have also created some grayscale images of some characters such as '#', '&' and '@' using photo editor and concatenated them as side information to the MNIST digits in different experiments. We resized CelebA dataset to 64×64 size. We did not crop the celebrity faces before resizing meaning that our images consists of

considerable portion of background in addition to the faces. For some experiments with celebrity images, we have created some images of name tags with the last names of the corresponding celebrities using photo editor. Even though the name tags are basically comprised of black background and white text, we used RGB version of them because CelebA images are RGB. As we are concatenating different images together, we assume that we know the region for each patch for patchwise projection.

Measurements: We take the magnitude of the 2D Fourier transform as the measurements. We did not do any oversampling (i.e. $M = N + N_s$). For RGB images (CelebA), we take three different sets of measurements for three different channels. For an image $X \in \mathbb{R}^{h \times w \times c}$, the measurement corresponding to c^{th} channel is

$$Y_c = |F_h X_c F_w| \tag{8.12}$$

where F_h and F_w are h and w point DFT matrices, respectively. In our experiments, we consider our measurements noiseless.

Generator: We use generator architecture from DCGAN[140] as our generative network except that we did not use batchnormalization. We have one trained generator on MNIST training digits (32×32) and another trained generator on CelebA dataset (64×64) using approach from [49]. The dimension of latent code for MNIST digit generator is 20 and CelebA generator is 256.

Optimization: In our reconstruction process using projected gradient descent, we optimize x and z starting with some initialization. We use zero phase initialization for x. In zero phase initialization, we assume the initial phase of the given measurement as zero and then take the real part of the inverse Fourier transform of the zero phase measurements as

initialization. The initial point, x_0 can be written as the following

$$\mathrm{x}_0 = \mathrm{Re}(F^{-1}\,\mathrm{y})$$

We randomly initialized z with samples drawn from $N(0, 1)$ distribution. We then projected the initial z on a unit norm ball. We used gradient descent to optimize x and z. In the experiments with MNIST digits, we used 0.6 as the learning rate for x update and 0.1 as the learning rate for z update in the projection step. In the experiments with CelebA dataset, we used 0.9 as the learning rate for x update and 3 as the learning rate for z update in the projection step. Furthermore, we used early stopping to halt the optimization in the earlier projection step in the experiments with CelebA. The intuition behind this is that the earlier steps of the gradient descent update does not give much meaningful estimation of x. Unlike MNIST generator which is trained on sparse MNIST digits, CelebA generator is trained on variety of images which gives it a wide range of generation capacity. If we optimize over the latent code for enough iteration in the early stages of gradient descent, it may be able to get a close projection of that initial estimates. But this will hurt the purpose of using generator which is to guide the gradient descent to a meaningful solution.

8.4 Results and Discussion

In this section, we report the reconstruction performance of different Fourier phase retrieval experiments with side information and demonstrate comparison with the reconstructions without side information. In all of our experiments, we use projected gradient descent with generative prior. The projection onto the range of the generator guides the

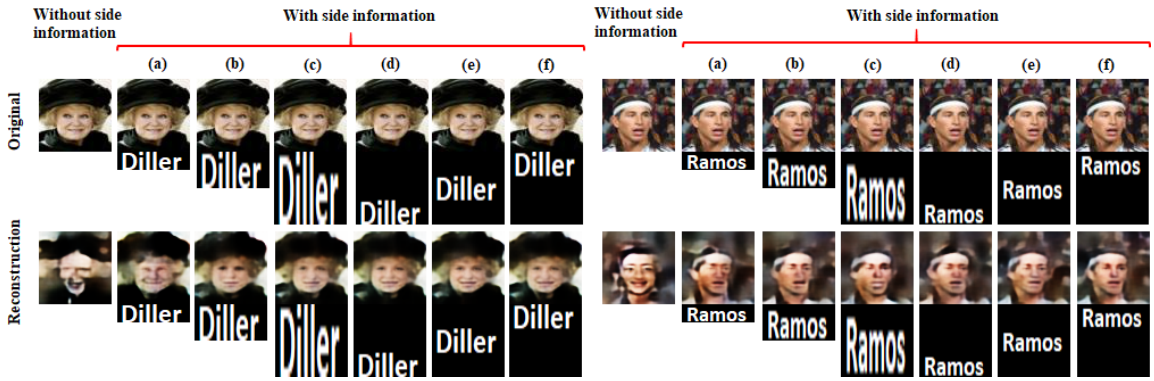


Figure 8.4: Reconstruction with known name tags as side information. We have concatenated the last names of the corresponding celebrities as side information to the original image. Fourier phase retrieval performance significantly increases with such side information. We have shown reconstruction for different size and position of the name tags. In (a), (b) and (c) name tags are of size 16×64, 32×64 and 64×64 respectively. The target celebrity image is 64×64. So the ratio of side information to the unknown part is $1:4$, $1:2$ and $1:1$ respectively. In (d), (e) and (f) we placed the text on the bottom, middle and the top of the known name tag with size 64×64.

reconstruction to some natural image output. We report two different experiments on MNIST digits and two others on CelebA images.

In our first experiment we concatenated five MNIST test digits and a known image patch of character ('#', '&' or '@'). Each digit and the known patch are of size 32×32. We also experiment with the case when we do not have any side information. We report the reconstruction result in Fig. 8.2. From equation 8.1, we know different shifted versions of the images will give us the same Fourier magnitude. So, the optimization problem with side information will have different optimal point with the same minima. So, there is high probability that the gradient descent step leads to some shifted version when we do not have any side information. However, as we are dividing the estimate into fixed patches and

projecting onto the range of the generator, the generator will try to approximate each patch with a digit as closely as possible. This will give us some reconstructions resembling some digits but they will not necessarily be the target digit. We observe this phenomena in the reconstruction without side information. However, when we have some side information, we can rule out the trivial ambiguities which leads to a good solution. we can observe from Fig. 8.2 that different characters as side information gave us very good reconstructions.

In our second set of experiments with MNIST digits, we concatenate four different MNIST test digits. We then fix one of the position and we assume that we know the digit in that position. We change the position of the known patch in the image and observe

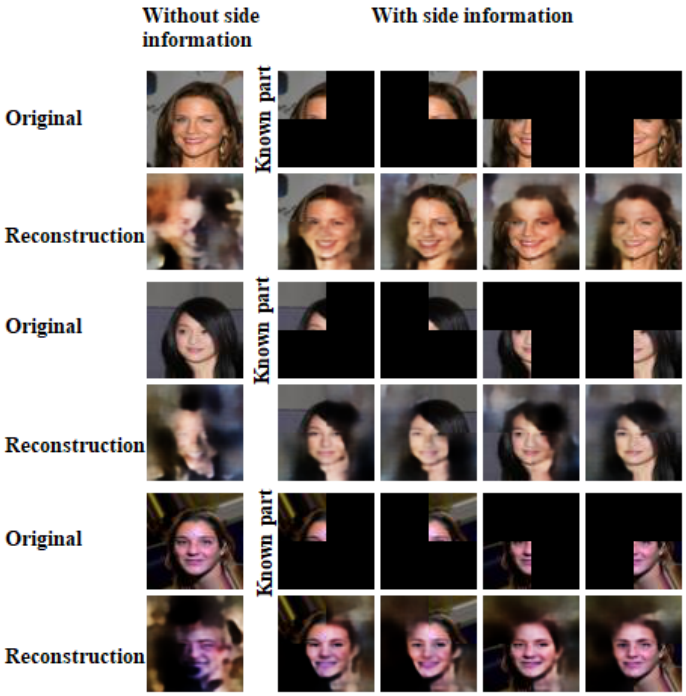


Figure 8.5: Reconstruction when some part of the image is known. We have shown cases when top-left, top-right, bottom-left, bottom-right 32×32 patches of 64×64 celebrity images are known. We have also shown comparison with the reconstruction without any side information.

reconstruction performance. We demonstrate the reconstruction result for different set of MNIST digits in Fig. 8.3. We also show comparison with the reconstruction without side information. From Fig. 8.3, we can observe that knowing a part of the image always leads to a better solution than the case when we do not have any knowledge about the image. Most of the reconstructions without side information lead to reconstructions of wrong digits. However, even without side information, one reconstruction (second row) reached a good solution which is rare but not improbable.

In our setup with RGB Fourier phase retrieval, we encounter additional ambiguities as the each channels now can be shifted independently without changing the Fourier magnitude measurements. This ambiguity can lead to some visually meaningless solution to be equally *good* as the original one. In our experiments with RGB CelebA images we can observe such ambiguities much often. Furthermore, as we discussed before, the CelebA generator can generate wide range of images which prevents generative network from providing similar aid as MNIST digit generator in guiding the gradient descent to some meaningful output. We mitigated this phenomena using early stopping in projection step after the early gradient descent steps.

We report two different set of experiments with CelebA. In our first set of experiments with CelebA, we assume that we know some part of that image. In Fig. 8.5, we demonstrate the cases when we know top-left, top-right, bottom-left, bottom-right 32×32 patches of the images. In this case as well, we show reconstruction without any known side information. As we can observe, without side information the Fourier phase retrieval did not give us any good reconstruction for color images. Furthermore, it faced some aforementioned

ambiguities and lead to some reconstructions which visually does not resemble any face. However, with side information, we could avoid those ambiguities and reach to some good reconstructions. We can also observe that the reconstructions are comparable for different position of the known patch.

In our second experiment with CelebA images, we have concatenated an image of text description at the bottom of a CelebA image. As we used the last name of the corresponding celebrity as the text description, we can think of it as a name tag. We assume that we know the name tag as the side information. In Fig. 8.4, we show reconstruction for different size of name tags and different position of text in the name tag. We also showed reconstruction without side information for comparison. Like before, Fourier phase retrieval without side information faced the aforementioned ambiguities and resulted in *bad* reconstruction. We experimented with three different size of name tags: 16×64, 32×64 and 64×64. We can observe that the reconstruction performance improves with the increasing size of the known name tag as the known and unknown part ratio also increases (1:4,1:2 and 1:1). We also changed the position of text (bottom, middle and top) in a 64×64 name tag. Changing the position of the name tag apparently does not have much effect on the reconstructions.

Chapter 9

Unrolling Network to Learn Reference for Phase Retrieval

.

9.1 Introduction

The problem of *phase retrieval* refers to the challenge of recovering a real- or complex-valued signal from its amplitude measurements. This problem arises in diffraction imaging, X-ray crystallography, and ptychography [72, 75, 189, 74, 70]. Fourier phase retrieval is a special class of phase retrieval problems aimed at the recovery of a signal from the amplitude of its Fourier coefficients. Let us assume that Fourier amplitude measurements are given as

$$y = |Fx| + \eta, \tag{9.1}$$

This work has been published in European Conference on Computer Vision 2020[110]

where F denotes the Fourier transform operator, x denotes the unknown signal or image, and η denotes the measurement noise. Our goal is to recover x given y.

Fourier phase retrieval is essential in many applications, especially in optical coherent imaging. Classical methods for phase retrieval utilize the prior knowledge about the support and positivity of the signals [72, 75]. Subsequent work has considered the case where the unknown signal is *structured* and belongs to a low-dimensional manifold that is known *a priori*. Examples of such low-dimensional structures include sparsity [82, 93], low-rank [106, 190], or neural generative models [191, 95]. Other techniques like Amplitude flow [192] and Wirtinger flow use alternating minimization [84]. Many of these newer algorithms involve solving a *non-convex* problem using iterative, gradient-based methods; therefore, they need to be carefully initialized. The initialization technique of choice is spectral initialization, first proposed in the context of phase retrieval in [92], and extended to the sparse signal case in [82, 93].

Fourier phase retrieval problem does not satisfy the assumptions needed for successful spectral initialization and remains highly sensitive to the initialization choice. Furthermore, Fourier amplitude measurements have the so-called trivial ambiguities about possible shifts and flips of the images. Therefore, many Fourier phase retrieval methods test a number of random initializations with all possible flips and shifts and select the estimate with the best recovery error [107].

In this work [110], we assume that a known (learned) reference is added to the signal before capturing the Fourier amplitude measurements. The main motivation for this comes from the empirical observation that knowing a part of the image can often help

resolve the trivial ambiguities [99, 101, 102]. We extend this concept and assume that a known reference signal is added to the target signal and aim to recover the target signal from the Fourier amplitude of the combined signal. Adding a reference may not feasible in all cases, but our method will be applicable whenever we can add a reference or split the target signal into known and unknown parts. We can describe the Fourier amplitude (phaseless) measurements with a known reference signal u as

$$y = |F(x + u)| + \eta. \tag{9.2}$$

Similar reference-based measurements and phase retrieval problems also arise in holographic optical coherence imaging [193].

Our goal is to recover the signal x from the amplitude measurements in (9.2). To do that, we implement a gradient descent method for phase retrieval. We present the algorithm as an unrolled network for a general system in Fig. 9.1. Every layer of the network implements one step of the gradient descent update. To minimize the computational complexity of the recovery algorithm, we seek to minimize the number of iterations (hence the layers in the network). In addition, we seek to learn the reference u to maximize the accuracy of the recovered signal for a given number of iterations. The learned u and reconstruction results for different datasets are summarized in Fig. 9.2.

We present an iterative method to efficiently recover a signal from the Fourier amplitude measurements using a fixed number of iterations. To achieve this goal, we first learn a reference signal that can be added to the phaseless Fourier measurements to enable the exact solution of the phase retrieval problem. We demonstrate that the reference learned on a very small training set perform remarkably well on the test dataset.

Our main contributions can be summarized as follows.

- The proposed method uses a fixed number of gradient descent iterations (i.e., fixed computational cost) to solve the Fourier phase retrieval problem.
- We formulate the gradient descent method as an unrolled network that allows us to learn a robust reference signal for a class of images. We demonstrate that reference learned on a very small dataset performs remarkably well on diverse and large test datasets. To the best of our knowledge, this is the first work on learning a reference for phase retrieval problems.
- We tested our method extensively on different challenging datasets and demonstrated the superiority of our method.
- We demonstrate the robustness of our approach by testing it with the noisy measurements using the reference that was trained on noise-free measurements.

In this work, we consider the reference signal to be additive and overlapping with the target signal. To the best of our knowledge, there has not been any study on such unrestricted reference design. While driven by data, our approach for reference design uses training samples in a very efficient way. The number of training images required by our network is parsimonious without limiting its generalizability. The reference learned by our network provides robust recovery test images with different sizes. Apart from the great flexibility, our unrolled network uses a well-defined routine in each layer and demonstrates excellent interpretability as opposed to black-box deep neural networks.

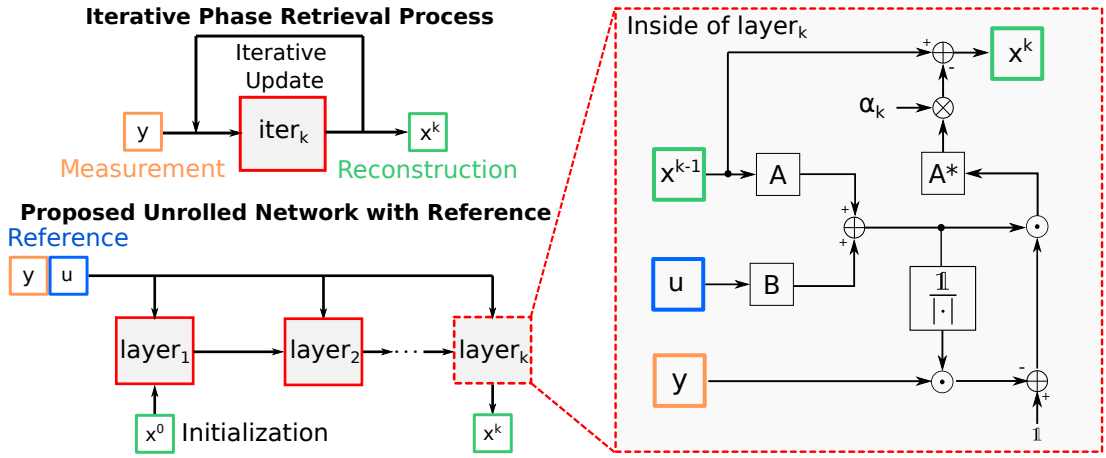


Figure 9.1: Our proposed approach for learning reference signal by solving phase retrieval using an unrolled network. Unrolled network has K layers. Each layer_k gets amplitude measurements y, reference u, and estimate x^{k-1} as inputs, and updates the estimate to x^k. The operations inside layer_k are shown in the dashed box on the right, where A and B are both linear measurement operators, and A^* is the adjoint operator of A.

9.2 Proposed Approach

We use the general formulation for the phase retrieval from amplitude measurements. The formulation can be extended for phase retrieval with squared amplitude measurement as well. In our setup, we model amplitude measurements of a target signal x and a reference signal u as $y = |Ax + Bu|$, where A and B are linear measurement operators. Our goal is to learn a reference signal that provides us the best recovery of the target signal. We formulate this overall task as the following optimization problem:

$$\underset{\hat{x}(u)}{\text{minimize}} \; \|x - \hat{x}(u)\|_2^2 \quad \text{s.t.} \;\; y = |A\hat{x}(u) + Bu|, \tag{9.3}$$

where $\hat{x}(u)$ denotes the solution of the phase retrieval problem for a given reference u. Our approach to learn u and solve (9.3) can be divided into two nested steps: (1) Outer step updates u to minimize the recovery error for phase retrieval and (2) inner step uses the learned u to recover target images by solving phase retrieval.

To solve the (inner step) of phase retrieval problem, we use an unrolled network. Figure 9.1 depicts the structure of our phase retrieval algorithm. In the unrolled phase retrieval network, we have K blocks to represent K iterations of the phase retrieval algorithm. We minimize the following loss to solve the phase retrieval problem:

$$L_x(x, u) = \|y - |Ax + Bu|\|_2^2. \tag{9.4}$$

Every block of the unrolled phase retrieval network is equivalent to one gradient descent step for (9.4). For some value of reference estimate, u, we can represent the target signal estimate after $k + 1^{th}$ block of the unrolled network as

$$x^{k+1} = x^k - \alpha_k \nabla_x L_x(x^k, u), \tag{9.5}$$

where $\nabla_x L_x(x^k, u)$ is the gradient of L_x with respect to x at the given values of x^k, u. As the loss function in (9.4) is not differentiable, we can redefine it as

$$L_x(x, u) = \|y \odot p - (Ax + Bu)\|_2^2, \tag{9.6}$$

where $p = \angle(Ax^k + Bu) = (Ax^k + Bu)/|Ax^k + Bu|$. The expression of gradient can be written as

$$\nabla_x L_x(x^k, u) = 2A^*[p \odot (p^* \odot (Ax^k + Bu) - y)], \tag{9.7}$$

where A^* denotes the adjoint of A. After K blocks, we get the estimate of the target signal that we denote as $\hat{x}(u) = x^K$.

In the learning phase, we are given a set of training signals, $\{x_1, x_2, ..., x_N\}$, which share the same distribution as our target signals. We initialize x^0 and u^0 with some initial (feasible) values. First we minimize the following loss with respect to u:

$$L_u(u) = \sum_{i=1}^{N} \|x_i - \hat{x}_i\|_2^2 = \sum_{i=1}^{N} \|x_i - x_i^K\|_2^2. \tag{9.8}$$

We can rewrite (9.8) using the gradient recursion in (9.5) as

$$L_u(u) = \sum_{i=1}^{N} \|x_i - x_i^0 + \sum_{k=0}^{K-1} \alpha_k \nabla_x L_x(x_i^k, u)\|_2^2. \tag{9.9}$$

We can then use gradient descent to to minimize $L_u(u)$. We can represent the $j + 1^{th}$ iteration of gradient descent step as

$$u^{j+1} = u^j - \beta \nabla_u L_u(u^j). \tag{9.10}$$

The expression for $\nabla_u L_u(u)$ can be written as

$$\nabla_u L_u(u) = 2 \sum_{i=1}^{N} \left[\sum_{k=0}^{K-1} \alpha_k J_u(x_i^k, u) \right] \left[x_i - x_i^0 + \sum_{k=0}^{K-1} \alpha_k \nabla_x L_x(x_i^k, u) \right], \tag{9.11}$$

where $J_u(x_i^k, u) = \nabla_u \nabla_x L_x(x_i^k, u)$ is a Jacobian matrix with rows and columns of the same size as u and x, respectively. The measurement vector $y = |Ax + Bu|$ is a function of u during training. Since we model $\hat{x}(u)$ as an unrolled network, we can think of the gradient step as a backpropagation step. To compute $\nabla_u L_u(u)$, we backpropagate through the entire unrolled network. At the end of J^{th} outer iteration, we will get our learned reference $\hat{u} = u^J$.

Once we have learned a reference, $\hat{u}$, we can use it to capture (phaseless) amplitude measurements as $y = |Ax^* + B\hat{u}|$ for target signal x^*. To solve the phase retrieval problem, we perform one forward pass through the unrolled network. Pseudocodes for training and testing are provided in Algorithms 7,8.

Algorithm 7 Learning Reference Signal

Input: Training signals $\{x_1, x_2, ..., x_N\}$, measurement operators, A and B

Initialize $\{x_1^0, x_2^0, ..., x_N^0\}, u^0$.

for $j = 0, 1, ..., J-1$ **do**

  for $i = 1, 2, ..., N$ **do**

    $y_i = |Ax_i^* + Bu^j|$

    for $k = 0, 1, ..., K-1$ **do**

      $L_x(x_i^k, u^j) = \|y_i - |Ax_i^k + Bu^j|\|_2^2$

      $x_i^{k+1} \leftarrow x_i^k - \alpha_k \nabla_x L_x(x_i^k, u^j)$

    end for

  end for

  $L_u(u^j) = \sum_{i=1}^N \|x_i^* - x_i^0 + \sum_{k=1}^K \alpha_k \nabla_x L_x(x_i^{k-1}, u^j)\|_2^2$

  $u^{j+1} \leftarrow u^j - \beta \nabla_u L_u(u^j)$

end for

Output: Optimal reference, $\hat{u} = u^J$

In our Fourier phase retrieval experiments $A = B = F$, where F is the Fourier transform operation. To implement similar method for squared amplitude measurements, we can simply replace $p = \angle(Ax^k + Bu^j)$ with $p = Ax^k + Bu^j$. In all our experiments, we initialized x^0 as a zero vector whenever $\hat{u} \neq 0$. We can also add additional constraints on the reference while minimizing the loss function in (9.9). In our experiments, we used target signals with intensity values in the range $[0, 1]$; therefore, we restricted the range of entries in u to $[0, 1]$ as well. We discuss other constraints in the experiment section.

Algorithm 8 Solving Phase Retrieval via Unrolled Network

Input: Measurements y, learned reference $\hat{u}$, measurement operators, A and B

Initialize x^0.

for $k = 0, 1, ..., K-1$ **do**

$L_x(x^k, \hat{u}) = \|y - |Ax^k + B\hat{u}|\|_2^2$

$x^{k+1} \leftarrow x^k - \alpha_k \nabla_x L_x(x^k, \hat{u})$

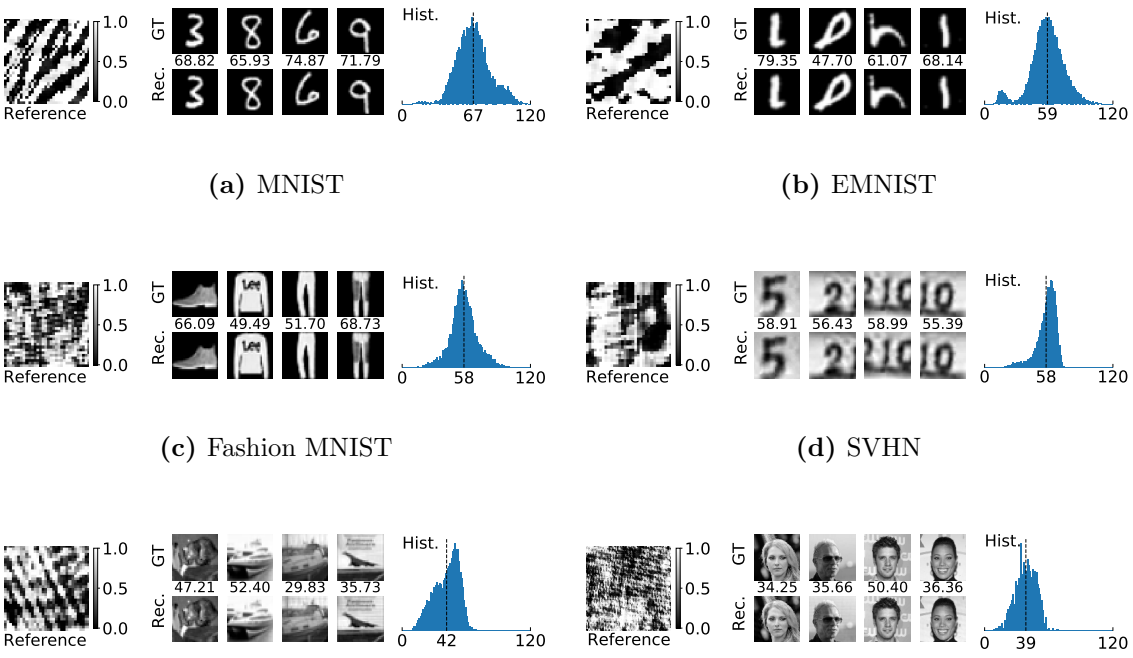
end for

Output: Estimation of target signal $\hat{x} = x^K$

9.3 Experiments

Datasets. We have used MNIST digits, EMNIST letters, Fashion MNIST, CIFAR10, SVHN, CelebA datasets, and different well-known standard images for our experiments. We convert all images to grayscale and resize 28×28 images to 32×32. Although there are tens of thousands training images in MNIST, EMNIST letters, Fashion MNIST, CIFAR10, and SVHN dataset, we have used only a few (i.e.. 32) of them in training. We have shown that the references learned on the small number of training images perform remarkably well on the entire test dataset. MNIST, Fashion MNIST, and CIFAR10 test datasets contain 10000 test images each; EMNIST letters dataset contains 24800 test images; SVHN test dataset contains 26032 test images. We used 1032 images from CelebA and center-cropped and resized all of them to 200×200. We selected 32 images for training and the rest for testing.

We present the results for these different datasets using references learned from 32



(a) MNIST **(b)** EMNIST

(c) Fashion MNIST **(d)** SVHN

(e) CIFAR10 **(f)** CelebA

Figure 9.2: Reconstruction results using learned references. Each block **(a)-(f)** shows results for different dataset: (**left**) learned reference with a colorbar; (**middle**) sample original images and reconstruction with PSNR on top; (**right**) histogram of PSNR over the entire test dataset (vertical dashed line represents the mean PSNR).

images from the same dataset in Fig. 9.2. We present results for six standard images of size 512×512 from [107] using a resized reference learned from CelebA dataset in Fig. 9.3.

Measurements. We simulated amplitude measurements of the 2D Fourier transform. We performed 4 times oversampling in the spatial domain for both reference and target signal. Unless otherwise mentioned, we consider our measurements to be noise-free. We also report results for noisy measurements.

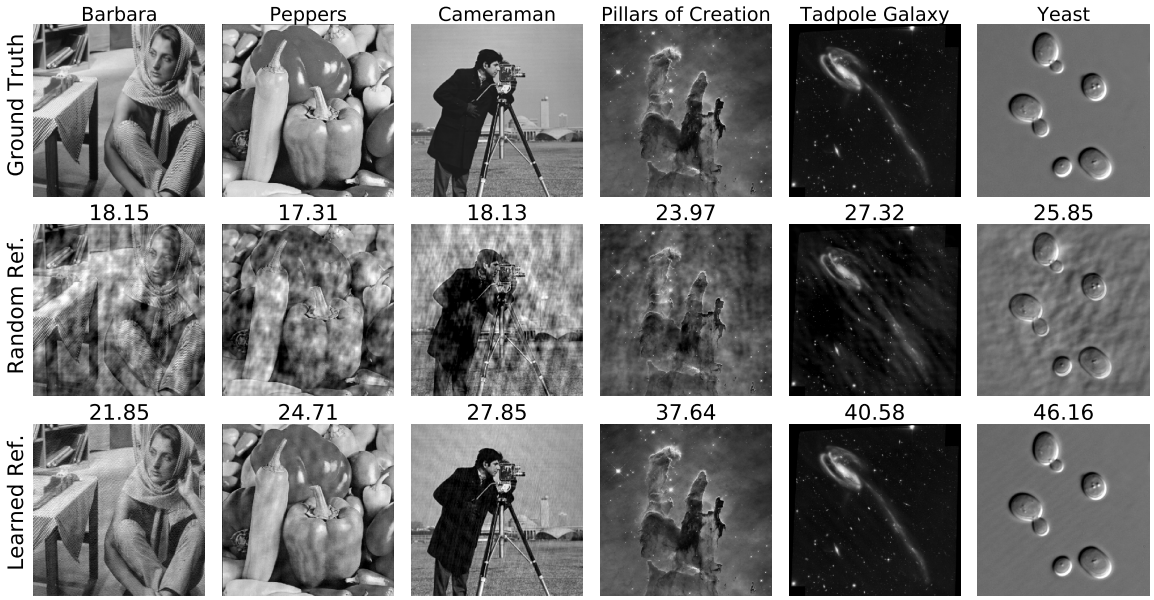


Figure 9.3: Phase retrieval results using learned and random references. **First Row:** Original 512×512 test images. **Second Row:** Reconstruction using random references with uniform distribution between $[0, 1]$ **best result out of 100 trials**. **Third Row:** Reconstruction using the reference learned on CelebA dataset and resized from 200×200 to 512×512. (PSNR shown on top of images).

9.3.1 Configurations of Reference (u)

The reference signal u, which we are trying to learn, has a number of hyper-parameters that inherently affect the performance of the phase retrieval process. We considered several constraints on u, including the support, size, range, position, and sparsity.

We tested reference signals with both complex and real values and found that u has comparable results in the two domains. Since it is easy to physically create amplitude or phase-only reference signals, we constrain u to be in the real domain; thus, $u \in \mathbb{R}^{m \times n}$

and m, n represent height and width, respectively. The height and width of u determine the overlapping area between the target signal and the reference. We found that u with larger size tends to have better performance, especially when the value of u is constrained to a small range. The intensity values of u play a major role in its performance. If we constrain the value of u to be within a certain range: $u[i,j] \in [u_{min}, u_{max}]$, for all i, j, we observed that bigger range of u yields better performance. This is because when u is unconstrained then we can construct a u with a large norm. Consider the noiseless setting with quadratic measurements $|F(x+u)|^2 = |Fx|^2 + |Fu|^2 + 2\text{Re}(Fx \odot Fu)$, the last term is the real value of the element-wise product of target and reference Fourier transforms. We can remove $|Fu|^2$ because it is known. If u is large compared to x, then we can also ignore the quadratic term $|Fx|^2$ and recover x in a single iteration if all entries of Fu are nonzero. To avoid this situation and make the problem stable in the presence of noise, we restricted the values in the reference u to be in [0,1] range.

9.3.2 Setup of Training Samples and Sample Size

We observed that we can learn the reference signal from a small number of training images. In Table 9.1, we report test results for different reference signals learned on first N images from MNIST training dataset for $N = 32, 128, 512$. We kept the signal and reference strength (i.e., the range of the signal) equal for this experiment. We observe that increasing the training size improves test performance. However, we can get reasonable reconstruction performance on large test datasets (10k+ images) with reference learned using only 32 images.

Table 9.1: PSNR for different training size

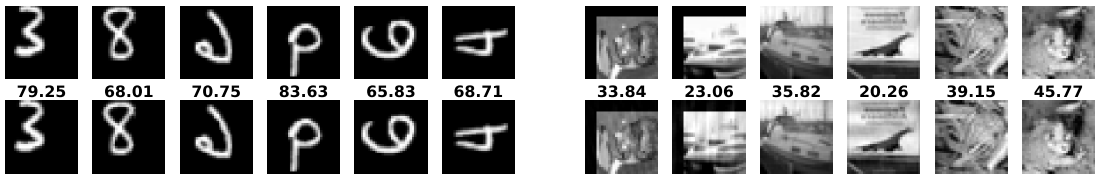
Train/Test	MNIST	EMNIST	F. MNIST	SVHN	CIFAR10
Training size=32	66.54	58.72	57.81	57.51	41.60
Training size=128	76.25	64.16	55.86	59.50	44.34
Training size=512	79.14	62.34	52.01	59.78	48.90

9.3.3 Generalization of Reference on Different Classes

We are interested in evaluating the generalization of our learned reference. (i.e., how the reference performs when trained on one dataset and tested on another). In the comparison study, we took the reference u trained on each dataset and then tested them on the remaining 4 datasets. The value range of the reference is between $[0, 1]$, the number of steps in the unrolled network is $K = 50$. We observed that when the datasets share great similarity (e.g., MNIST and EMNIST are both sparse digits or letters), the reference signal tends to work well on both datasets. Even when the datasets differ greatly in their distributions, the reference trained on one dataset provides good results on other datasets (with only a few dB of PSNR decrease in performance).

We also tested our method on shifted and rotated versions of test images. Results in Fig. 9.4 demonstrate that even though the reference was trained on upright and centered images, we can perfectly recover shifted and rotated images.

Our key insight about this generalization phenomenon is that the main challenge in Fourier phase retrieval methods is initialization and ambiguities that arise because of symmetries. We are able to solve these issues using a learned reference because of the following reasons: (1) A reference gives us a good initialization for the phase retrieval



(a) MNIST **(b)** CIFAR10

Figure 9.4: Test results on shifted/flipped/rotated images using the reference learned on upright-centered (canonical) images. PSNR shown on top of images.

Table 9.2: PSNR of the Same Reference Tested on Different Datasets

Train/Test	MNIST	EMNIST	F. MNIST	SVHN	CIFAR10
MNIST	66.54	55.12	40.87	41.87	31.72
EMNIST	**72.84**	**58.72**	52.18	55.42	48.16
F. MNIST	40.87	55.67	**57.81**	50.70	42.85
SVHN	41.87	46.76	49.60	**57.51**	**51.54**
CIFAR10	31.72	38.93	36.40	40.36	41.60

iterations. (2) The presence of a reference breaks the symmetries that arise in Fourier amplitude measurements. Moreover, we are not learning to solve the phase retrieval problem in an end-to-end manner or learn a signal-dependent denoiser to solve the inverse problem [107, 109]. We are learning reference signals to primarily help a predefined phase retrieval algorithm to recover the true signal from the phaseless measurements. Thus, the references learned on one class of images provide good results on other images, see Table 9.2. This study shows that the reference learned using our network has the ability to generalize to new datasets, thus making our method suitable for real-life applications where new test cases keep emerging.

9.3.4 Noise Response

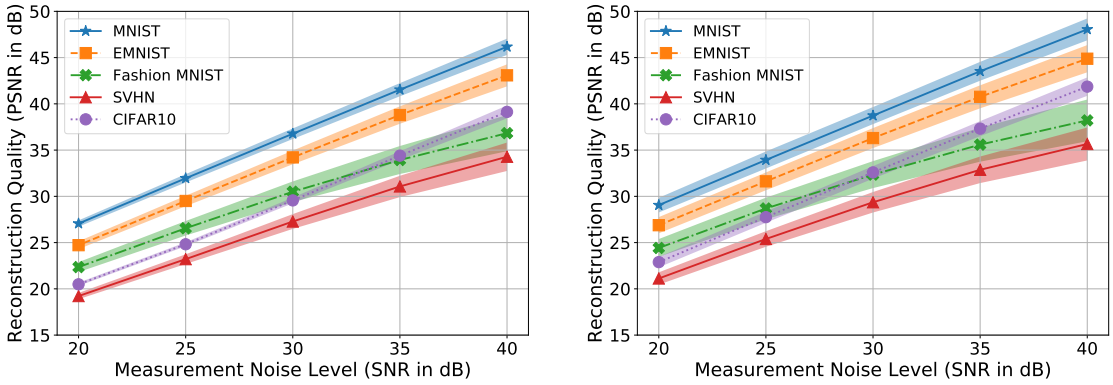
To test the robustness of our method in the presence of noise, we added Gaussian and Poisson noise at different levels to the measurements. Poisson noise or shot noise is the most common in the practical systems. We model the Poisson noise following the same approach as in [107]. We simulate the measurements as

$$y(i) = |z(i)| + \eta(i) \quad \text{for all } i = 1, 2, \ldots, m, \tag{9.12}$$

where $\eta(i) \sim \mathcal{N}(0, \sigma^2)$ for Gaussian noise and $\eta(i) \sim \mathcal{N}(0, \lambda|z(i)|)$ for Poisson noise with $z = Ax + Bu$. We varied σ, λ to generate noise at different signal-to-noise ratios. Poisson noise affects the larger measurements with higher strength than the smaller measurements. As the sensors can measure only positive measurements, we kept the measurements positive by applying ReLU function after noise addition. We can observe the effect of noise in Fig. 9.5. Even though we did not add noise during training, we get reasonable reconstruction and performance degrades gracefully with increased noise.

9.3.5 Random Reference versus Learned Reference

To demonstrate the advantage of the learned reference signal, we compared the performance of learned reference and random reference on some standard images. The results are shown in Fig. 9.3. The learned reference is trained using 32 images from CelebA dataset which we resized to 200×200. The test images used in Fig. 9.3 are 512×512, so we resized the learned reference from 200×200 to 512×512. For random reference, we selected the entries of the reference uniformly at random from $[0, 1]$. We selected the best result out of 100 trials for every test image with random reference. We can observe from



(a) Gaussian **(b)** Poisson

Figure 9.5: Reconstruction quality of the test images vs noise level of the measurements for different datasets. We learned the reference using noise-free measurements.

the results that our learned reference significantly outperforms the random reference even though the test image distribution is distinct from the training data. The number of steps of the unrolled network is $K = 50$.

9.3.6 Comparison with Existing Phase Retrieval Methods

We have shown comparison with other approaches in Table 9.3. We selected Kaczmarz [194] and Amplitude flow [86] for comparison using PhasePack package [104]. We also show Hybrid Input Output (HIO), which is similar to our phase retrieval routine without any reference. We observe that our approach with learned reference can outperform all other approaches on all the datasets. All the traditional phase retrieval methods suffer from the trivial circular shift, rotation, and flip ambiguities, thus produce significantly worse reconstruction than our method does. Our method uses a reference signal to simplify the initialization and removes the shift/reflect ambiguities. To mathematically explain this

Table 9.3: Comparison with Existing Phase Retrieval Methods

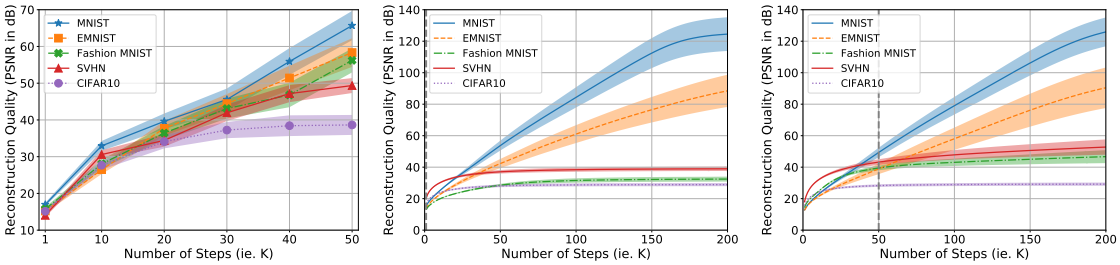
Methods	MNIST	EMNIST	F. MNIST	SVHN	CIFAR10
HIO	9.04	8.42	9.65	19.87	14.70
Amplitude Flow	9.99	9.79	11.90	20.25	15.04
Kaczmarz	11.81	11.47	13.44	19.48	15.01
Flat Reference	18.21	17.24	16.56	20.89	15.81
Random Reference	36.87	28.41	27.27	36.45	25.57
Learned Reference (Ours)	**66.54**	**58.72**	**57.81**	**57.51**	**41.60**

fact, a shifted or flipped version of x would not give us the same Fourier measurements as $|F(x+u)|$ if u is chosen appropriately as we do with the learning procedure. As we showed in Fig. 9.5, our method can perfectly recover the shifted and flipped versions of the images using the reference that was trained with upright and centered images.

9.3.7 Effects of Number of Layers (K)

We tested our unrolled network with different numbers of layers (i.e., K) at training and test time. The results are summarized in Fig. 9.6. We first used the same values of K for training and testing. We observed that as K increases, the reconstruction quality (measured in PSNR) improves. Then we fixed $K = 1$ or $K = 10$ at training, but used different values of K at testing. We observed that if we increase K at the test time, PSNR improves up to a certain level and then it plateaus. The PSNR achieved with reference trained with $K = 10$ is better than what the referenced trained with $K = 1$ provided. These results provide us a trade-off between the reconstruction speed and quality. As we increase K, the reconstruction quality improves but the reconstruction requires more steps (computations and time).

Finally, we learned a reference using $K = 1$ and tested it on different images with



(a) Training K=Testing K **(b)** Training K=1 **(c)** Training K=50

Figure 9.6: Reconstruction PSNR vs the number of blocks (K) in the unrolled network at training and testing. (a) K is same for training and testing (shaded region shows ± 0.25 times `std` of PSNR). (b) $K = 1$ and (c) $K = 10$, but tested using different K.

$K = 1$. To our surprise, our method was able to produce reasonable quality reconstruction with this extreme setting. We present some single-step reconstructions of each data set in Fig. 9.7.

9.3.8 Localizing the Reference

We also evaluated the effect of localizing the reference to a small region. For example, the reference is constrained to be within a small block in the corner or the center of the target signal. We restricted u to be an 8×8 block and placed it in different positions. We found that corner positions provide better results as shown in Fig. 9.8. As we bring the reference support closer to the center, the quality of reconstruction deteriorates. This observation is related to the method in [99, 101, 103], where if the known reference signal is separated from the target signal, then the phase retrieval problem can be solved as a linear inverse problem.

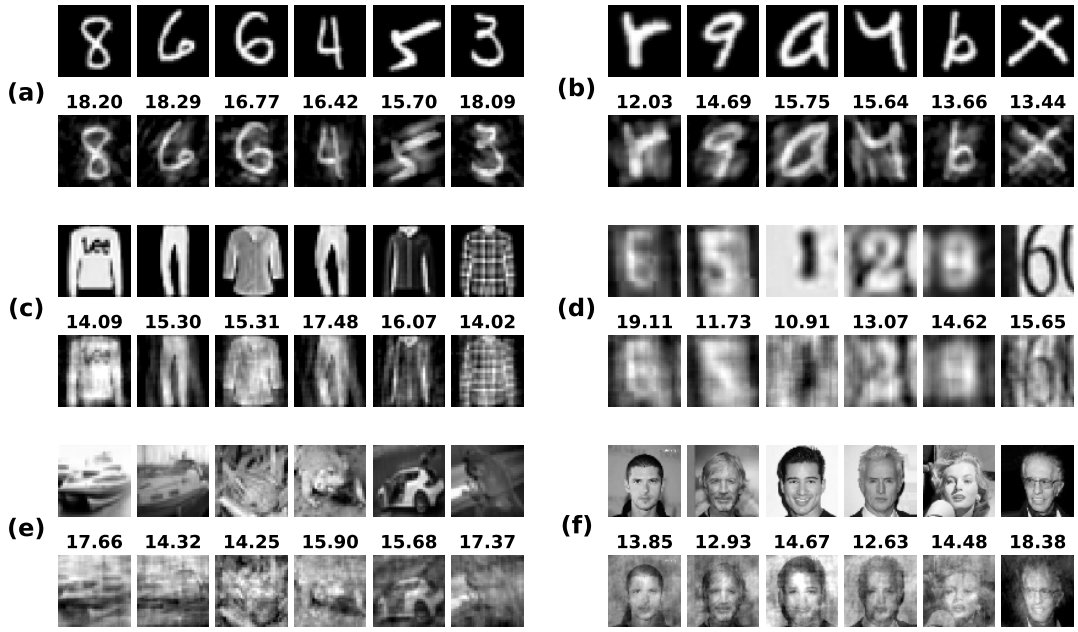
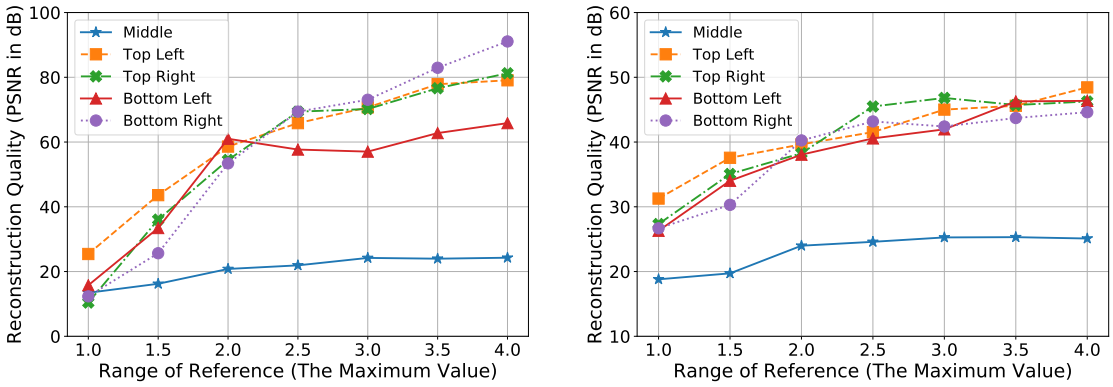


Figure 9.7: Single step reconstruction with reference in range $[0, 1]$. Each of the **6 sets (a)-(f)** has the the ground truth in the first row. Second row is the reconstruction (PSNR values on top).

Note that signal recovery from Fourier phase retrieval is equivalent to signal recovery from its autocorrelation. We can write the autocorrelation of target plus reference signals as $(x+u)\star(x+u) = x\star x + u\star u + x\star u + u\star x$. The first term is a quadratic function of x, the second term is known, and the last two terms are linear functions of x. If the supports for x and u are sufficiently separated, then we can separate the last two linear terms from the first two quadratic terms and recover x by solving a linear problem. However, if x and u have a significant overlap, then we need to solve a nonlinear inverse problem as we do in this work.



(a) MNIST **(b)** CIFAR10

Figure 9.8: Performance of our method if the reference is an 8×8 block placed at different positions. Fixing the minimum value at 0, we increased the maximum value of the reference we learn. We observe that the small reference placed in the corners performs better than the ones placed in the center.

Chapter 10

Unrolling Network to Learn Coded Illumination Patterns

10.1 Introduction

The problem of signal recovery from nonlinear measurements arises in various imaging and signal processing tasks [70, 71, 74, 73, 81]. Conventional methods for solving such inverse problems use an iterative method to recover the signal from given measurements. In this chapter, we present a framework to optimize over the measurement parameters to improve the quality of signals recovered by the given iterative method. In particular, we learn illumination patterns to recover the signal from coded diffraction patterns (CDP) using a fixed-cost alternating minimization method.

Coded diffraction imaging is a specific instance of Fourier phase retrieval problems. Phase retrieval refers to a broad class of nonlinear inverse problems where we seek to

Part of this work has been accepted to IEEE International Conference on Image Processing 2021 [195].

recover a complex- (or real-) valued signal from its phase-less (or sign-less) measurements [196, 79, 72, 197, 70, 198]. In practice, these problems often arise in coherent optical imaging where an image sensor records the intensity of the Fourier measurements of the object of interest. In coded diffraction imaging, the signal of interest gets modulated by a sequence of known illumination patterns/masks before observing the Fourier intensity at the sensor [79, 197, 70]. Applications include X-ray crystallography [199, 74, 189], astronomy [200, 201], microscopy [202, 105, 73, 203], speech processing and acoustics [204, 205, 206], and quantum mechanics [207, 208].

We can model the sensor measurements for coded diffraction imaging as follows. Let us denote the signal of interest as $x \in \mathbb{R}^n$ or $\mathbb{C}^n$ that is modulated by T illumination patterns $D = \{d_1, \ldots, d_T\}$, where $d_t \in \mathbb{R}^n$ or $\mathbb{C}^n$. The amplitude of sensor measurements for t^{th} illumination pattern can be written as

$$y_t = |\mathcal{F}(d_t \odot x)|, \tag{10.1}$$

where $\mathcal{F}$ denotes the Fourier transform operator and $\odot$ denotes an element-wise product. We note that real sensor measurements are proportional to the intensity of the incoming signal (i.e., square of the Fourier transform). In practice, however, solving the inverse problem with (non-square) amplitude measurements provides better results [209, 107]; therefore, we use the amplitude measurements throughout the chapter.

To recover the signal x from the the observed measurements, we can solve the following optimization problem:

$$\min_x \sum_{t=1}^{T} \|y_t - |\mathcal{F}(d_t \odot x)|\|_2^2. \tag{10.2}$$

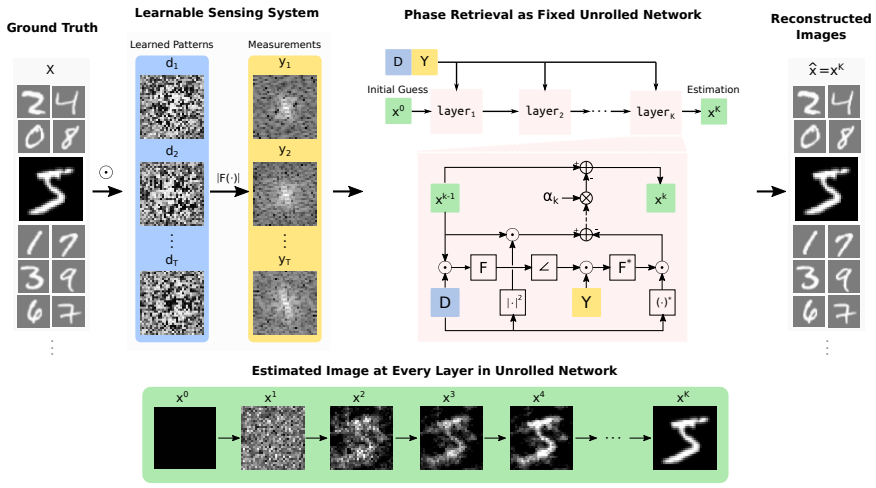


Figure 10.1: Pipeline of our proposed framework at inference time. Our framework mainly contains two components: (*1*) a learnable sensing system that updates the illumination patterns during training time, but at inference time the learned illumination patterns are fixed; (*2*) a fixed unrolled network that runs phase retrieval process to recover the original signal x form measurements Y. The number of layers in the network is fixed to K. Steps at every iteration are fixed and depicted as an unrolled network (details can be found in Algorithm 10). We illustrate the steps of k^{th} layer of the unrolling network. Phase retrieval algorithm uses the measurements $Y = \{y_t\}$ and illumination patterns $D = \{d_t\}$ to provide an estimate x^K after K iterations. During training time, our goal is to learn the illumination patterns D to minimize the error between the estimated x^K and the ground truth. More details can be found in section 10.2.

In recent years, a number of iterative algorithms have been proposed for solving the problem in (10.2), which includes lifting-based convex methods, alternating minimization-based nonconvex methods, and greedy methods [79, 80, 92, 78, 93].

Our goal is to learn a set of illumination patterns to optimize the recovery of an alternating minimization (AltMin) algorithm for solving the problem in (10.2). The AltMin method can be viewed as an unrolled gradient descent network, as shown in Fig. 10.1,

where we fix the steps at every iteration and the total number of iterations for AltMin. One forward pass through the unrolled network is equivalent to K iterations of the AltMin algorithm. We can increase or decrease the number of iterations for better accuracy or faster run-time. To keep the computational complexity of the recovery algorithm low, we keep the total number of iterations small (e.g., $K = 50$). At the training stage, we optimize over the illumination patterns to minimize the error between the AltMin outputs after K iterations and the ground truth training images. At the test time, we solve the problem in (10.2) using K AltMin iterations with the learned illumination patterns (equivalent to one forward pass). We evaluated our method on different image datasets and compared against existing methods for coded diffraction imaging. We demonstrate that our proposed method of designing illumination patterns for a fixed-cost algorithm outperforms existing methods both in terms of accuracy and speed.

The key contributions of this work [195] are as follows.

- We learned illumination patterns for coded diffraction imaging using unrolled network formulation of a classical AltMin method.
- We showed that with our designed patterns and unrolled AltMin method outperform computationally complex algorithms and provide superior image reconstruction.
- Our algorithm requires only a small number of training images to learn the illumination patterns. It is crucial in applications because finding training samples is difficult in practice.
- The patterns learned on a given dataset generalize to different datasets and provide robust reconstruction for shifted and flipped versions of the target samples.

- Our learned illumination patterns can also help other algorithms achieve better performance even though they are not used for training.

10.2 Proposed Method

We use N training images $(x_1, \ldots, x_N)$ to learn T illumination patterns that provide best reconstruction using a predefined (iterative) phase retrieval algorithm. Furthermore, to ensure that the illumination patterns are physically realizable, we constrain their values to be in the range $[0, 1]$. We use a sigmoid function over unconstrained parameters $\Theta = \{\theta_1, \ldots, \theta_T\}$ to define the illumination patterns; that is, $d_t = \text{sigmoid}(\theta_t)$ for all $t = 1, \ldots, T$.

Our proposed method for learning illumination patterns can be divided into two parts: The first (inner) part involves solving the phase retrieval problem with given coded diffraction patterns using AltMin as an unrolled network (see block diagram in Fig. 10.1); Second part is updating the illumination patterns based on backpropagating the image reconstruction loss. These two parts provide optimized image reconstruction and illumination patterns. Pseudocodes for both parts are listed in Algorithms 9,10.

Phase retrieval as alternating minimization (AltMin). Given measurements $Y = \{y_1, \ldots, y_T\}$ and illumination patterns $D = \{d_1, \ldots, d_T\}$, we seek to solve the CDP phase retrieval problem by minimizing the loss function defined in (10.2) as

$$L_x = \frac{1}{2} \sum_{t=1}^{T} \| y_t - |\mathcal{F}(d_t \odot x)| \|_2^2. \tag{10.3}$$

Even though the loss function in (10.3) is nonconvex and nonsmooth with respect to x, we can minimize it using the well-known alternating minimization (AltMin) with gradient

Algorithm 9 Learning illumination patterns

Input: Training set X with N images $X = \{x_1, \ldots, x_N\}$.

Initialize: Initialize the optimization variables for T patterns as $\Theta = \{\theta_1, \ldots, \theta_T\}$ from a uniform distribution.

for epoch $= 1, 2, ..., M$ **do** ▷ M epochs

Generate illumination patterns $d_t = \text{sigmoid}(\theta_t)$

for all t

for $n = 1, 2, ..., N$ **do** ▷ N samples

$Y_n = \{y_{1,n}, \ldots, y_{T,n} \mid y_{t,n} = |\mathcal{F}(d_t \odot x_n)|\}$

$x_n^K(\Theta) \leftarrow$ `solveCDP(`Y_n`,`D`)`

end for

$L_\Theta = \sum_{n=1}^{N} \|x_n - x_n^K(\Theta)\|_2^2$

$\Theta \leftarrow \Theta - \beta \nabla_\Theta L_\Theta$ ▷ Update Θ with stepsize β

end for

Output: Learned illumination patterns

$$D = \{d_1, \ldots, d_T \mid d_t = \text{sigmoid}(\theta_t)\}.$$

descent [92, 85]. We define a new variable for the estimated phase of linear measurements as $p_t = \text{phase}[\mathcal{F}(d_t \odot x)]$ and reformulate the loss function in (10.3) into

$$L_{x,p} = \frac{1}{2} \sum_{t=1}^{T} \|p_t \odot y_t - \mathcal{F}(d_t \odot x)\|_2^2. \tag{10.4}$$

The gradient with respect to x can be computed as

$$\nabla_x L_{x,p} = \sum_{t=1}^{T} |d_t|^2 \odot x - d_t^* \odot \mathcal{F}^*(p_t \odot y_t), \tag{10.5}$$

Algorithm 10 `solveCDP(`Y, D`)` via alternating minimization using single-step gradient descent

Input: Measurements $Y = \{y_1, \ldots, y_t\}$ and illumination patterns $D = \{d_1, \ldots, d_T\}$.

Initialization: Zero initialization of estimate x^0.

for $k = 1, 2, ..., K$ **do** ▷ K iterations of AltMin

$p_t^{k-1} \leftarrow \text{phase}(\mathcal{F}(d_t \odot x^{k-1})$ for all t.

$\nabla_x L_{x,p} = \frac{2}{T} \sum_{t=1}^{T} [|d_t|^2 \odot x^{k-1} - d_t^* \odot \mathcal{F}^*(p_t^{k-1} \odot y_t)]$

$x^k \leftarrow x^{k-1} - \alpha \nabla_x L_{x,p}$

Project x^k onto feasible range.

end for

Output: Estimated signal x^K.

where $\mathcal{F}^*$ denotes the inverse Fourier transform and d_t^* is the conjugate of pattern d_t. We can update the estimate at every iteration as

$$x^k = x^{k-1} - \alpha_{k-1} \nabla_x L_{x,p}, \tag{10.6}$$

where α_{k-1} denotes the step size. Another way is to directly solve for x^k such that $\nabla_x L_{x,p} = 0$. The closed-form solution is

$$x^k = (\sum_{t=1}^{T} |d_t|^2)^{-1} \odot [\sum_{t=1}^{T} d_t^* \odot \mathcal{F}^*(p_t^{k-1} \odot y_t)]. \tag{10.7}$$

We compared these two strategies and found that single-step gradient descent tends to work well in practice and the closed-form solution does not show advantage over the single-step gradient descent. In our implementation, we used the former strategy (Algorithm 10) and fixed a step size α for all iterations. The unrolled network has K layers that implement K iterations of the gradient descent, and the final estimate is denoted as x^K.

Choice of initialization is important, and our method can handle different types of initialization. Zero initialization, where every pixel of the initial guess of x^0 is 0, is the simplest and cost-free method. Many recent phase retrieval algorithms [84, 86, 85, 210] use spectral initialization, which tries to find a good initial estimate. However, it requires computing the principal eigenvector of the following positive semidefinite matrix, $\sum_{t=1}^{T} \text{diag}(d_t^*)\mathcal{F}^*\text{diag}(|y_t|^2)\mathcal{F}\text{diag}(d_t)$. In our experiments, we observed that spectral initialization does not provide a significant improvement in terms of image reconstruction, and that our algorithm can perform very well using the overhead-free zero initialization.

Learning illumination patterns. To learn a set of illumination patterns that provide the best reconstruction with the predefined iterative method (or the unrolled network), we seek to minimize the difference between the original training images and their estimates. In this regard, we minimize the following quadratic loss function with respect to Θ:

$$L_\Theta = \frac{1}{2}\sum_{n=1}^{N} \|x_n - x_n^K(\Theta)\|_2^2, \tag{10.8}$$

where $x_n^K(\Theta)$ denotes the `solveCDP` estimate of nth training image for the given values of Θ. Note that for given real values of $\Theta = \{\theta_1, \ldots, \theta_T\}$, we can define illumination patterns as $d_t = \sigma(\theta_t)$, where $\sigma(\cdot)$ is the *sigmoid* function. We can define sensor measurements for x_n as $y_{t,n} = |\mathcal{F}(d_t \odot x_n)| = p_{t,n}^* \odot \mathcal{F}(d_t \odot x_n)$ for $t = 1, \ldots, T$ and $n = 1, \ldots, N$, where $p_{t,n} = \text{phase}[\mathcal{F}(d_t \odot x_n)]$ is the phase of the original complex-valued signal.

We can use the recursive expression of the signal estimate in (10.6) and the gradient in (10.5) to represent the estimate of x_n at iteration/layer k with the given values of Θ as

$$x_n^k(\Theta) = (1 - \alpha\sum_{t=1}^{T} |d_t|^2)x_n^{k-1}(\Theta) + \alpha\sum_{t=1}^{T} d_t^* \odot \mathcal{F}^*(p_{t,n}^{k-1} \odot y_{t,n}), \tag{10.9}$$

where $p_{t,n}^k = \text{phase}[\mathcal{F}(d_t \odot x_n^k(\Theta))]$. We can compute the gradient of the loss function in (10.8) with respect to any θ_t in a recursive manner as follows.

$$\nabla_{\theta_t} L_\Theta = \sum_{n=1}^{N} J_{\theta_t}(x_n^K(\Theta))[x_n^K(\Theta) - x_n], \tag{10.10}$$

where $J_{\theta_t}(x_n^K(\Theta))$ denotes the Jacobian matrix of the signal estimate with respect to θ_τ. We can now write the product of the Jacobian matrix with a vector u as

$$\begin{aligned} J_{\theta_\tau}(x_n^K(\Theta))[u] &= J_{\theta_\tau}(x_n^{K-1}(\Theta))[(1 - \alpha \sum_{t=1}^{T} |d_t|^2) \odot u] \\ &- 2\alpha |d_\tau|^2 \odot (1 - d_\tau) \odot x_n^{K-1*}(\Theta) \odot u \\ &+ \alpha d_\tau \odot (1 - d_\tau) \odot \mathcal{F}^*(p_{\tau,n}^K \odot y_{\tau,n}) \odot u \\ &+ \alpha d_\tau \odot (1 - d_\tau) \odot x_n \odot \mathcal{F}^*(p_{\tau,n} \odot p_{\tau,n}^{K*} \odot \mathcal{F}(d_\tau \odot u)), \end{aligned} \tag{10.11}$$

where $J_{\theta_\tau}(x_n^0) = 0$ for all n, τ. Here we assume initial estimate $x_n^0 = 0$ and $\alpha_k = \alpha$ for $k = 1, \ldots, K$. We also assume that the phase of the measurements or the signal estimates do not change with small changes in Θ. The overall gradient of the reconstruction loss with respect to the parameters Θ can be computed in a recursive manner (back-propagation) using element-wise products and forward/inverse Fourier transform operations at every iteration/layer.

We can use gradient descent to find the optimal Θ using equation (10.10). We can update the estimate at every iteration of gradient descent as

$$\Theta_m = \Theta_{m-1} - \beta \nabla_\Theta L_\Theta, \tag{10.12}$$

where β denotes the learning rate for the gradient descent.

In practice, we can also compute the gradient using auto-differentiation. In our experiments, we used Adam optimizer in PyTorch [161, 211] to minimize the loss function

Table 10.1: PSNR (`mean` ± `std`) for random and learned illumination patterns tested on different datasets.

Dataset	2 Illumination Patterns		3 Illumination Patterns		4 Illumination Patterns		8 Illumination Patterns	
	Random	Learned	Random	Learned	Random	Learned	Random	Learned
MNIST	14 ± 6	28 ± 9	20 ± 11	75 ± 19	32 ± 14	102 ± 10	61 ± 19	113 ± 11
F. MNIST	17 ± 4	26 ± 6	20 ± 6	49 ± 15	33 ± 9	94 ± 13	67 ± 14	111 ± 12
CIFAR10	15 ± 3	26 ± 4	20 ± 3	34 ± 10	30 ± 8	86 ± 18	64 ± 15	108 ± 18
SVHN	17 ± 3	28 ± 6	24 ± 4	45 ± 15	35 ± 7	93 ± 21	73 ± 15	118 ± 21
CelebA	13 ± 2	19 ± 3	14 ± 4	28 ± 2	23 ± 5	81 ± 4	43 ± 8	98 ± 15

in (10.8). A summary of the algorithm for learning the illumination patterns is also listed in Algorithm 9.

10.3 Experiments

Datasets. We used MNIST digits, Fashion MNIST (F. MNIST), CIFAR10, SVHN, and CelebA datasets for training and testing in our experiments. We used 128 images from each of the datasets for training and another 1000 images for testing. To make the tiny-image datasets uniform, we reshaped all of them to 32×32 size with grayscale values. Images in CelebA dataset have 218×178 pixels, we first converted all the images to grayscale, cropped 178×178 region in the center, and resized to 200×200. We report the performance of our method on images used in [107] in Fig. 10.6.

Measurements. We used the amplitude of the 2D Fourier transform of the images modulated with T illumination patterns as the measurements. Unless otherwise mentioned, we used noiseless measurements. We report results for measurements with Gaussian and Poisson noise in Fig. 10.7.

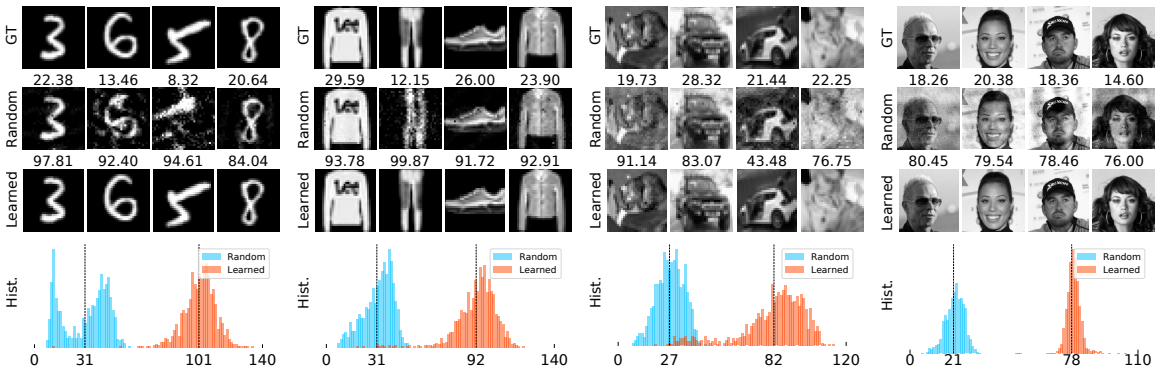
Computing platform. We performed all the experiments using a computer equipped with Intel Core i7-8700 CPU and NVIDIA TITAN Xp GPU. We learned the illumination patterns using a PyTorch implementation, but we also implemented our algorithm in Matlab to provide a fair runtime comparison with existing phase retrieval methods.

10.3.1 Setup and hyper-parameter search

The hyper-parameters include the number of iterations (K), step size α, and the number of training samples N. We set the default value of $K = 50$, but we will show later that K can be adjusted as a trade-off between better reconstruction quality and shorter run time. We tested all methods for $T = \{2, 3, 4, 8\}$ to evaluate cases where signal recovery is hard, moderate, and easy. Through grid search, we found that it provides the best results over all datasets when $\alpha = 4/T$. We also studied the effect of the number of training images and found that illumination patterns learned on 32 randomly selected images provide good recovery over the entire dataset. The test accuracy improves slightly as we increase the number of training samples. To be safe, we used 128 training images in all our experiments. Unless otherwise mentioned, the images are constrained to be in $[0, 1]$ range for our experiments.

10.3.2 Comparison between random and learned illumination patterns

To demonstrate the advantages of our learned illumination patterns, we compare the performance of learned and random illumination patterns on five different datasets. We learn a set of $T = \{2, 3, 4, 8\}$ illumination patterns on 128 training images from a dataset

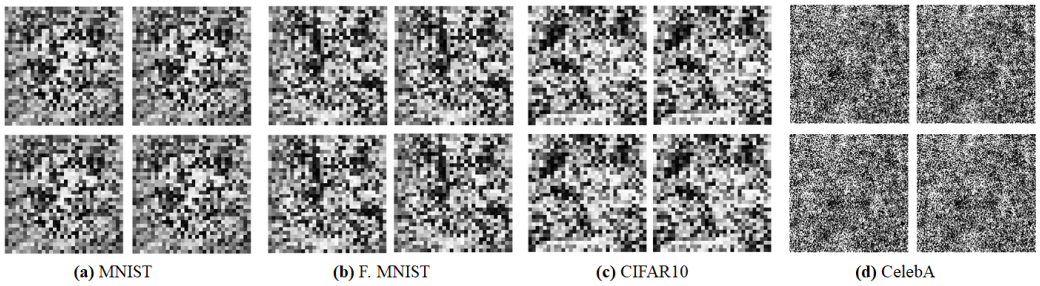


(a) MNIST **(b)** F. MNIST **(c)** CIFAR10 **(d)** CelebA

Figure 10.2: Selected ground truth (GT) images, corresponding reconstructed images using random and learned illumination patterns. PSNR is shown on top of every reconstruction. Below each dataset, we show the histograms of the PSNRs of all images with random patterns (shown in blue) and learned patterns (shown in orange). The dashed vertical line indicates the mean of all PSNRs. We used $T = 4$ illumination patterns. Random illumination patterns are selected best out of 30 trials. The learned illumination patterns are trained on 128 training images.

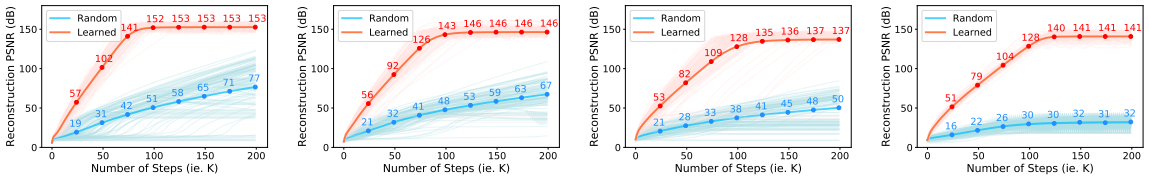
and test them on 1000 test images from the same dataset. For random patterns, we draw T independent patterns from Uniform(0,1) distribution and test their performance on the same 1000 samples that we used for the learned case. We repeat this process 30 times and choose the best result to compare with the results for the learned illumination patterns. The average peak signal-to-noise ratio (PSNR) over all 1000 test image reconstructions is presented in Table 10.1, which shows that the learned illumination patterns perform significantly better than the random patterns for all values of T. In addition to that, we can observe a transition in the performance for $T = 3$, where random patterns provide poor quality reconstructions and learned patterns provide reasonably high quality reconstructions. Furthermore, the learned patterns provide very high quality reconstructions for $T \geq 4$.

To highlight this effect, we show a small set of reconstructed images and histograms of PSNRs of all the reconstructed images from learned and random illumination patterns in Fig. 10.2 for $T = 4$ patterns. The result suggests that the learned illumination patterns demonstrate consistently better performance compared to random illumination patterns. We demonstrate the corresponding learned illumination patterns in Fig. 10.3.



(a) MNIST **(b)** F. MNIST **(c)** CIFAR10 **(d)** CelebA

Figure 10.3: Learned illumination patterns corresponding to the reported results for MNIST, F. MNIST, CIFAR10 and CelebA in Table 10.2.

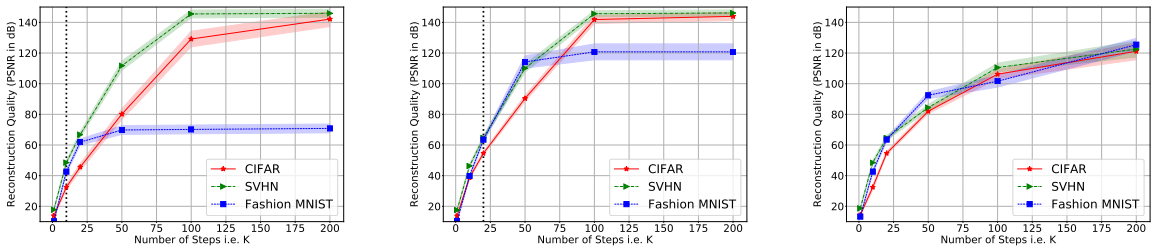


(a) MNIST **(b)** F. MNIST **(c)** CIFAR10 **(d)** CelebA

Figure 10.4: Comparison of the reconstruction quality with random and learned illumination patterns for different values of $K = 1, \ldots, 200$. We plot the average PSNR in bright color and the PSNR of randomly selected 100 samples in light shadows. **Learned** represents the reconstruction PSNR with learned illumination patterns (shown in red), and **Random** represents PSNR for random illumination patterns (shown in blue). The number of illumination patterns is $T = 4$. Random illumination patterns are selected best out of 30 trials. The learned illumination patterns are trained on 128 training images and number of iterations $K = 50$ during training.

10.3.3 Effect of number of iterations/layers (K)

Figure 10.4 shows the performance of the learned and random illumination patterns as we increase K to 200 at test time using the patterns learned for $K = 50$. We observed that with the learned patterns the image reconstruction process converges faster and is more stable (smaller variance) compared to the case with random patterns. The red curve in Fig. 10.4 has a steeper slope and narrower shades. Besides the default setting for $K = 50$, we also learn the illumination patterns for different values of K. Figure 10.5 shows that we can recover images in a small number of iterations if we use learned illumination patterns. We also observe that we can perform better if we use more iterations in testing than in training. We have chosen $K = 50$ for most of the experiments as a trade-off between computational cost and reconstruction performance.



(a) Training K=10 **(b)** Training K=20 **(c)** Training K=Test K

Figure 10.5: Reconstruction quality vs number of iterations (layers) at test time (i.e., K is different for training and testing with $T = 4$). We show error bar of $\pm 0.25\sigma$ for each dataset. In (a) and (b), we fixed K (K=10, 20) and tested using different K. In (c), we trained and tested using the same number of layers.

Table 10.2: Reconstruction PSNR (`mean` ± `std`) of different algorithms using random patterns and our learned patterns. The number of patterns is 4 in each case. Here we round the PSNR values to integers to fit the width of the page. We let all the algorithms to run until convergence.

*For Deep Model [3] experiments, patterns are normalized to $[-1, 1]$ range. **For Deep Model, the image size for CelebA generator is 64×64.

	MNIST		F. MNIST		CIFAR10		SVHN		CelebA	
	Random	Learned	Random	Learned	Random	Learned	Random	Learned	Random	Learned
HIO [196]	16 ± 9	37 ± 19	32 ± 14	61 ± 24	49 ± 20	99 ± 25	60 ± 22	114 ± 27	38 ± 5	102 ± 5
GS [75]	16 ± 9	37 ± 19	33 ± 15	61 ± 24	48 ± 20	99 ± 25	60 ± 22	114 ± 27	38 ± 4	102 ± 5
WirtFlow [84]	22 ± 16	48 ± 25	33 ± 14	51 ± 19	41 ± 10	57 ± 10	41 ± 10	58 ± 10	20 ± 2	39 ± 3
AmpFlow [86]	42 ± 32	74 ± 48	64 ± 38	109 ± 43	86 ± 37	138 ± 25	97 ± 33	144 ± 21	42 ± 8	138 ± 11
PhaseMax [210]	14 ± 4	24 ± 8	21 ± 4	45 ± 20	26 ± 4	97 ± 41	32 ± 5	115 ± 33	32 ± 2	148 ± 2
Ours - K=20	17 ± 6	49 ± 8	20 ± 6	49 ± 8	21 ± 6	49 ± 9	26 ± 5	55 ± 11	16 ± 4	46 ± 3
Ours - K=50	32 ± 14	102 ± 10	33 ± 9	94 ± 13	30 ± 8	86 ± 18	35 ± 7	93 ± 21	23 ± 5	81 ± 4
Ours - K=100	51 ± 19	186 ± 15	49 ± 11	162 ± 22	40 ± 10	139 ± 30	45 ± 10	149 ± 35	33 ± 4	132 ± 7
Deep Model [3]*	31 ± 2	32 ± 3	22 ± 4	22 ± 4	28 ± 3	25 ± 3	26 ± 3	28 ± 4	22 ± 3**	23 ± 2**

Table 10.3: Average runtime (sec) per image of different algorithms corresponding to the performance reported in Table 10.2. The reported runtime corresponds to the time required for convergence of each algorithm.

**For Deep Model, the image size for CelebA is 64×64.

	Max iterations	Image size	
		32×32	200×200
HIO [196]	100	0.473	7.353
GS [75]	100	0.461	7.269
WirtFlow [84]	2000	0.459	10.90
AmpFlow [86]	2000	0.080	2.377
PhaseMax [210]	2000	0.563	10.84
Deep Model [3]	2000	8.422	10.55**
Ours - K=20	20	0.008	0.061
Ours - K=50	50	0.011	0.124
Ours - K=100	100	0.017	0.238

10.3.4 Comparision with existing methods

We compare our method with various existing methods using different datasets. These existing methods fall into 4 categories:

- Hybrid input output (HIO) [196] and Gerchberg-Saxton (GS) [75] (alternating minimization methods)
- Wirtinger Flow [84] and Amplitude Flow [212] (non-convex, gradient descent-based methods)
- PhaseMax [210] (a convex method)
- Deep S^3PR [3] (deep model-based method).

We compare the performance of our method with these methods in terms of reconstruction quality and computation time.

For algorithms in [196, 75, 84, 212, 210], we used PhasePack [104] package. In our comparison, we used 4 illumination patterns and restricted all the illumination patterns in the range of $[0, 1]$. For all the PhasePack algorithms, we used the default spectral initialization. We observed that different algorithms have different computational complexity in each iteration. Thus, a comparison in terms of the number of maximum iterations in all algorithms is not fair. To overcome this issue, we set the error tolerance ($\texttt{tol} = 10^{-6}$) and customize the maximum number iterations in each algorithm to have comparable computations or performance. Specifically, we set the maximum iterations to be 100 for HIO and GS, and 2000 for Wirtinger Flow, Amplitude Flow, and PhaseMax. For our proposed method, we want to keep the number of iterations low (20, 50, 100). To make our runtime comparable with PhasePack algorithms, we implemented our original Python code in Matlab.

For deep generative models, we used a modified version of the publicly available code for [3]. The code only provided pretrained DCGAN models for MNIST and F. MNIST;

therefore, we trained our DCGAN models on the other datasets. This method is noticeably time-consuming because it optimizes over the latent vector for the deep model and uses 2000 iterations for each image where each iteration requires a forward and backward pass through the deep model. The patterns drawn from Uniform(0,1) range did not provide us good reconstruction with Deep Model; therefore, we tested this method using random patterns drawn uniformly from $[-1, 1]$ range and learned patterns that we manually scaled to $[-1, 1]$. The reconstruction results for the Deep Model also directly depend on the quality of the trained generative models. In our experiments, we were not able to generate images with PSNR higher than 30dB using the generative models.

We tested all the methods using Random illumination patterns and the Learned illumination patterns using $K = 50$ in our method. For the case of Random illumination, we selected the best PSNR from 5 independent trials and report the average computation time for each experiment. In all the cases, we tuned the parameters that provide best results.

The reconstruction PSNR (in dB) and run time (in seconds) per image is reported in Table 10.2 and Table 10.3, respectively. We observe that our proposed method with learned patterns performs significantly better than all other algorithms in terms of both reconstruction quality and runtime. We also observed that if we increase the number of iterations for other methods, their reconstruction quality improves beyond the numbers reported in Table 10.2, but this happens at the expense of much longer computation time.

An interesting attribute of our learned patterns is that they can be used with different algorithms. We observe in Table 10.2 that our learned patterns provide better results compared to Random patterns with all the phase retrieval algorithms, even though

the patterns were not optimized for those algorithms.

10.3.5 Generalization of learned patterns on different datasets

To explore the generalizability of our learned illumination patterns, we use patterns learned on one dataset to recover images from another. The results are shown in Table. 10.4. As we can see in the table, the diagonal numbers are generally the best, and off-diagonal numbers are generally better than the ones with random illumination patterns.

We have also tested the learned illumination patterns on several classical images. Some results are shown in Fig. 10.6. We used illumination patterns learned on 128 celebA images, but we can see that the learned illumination patterns perform better than the randomly chosen illumination patterns for classical images which further supports the generalizability of our learned illumination patterns.

Table 10.4: Reconstruction PSNR (`mean` $\pm$ `std`) of illumination patterns learned and tested on different datasets for $K = 50$. Every column corresponds to patterns learned on a fixed dataset and tested on all. Random column reports the performance of random illumination patterns.

Test \ Train	4 Illumination Patterns					8 Illumination Patterns				
	MNIST	F. MNIST	CIFAR10	SVHN	Random	MNIST	F. MNIST	CIFAR10	SVHN	Random
MNIST	102 ± 10	66 ± 16	34 ± 15	48 ± 15	32 ± 14	113 ± 11	84 ± 13	56 ± 20	74 ± 19	61 ± 19
F. MNIST	84 ± 24	94 ± 13	50 ± 20	64 ± 19	33 ± 9	94 ± 23	111 ± 12	89 ± 20	108 ± 21	67 ± 14
CIFAR10	79 ± 27	87 ± 13	86 ± 18	96 ± 17	30 ± 8	84 ± 18	88 ± 17	108 ± 18	113 ± 17	64 ± 15
SVHN	56 ± 28	78 ± 16	72 ± 21	93 ± 21	35 ± 7	76 ± 19	95 ± 12	91 ± 24	118 ± 21	73 ± 15

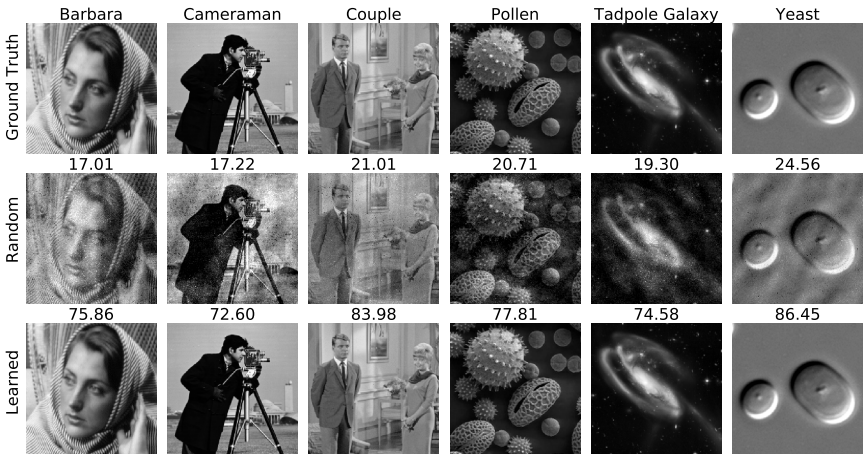
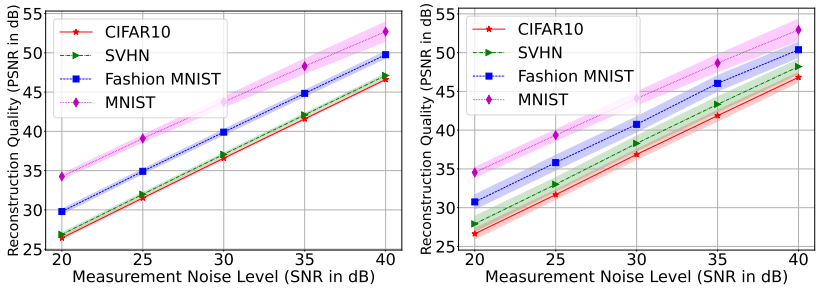


Figure 10.6: First Row: Ground truth images from image processing standard test datasets. **Second Row:** Reconstruction using random illumination patterns with uniform random distribution [0, 1] (we selected $T = 4$ patterns that provided best results on celebA test images in **30 trials**). PSNR numbers are shown on the top of reconstructed images. **Third Row:** Reconstruction using the patterns trained on celebA dataset. Each image has 200×200 pixels and the number of illumination patterns is $T = 4$.



(a) Gaussian **(b)** Poisson

Figure 10.7: Reconstruction quality of the test images vs noise level of the measurements for different datasets. Here we show shaded error bar of $\pm 0.25\sigma$ for each dataset. We learn the illumination patterns ($T = 4$) on 128 noiseless training images of corresponding datasets.

Table 10.5: Reconstruction PSNR (`mean` ± `std`) of different algorithms using random patterns (best out of 5 trials) and our learned patterns at different Poisson noise levels for MNIST and CIFAR10 dataset. The number of patterns is 4 in each case. We let all the algorithms to run until convergence. Here we round the PSNR values to integers to fit the width of the page.

MNIST												
Noise SNR	0		5		10		20		30		40	
	Random	Learned	Random	Learned	Random	Learned	Random	Learned	Random	Learned	Random	Learned
HIO [196]	23±13	25±15	17±10	19±12	22±12	18±10	18±11	23±16	22±11	10±3	20±11	11±4
GS [75]	16±9	25±15	19±11	18±12	22±13	18±10	20±11	22±16	21±11	10±3	17±8	11±4
WirtFlow [84]	20±16	25±15	23±19	23±18	27±20	25±16	29±20	28±22	30±19	14±9	31±19	24±16
PhaseMax [210]	16±5	18±6	13±3	16±6	15±5	16±5	17±5	17±6	17±5	11±2	16±4	11±2
Ours - K=50	28±16	24±3	21±13	28±5	28±12	31±5	16±11	48±13	22±13	65±21	27±13	61±17
CIFAR10												
Noise SNR	0		5		10		20		30		40	
	Random	Learned	Random	Learned	Random	Learned	Random	Learned	Random	Learned	Random	Learned
HIO [196]	28±26	18±16	28±28	20±18	27±25	31±31	28±25	41±42	28±26	47±43	29±27	51±44
GS [75]	27±27	17±15	26±25	19±18	32±30	33±32	27±26	45±42	27±26	47±43	29±26	51±44
WirtFlow [84]	23±20	16±14	23±19	18±16	23±18	23±22	23±20	31±28	23±19	30±28	24±20	33±30
PhaseMax [210]	17±12	23±22	16±10	23±22	16±11	29±32	16±11	50±55	17±11	48±52	18±12	58±60
Ours - K=50	29±6	26±9	28±7	30±12	29±7	38±10	28±5	51±10	30±8	68±12	31±7	71±9

10.3.6 Noise response

To investigate the robustness of our method to noise, we train our illumination patterns on noiseless measurements obtained from the training datasets. We then added Gaussian and Poisson noise at different levels to the measurements from the test datasets. Poisson noise or shot noise is the most common in the imaging systems, which we add following the approach in [107, 213]. Let us denote the i^{th} element of measurement vector corresponding to t^{th} illumination pattern, y_t as

$$y_t(i) = |z_t(i)| + \eta_t(i), \quad \text{for } i = 1, 2, \ldots, m, \tag{10.13}$$

where $\eta_t(i) \sim \mathcal{N}(0, \lambda|z_t(i)|)$ and $z_t = \mathcal{F}(d_t \odot x)$. We varied λ to generate noise at different signal-to-noise ratio (SNR) levels. Poisson noise affects larger values in measurements with higher strength than the smaller values. Since the sensors can measure only positive measurements, we kept the measurements positive by applying ReLU function after noise

addition. We expect the reconstruction to be affected by noise as we did not use any denoiser. We observe the effect of noise in Figure 10.7. Even though noise affects the reconstructions, we can get reasonable reconstruction up to a certain level of noise. The relationship between noise level and reconstruction performance also indicates that our phase retrieval system is quite stable.

We ran another set of experiments where we learned different set of illumination patterns at different noise level by introducing measurement noise during training. In Table 10.5, we report results for MNIST and CIFAR10 dataset at different level of Poisson noise introduced during training and testing. We show the performance of some comparing approaches with our learned patterns and random patterns. For random patterns, we reported the results for the best out of 5 runs. We can observe that even under the presence of high noise (0-20dB), the learned illumination patterns using our approach performs reasonably well. We observe performance boost with our learned patterns for 5dB or higher SNR.

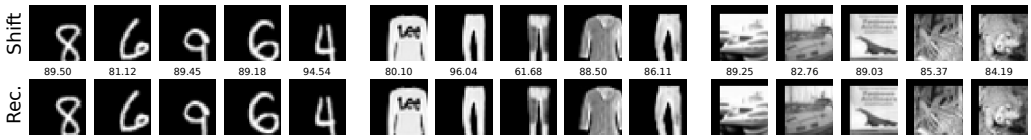


Figure 10.8: Test results on images shifted to bottom right by 5 pixels. From left to right: MNIST, F. MNIST, and CIFAR10.

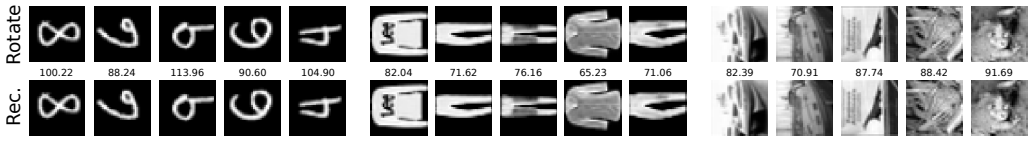


Figure 10.9: Test results on images rotated by 90°. From left to right: MNIST, F. MNIST, and CIFAR10.

10.3.7 Mismatch in training and test images

In our final experiment, we tested illumination patterns trained on upright images to recover shifted and rotated images. Our results in Fig. 10.8 and Fig. 10.9 show that the learned patterns reliably recover images regardless of the position or orientation. This is not surprising because we do not learn to represent images or solve the phase retrieval problem using the training data; instead, we only learn the illumination patterns using a predefined AltMin-based recovery algorithm. In contrast, data-driven methods that learn to solve the inverse problem may suffer if the distribution of test images differ significantly from the training images.

Chapter 11

Conclusion

The work presented in this thesis explores different structures in deep networks and formulates efficient algorithms to utilize these structures for solving continual learning and different linear and nonlinear inverse problems. The models used in these algorithms are static and deterministic. The main focus of this work is to achieve good performance with reduced computational complexity while applying deep networks for solving different problems.

11.1 Continual Learning with Low-Rank Increment

We proposed a new incremental task learning method in which we update the network weights using low rank increments as we learn new tasks. Network layers are represented as a linear combination of low-rank factors. To update the network for a new task, we freeze the factors learned for previous tasks, add a new low-rank (or rank-1) factor, and combine that with the previous factors using a learned combination. The proposed

method offered considerable improvement in performance compared to the state-of-the-art methods for ITL in image classification tasks. In addition, the proposed low-rank ITL circumvents the use of memory buffer or large memory overhead while achieving zero forgetting. The need for task identity knowledge is a general limitation of our and other ITL methods. Such methods can be useful for incremental multitask learning where task identity is available during inference but training data is only available in a short window.

11.2 Inverse Problems with Deep Networks

11.2.1 Solving Linear Inverse Problems with Untrained Generative Prior

We used untrained generative network as a prior to solve ill posed linear inverse problems. Usually in the untrained generative priors, the latent codes are kept fixed at random initialization where the network parameters are optimized to provide solution. In our experiments, we observe that joint optimization of network weights and latent codes performs remarkably well for compressive measurements. Even though the number of measurements are extremely small compared to the number of parameters in the network, the solution almost always converges to a good sequence. Introducing low-rank constraint in the optimization, we get additional degree of compression with comparable performance. We show comparison with classical and generative prior based techniques in terms of reconstruction performance and computational complexity. We also demonstrate one application of joint optimization in coded flutter shutter problem where its tractable memory requirement and lower computational cost makes it more suitable than other generative prior based approaches.

We extend this setup with tensor factorization replacing low-rank matrix factorization in the latent space. We formulated a tensor ring factorized autoencoder to reconstruct structured datasets from their corrupted versions utilizing the structural similarity between the images in the dataset. We demonstrate that utilizing the structural information in the latent space can significantly improve performance of generative prior based approaches.

Finally, we present a consensus equilibrium framework to incorporate untrained generative prior with other denoiser based priors for solving image deblurring problem. We show that we can achieve robust performance under the presence of high noise and blurring using consensus equilibrium formulation.

11.2.2 Solving Phase Retrieval with Trained Generative Prior

We developed a projected gradient descent based approach to use trained generative models as a prior to solve compressive phase retrieval problems. We show comparisons with other generative prior based approaches and demonstrate that we can reconstruct images with very low number of measurements. We then show that this approach can be benefited more if we know some side information about the target signal. We show qualitative results on Fourier phase retrieval problem which suggests that generative prior based approaches can perform significantly better with side information.

11.2.3 Learning Sensing Parameters Using Unrolling Networks

We presented an unrolling network based framework for learning a reference signal for holographic imaging and illumination patterns for coded diffraction imaging. Both of these imaging problems are variations of Fourier phase retrieval problem. The reference signal

and the illumination patterns can be considered as the sensing parameters for corresponding imaging problems. The sensing parameters are learned via backpropagation using a small number of training images by formulating an iterative phase retrieval algorithm as a fixed unrolled network. Once learned, the sensing parameters significantly improve the efficiency of the signal reconstruction in the phase retrieval process. The number of iterations in our algorithm provides a clear trade-off between reconstruction accuracy and run time. The learned parameters generalizes to a broad class of datasets with different distribution compared to the training samples. We demonstrated the robustness and efficiency of our method through extensive experiments.

11.3 Future Directions

11.3.1 Continual Learning with Low-Rank Networks

Class Incremental Learning: We discussed about incremental task learning (ITL) setup in our work. We can extend it to incremental class learning (ICL) setup where we do not have access to task identity of a data during inference. It is a harder and more practical continual learning scenario. We can extend our formulation for ICL setup by applying an approach for task identification during inference and using the task specific factors accordingly. Different approaches have been proposed for task identification in the literature [33, 41].

Low-rank Tensor Factorization based Network: We used low-rank matrix factorization to factorize both fully connected layers and convolutional layers. Although matrix factorization is intuitive for 2-d weights of fully connected layers, for 4-d weights of convolu-

tional layers, it is more intuitive to use tensor factorization instead. Tensor factorization can potentially provide more compression and more expressive power than matrix factorization for convolutional networks.

11.3.2 Inverse Problems with Structured Networks

Extended Applications: We discussed different applications for our proposed approaches including phase retrieval, deblurring, denoising, inpainting etc. We can extend the proposed formulations for other applications as well. For example, we showed the application of unrolled networks in learning coded illumination patterns in Chapter 10. It can be extended to Fourier ptychography which is another related Fourier phase retrieval problem. We can also potentially introduce denoisers in between the layers of unrolling network to increase noise robustness. In Chapter 5, we used tensor factorized network to recover corrupted images with Gaussian noise or missing pixels. This formulation can be extended to hyperspectral imaging or image super resolution setup which are practically more realistic setups. In Chapter 6, we discussed consensus equilibrium for non-blind deblurring which can be extended to blind deblurring scenario.

Novel Image Generation: We demonstrated image recovery from corrupted images in Chapters 4, 5 and 6. In one extreme case, we can have some images entirely missing from the training dataset. Using the structure in the dataset and generative prior we can potentially try to recover the missing images given the available images. The tensor factorization based formulation discussed in Chapter 5 can be helpful in this respect. If we can learn a network which can disentangle different visual attributes successfully with different factors, we can

potentially generate novel images using the learnt factors with the given images. This research direction has different potential applications including dataset augmentation, image editing etc.

Bibliography

[1] Chengbo Li, Wotao Yin, Hong Jiang, and Yin Zhang. An efficient augmented lagrangian method with applications to total variation minimization. *Computational Optimization and Applications*, 56(3):507–530, 2013.

[2] R. Heckel and P. Hand. Deep decoder: Concise image representations from untrained non-convolutional networks. *Proc. Int. Conf. Learning Representations (ICLR)*, 2018.

[3] Christopher A Metzler and Gordon Wetzstein. Deep s 3 pr: Simultaneous source separation and phase retrieval using deep generative models. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1370–1374. IEEE, 2021.

[4] Matthias Delange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Ales Leonardis, Greg Slabaugh, and Tinne Tuytelaars. A continual learning survey: Defying forgetting in classification tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[5] Daniel L Silver and Robert E Mercer. The task rehearsal method of life-long learning: Overcoming impoverished data. In *Conference of the Canadian Society for Computational Studies of Intelligence*, pages 90–101. Springer, 2002.

[6] Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pages 109–165. Elsevier, 1989.

[7] Arslan Chaudhry, Ranzato Marc'Aurelio, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with a-gem. In *International Conference on Learning Representations, ICLR*, 2019.

[8] Arslan Chaudhry, Naeemullah Khan, Puneet Dokania, and Philip Torr. Continual learning in low-rank orthogonal subspaces. *Advances in Neural Information Processing Systems*, 33, 2020.

[9] Julio Hurtado, Alain Raymond-Saez, and Alvaro Soto. Optimizing reusable knowledge for continual learning via metalearning. *Advances in Neural Information Processing Systems*, 34, 2021.

[10] Danruo Deng, Guangyong Chen, Jianye Hao, Qiong Wang, and Pheng-Ann Heng. Flattening sharpness for dynamic gradient projection memory benefits continual learning. *Advances in Neural Information Processing Systems*, 34, 2021.

[11] Tom Veniat, Ludovic Denoyer, and MarcAurelio Ranzato. Efficient continual learning with modular networks and task-driven priors. In *International Conference on Learning Representations*, 2021.

[12] Gobinda Saha, Isha Garg, and Kaushik Roy. Gradient projection memory for continual learning. In *International Conference on Learning Representations*, 2021.

[13] Haiyan Yin, Ping Li, et al. Mitigating forgetting in online continual learning with neuron calibration. *Advances in Neural Information Processing Systems*, 34, 2021.

[14] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Deep image prior. In *Proc. IEEE Conf. Comp. Vision and Pattern Recog. (CVPR)*, pages 9446–9454, 2018.

[15] Cuong V Nguyen, Yingzhen Li, Thang D Bui, and Richard E Turner. Variational continual learning. In *International Conference on Learning Representations*, 2018.

[16] Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947, 2017.

[17] Rahaf Aljundi, Punarjay Chakravarty, and Tinne Tuytelaars. Expert gate: Lifelong learning with a network of experts. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3366–3375, 2017.

[18] Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. Memory aware synapses: Learning what (not) to forget. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 139–154, 2018.

[19] Rahaf Aljundi, Min Lin, Baptiste Goujaud, and Yoshua Bengio. Gradient based sample selection for online continual learning. *Advances in Neural Information Processing Systems*, 32:11816–11825, 2019.

[20] Arslan Chaudhry, Puneet K Dokania, Thalaiyasingam Ajanthan, and Philip HS Torr. Riemannian walk for incremental learning: Understanding forgetting and intransigence. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 532–547, 2018.

[21] Matthew Riemer, Ignacio Cases, Robert Ajemian, Miao Liu, Irina Rish, Yuhai Tu, and Gerald Tesauro. Learning to learn without forgetting by maximizing transfer and minimizing interference. In *International Conference on Learning Representations*, 2019.

[22] David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy Lillicrap, and Gregory Wayne. Experience replay for continual learning. *Advances in Neural Information Processing Systems*, 32:350–360, 2019.

[23] Mehrdad Farajtabar, Navid Azizan, Alex Mott, and Ang Li. Orthogonal gradient descent for continual learning. In *International Conference on Artificial Intelligence and Statistics*, pages 3762–3773. PMLR, 2020.

[24] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

[25] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *NeurIPS*, 2014.

[26] Pietro Buzzega, Matteo Boschini, Angelo Porrello, Davide Abati, and Simone Calderara. Dark experience for general continual learning: a strong, simple baseline. *Advances in neural information processing systems*, 33:15920–15930, 2020.

[27] Shipeng Wang, Xiaorong Li, Jian Sun, and Zongben Xu. Training networks in null space of feature covariance for continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 184–193, 2021.

[28] Shixiang Tang, Dapeng Chen, Jinguo Zhu, Shijie Yu, and Wanli Ouyang. Layerwise optimization by gradient decomposition for continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9634–9643, 2021.

[29] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 2001–2010, 2017.

[30] Arslan Chaudhry, Marcus Rohrbach, Mohamed Elhoseiny, Thalaiyasingam Ajanthan, Puneet K Dokania, Philip HS Torr, and Marc'Aurelio Ranzato. On tiny episodic memories in continual learning. *arXiv preprint arXiv:1902.10486*, 2019.

[31] Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016.

[32] Arun Mallya and Svetlana Lazebnik. Packnet: Adding multiple tasks to a single network by iterative pruning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7765–7773, 2018.

[33] Mitchell Wortsman, Vivek Ramanujan, Rosanne Liu, Aniruddha Kembhavi, Mohammad Rastegari, Jason Yosinski, and Ali Farhadi. Supermasks in superposition. *Advances in Neural Information Processing Systems*, 33, 2020.

[34] Yeming Wen, Dustin Tran, and Jimmy Ba. Batchensemble: an alternative approach to efficient ensemble and lifelong learning. In *International Conference on Learning Representations*, 2020.

[35] Joan Serra, Didac Suris, Marius Miron, and Alexandros Karatzoglou. Overcoming catastrophic forgetting with hard attention to the task. In *International Conference on Machine Learning*, pages 4548–4557. PMLR, 2018.

[36] Jaehong Yoon, Eunho Yang, Jeongtae Lee, and Sung Ju Hwang. Lifelong learning with dynamically expandable networks. In *International Conference on Learning Representations*, 2018.

[37] Johannes von Oswald, Christian Henning, João Sacramento, and Benjamin F Grewe. Continual learning with hypernetworks. In *International Conference on Learning Representations*, 2019.

[38] Nicolas Y. Masse, Gregory D. Grant, and David J. Freedman. Alleviating catastrophic forgetting using context-dependent gating and synaptic stabilization. *Proceedings of the National Academy of Sciences*, 115(44):E10467–E10475, 2018.

[39] Arun Mallya, Dillon Davis, and Svetlana Lazebnik. Piggyback: Adapting a single network to multiple tasks by learning to mask weights. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 72–88, 2018.

[40] Davide Abati, Jakub Tomczak, Tijmen Blankevoort, Simone Calderara, Rita Cucchiara, and Babak Ehteshami Bejnordi. Conditional channel gated networks for task-aware continual learning. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition*, pages 3930–3939, 2020.

[41] Nikhil Mehta, Kevin Liang, Vinay Kumar Verma, and Lawrence Carin. Continual learning using a bayesian nonparametric dictionary of weight factors. In *International Conference on Artificial Intelligence and Statistics*, pages 100–108. PMLR, 2021.

[42] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Proc. Adv. in Neural Inf. Proc. Sys. (NIPS)*, pages 2672–2680, 2014.

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

[44] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

[45] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE Trans. Pattern Anal. Machine Intell.*, 35(8):1798–1828, 2013.

[46] Zachary C Lipton and Subarna Tripathi. Precise recovery of latent vectors from generative adversarial networks. *arXiv preprint arXiv:1702.04782*, 2017.

[47] Jun-Yan Zhu, Philipp Krähenbühl, Eli Shechtman, and Alexei A. Efros. Generative visual manipulation on the natural image manifold. In *Proc. European Conf. Comp. Vision (ECCV)*, 2016.

[48] A. Creswell and A. A. Bharath. Inverting the generator of a generative adversarial network. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–8, 2018.

[49] P. Bojanowski, A. Joulin, D. Lopez-Paz, and A. Szlam. Optimizing the latent space of generative networks. In *Proc. Int. Conf. Machine Learning*, 2018.

[50] Femke van Belzen and Siep Weiland. A tensor decomposition approach to data compression and approximation of nd systems. *Multidimensional Systems and Signal Processing*, 23(1-2):209–236, 2012.

[51] Wenqi Wang, Yifan Sun, Brian Eriksson, Wenlin Wang, and Vaneet Aggarwal. Wide compression: Tensor ring nets. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9329–9338, 2018.

[52] Andros Tjandra, Sakriani Sakti, and Satoshi Nakamura. Tensor decomposition for compressing recurrent neural network. In *2018 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2018.

[53] Xinyu Chen, Zhaocheng He, and Lijun Sun. A bayesian tensor decomposition approach for spatiotemporal traffic data imputation. *Transportation research part C: emerging technologies*, 98:73–84, 2019.

[54] Sriram Krishnaswamy and Mrinal Kumar. Tensor decomposition approach to data association for multitarget tracking. *Journal of Guidance, Control, and Dynamics*, 42(9):2007–2025, 2019.

[55] Maxim Kuznetsov, Daniil Polykovskiy, Dmitry P Vetrov, and Alex Zhebrak. A prior of a googol gaussians: a tensor ring induced prior for generative models. In *Advances in Neural Information Processing Systems*, pages 4104–4114, 2019.

[56] Zhiwei Deng, Rajitha Navarathna, Peter Carr, Stephan Mandt, Yisong Yue, Iain Matthews, and Greg Mori. Factorized variational autoencoders for modeling audience reactions to movies. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2577–2586, 2017.

[57] Li Jing, Jure Zbontar, et al. Implicit rank-minimizing autoencoder. *Advances in Neural Information Processing Systems*, 33, 2020.

[58] Emmanuel J Candes, Yonina C Eldar, Deanna Needell, and Paige Randall. Compressed sensing with coherent and redundant dictionaries. *Applied and Computational Harmonic Analysis*, 31(1):59–73, 2011.

[59] David L Donoho. Compressed sensing. *IEEE Transactions on information theory*, 52(4):1289–1306, 2006.

[60] Emmanuel J Candes and Terence Tao. Decoding by linear programming. *IEEE transactions on information theory*, 51(12):4203–4215, 2005.

[61] Marco F Duarte, Mark A Davenport, Dharmpal Takhar, Jason N Laska, Ting Sun, Kevin F Kelly, and Richard G Baraniuk. Single-pixel imaging via compressive sampling. *IEEE signal processing magazine*, 25(2):83–91, 2008.

[62] Richard Baraniuk and Philippe Steeghs. Compressive radar imaging. In *Radar Conference, 2007 IEEE*, pages 128–133. IEEE, 2007.

[63] Fei Yang, Hong Jiang, Zuowei Shen, Wei Deng, and Dimitris Metaxas. Adaptive low rank and sparse decomposition of video using compressive sensing. In *Proc. IEEE Int. Conf. Image Processing (ICIP)*, pages 1016–1020. IEEE, 2013.

[64] Jianing V Shi, Aswin C Sankaranarayanan, Christoph Studer, and Richard G Baraniuk. Video compressive sensing for dynamic mri. *BMC neuroscience*, 13(1):P183, 2012.

[65] Chen Zhao, Siwei Ma, Jian Zhang, Ruiqin Xiong, and Wen Gao. Video compressive sensing reconstruction via reweighted residual sparsity. *IEEE Transactions on Circuits and Systems for Video Technology*, 27(6):1182–1195, 2017.

[66] A. Bora, A. Jalal, E. Price, and A. Dimakis. Compressed sensing using generative models. *Proc. Int. Conf. Machine Learning*, 2017.

[67] Anna C Gilbert, Yi Zhang, Kibok Lee, Yuting Zhang, and Honglak Lee. Towards understanding the invertibility of convolutional neural networks. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pages 1703–1710. AAAI Press, 2017.

[68] D. Van Veen, A. Jalal, E. Price, S. Vishwanath, and Alexandros G. Dimakis. Compressed sensing with deep image prior and learned regularization. *arXiv preprint arXiv:1806.06438*, 2018.

[69] V. Shah and C. Hegde. Solving linear inverse problems using gan priors: An algorithm with provable guarantees. *Proc. IEEE Int. Conf. Acoust., Speech, and Signal Processing (ICASSP)*, 2018.

[70] Y. Shechtman, Y. Eldar, O. Cohen, H. Chapman, J. Miao, and M. Segev. Phase retrieval with application to optical imaging: a contemporary overview. *IEEE Signal Processing Mag.*, 32(3):87–109, 2015.

[71] A. Maiden and J. Rodenburg. An improved ptychographical phase retrieval algorithm for diffractive imaging. *Ultramicroscopy*, 109(10):1256–1262, 2009.

[72] J. R. Fienup. Phase retrieval algorithms: a comparison. *Applied optics*, 21(15):2758–2769, 1982.

[73] John M Rodenburg. Ptychography and related diffractive imaging methods. *Advances in imaging and electron physics*, 150:87–184, 2008.

[74] R. Millane. Phase retrieval in crystallography and optics. *JOSA A*, 7(3):394–411, 1990.

[75] R. W. Gerchberg. A practical algorithm for the determination of phase from image and diffraction plane pictures. *Optik*, 35:237–246, 1972.

[76] P. Hand, O. Leong, and V. Voroninski. Phase retrieval under a generative prior. In *Proc. Adv. in Neural Inf. Proc. Sys. (NeurIPS)*, pages 9154–9164, 2018.

[77] Seyedehsara Nayer, Praneeth Narayanamurthy, and Namrata Vaswani. Phaseless pca: Low-rank matrix recovery from column-wise phaseless measurements. In *International Conference on Machine Learning*, pages 4762–4770, 2019.

[78] R. Hyder, Viraj S., C. Hegde, and M.S. Asif. Alternating phase projected gradient descent with generative priors for solving compressive phase retrieval. In *Proc. IEEE Int. Conf. Acoust., Speech, and Signal Processing (ICASSP)*, pages 7705–7709. IEEE, 2019.

[79] E. Candes, T. Strohmer, and V. Voroninski. Phaselift: Exact and stable signal recovery from magnitude measurements via convex programming. *Comm. Pure Appl. Math.*, 66(8):1241–1274, 2013.

[80] D. Gross, F. Krahmer, and R. Kueng. Improved recovery guarantees for phase retrieval from coded diffraction patterns. *Appl. Comput. Harmon. Anal.*, 42(1):37–64, 2017.

[81] E. Candes, X. Li, and M. Soltanolkotabi. Phase retrieval from coded diffraction patterns. *Appl. Comput. Harmon. Anal.*, 39(2):277–299, 2015.

[82] G. Wang, L. Zhang, G. B. Giannakis, M. Akcakaya, and J. Chen. Sparse phase retrieval via truncated amplitude flow. *IEEE Trans. Signal Processing*, 66:479–491, 2018.

[83] G. Wang and G. Giannakis. Solving random systems of quadratic equations via truncated generalized gradient flow. In *Proc. Adv. in Neural Inf. Proc. Sys. (NeurIPS)*, pages 568–576, 2016.

[84] E. Candes, X. Li, and M. Soltanolkotabi. Phase retrieval via wirtinger flow: theory and algorithms. *IEEE Trans. Inform. Theory*, 61(4):1985–2007, 2015.

[85] H. Zhang and Y. Liang. Reshaped wirtinger flow for solving quadratic system of equations. In *Proc. Adv. in Neural Inf. Proc. Sys. (NeurIPS)*, pages 2622–2630, 2016.

[86] Y. Chen and E. Candes. Solving random quadratic systems of equations is nearly as easy as solving linear systems. In *Proc. Adv. in Neural Inf. Proc. Sys. (NeurIPS)*, pages 739–747, 2015.

[87] T. Cai, X. Li, Z. Ma, et al. Optimal rates of convergence for noisy sparse phase retrieval via thresholded wirtinger flow. *Ann. Stat.*, 44(5):2221–2251, 2016.

[88] H. Ohlsson, A. Yang, R. Dong, and S. Sastry. Cprl–an extension of compressive sensing to the phase retrieval problem. In *Proc. Adv. in Neural Inf. Proc. Sys. (NeurIPS)*, pages 1367–1375, 2012.

[89] X. Li and V. Voroninski. Sparse signal recovery from quadratic measurements via convex programming. *SIAM J. on Math. Analysis*, 45(5):3019–3033, 2013.

[90] S. Bahmani and J. Romberg. Efficient compressive phase retrieval with constrained sensing vectors. In *Proc. Adv. in Neural Inf. Proc. Sys. (NeurIPS)*, pages 523–531, 2015.

[91] K. Jaganathan, S. Oymak, and B. Hassibi. Recovery of sparse 1-d signals from the magnitudes of their fourier transform. In *Proc. IEEE Int. Symp. Inform. Theory (ISIT)*, pages 1473–1477. IEEE, 2012.

[92] P. Netrapalli, P. Jain, and S. Sanghavi. Phase retrieval using alternating minimization. In *Proc. Adv. in Neural Inf. Proc. Sys. (NeurIPS)*, pages 2796–2804, 2013.

[93] G. Jagatap and C. Hegde. Fast, sample-efficient algorithms for structured phase retrieval. In *Advances in Neural Information Processing Systems*, pages 4917–4927, 2017.

[94] H. Chang, Y. Lou, M.K. Ng, and T. Zeng. Phase retrieval from incomplete magnitude information via total variation regularization. *SIAM Journal on Scientific Computing*, 38(6):A3672–A3695, 2016.

[95] Gauri Jagatap and Chinmay Hegde. Algorithmic guarantees for inverse imaging with untrained network priors. In *Advances in Neural Information Processing Systems*, pages 14832–14842, 2019.

[96] F. Shamshad and A. Ahmed. Robust compressive phase retrieval via deep generative priors. *arXiv preprint arXiv:1808.05854*, 2018.

[97] IS Park, RJC Middleton, Charles R Coggrave, Pablo D Ruiz, and Jeremy M Coupland. Characterization of the reference wave in a compact digital holographic camera. *Applied optics*, 57(1):A235–A241, 2018.

[98] Tatsuki Tahara, Xiangyu Quan, Reo Otani, Yasuhiro Takaki, and Osamu Matoba. Digital holography and its multidimensional imaging applications: a review. *Microscopy*, 67(2):55–67, 2018.

[99] D.A. Barmherzig, J. Sun, P. Li, T.J. Lane, and E. Candès. Holographic phase retrieval and reference design. *Inverse Problems*, 2019.

[100] Z. Yuan and H. Wang. Phase retrieval with background information. *Inverse Problems*, 35(5):054003, may 2019.

[101] M. Guizar-Sicairos and J.R. Fienup. Holography with extended reference by autocorrelation linear differential operation. *Optics express*, 15(26):17592–17612, 2007.

[102] R. Hyder, C. Hegde, and M.S. Asif. Fourier phase retrieval with side information using generative prior. In *Proc. Asilomar Conf. Signals, Systems, and Computers*. IEEE, 2019.

[103] Fahimeh Arab and M Salman Asif. Fourier phase retrieval with arbitrary reference signal. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1479–1483. IEEE, 2020.

[104] Rohan Chandra, Ziyuan Zhong, Justin Hontz, Val McCulloch, Christoph Studer, and Tom Goldstein. Phasepack: A phase retrieval library. *Asilomar Conference on Signals, Systems, and Computers*, 2017.

[105] Jianwei Miao, Tetsuya Ishikawa, Qun Shen, and Thomas Earnest. Extending x-ray crystallography to allow the imaging of noncrystalline materials, cells, and single protein complexes. *Annu. Rev. Phys. Chem.*, 59:387–410, 2008.

[106] G. Jagatap, Z. Chen, S. Nayer, C. Hegde, and N. Vaswani. Sample efficient fourier ptychography for structured data. *IEEE Transactions on Computational Imaging*, 6:344–357, 2020.

[107] Christopher A Metzler, Philip Schniter, Ashok Veeraraghavan, and Richard G Baraniuk. prdeep: Robust phase retrieval with a flexible deep network. In *Proc. Int. Conf. Machine Learning*, 2018.

[108] Michael Kellman, Emrah Bostan, Michael Chen, and Laura Waller. Data-driven design for fourier ptychographic microscopy. *International Conference for Computational Photography*, pages 1–8, 2019.

[109] Yair Rivenson, Yibo Zhang, Harun Günaydın, Da Teng, and Aydogan Ozcan. Phase recovery and holographic image reconstruction using deep learning in neural networks. *Light: Science & Applications*, 7(2):17141–17141, 2018.

[110] Rakib Hyder, Zikui Cai, and M Salman Asif. Solving phase retrieval with a learned reference. In *Proc. European Conf. Comp. Vision (ECCV)*, 2020.

[111] Michael R Kellman, Emrah Bostan, Nicole A Repina, and Laura Waller. Physics-based learned design: optimized coded-illumination for quantitative phase imaging. *IEEE Transactions on Computational Imaging*, 5(3):344–353, 2019.

[112] Steven Diamond, Vincent Sitzmann, Felix Heide, and Gordon Wetzstein. Unrolled optimization with deep priors. *arXiv preprint arXiv:1705.08041*, 2017.

[113] Karol Gregor and Yann LeCun. Learning fast approximations of sparse coding. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, pages 399–406, 2010.

[114] Shenlong Wang, Sanja Fidler, and Raquel Urtasun. Proximal deep structured models. In *Advances in Neural Information Processing Systems*, pages 865–873, 2016.

[115] Kerstin Hammernik, Teresa Klatzer, Erich Kobler, Michael P Recht, Daniel K Sodickson, Thomas Pock, and Florian Knoll. Learning a variational network for reconstruction of accelerated mri data. *Magnetic resonance in medicine*, 79(6):3055–3071, 2018.

[116] Yan Yang, Jian Sun, Huibin Li, and Zongben Xu. Deep admm-net for compressive sensing mri. In *Advances in neural information processing systems*, pages 10–18, 2016.

[117] Ulugbek S Kamilov and Hassan Mansour. Learning optimal nonlinearities for iterative thresholding algorithms. *IEEE Signal Processing Letters*, 23(5):747–751, 2016.

[118] Emrah Bostan, Ulugbek S Kamilov, and Laura Waller. Learning-based image reconstruction via parallel proximal algorithm. *IEEE Signal Processing Letters*, 25(7):989–993, 2018.

[119] Ali Mousavi and Richard G Baraniuk. Learning to invert: Signal recovery via deep convolutional networks. *arXiv preprint arXiv:1701.03891*, 2017.

[120] Shanshan Wu, Alex Dimakis, Sujay Sanghavi, Felix Yu, Daniel Holtmann-Rice, Dmitry Storcheus, Afshin Rostamizadeh, and Sanjiv Kumar. Learning a compressed sensing measurement matrix via gradient unrolling. In *Proc. Int. Conf. Machine Learning*, 2019.

[121] Alexander W. Bergman, David B. Lindell, and Gordon Wetzstein. Deep Adaptive LiDAR: End-to-end Optimization of Sampling and Depth Completion at Low Sampling Rates. *Proc. IEEE ICCP*, 2020.

[122] Jie Wang, Qinhua Gao, Xiaorui Ma, Yunong Zhao, and Yuguang Fang. Learning to sense: Deep learning for wireless sensing with less training efforts. *IEEE Wireless Communications*, 2020.

[123] Cagla Deniz Bahadir, Adrian V Dalca, and Mert R Sabuncu. Learning-based optimization of the under-sampling pattern in mri. In *International Conference on Information Processing in Medical Imaging*, pages 780–792. Springer, 2019.

[124] Tomer Weiss, Ortal Senouf, Sanketh Vedula, Oleg Michailovich, Michael Zibulevsky, and Alex Bronstein. Pilot: Physics-informed learned optimized trajectories for accelerated mri. *MELBA*, pages 1–23, 2021.

[125] Hemant Kumar Aggarwal and Mathews Jacob. J-modl: Joint model-based deep learning for optimized sampling and reconstruction. *IEEE Journal of Selected Topics in Signal Processing*, 14(6):1151–1162, 2020.

[126] Vincent Sitzmann, Steven Diamond, Yifan Peng, Xiong Dun, Stephen Boyd, Wolfgang Heidrich, Felix Heide, and Gordon Wetzstein. End-to-end optimization of optics and image processing for achromatic extended depth of field and super-resolution imaging. *ACM Transactions on Graphics (TOG)*, 37(4):1–13, 2018.

[127] Julie Chang, Vincent Sitzmann, Xiong Dun, Wolfgang Heidrich, and Gordon Wetzstein. Hybrid optical-electronic convolutional neural networks with optimized diffractive optics for image classification. *Scientific reports*, 8(1):1–10, 2018.

[128] Rakib Hyder, Ken Shao, Boyu Hou, Panos Markopoulos, Ashley Prater-Bennette, and M Salman Asif. Incremental task learning with incremental rank updates. In *Proc. European Conf. Comp. Vision (ECCV)*, 2022.

[129] German I Parisi, Ronald Kemker, Jose L Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *Neural Networks*, 113:54–71, 2019.

[130] Roger Ratcliff. Connectionist models of recognition memory: constraints imposed by learning and forgetting functions. *Psychological review*, 97(2):285, 1990.

[131] Andrew M Saxe, James L McClelland, and Surya Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv preprint arXiv:1312.6120*, 2013.

[132] David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. *Advances in neural information processing systems*, 30:6467–6476, 2017.

[133] Rakib Hyder and M Salman Asif. Generative models for low-dimensional video representation and reconstruction. *IEEE Transactions on Signal Processing*, 68:1688–1701, 2020.

[134] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[135] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, 2017.

[136] Carl Vondrick, Hamed Pirsiavash, and Antonio Torralba. Generating videos with scene dynamics. In *Proc. Adv. in Neural Inf. Proc. Sys. (NIPS)*, pages 613–621, 2016.

[137] Rakib Hyder and M. Salman Asif. Generative Models for Low-Dimensional Video Representation and Reconstruction. *IEEE Transactions on Signal Processing*, 68:1688–1701, 2020.

[138] R. Hyder and M. S. Asif. Generative models for low-rank video representation and reconstruction from compressive measurements. In *IEEE International Workshop on Machine Learning for Signal Processing*, (accepted) 2019.

[139] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[140] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *Proc. Int. Conf. Learning Representations (ICLR)*, 2016.

[141] Reinhard Heckel. Regularizing linear inverse problems with convolutional neural networks. *arXiv preprint arXiv:1907.03100*, 2019.

[142] Christian Schuldt, Ivan Laptev, and Barbara Caputo. Recognizing human actions: a local svm approach. In *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, volume 3, pages 32–36. IEEE, 2004.

[143] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012.

[144] Ramesh Raskar, Amit Agrawal, and Jack Tumblin. Coded exposure photography: motion deblurring using fluttered shutter. In *ACM transactions on graphics (TOG)*, volume 25, pages 795–804. ACM, 2006.

[145] Yasunobu Hitomi, Jinwei Gu, Mohit Gupta, Tomoo Mitsunaga, and Shree K Nayar. Video from a single coded exposure photograph using a learned over-complete dictionary. In *2011 International Conference on Computer Vision*, pages 287–294. IEEE, 2011.

[146] Ashok Veeraraghavan, Dikpal Reddy, and Ramesh Raskar. Coded strobing photography: Compressive sensing of high speed periodic videos. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(4):671–686, 2010.

[147] Amit Agrawal, Mohit Gupta, Ashok Veeraraghavan, and Srinivasa G Narasimhan. Optimal coded sampling for temporal super-resolution. In *Proc. IEEE Conf. Comp. Vision and Pattern Recog. (CVPR)*, pages 599–606. IEEE, 2010.

[148] Guangming Shi, Dahua Gao, Xiaoxia Song, Xuemei Xie, Xuyang Chen, and Danhua Liu. High-resolution imaging via moving random exposure and its simulation. *IEEE Transactions on Image Processing*, 20(1):276–282, 2010.

[149] Tamara G Kolda and Brett W Bader. Tensor decompositions and applications. *SIAM review*, 51(3):455–500, 2009.

[150] D Bacciu and DP Mandic. Tensor decompositions in deep learning. In *28th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, ESANN 2020*, pages 441–450. ESANN (i6doc. com), 2020.

[151] Yuwang Ji, Qiang Wang, Xuan Li, and Jie Liu. A survey on tensor techniques and applications in machine learning. *IEEE Access*, 7:162950–162990, 2019.

[152] Nicholas D Sidiropoulos, Lieven De Lathauwer, Xiao Fu, Kejun Huang, Evangelos E Papalexakis, and Christos Faloutsos. Tensor decomposition for signal processing and machine learning. *IEEE Transactions on Signal Processing*, 65(13):3551–3582, 2017.

[153] Daniel Vlasic, Matthew Brand, Hanspeter Pfister, and Jovan Popovic. Face transfer with multilinear models. In *ACM SIGGRAPH 2006 Courses*, pages 24–es. ACM, 2006.

[154] Stuart Perry. Image and video noise: An industry perspective. In *Denoising of Photographic Images and Video*, pages 207–234. Springer, 2018.

[155] Zhou Xue, Jingyu Yang, Qionghai Dai, and Naiyao Zhang. Multi-view image denoising based on graphical model of surface patch. In *2010 3DTV-Conference: The True Vision-Capture, Transmission and Display of 3D Video*, pages 1–4. IEEE, 2010.

[156] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[157] Tatsuya Yokota and Andrzej Cichocki. Tensor completion via functional smooth component deflation. In *2016 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 2514–2518. IEEE, 2016.

[158] Yann LeCun, Fu Jie Huang, and Leon Bottou. Learning methods for generic object recognition with invariance to pose and lighting. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, volume 2, pages II–104. IEEE, 2004.

[159] Oliver Langner, Ron Dotsch, Gijsbert Bijlstra, Daniel HJ Wigboldus, Skyler T Hawk, and AD Van Knippenberg. Presentation and validation of the radboud faces database. *Cognition and emotion*, 24(8):1377–1388, 2010.

[160] Chris Burgess and Hyunjik Kim. 3d shapes dataset. https://github.com/deepmind/3dshapes-dataset/, 2018.

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

[162] Hyunjik Kim and Andriy Mnih. Disentangling by factorising. In *Proceedings of the 35th International Conference on Machine Learning*, pages 2649–2658, 2018.

[163] Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. *Iclr*, 2(5):6, 2017.

[164] Rakib Hyder, Hassan Mansour, Yanting Ma, Petros T Boufounos, and Pu Wang. A consensus equilibrium solution for deep image prior powered by red. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1380–1384. IEEE, 2021.

[165] Antoni Buades, Bartomeu Coll, and J-M Morel. A non-local algorithm for image denoising. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 2, pages 60–65. IEEE, 2005.

[166] Kostadin Dabov, Alessandro Foi, Vladimir Katkovnik, and Karen Egiazarian. Image denoising by sparse 3-d transform-domain collaborative filtering. *IEEE Transactions on image processing*, 16(8):2080–2095, 2007.

[167] Julien Mairal, Francis Bach, Jean Ponce, Guillermo Sapiro, and Andrew Zisserman. Non-local sparse models for image restoration. In *2009 IEEE 12th international conference on computer vision*, pages 2272–2279. IEEE, 2009.

[168] Michael Elad and Michal Aharon. Image denoising via sparse and redundant representations over learned dictionaries. *IEEE Transactions on Image processing*, 15(12):3736–3745, 2006.

[169] Leonid I Rudin, Stanley Osher, and Emad Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D: Nonlinear Phenomena*, 60(1-4):259–268, 1992.

[170] Stanley Osher, Martin Burger, Donald Goldfarb, Jinjun Xu, and Wotao Yin. An iterative regularization method for total variation-based image restoration. *Multiscale Modeling & Simulation*, 4(2):460–489, 2005.

[171] Yair Weiss and William T Freeman. What makes a good model of natural images? In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2007.

[172] Xiangyang Lan, Stefan Roth, Daniel Huttenlocher, and Michael J Black. Efficient belief propagation with learned higher-order markov random fields. In *European conference on computer vision*, pages 269–282. Springer, 2006.

[173] Kai Zhang, Wangmeng Zuo, Yunjin Chen, Deyu Meng, and Lei Zhang. Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising. *IEEE Transactions on Image Processing*, 26(7):3142–3155, 2017.

[174] Dong Yang and Jian Sun. Bm3d-net: A convolutional neural network for transform-domain collaborative filtering. *IEEE Signal Processing Letters*, 25(1):55–59, 2017.

[175] Weisheng Dong, Lei Zhang, Guangming Shi, and Xin Li. Nonlocally centralized sparse representation for image restoration. *IEEE transactions on Image Processing*, 22(4):1620–1630, 2012.

[176] S. Venkatakrishnan, C. Bouman, and B. Wohlberg. Plug-and-play priors for model based reconstruction. In *2013 IEEE Global Conf. on Signal and Inf. Processing*, pages 945–948. IEEE, 2013.

[177] Yaniv Romano, Michael Elad, and Peyman Milanfar. The little engine that could: Regularization by denoising (red). *SIAM Journal on Imaging Sciences*, 10(4):1804–1844, 2017.

[178] E. T. Reehorst and P. Schniter. Regularization by denoising: Clarifications and new interpretations. *IEEE Transactions on Computational Imaging*, 5(1):52–67, 2019.

[179] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Deep image prior. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9446–9454, 2018.

[180] Reinhard Heckel and Mahdi Soltanolkotabi. Denoising and regularization via exploiting the structural bias of convolutional generators. In *International Conference on Learning Representations*, 2019.

[181] Gary Mataev, Peyman Milanfar, and Michael Elad. Deepred: Deep image prior powered by red. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 0–0, 2019.

[182] Gregery T Buzzard, Stanley H Chan, Suhas Sreehari, and Charles A Bouman. Plug-and-play unplugged: Optimization-free reconstruction using consensus equilibrium. *SIAM Journal on Imaging Sciences*, 11(3):2001–2020, 2018.

[183] Md Zulfiquar Ali Bhotto, M Omair Ahmad, and MNS Swamy. An improved fast iterative shrinkage thresholding algorithm for image deblurring. *SIAM journal on imaging sciences*, 8(3):1640–1657, 2015.

[184] Praveen Kumar Pokala and Chandra Sekhar Seelamantula. Projected improved fista and application to image deblurring. In *2020 IEEE International Conference on Image Processing (ICIP)*, pages 1043–1047. IEEE, 2020.

[185] Heinz H Bauschke, Patrick L Combettes, et al. *Convex analysis and monotone operator theory in Hilbert spaces*, volume 408. Springer, 2011.

[186] Noam Yair and Tomer Michaeli. Multi-scale weighted nuclear norm image restoration. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3165–3174, 2018.

[187] D. Berthelot, T. Schumm, and L. Metz. Began: Boundary equilibrium generative adversarial networks. *arXiv preprint arXiv:1703.10717*, 2017.

[188] M.S. Asif and C. Hegde. Phase retrieval for signals in union of subspaces. In *2018 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, pages 356–359. IEEE, 2018.

[189] R. Harrison. Phase problem in crystallography. *JOSA a*, 10(5):1046–1055, 1993.

[190] Z. Chen, G. Jagatap, S. Nayer, C. Hegde, and N. Vaswani. Low rank fourier ptychography. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6538–6542, April 2018.

[191] G. Jagatap, Z. Chen, C. Hegde, and N. Vaswani. Sub-diffraction imaging using fourier ptychography and structured sparsity. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6493–6497, April 2018.

[192] Gang Wang, Georgios Giannakis, Yousef Saad, and Jie Chen. Solving most systems of random quadratic equations. In *Advances in Neural Information Processing Systems*, pages 1867–1877, 2017.

[193] David D Nolte. *Optical interferometry for biology and medicine*, volume 1. Springer Science & Business Media, 2011.

[194] Ke Wei. Solving systems of phaseless equations via kaczmarz methods: A proof of concept study. *Inverse Problems*, 31(12):125008, 2015.

[195] Zikui Cai, Rakib Hyder, and M Salman Asif. Data-driven illumination patterns for coded diffraction imaging. In *2021 IEEE International Conference on Image Processing (ICIP)*, pages 2818–2822. IEEE, 2021.

[196] James R Fienup. Reconstruction of an object from the modulus of its fourier transform. *Optics letters*, 3(1):27–29, 1978.

[197] Kishore Jaganathan, Yonina C Eldar, and Babak Hassibi. Phase retrieval: An overview of recent developments. *arXiv preprint arXiv:1510.07713*, 2015.

[198] Baurzhan Muminov and Luat T Vuong. Small-brain neural networks rapidly solve inverse problems with vortex fourier encoders. *arXiv preprint arXiv:2005.07682*, 2020.

[199] James D Watson and Francis HC Crick. Molecular structure of nucleic acids: a structure for deoxyribose nucleic acid. *Nature*, 171(4356):737–738, 1953.

[200] C Fienup and J Dainty. Phase retrieval and image reconstruction for astronomy. *Image recovery: theory and application*, 231:275, 1987.

[201] Robert A Gonsalves. Perspectives on phase retrieval and phase diversity in astronomy. In *Adaptive Optics Systems IV*, volume 9148, page 91482P. International Society for Optics and Photonics, 2014.

[202] DL Misell. A method for the solution of the phase problem in electron microscopy. *Journal of Physics D: Applied Physics*, 1973.

[203] Lei Tian, Xiao Li, Kannan Ramchandran, and Laura Waller. Multiplexed coded illumination for fourier ptychography with an led array microscope. *Biomedical optics express*, 5(7):2376–2389, 2014.

[204] Lawrence Rabiner. Fundamentals of speech recognition. *Fundamentals of speech recognition*, 1993.

[205] Radu Balan, Pete Casazza, and Dan Edidin. On signal reconstruction without phase. *Applied and Computational Harmonic Analysis*, 20(3):345–356, 2006.

[206] Kishore Jaganathan, Yonina C Eldar, and Babak Hassibi. Stft phase retrieval: Uniqueness guarantees and recovery algorithms. *IEEE Journal of selected topics in signal processing*, 10(4):770–781, 2016.

[207] John V Corbett. The pauli problem, state reconstruction and quantum-real numbers. *Reports on Mathematical Physics*, 1(57), 2006.

[208] Hans Reichenbach. *Philosophic foundations of quantum mechanics*. Courier Corporation, 1998.

[209] Li-Hao Yeh, Jonathan Dong, Jingshan Zhong, Lei Tian, Michael Chen, Gongguo Tang, Mahdi Soltanolkotabi, and Laura Waller. Experimental robustness of fourier ptychography phase retrieval algorithms. *Optics express*, 23(26):33214–33240, 2015.

[210] Sohail Bahmani and Justin Romberg. Phase retrieval meets statistical learning theory: A flexible convex relaxation. In *Artificial Intelligence and Statistics*, pages 252–260, 2017.

[211] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *Proc. Adv. in Neural Inf. Proc. Sys.*, pages 8024–8035, 2019.

[212] Gang Wang, Georgios B Giannakis, and Yonina C Eldar. Solving systems of random quadratic equations via truncated amplitude flow. *IEEE Transactions on Information Theory*, 64(2):773–794, 2017.

[213] Alessandro Foi, Mejdi Trimeche, Vladimir Katkovnik, and Karen Egiazarian. Practical poissonian-gaussian noise modeling and fitting for single-image raw-data. *IEEE Transactions on Image Processing*, 17(10):1737–1754, 2008.

[214] Qi Xu, Ming Zhang, Zonghua Gu, and Gang Pan. Overfitting remedy by sparsifying regularization on fully-connected layers of cnns. *Neurocomputing*, 328:69–74, 2019.

[215] Adrian Bulat, Jean Kossaifi, Georgios Tzimiropoulos, and Maja Pantic. Incremental multi-domain learning with network latent tensor factorization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 10470–10477, 2020.

[216] Jean Kossaifi, Aran Khanna, Zachary Lipton, Tommaso Furlanello, and Anima Anandkumar. Tensor contraction layers for parsimonious deep nets. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 26–32, 2017.

[217] Timur Garipov, Dmitry Podoprikhin, Alexander Novikov, and Dmitry Vetrov. Ultimate tensorization: compressing convolutional and fc layers alike. *arXiv preprint arXiv:1611.03214*, 2016.

[218] Tara N Sainath, Brian Kingsbury, Vikas Sindhwani, Ebru Arisoy, and Bhuvana Ramabhadran. Low-rank matrix factorization for deep neural network training with high-dimensional output targets. In *2013 IEEE international conference on acoustics, speech and signal processing*, pages 6655–6659. IEEE, 2013.

[219] Huanrui Yang, Minxue Tang, Wei Wen, Feng Yan, Daniel Hu, Ang Li, Hai Li, and Yiran Chen. Learning low-rank deep neural networks via singular vector orthogonality regularization and singular value sparsification. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pages 678–679, 2020.

[220] Daniel Povey, Gaofeng Cheng, Yiming Wang, Ke Li, Hainan Xu, Mahsa Yarmohammadi, and Sanjeev Khudanpur. Semi-orthogonal low-rank matrix factorization for deep neural networks. In *Interspeech*, pages 3743–3747, 2018.

[221] Shuang Xu, Chunxia Zhang, and Jiangshe Zhang. Bayesian deep matrix factorization network for multiple images denoising. *Neural Networks*, 123:420–428, 2020.

[222] Cho-Jui Hsieh, Kai-Yang Chiang, and Inderjit S Dhillon. Low rank modeling of signed networks. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 507–515, 2012.

[223] Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning. *Advances in Neural Information Processing Systems*, 33, 2020.

[224] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The Caltech-UCSD Birds-200-2011 Dataset. Technical Report CNS-TR-2011-001, California Institute of Technology, 2011.

[225] Lomonaco Vincenzo and Davide Maltoni. Core50: a new dataset and benchmark for continuous object recognition. *arXiv preprint arXiv:1705.03550v1*, 2017.

[226] Martín Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *ICML*, pages 214–223, 2017.

[227] Ta-Chu Kao, Kristopher T Jensen, Alberto Bernacchia, and Guillaume Hennequin. Natural continual learning: success is a journey, not (just) a destination. *Advances in Neural Information Processing Systems*, 34, 2021.

[228] Gido M Van de Ven and Andreas S Tolias. Three scenarios for continual learning. *arXiv preprint arXiv:1904.07734*, 2019.

[229] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Proc. Adv. in Neural Inf. Proc. Sys. (NIPS)*, pages 2172–2180, 2016.

[230] Charilaos Christopoulos, Athanassios Skodras, and Touradj Ebrahimi. The jpeg2000 still image coding system: an overview. *IEEE transactions on consumer electronics*, 46(4):1103–1127, 2000.

[231] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. In *Proc. Adv. in Neural Inf. Proc. Sys. (NIPS)*, pages 5767–5777, 2017.

[232] Roy R Lederman and Ronen Talmon. Learning the geometry of common latent variables using alternating-diffusion. *Applied and Computational Harmonic Analysis*, 44(3):509–536, 2018.

[233] Rongqun Lin, Yongbing Zhang, Haoqian Wang, Xingzheng Wang, and Qionghai Dai. Deep convolutional neural network for decompressed video enhancement. In *Data Compression Conference (DCC), 2016*, pages 617–617. IEEE, 2016.

[234] Pingbo Pan, Zhongwen Xu, Yi Yang, Fei Wu, and Yueting Zhuang. Hierarchical recurrent neural encoder for video representation with application to captioning. In *Proc. IEEE Conf. Comp. Vision and Pattern Recog. (CVPR)*, pages 1029–1038, 2016.

[235] Atul Puri and Alexandros Eleftheriadis. Mpeg-4: An object-based multimedia coding standard supporting mobile applications. *Mobile Networks and Applications*, 3(1):5–32, 1998.

[236] Shibani Santurkar, David Budden, and Nir Shavit. Generative compression. In *2018 Picture Coding Symposium (PCS)*, pages 258–262. IEEE, 2018.

[237] Ashish Shrivastava, Tomas Pfister, Oncel Tuzel, Joshua Susskind, Wenda Wang, and Russell Webb. Learning from simulated and unsupervised images through adversarial training. In *Proc. IEEE Conf. Comp. Vision and Pattern Recog. (CVPR)*, pages 2242–2251. IEEE, 2017.

[238] Nitish Srivastava, Elman Mansimov, and Ruslan Salakhudinov. Unsupervised learning of video representations using lstms. In *Proc. Int. Conf. Machine Learning*, pages 843–852, 2015.

[239] Gary J Sullivan, Pankaj N Topiwala, and Ajay Luthra. The h. 264/avc advanced video coding standard: Overview and introduction to the fidelity range extensions. In *Applications of Digital Image Processing XXVII*, volume 5558, pages 454–475. International Society for Optics and Photonics, 2004.

[240] Gregory K Wallace. The jpeg still picture compression standard. *IEEE transactions on consumer electronics*, 38(1):xviii–xxxiv, 1992.

[241] Federico Perazzi, Jordi Pont-Tuset, Brian McWilliams, Luc Van Gool, Markus Gross, and Alexander Sorkine-Hornung. A benchmark dataset and evaluation methodology for video object segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 724–732, 2016.

[242] D. Needell and J. Tropp. Cosamp: iterative signal recovery from incomplete and inaccurate samples. *Comm. of the ACM*, 53(12):93–100, 2010.

[243] S. Chen, D. Donoho, and M. Saunders. Atomic decomposition by basis pursuit. *SIAM review*, 43(1):129–159, 2001.

[244] A. Lacoste, T. Boquet, N. Rostamzadeh, B. Oreshki, W. Chung, and D. Krueger. Deep prior. *arXiv preprint arXiv:1712.05016*, 2017.

[245] M. Cucuringu and H. Tyagi. On denoising modulo 1 samples of a function. In *Proc. Int. Conf. Art. Intell. Stat. (AISTATS)*, 2018.

[246] E. van den Berg and M. P. Friedlander. Probing the pareto frontier for basis pursuit solutions. *SIAM J. on Sci. Computing*, 31(2):890–912, 2008.

[247] E. van den Berg and M. P. Friedlander. SPGL1: A solver for large-scale sparse reconstruction, June 2007.

[248] J. Bioucas-Dias and G. Valadão. Phase unwrapping via graph cuts. *IEEE Trans. Image Processing*, 16(3), 2007.

[249] A. Hooper and H. Zebker. Phase unwrapping in three dimensions with application to InSAR time series. *J. of the Optical Soc. of America A*, 24(9):2737, 2007.

[250] Mihai Cucuringu and Hemant Tyagi. Provably robust estimation of modulo 1 samples of a smooth function with applications to phase unwrapping. *Journal of Machine Learning Research*, 21(32), 2020.

[251] V. Shah, M. Soltani, and C. Hegde. Reconstruction from periodic nonlinearities, with applications to hdr imaging. In *Proc. Asilomar Conf. Signals, Systems, and Computers*, pages 863–867. IEEE, 2017.

[252] F. Lang, T. Plötz, and S. Roth. Robust multi-image HDR reconstruction for the modulo camera. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 10496 LNCS, pages 78–89, 2017.

[253] X. Li. Compressed sensing and matrix completion with constant proportion of corruptions. *Const. Approx.*, 37(1):73–99, 2013.

[254] L. Jacques, J. Laska, P. Boufounos, and R. Baraniuk. Robust 1-Bit compressive sensing via binary stable embeddings of sparse vectors. *IEEE Trans. Inform. Theory*, 59(4):2082–2102, 2013.

[255] J. Rhee and Y. Joo. Wide dynamic range cmos image sensor with pixel level adc. *Electron. Lett.*, 39:360–361, 2010.

[256] K. Sasagawa, T. Yamaguchi, M. Haruta, Y. Sunaga, H. Takehara, H. Takehara, T. Noda, T. Tokuda, and J. Ohta. An implantable cmos image sensor with self-reset pixels for functional brain imaging. *IEEE Trans. on Electron Devices*, 63(1):215–222, 2016.

[257] T. Yamaguchi, H. Takehara, Y. Sunaga, M. Haruta, M. Motoyama, Y. Ohta, T. Noda, K. Sasagawa, T. Tokuda, and J. Ohta. Implantable self-reset cmos image sensor and its application to hemodynamic response detection in living mouse brain. *Japanese J. of Appl. Physics*, 55(4S):04EM02, 2016.

[258] S. Kavusi and A. El Gamal. Quantitative study of high-dynamic-range image sensor architectures. In *Sensors and Camera Systems for Sci., Indust., and Digi. Photography Applications V*, volume 5301, pages 264–276. Intl. Soc. for Optics and Photonics, 2004.

[259] R. Vershynin. Introduction to the non-ptotic analysis of random matrices. *arXiv preprint arXiv:1011.3027*, 2010.

[260] J. Laska, M. Davenport, and R. Baraniuk. Exact signal recovery from sparsely corrupted measurements through the pursuit of justice. In *Proc. Asilomar Conf. Signals, Systems, and Computers*, pages 1556–1560, 2009.

[261] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proc. of the IEEE*, 86(11):2278–2324, 1998.

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

[263] Junbo Zhao, Michael Mathieu, and Yann LeCun. Energy-based generative adversarial network. *arXiv preprint arXiv:1609.03126*, 2016.

[264] Parikshit Shah and Venkat Chandrasekaran. Iterative projections for signal identification on manifolds: Global recovery guarantees. In *Proc. Allerton Conf. Communication, Control, and Computing*, pages 760–767. IEEE, 2011.

[265] Emmanuel J Candès et al. Compressive sampling. In *Proceedings of the international congress of mathematicians*, volume 3, pages 1433–1452. Madrid, Spain, 2006.

[266] Emmanuel J Candes, Justin K Romberg, and Terence Tao. Stable signal recovery from incomplete and inaccurate measurements. *Communications on pure and applied mathematics*, 59(8):1207–1223, 2006.

[267] Z. Liu, P. Luo, X. Wang, and X. Tang. Deep learning face attributes in the wild. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3730–3738, 2015.

[268] David L Donoho. De-noising by soft-thresholding. *IEEE transactions on information theory*, 41(3):613–627, 1995.

[269] Zongben Xu and Jian Sun. Image inpainting by patch propagation using patch sparsity. *IEEE transactions on image processing*, 19(5):1153–1165, 2010.

[270] Weisheng Dong, Lei Zhang, Guangming Shi, and Xiaolin Wu. Image deblurring and super-resolution by adaptive sparse domain selection and adaptive regularization. *IEEE Transactions on Image Processing*, 20(7):1838–1857, 2011.

[271] Michal Aharon, Michael Elad, and Alfred Bruckstein. *rmk*-svd: An algorithm for designing overcomplete dictionaries for sparse representation. *IEEE Transactions on signal processing*, 54(11):4311–4322, 2006.

[272] Antonin Chambolle. An algorithm for total variation minimization and applications. *Journal of Mathematical imaging and vision*, 20(1):89–97, 2004.

[273] Tony F Chan, Jianhong Shen, and Hao-Min Zhou. Total variation wavelet inpainting. *Journal of Mathematical imaging and Vision*, 25(1):107–125, 2006.

[274] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[275] Kuldeep Kulkarni, Suhas Lohit, Pavan Turaga, Ronan Kerviche, and Amit Ashok. Reconnet: Non-iterative reconstruction of images from compressively sensed measurements. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 449–458, 2016.

[276] Ali Mousavi, Ankit B Patel, and Richard G Baraniuk. A deep learning approach to structured signal recovery. In *Communication, Control, and Computing (Allerton), 2015 53rd Annual Allerton Conference on*, pages 1336–1343. IEEE, 2015.

[277] Li Xu, Jimmy SJ Ren, Ce Liu, and Jiaya Jia. Deep convolutional neural network for image deconvolution. In *Advances in Neural Information Processing Systems*, pages 1790–1798, 2014.

[278] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017.

[279] Raymond Yeh, Chen Chen, Teck Yian Lim, Mark Hasegawa-Johnson, and Minh N Do. Semantic image inpainting with perceptual and contextual losses. *arXiv preprint arXiv:1607.07539*, 2016.

[280] C. Dong, C.C. Loy, K. He, and X. Tang. Image super-resolution using deep convolutional networks. *IEEE transactions on pattern analysis and machine intelligence*, 38(2):295–307, 2016.

[281] Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. *arXiv preprint arXiv:1609.04802*, 2016.

[282] Brendan Kelly, Thomas P Matthews, and Mark A Anastasio. Deep learning-guided image reconstruction from incomplete data. *arXiv preprint arXiv:1709.00584*, 2017.

[283] JH Rick Chang, Chun-Liang Li, Barnabas Poczos, BVK Vijaya Kumar, and Aswin C Sankaranarayanan. One network to solve them all–solving linear inverse problems using deep projection models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5888–5897, 2017.

[284] Jiwon Kim, Jung Kwon Lee, and Kyoung Mu Lee. Accurate image super-resolution using very deep convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1646–1654, 2016.

[285] A. Bhandari, F. Krahmer, and R. Raskar. On unlimited sampling. *Proc. Sampling Theory and Applications (SampTA)*, pages 31–35, 2017.

[286] H. Zhao, B. Shi, C. Fernandez-Cull, S. Yeung, and R. Raskar. Unbounded high dynamic range photography using a modulo camera. In *Intl. Conf. on Comp. Photography (ICCP)*, 2015.

[287] J. Bioucas-Dias and G. Valadao. Phase unwrapping via graph cuts. *IEEE Trans. Image Proc.*, 16(3):698–709, 2007.

[288] U. Kamilov, V. Goyal, and S. Rangan. Message-passing de-quantization with applications to compressed sensing. *IEEE Trans. Sig. Proc.*, 60(12):6270–6281, 2012.

[289] J. Laurent and C. Valerio. Time for dithering: fast and quantized random embeddings via the restricted isometry property. *CoRR*, abs/1607.00816, 2016.

[290] M. Soltani and C. Hegde. Stable recovery of sparse vectors from random sinusoidal feature maps. *arXiv preprint arXiv:1701.06607*, 2017.

[291] R. Baraniuk, V. Cevher, M. Duarte, and C. Hegde. Model-based compressive sensing. *IEEE Trans. Inform. Theory*, 56(4):1982–2001, 2010.

[292] B. Recht, M. Fazel, and P. Parrilo. Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization. *SIAM review*, 52(3):471–501, 2010.

[293] M. Soltani and C. Hegde. Fast algorithms for demixing sparse signals from nonlinear observations. *arXiv preprint arXiv:1608.01234*, 2016.

[294] Salman Asif, Ali Ayremlou, Aswin Sankaranarayanan, Ashok Veeraraghavan, and Richard Baraniuk. Flatcam: Thin, lensless cameras using coded aperture and computation. *IEEE Trans. on Comp. Imaging*, 2017.

[295] J. Laurent, D. Hammond, and J. Fadili. Dequantizing compressed sensing: When oversampling and non-gaussian constraints combine. *IEEE Trans. Inform. Theory*, 57(1):559–571, 2011.

[296] T. Tirer and R. Giryes. Image restoration by iterative denoising and backward projections. *IEEE Trans. Image Processing*, 28(3):1220–1234, 2019.

[297] K. Zhang, W. Zuo, S. Gu, and L. Zhang. Learning deep cnn denoiser prior for image restoration. In *Proc. IEEE Conf. Comp. Vision and Pattern Recog. (CVPR)*, pages 3929–3938, 2017.

[298] Paul Hand and Vladislav Voroninski. Compressed sensing from phaseless gaussian measurements via linear programming in the natural parameter space. *arXiv preprint arXiv:1611.05985*, 2016.

[299] Tatiana Latychevskaia. Iterative phase retrieval for digital holography. *JOSA A*, 36(12):D31–D40, 2019.

[300] Milen Shishkov, Brett Eugene Bouma, and Guillermo J Tearney. System and method for optical coherence imaging, Apr. 29 2008. US Patent 7,366,376.

[301] Heinz H Bauschke, Patrick L Combettes, and D Russell Luke. Phase retrieval, error reduction algorithm, and fienup variants: a view from convex optimization. *JOSA A*, 19(7):1334–1345, 2002.

[302] Jian Sun, Huibin Li, and Zongben Xu. Deep ADMM-Net for compressive sensing MRI. In *Proc. Adv. in Neural Inf. Proc. Sys.*, pages 10–18, 2016.

[303] Radu Balan. On signal reconstruction from its spectrogram. In *2010 44th Annual Conference on Information Sciences and Systems (CISS)*, pages 1–4. IEEE, 2010.

[304] Christopher A Metzler, Felix Heide, Prasana Rangarajan, Muralidhar Madabhushi Balaji, Aparna Viswanath, Ashok Veeraraghavan, and Richard G Baraniuk. Deep-inverse correlography: towards real-time high-resolution non-line-of-sight imaging. *Optica*, 7(1):63–71, 2020.

[305] Vishal Monga, Yuelong Li, and Yonina C Eldar. Algorithm unrolling: Interpretable, efficient deep learning for signal and image processing. *arXiv preprint arXiv:1912.10557*, 2019.

[306] Dong Liang, Jing Cheng, Ziwen Ke, and Leslie Ying. Deep mri reconstruction: Unrolled optimization algorithms meet neural networks. *arXiv preprint arXiv:1907.11711*, 2019.

[307] Maryam Fazel, E Candes, Benjamin Recht, and P Parrilo. Compressed sensing and robust recovery of low rank matrices. In *2008 42nd Asilomar Conference on Signals, Systems and Computers*, pages 1043–1047. IEEE, 2008.

[308] Tom Goldstein and Christoph Studer. Phasemax: Convex phase retrieval via basis pursuit. *IEEE Transactions on Information Theory*, 64(4):2675–2689, 2018.

[309] Ramina Ghods, Andrew S Lan, Tom Goldstein, and Christoph Studer. Phaselin: Linear phase retrieval. In *2018 52nd Annual Conference on Information Sciences and Systems (CISS)*, pages 1–6. IEEE, 2018.

[310] Emrah Bostan, Reinhard Heckel, Michael Chen, Michael Kellman, and Laura Waller. Deep phase decoder: self-calibrating phase microscopy with an untrained deep neural network. *Optica*, 7(6):559–562, 2020.

[311] Yinhao Ren, Zhe Zhu, Yingzhou Li, and Joseph Lo. Mask Embedding in Conditional GAN for Guided Synthesis of High Resolution Images. *arXiv preprint arXiv:1907.01710*, 2019.

[312] Jianmin Bao, Dong Chen, Fang Wen, Houqiang Li, and Gang Hua. CVAE-GAN: Fine-Grained Image Generation through Asymmetric Training. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2745–2754, 2017.

[313] Tao Xu, Pengchuan Zhang, Qiuyuan Huang, Han Zhang, Zhe Gan, Xiaolei Huang, and Xiaodong He. ATTNGAN: Fine-Grained Text-to-Image Generation with Attentional Generative Adversarial Networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1316–1324, 2018.

[314] Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A Efros. Context Encoders: Feature Learning by Inpainting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2536–2544, 2016.

[315] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-Image Translation with Conditional Adversarial Networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1125–1134, 2017.

[316] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2223–2232, 2017.

[317] Ryan Dahl, Mohammad Norouzi, and Jonathon Shlens. Pixel Recursive Super Resolution. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5439–5448, 2017.

[318] Fujun Luan, Sylvain Paris, Eli Shechtman, and Kavita Bala. Deep Painterly Harmonization. In *Computer Graphics Forum*, volume 37, pages 95–106. Wiley Online Library, 2018.

[319] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Guilin Liu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. Video-to-Video Synthesis. In *Advances in Neural Information Processing Systems*, pages 1144–1156, 2018.

[320] Xiaodan Liang, Lisa Lee, Wei Dai, and Eric P Xing. Dual-Motion GAN for Future-Flow Embedded Video Prediction. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1744–1752, 2017.

[321] Joost van Amersfoort, Wenzhe Shi, Alejandro Acosta, Francisco Massa, Johannes Totz, Zehan Wang, and Jose Caballero. Frame Interpolation with Multi-scale Deep Loss Functions and Generative Adversarial Networks. *arXiv preprint arXiv:1711.06045*, 2017.

[322] Piotr Bojanowski, Armand Joulin, David Lopez-Pas, and Arthur Szlam. Optimizing the Latent Space of Generative Networks. In *Proceedings of the International Conference on Machine Learning*, pages 599–608, 2018.

[323] Yedid Hoshen, Ke Li, and Jitendra Malik. Non-Adversarial Image Synthesis with Generative Latent Nearest Neighbors. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5811–5819, 2019.

[324] Ke Li and Jitendra Malik. Implicit Maximum Likelihood Estimation. *arXiv preprint arXiv:1809.09087*, 2018.

[325] Sergey Tulyakov, Ming-Yu Liu, Xiaodong Yang, and Jan Kautz. MOCOGAN: Decomposing Motion and Content for Video Generation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1526–1535, 2018.

[326] Jiawei He, Andreas Lehrmann, Joseph Marino, Greg Mori, and Leonid Sigal. Probabilistic Video Generation using Holistic Attribute Control. In *Proceedings of the European Conference on Computer Vision*, pages 452–467, 2018.

[327] Masaki Saito, Eiichi Matsumoto, and Shunta Saito. Temporal Generative Adversarial Nets with Singular Value Clipping. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2830–2839, 2017.

[328] Tianfan Xue, Jiajun Wu, Katherine Bouman, and William Freeman. Visual Dynamics: Stochastic Future Generation via Layered Cross-Convolutional Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2018.

[329] Simon Niklaus, Long Mai, and Feng Liu. Video Frame Interpolation via Adaptive Separable Convolution. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 261–270, 2017.

[330] Piotr Bojanowski and Armand Joulin. Unsupervised Learning by Predicting Noise. In *Proceedings of the International Conference on Machine Learning*, pages 517–526, 2017.

[331] Christoph Feichtenhofer, Axel Pinz, and Richard Wildes. Spatiotemporal Residual Networks for Video Action Recognition. In *Advances in Neural Information Processing Systems*, pages 3468–3476, 2016.

[332] Christoph Feichtenhofer, Axel Pinz, and Richard P Wildes. Spatiotemporal Multiplier Networks for Video Action Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4768–4777, 2017.

[333] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved Techniques for Training GANs. In *Advances in Neural Information Processing Systems*, pages 2234–2242, 2016.

[334] Mathieu Aubry, Daniel Maturana, Alexei A Efros, Bryan C Russell, and Josef Sivic. Seeing 3D Chairs: Exemplar part-based 2D-3D Alignment using a Large Dataset of CAD Models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3762–3769, 2014.

[335] Lena Gorelick, Moshe Blank, Eli Shechtman, Michal Irani, and Ronen Basri. Actions as Space-Time Shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(12):2247–2253, 2007.

[336] Mehdi Mirza and Simon Osindero. Conditional Generative Adversarial Networks. *arXiv preprint arXiv:1411.1784*, 2014.

[337] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[338] Haibin Ling and Kazunori Okada. Diffusion Distance for Histogram Comparison. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 246–253, 2006.

[339] Mario Lucic, Karol Kurach, Marcin Michalski, Sylvain Gelly, and Olivier Bousquet. Are GANs created equal? A Large-scale Study. In *Advances in Neural Information Processing Systems*, pages 700–709, 2018.

[340] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the Inception Architecture for Computer Vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2818–2826, 2016.

[341] Karen Simonyan and Andrew Zisserman. Very Deep Convolutional Networks for Large-scale Image Recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[342] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. ImageNet: Large-Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[343] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual Losses for Real-time Style Transfer and Super-Resolution. In *European Conference on Computer Vision*, pages 694–711. Springer, 2016.

[344] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning Spatiotemporal Features with 3D Convolutional Networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4489–4497, 2015.

[345] Yong-Hoon Kwon and Min-Gyu Park. Predicting Future Frames using Retrospective Cycle-GAN. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1811–1820, 2019.

[346] William Lotter, Gabriel Kreiman, and David Cox. Deep Predictive Coding Networks for Video Prediction and Unsupervised Learning. *arXiv preprint arXiv:1605.08104*, 2016.

[347] Jerry Li, Aleksander Madry, John Peebles, and Ludwig Schmidt. Towards Understanding the Dynamics of Generative Adversarial Networks. *arXiv preprint arXiv:1706.09884*, 2017.

[348] Shane Barratt and Rishi Sharma. A Note on the Inception Score. *arXiv preprint arXiv:1801.01973*, 2018.

[349] Xu Cheng, Cameron Dale, and Jiangchuan Liu. Statistics and Social Network of YouTube Videos. In *Proceedings of the International Workshop on Quality of Service*, pages 229–238, 2008.

[350] Tsun-Hsuan Wang, Yen-Chi Cheng, Chieh Hubert Lin, Hwann-Tzong Chen, and Min Sun. Point-to-Point Video Generation. In *Proceedings of the IEEE International Conference on Computer Vision*, 2019.

[351] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic Differentiation in PyTorch. In *NIPS AutoDiff Workshop*, 2017.

[352] Pierre Sermanet, Corey Lynch, Yevgen Chebotar, Jasmine Hsu, Eric Jang, Stefan Schaal, Sergey Levine, and Google Brain. Time-Contrastive Networks: Self-Supervised Learning from Video. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1134–1141, 2018.

[353] Florian Schroff, Dmitry Kalenichenko, and James Philbin. FACENET: A Unified Embedding for Face Recognition and Clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 815–823, 2015.

[354] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The Elements of Statistical Learning*, volume 1. Springer Series in Statistics New York, 2001.

[355] L Theis, A van den Oord, and M Bethge. A Note on the Evaluation of Generative Models. In *Proceedings of the International Conference on Learning Representations*, pages 1–10, 2016.

[356] Hang Zhao, Orazio Gallo, Iuri Frosio, and Jan Kautz. Loss Functions for Neural Networks for Image Processing. *arXiv preprint arXiv:1511.08861*, 2015.

[357] Sebastian Ruder. An Overview of Gradient Descent Optimization Algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

[358] Aliaksandr Siarohin, Stéphane Lathuilière, Sergey Tulyakov, Elisa Ricci, and Nicu Sebe. Animating Arbitrary Objects via Deep Motion Transfer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2377–2386, 2019.

[359] Tian Qi Chen, Xuechen Li, Roger B Grosse, and David K Duvenaud. Isolating sources of disentanglement in variational autoencoders. In *Advances in Neural Information Processing Systems*, pages 2610–2620, 2018.

[360] Elliot Creager, David Madras, Joern-Henrik Jacobsen, Marissa Weis, Kevin Swersky, Toniann Pitassi, and Richard Zemel. Flexibly fair representation learning by disentanglement. In *International Conference on Machine Learning*, pages 1436–1445, 2019.

[361] Diane Bouchacourt, Ryota Tomioka, and Sebastian Nowozin. Multi-level variational autoencoder: Learning disentangled representations from grouped observations. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[362] Xianxu Hou, Linlin Shen, Ke Sun, and Guoping Qiu. Deep feature consistent variational autoencoder. In *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1133–1141. IEEE, 2017.

[363] Ivan V Oseledets. Tensor-train decomposition. *SIAM Journal on Scientific Computing*, 33(5):2295–2317, 2011.

[364] Tatsuya Yokota and Andrzej Cichocki. Tensor completion via functional smooth component deflation. In *2016 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 2514–2518. IEEE, 2016.

[365] Kenan E Ak, Joo Hwee Lim, Jo Yew Tham, and Ashraf A Kassim. Attribute manipulation generative adversarial networks for fashion images. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 10541–10550, 2019.

[366] Abhishek Aich, Akash Gupta, Rameswar Panda, Rakib Hyder, M Salman Asif, and Amit K Roy-Chowdhury. Non-adversarial video synthesis with learned priors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6090–6099, 2020.

[367] Yujun Shen, Jinjin Gu, Xiaoou Tang, and Bolei Zhou. Interpreting the latent space of gans for semantic face editing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9243–9252, 2020.

[368] Yu Deng, Jiaolong Yang, Dong Chen, Fang Wen, and Xin Tong. Disentangled and controllable face image generation via 3d imitative-contrastive learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5154–5163, 2020.

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

[370] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. In *International Conference on Learning Representations*, 2018.

[371] Thu Nguyen-Phuoc, Chuan Li, Lucas Theis, Christian Richardt, and Yong-Liang Yang. Hologan: Unsupervised learning of 3d representations from natural images. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 7588–7597, 2019.

[372] Luan Tran, Xi Yin, and Xiaoming Liu. Disentangled representation learning gan for pose-invariant face recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1415–1424, 2017.

[373] Abhishek Kumar, Prasanna Sattigeri, and Avinash Balakrishnan. Variational inference of disentangled latent concepts from unlabeled observations. In *International Conference on Learning Representations*, 2018.

[374] Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models. In *Advances in neural information processing systems*, pages 3483–3491, 2015.

[375] Tejas D Kulkarni, William F Whitney, Pushmeet Kohli, and Josh Tenenbaum. Deep convolutional inverse graphics network. In *Advances in neural information processing systems*, pages 2539–2547, 2015.

[376] Narayanaswamy Siddharth, Brooks Paige, Jan-Willem Van de Meent, Alban Desmaison, Noah Goodman, Pushmeet Kohli, Frank Wood, and Philip Torr. Learning disentangled representations with semi-supervised deep generative models. In *Advances in Neural Information Processing Systems*, pages 5925–5935, 2017.

[377] Michael F Mathieu, Junbo Jake Zhao, Junbo Zhao, Aditya Ramesh, Pablo Sprechmann, and Yann LeCun. Disentangling factors of variation in deep representation using adversarial training. In *Advances in neural information processing systems*, pages 5040–5048, 2016.

[378] Muhammad Waleed Gondal, Manuel Wuthrich, Djordje Miladinovic, Francesco Locatello, Martin Breidt, Valentin Volchkov, Joel Akpo, Olivier Bachem, Bernhard Scholkopf, and Stefan Bauer. On the transfer of inductive bias from simulation to the real world: a new disentanglement dataset. In *Advances in Neural Information Processing Systems*, pages 15740–15751, 2019.

[379] Moshe Blank, Lena Gorelick, Eli Shechtman, Michal Irani, and Ronen Basri. Actions as space-time shapes. In *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, volume 2, pages 1395–1402. IEEE, 2005.

[380] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. In *International Conference on Learning Representations*, 2018.

[381] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in neural information processing systems*, pages 6626–6637, 2017.

[382] Xin Geng, Kate Smith-Miles, Zhi-Hua Zhou, and Liang Wang. Face image modeling by multilinear subspace analysis with missing values. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 41(3):881–892, 2010.

[383] Emily L. Denton, Soumith Chintala, Arthur Szlam, and Rob Fergus. Deep generative image models using a laplacian pyramid of adversarial networks. In *NIPS*, 2015.

[384] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. Image-to-image translation with conditional adversarial networks. *Proc. IEEE Conf. Comp. Vision and Pattern Recog. (CVPR)*, pages 5967–5976, 2016.

[385] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *ArXiv*, abs/1411.1784, 2014.

[386] M Salman Asif, Felix Fernandes, and Justin Romberg. Low-complexity video compression and compressive sensing. In *2013 Asilomar Conference on Signals, Systems and Computers*, pages 579–583. IEEE, 2013.

[387] Michael Iliadis, Leonidas Spinoulas, and Aggelos K. Katsaggelos. Deep fully-connected networks for video compressive sensing. *ArXiv*, abs/1603.04930, 2016.

[388] Angshul Majumdar and Aditay Tripathi. Asymmetric stacked autoencoder. *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 911–918, 2017.

[389] Antonia Creswell and Anil Anthony Bharath. Denoising adversarial autoencoders. *IEEE transactions on neural networks and learning systems*, 30(4):968–984, 2018.

[390] Yong Li, Wenrui Dai, Junni Zou, Hongkai Xiong, and Yuan F Zheng. Structured sparse representation with union of data-driven linear and multilinear subspaces model for compressive video sampling. *IEEE Transactions on Signal Processing*, 65(19):5062–5077, 2017.

[391] Wenrui Dai, Yong Li, Junni Zou, Hongkai Xiong, and Yuan F Zheng. Fully decomposable compressive sampling with joint optimization for multidimensional sparse representation. *IEEE Transactions on Signal Processing*, 66(3):603–616, 2017.

[392] João FC Mota, Nikos Deligiannis, Aswin C Sankaranarayanan, Volkan Cevher, and Miguel RD Rodrigues. Adaptive-rate reconstruction of time-varying signals with application in compressive foreground extraction. *IEEE Transactions on Signal Processing*, 64(14):3651–3666, 2016.

[393] C. A. Metzler, A. Maleki, and R. G. Baraniuk. From denoising to compressed sensing. *IEEE Transactions on Information Theory*, 62(9):5117–5144, 2016.