# Diffusion Models for Large-Scale and 3D Image Reconstruction Problems

by

Jason Hu

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Electrical and Computer Engineering)
in the University of Michigan
2026

Doctoral Committee:

Professor Jeffrey A. Fessler, Co-Chair
Assistant Professor Liyue Shen, Co-Chair
Professor Douglas Noll
Assistant Professor Qing Qu

Jason Hu

jashu@umich.edu

ORCID iD: 0000-0001-5709-1909

© Jason Hu 2026

# ACKNOWLEDGMENTS

It would have been impossible for me to have reached this day without the incredible support of many people around me. Their unwavering guidance, encouragement, and insights have impacted the way I think, my goals, and research, and played a significant role in the completion of my PhD thesis.

First of all, I would like to extend my thanks to my advisors Jeff Fessler and Liyue Shen for their continued and supportive mentorship as well as wide domain expertise that have shaped both the direction of my research and my development as an independent thinker. Your constructive criticism and insightful discussions consistently challenged me to refine my ideas, approach problems with greater clarity and rigor, and ultimately helped me develop into the researcher I am today.

Additionally, I would like to thank my fellow researchers and collaborators in both of my research groups. Your discussions, feedback, and shared curiosity created an environment that made tackling difficult problems engaging and rewarding. Thank you for the late night debugging and paper writing sessions as well as coming to celebrate my successes and navigating this journey together. Your support made the day-to-day process of research far more enjoyable.

Finally, my heartfelt gratitude goes out to my friends and family for their constant encouragement, patience, and understanding throughout this journey. Your emotional support was a source of stability and perspective that made it possible for me to persevere through the challenges along the way.

# TABLE OF CONTENTS

**Acknowledgments** . . . . . **ii**

**List of Figures** . . . . . **vi**

**List of Tables** . . . . . **xii**

**Abstract** . . . . . **xv**

**Chapter**

**1 Introduction** . . . . . **1**

1.1 Contributions . . . . . 2
1.2 Outline . . . . . 4

**2 Background** . . . . . **7**

2.1 Inverse Problems . . . . . 7
2.1.1 Phase Retrieval . . . . . 8
2.1.2 CT Reconstruction . . . . . 9
2.1.3 MRI Reconstruction . . . . . 10
2.2 Diffusion Models . . . . . 10
2.2.1 Diffusion Inverse Solvers . . . . . 11
2.2.2 Large-scale Diffusion Models . . . . . 12
2.2.3 3D Diffusion Models . . . . . 13
2.3 Distribution Mismatch Problems . . . . . 13

**3 Accelerated Wirtinger Flow with Score-based Image Priors for Holographic Phase Retrieval in Poisson-Gaussian Noise Conditions** . . . . . **16**

3.1 Motivation . . . . . 16
3.1.1 Poisson-Gaussian Log-likelihood for PR . . . . . 18
3.1.2 Accelerated Wirtinger Flow with Score-based Image Prior . . . . 20
3.2 Experiment . . . . . 22
3.2.1 Experiment Settings . . . . . 22
3.2.2 Results . . . . . 24
3.3 Discussion . . . . . 30
3.4 Conclusion . . . . . 32

**4 Learning Image Priors through Patch-based Diffusion Models for Solving Inverse Problems** . . . . . **34**

4.1 Motivation . . . . . 34

4.2 Methods . . . 37
4.2.1 Patch-wise training . . . 38
4.2.2 Sampling and reconstruction . . . 39
4.3 A Unified Theory of Patch-based Diffusion Models . . . 41
4.3.1 Motivation . . . 41
4.3.2 Methods . . . 41
4.3.3 Examples . . . 46
4.4 Experiments . . . 48
4.5 Conclusion . . . 54

**5 DiffusionBlend: Learning 3D Image Prior through Position-aware Diffusion Score Blending for 3D Computed Tomography Reconstruction . . . 55**

5.1 Motivation . . . 55
5.2 Methods . . . 57
5.3 Experiments . . . 61
5.4 Conclusion . . . 67

**6 Local Patches Meet Global Context: Scalable 3D Diffusion Priors for Computed Tomography Reconstruction . . . 68**

6.1 Motivation . . . 68
6.2 Methods . . . 71
6.2.1 Global-Aware Patch Prior . . . 71
6.2.2 Sampling and Reconstruction Algorithm . . . 74
6.3 Results . . . 76
6.3.1 Experimental Setup . . . 76
6.3.2 Unconditional 3D Image Generation . . . 77
6.3.3 Solving Inverse Problems . . . 77
6.3.4 Ablation studies . . . 78
6.4 Conclusion . . . 81

**7 Test-Time Adaptation Improves Inverse Problem Solving with Patch-Based Diffusion Models . . . 82**

7.1 Motivation . . . 82
7.2 Methods . . . 84
7.2.1 Patch-based prior . . . 84
7.2.2 Diffusion self-supervision . . . 86
7.2.3 Patch-based training . . . 87
7.2.4 Theoretical Analysis . . . 88
7.3 Experiments . . . 90
7.4 Conclusion . . . 98

**8 SPAR: Refine a Single Pretrained Diffusion Model to Solve Inverse Problems in Many Modalities . . . 100**

8.1 Motivation . . . 100
8.2 Methods . . . 103
8.2.1 Patch-based Prior . . . 103

8.2.2 Model Pretraining . . . 104
8.2.3 Self-Supervised Model Finetuning . . . 105
8.2.4 Theoretical Analysis of Majorizer Loss . . . 107
8.3 Experiments . . . 110
8.4 Conclusion . . . 116
**9 Future Work . . . 117**
9.1 Improving Existing Works . . . 117
9.1.1 Generalizing Patch-Based Diffusion Models . . . 117
9.1.2 Test-time Adaptation for 3D Problems . . . 118
9.1.3 Diffusion Bridge Network Refining . . . 119
9.1.4 Provable Posterior Sampling . . . 120
9.2 Leveraging Video Diffusion Models . . . 120
**Appendices . . . 122**
**Bibliography . . . 185**

# LIST OF FIGURES


1.1 Dependency flowchart for the chapters of this thesis. . . . . 6

3.1 Illustration of Poisson and Gaussian noise statistics in Fourier transform holographic phase retrieval. . . . . 17

3.2 Reconstructed images by unregularized methods (Gaussian, Gaussian-Amplitude, Poisson and Poisson-Gaussian) on Histopathology dataset [2], celebA dataset [125] and CT-density dataset. The bottom left/right subfigures correspond to the zoomed in area and the error map for each image. We used $\alpha = 0.035$ and $\sigma = 1$. . . . . 23

3.3 Reconstructed images on dataset [2]. The bottom left/right subfigures correspond to the zoomed in area and the error map for each image. We used $\alpha = 0.02$ and $\sigma = 1$. . . . . 24

3.4 Reconstructed images on celebA dataset [125]. The bottom left/right subfigures correspond to the zoomed in area and the error map for each image. We used $\alpha = 0.035$ and $\sigma = 1$. . . . . 25

3.5 Reconstructed images on CT-density dataset. The bottom left/right subfigures correspond to the zoomed in area and the error map for each image. We used $\alpha = 0.035$ and $\sigma = 1$. . . . . 25

3.6 Reconstructed images by Gaussian, Poisson and Poisson-Gaussian log-likelihood model with AWFS image prior. Tested on Histopathology dataset [2], celebA dataset [125] and CT-density dataset. The bottom left/right subfigures correspond to the zoomed in area and the error map for each image. $\alpha$ and $\sigma$ were set to 0.025 and 1, respectively. . . . . 26

3.7 Comparison of SSIM and NRMSE varying scaling factor $\alpha \in [0.02, 0.035]$ and STD of Gaussian noise $\sigma \in [0.25, 1.5]$ defined in (3.1). . . . . 27

3.8 Reconstructed images by DOLPH [158] and our proposed AWFS method under different $\sigma$ values, for $\alpha = 0.02$. . . . . 29

3.9 Comparing AWF vs. WF with NRMSE vs. number of iterations under different noise levels. The curves and shadows represent the mean and standard deviation, respectively. . . . . 30

3.10 Reconstructed images by the unregularized Poisson method (the second column) as well as with the AWFS method for different scaling factors $\alpha$ (third to fifth columns). The top and bottom rows show reconstructions from different measurement realizations. . . . . 31

4.1 Training the proposed Patch Diffusion Inverse Solver (PaDIS) method. Different sized patches are used in each training iteration. The X and Y position arrays have the same size as the patch and consist of the normalized X and Y coordinates of each pixel of the patch. . . . . 35
4.2 Overview of reconstruction process for the proposed Patch Diffusion Inverse Solver (PaDIS) method. Starting at $t = T$, at each iteration we choose a random partition of the zero-padded image and use the neural network trained on patches to get the score function of the entire image. Due to the shifting patch locations, the output image has no boundary artifacts. . . . . 35
4.3 Schematic for zero padding and partitioning image into patches . . . . . 38
4.4 Unconditional generation of CT images. Top row: generation with a network trained on whole image; middle row: patch-only version of [183]; bottom row: proposed PaDIS method. . . . . 49
4.5 Results of CT reconstruction. 60 views are used for the top two rows, 20 views are used for the bottom two rows. To better show contrast between organs, we use modified Hounsfield units (HU) in the top figure, while we use the same scale the images were trained on in the bottom figure. . . . . 52
4.6 Results of deblurring with Gaussian noise ($\sigma = 0.01$). . . . . 53
4.7 Results of superresolution with Gaussian noise ($\sigma = 0.01$). . . . . 53
4.8 Results of 60 view CT reconstruction on $512 \times 512$ images using modified HU units. . . . . 54

5.1 Overview of DiffusionBlend++ compared to previous 3D image reconstruction works. Previous work used a hand-crafted TV term to "regularize" adjacent slices, whereas the proposed approach uses learned diffusion score blending between groups of slices. Here $i$ is the slice index, and $t$ is the reconstruction iteration. . . . . 56
5.2 Overview of slice blending process during reconstruction for DiffusionBlend++. At each iteration, we partition the slices of the volume in a different way; slices of the same color are inputted into the network independently. Positional encoding (PE) is also inputted to the network as information about the separation between the slices. . . . . 57
5.3 Results of CT reconstruction with 4 views on AAPM dataset, axial view. . . . 62
5.4 Results of DiffusionBlend++ reconstruction with multiple views on AAPM dataset, axial view. . . . . 62
5.5 Results of CT reconstruction with 8 views on AAPM dataset, coronal view. DiffusionPatch refers to Algorithm 1 with the same partition for every timestsep, and DiffusionBlend+ refers to Algorithm 1 only with partitions of adjacency slices. . . . . 66

6.1 Training the 3D patch diffusion model. Noisy 3D patch $\boldsymbol{G}_c\boldsymbol{x}_t$ (local details), downsampled volume $\boldsymbol{D}\boldsymbol{x}_t$ (global context), and positional encoding are concatenated as inputs to the denoiser network $D_\theta$ in each training iteration, to predict the noise $\boldsymbol{\epsilon}_\theta$ in the input noisy 3D patch. Positional encoding consists of voxel-based coordinates in $x, y, z$ that are normalized to $[-1, 1]$. . . . . 69

6.2 Schematic illustration for zero padding and partitioning image into 3D patches. Each index $i$ represents one of $P^3$ possible ways to choose a patch offset tuple. 71
6.3 Unconditional 3D image generation results using the LIDC-IDRI prior. The top row shows axial, coronal, and sagittal slices from a generated volume, and the bottom row shows the corresponding slices from its nearest-neighbor volume in the training dataset. The slice indices for the axial, coronal, and sagittal views are $[30, 80, 130, 180, 230]$, $[70, 100, 130, 160, 190]$, and $[60, 130, 160, 190, 210]$, respectively . . . . . 77
6.4 Results of our proposed method and comparison methods for 20 view CT recon on LIDC $256 \times 256 \times 256$ dataset. Images are shown in modified Hounsfield units. The top row shows the axial slice and the bottom row shows the sagittal slice from the reconstructed volume. . . . . 79

7.1 Schematic for zero padding and partitioning image into patches. Each index $i$ represents one of $P^2$ possible ways to choose a patch offset tuple. . . . . 85
7.2 Dataless setting: Results of 60 view CT reconstruction using self supervised (SS) loss. The display uses modified HU units to show more contrast between organs. . . . . 88
7.3 Dataless setting: Results of deblurring using test-time adaptation via the self supervised (SS) loss and several comparison methods. . . . . 91
7.4 Results of 60 view reconstruction on $512 \times 512$ images in dataless setting displayed in Hounsfield units. . . . . 95
7.5 Results of deblurring on $512 \times 512$ images in dataless setting. . . . . 95
7.6 Comparison of PSNR between patch-based model and whole-image model for overfitting in small dataset setting. . . . . 97
7.7 Comparison of SSIM between patch-based model and whole-image model for overfitting in small dataset setting. . . . . 97
7.8 **Small dataset setting**: Results of inverse problem solving. Top row is 60 view CT recon, middle row is deblurring, and bottom row is superresolution. 97
7.9 Unconditional generation of CT images from networks fine-tuned in the small dataset setting. Top two rows were generated with the whole image model; bottom two rows were generated with the patch-based model. . . . . 98

8.1 Comparison of traditional domain-specific diffusion models (DM) versus the proposed SPAR method. Left: previous works require separate pre-training for each domain/modality (CT, MRI, CelebA, Places, Dogs). Right: The proposed method enables domain adaptation at testing time through self-supervised learning, avoiding the need for domain-specific pre-training. . . . . 101
8.2 Schematic illustration for zero padding and partitioning image into patches. Each index $i$ represents one of $P^2$ possible ways to choose a patch offset tuple. 103
8.3 Results of our proposed method and comparison methods for various inverse problems. MRI images are complex-valued so we show the pixel magnitudes. 112
8.4 PSNR over time for different loss types in CS-MRI. . . . . 115

A.1 Results of PaDIS for 20 view CT reconstruction with different sized patches. . 126

A.2 Results for 20 view CT reconstruction with different dataset sizes. Top row shows recon performed by PaDIS; bottom row shows recon performed with the whole-image model. . . . . 127
A.3 Results of PaDIS for 20 view CT reconstruction for different positional encoding methods. . . . . 128
A.4 Results of PaDIS for 20 view CT reconstruction using different sampling and inverse problem solving algorithms. Top row is with PaDIS and bottom row is with the whole-image model. . . . . 129
A.5 Generation of CT images with various acceleration algorithms. Top row shows generation with the EDM sampler [93], middle row uses DDIM [165], bottom row is our proposed method. . . . . 136
A.6 Results of extra inverse problem experiments. From top to bottom: 60 view CT, fan beam CT, heavy deblurring. From left to right: baseline, ADMM-TV, whole image diffusion, PaDIS, ground truth. . . . . 137
A.7 Unconditionally generated CT images using patch stitching [152] for the top row and patch averaging [142] for the bottom row. Compare this to the images generated by our proposed method in Figure 4.4. . . . . 138
A.8 Comparison between PSNR of deblurring between whole-image model and proposed method for each image in the test dataset. . . . . 138
A.9 Comparison between PSNR of superresolution between whole-image model and proposed method for each image in the test dataset. . . . . 139
A.10 Quantitative results (axial view) of CT reconstruction with 8 views on AAPM dataset for different NFEs and slice blending methods. . . . . 145
A.11 Left: Performance of DiffusionBlend++ on more angles, Right: Reconstruction of DiffusionBlend++ with low-dose noise . . . . . 146
A.12 Comparison of DiffusionBlend++ with classical methods . . . . . 146
A.13 Results of 3D CT reconstruction with 8 views on LIDC dataset. Top row is axial view, middle row is sagittal view, bottom row is coronal view. . . . . 148
A.14 Results of 3D CT reconstruction with 6 views on LIDC dataset. Top row is axial view, middle row is sagittal view, bottom row is coronal view. . . . . 148
A.15 Results of 3D CT reconstruction with 4 views on LIDC dataset. Top row is axial view, middle row is sagittal view, bottom row is coronal view. . . . . 149
A.16 Results of limited angle 3D CT reconstruction on LIDC dataset. Top row is axial view, middle row is sagittal view, bottom row is coronal view. . . . . 149
A.17 Sagittal slices of generated 3D volumes for different numbers of different sampling steps. Generated image quality degrades if the number of steps is reduced too much. . . . . 153
A.18 Top row shows the first four axial slices, and the bottom row shows the last four axial slices of the generated volume . . . . . 154
A.19 Results of our proposed method and comparison methods for 8-view 3D CT reconstruction on AAPM dataset. The top row shows an axial slice and the bottom row shows a sagittal slice from each reconstructed volume. . . . . 156

A.20 Results of our proposed method and comparison methods for 60 view CT recon on $256 \times 256 \times 256$ LIDC dataset. Plots are shown in Hounsfield units. The top row shows the axial slice and the bottom row shows the sagittal slice from the reconstructed volume. . . . . 157
A.21 Results of our proposed method and comparison methods for 20 view CT recon on $512 \times 512 \times 256$ LIDC dataset. Plots are shown in Hounsfield units. The top row shows the axial slice and the bottom row shows the sagittal slice from the reconstructed volume. . . . . 158
A.22 Results of our proposed method for 60 view CT recon on $512 \times 512 \times 256$ LIDC dataset. Plots are shown in Hounsfield units. The top row shows our proposed method and the bottom row shows the ground truth. All slices are axial slices from the same volume. . . . . 158
A.23 Unconditional sampling result on $256 \times 256 \times 256$ LIDC-IDRI prior and its nearest neighbor from the training dataset sliced by axial, coronal, and sagittal view. . . . . 159
A.24 Unconditional sampling result on $256 \times 256 \times 256$ AAPM prior and its nearest neighbor from the training dataset sliced by axial, coronal, and sagittal view. . 159
A.25 Unconditional sampling examples for three different priors (axial slices of 3D volumes). . . . . 160
A.26 Unconditional sampling examples for three different priors (coronal slices of 3D volumes). . . . . 160
A.27 Unconditional sampling examples for three different priors (sagittal slices of 3D volumes). . . . . 161
A.28 Continuous axial slice of a volume sampled from $256 \times 256 \times 256$ LIDC-IDRI prior; every fourth slice is shown . . . . . 162
A.29 Continuous coronal slice of a volume sampled from $256 \times 256 \times 256$ LIDC-IDRI prior; every fourth slice is shown . . . . . 180
A.30 Continuous sagittal slice of a volume sampled from $256 \times 256 \times 256$ LIDC-IDRI prior; every fourth slice is shown . . . . . 181
A.31 Results of using LoRA module for 60 view CT reconstruction in a single measurement setting. All weights refers to adjusting all the weights of the network at reconstruction time. . . . . 182
A.32 Results of using LoRA module for deblurring in a single measurement setting. All weights refers to adjusting all the weights of the network at reconstruction time. . . . . 182
A.33 Results of 60 view CT recon using networks with different numbers of parameters in the single-measurement setting. The top numbers show the number of total parameters in the network. . . . . 182
A.34 Results of 60 view CT recon in the small dataset setting where the size of the small dataset is varied. . . . . 183
A.35 Comparison of PSNR between patch-based model and whole-image model for number of network refining iterations in single measurement setting. . . . . 183
A.36 Comparison of SSIM between patch-based model and whole-image model for number of network refining iterations in single measurement setting. . . . . 183

A.37 Six sample grayscale phantoms and colored phantoms used to train the mismatched distribution diffusion models . . . . . 183
A.38 Visual results of using various DIS to solve FBCT with 40 views (top) and deblurring (bottom). The same single pretrained network was used in all the experiments. Arrows point to some artifacts in the comparison methods. . . . 184

# LIST OF TABLES


3.1 SSIM and NRMSE for Poisson and Poisson-Gaussian likelihoods. Results were averaged across 7 different noise levels by varying $\alpha \in 0.02 : 0.005 : 0.035$ in (3.1), with $\sigma = 1$. . . . . 27

3.2 SSIM and NRMSE using Poisson Gaussian likelihood with different regularization/image prior approaches. Results were averaged across 7 different noise levels by varying $\alpha \in 0.02 : 0.005 : 0.035$ in (3.1). WFS$^*$ runs the same number of iterations as AWFS whereas WFS$^\dagger$ runs more iterations until convergence. 28

4.1 Comparison of quantitative results on three different inverse problems. Results are averages across all images in the test dataset. Best results are in bold. . . . . 51

4.2 Extra inverse problem experiments. Results are averages across all images in the test dataset. Best results are in bold. . . . . 52

4.3 Results of 60 view CT reconstruction with $512 \times 512$ images. Results are averages across all images in the test dataset. Best results are in bold. . . . . 54

5.1 Comprehensive comparison of quantitative results on Sparse-View CT Reconstruction on Axial View for AAPM and LIDC datasets. Best results are in bold. . . . . 64

5.2 Comprehensive comparison of quantitative results on Sparse-View CT Reconstruction on Sagittal View for AAPM and LIDC datasets. Best results are in bold. . . . . 64

5.3 Comprehensive comparison of quantitative results on Sparse-View CT Reconstruction on Coronal View for AAPM and LIDC datasets. Best results are in bold. . . . . 65

5.4 Comprehensive comparison of quantitative results on Limited-Angle CT Reconstruction on All Views for AAPM and LIDC datasets. Best results are in bold. . . . . 65

5.5 TV values of different reconstruction algorithms on the AAPM test set . . . . 66

5.6 Effectiveness of Blending Modules, Sagittal view performance on AAPM . . . 67

6.1 PSNR (dB) comparison of various methods for CT reconstruction on different datasets and resolutions with best results in bold. . . . . 78

6.2 Per volume runtimes of different methods for 8 view CT recon. . . . . 79

6.3 FID comparison with and without downsample channel . . . . . 80

6.4 PSNR in dB of different patch and volume size . . . . . 80

6.5 PSNR in dB of different $K$, with best results in bold . . . . . 81

7.1 Comparison of quantitative results on four different inverse problems in the dataless setting with the self-supervised loss (SS). Results are averages across all images in the test dataset. Best results for practical use are in bold. . . . . 93
7.2 Results of inverse problem solving in dataless setting for $512 \times 512$ images. . 94
7.3 Comparison of results for using self-supervised (SS) diffusion models in small dataset setting. The diffusion model is first trained on ellipse phantoms and then fine-tuned with the small dataset. Best results are in bold. . . . . 96
7.4 Comparison of results for using diffusion models trained on thousands of in-distribution images to solve inverse problems. Best results are in bold. Self-supervised refinement of network weights is beneficial even in this setting. . . . . 96
7.5 CT reconstruction results where the initial checkpoint was trained on LIDC dataset and refined on the fly with the AAPM measurement. . . . . 99

8.1 Comparison of quantitative results on four different medical imaging inverse problems. Results are averages across all images in the test dataset. Best results for practical use are in bold. . . . . 111
8.2 Comparison of quantitative results on different natural imaging inverse problems: deblurring and super-resolution. The second column indicates whether the method can handle images from different modalities. Best results for practical use are in bold. . . . . 111
8.3 Comparison between diffusion inverse solvers using the same network for CT reconstruction and image deblurring. . . . . 114

9.1 Comparison of different diffusion-based methods . . . . . 120

A.1 Effect of patch size $P$ on CT reconstruction . . . . . 123
A.2 Dataset size effect on CT reconstruction . . . . . 123
A.3 Positional encoding effect for CT reconstruction . . . . . 125
A.4 Dataset size effect on CT reconstruction . . . . . 125
A.5 Average runtimes of different methods across images in the test dataset for 20 view CT recon. . . . . 132
A.6 Effectiveness of Blending Modules, Sagittal view performance on AAPM . . . 144
A.7 Axial PSNR for 8 view SVCT recon for different NFEs . . . . . 145
A.8 Axial PSNR for 8 view SVCT recon for different slice jump frequencies . . . . 145
A.9 Wall times of various methods for 8 view 3D CT reconstruction . . . . . 146
A.10 Standard Deviation of Performance on Sparse-View CT Reconstruction on Sagittal View for AAPM and LIDC datasets. Best results are in bold. . . . . 147
A.11 3D prior modeling methods . . . . . 150
A.12 Model hyperparameters. . . . . 153
A.13 Trade-offs on different patch sizes . . . . . 155
A.14 Performance of 60 view CT recon using self-supervised network refining with LoRA module. Best results are in bold. . . . . 164
A.15 Performance of deblurring using self-supervised network refining with LoRA module. Best results are in bold. . . . . 164

A.16 Performance of 60 view CT recon using test-time adaptation with networks of different sizes. Best results are in bold. . . . 165
A.17 Performance of fine-tuning on 60 view CT using checkpoints fine-tuned from different dataset sizes. Best results are in bold. . . . 165
A.18 Performance of Algorithm 5 for 60 view CT reconstruction in single measurement setting with different numbers of backpropagation iterations. Best results are in bold. . . . 166
A.19 Average runtimes of different methods across images in the test dataset for 60 view CT recon. . . . 169
A.20 Comparison of using the proposed SPAR method while training only with CelebA images versus using a mixed dataset. Best results are in bold. . . . 172
A.21 Comparison between different thresholds $\sigma'$ when using SPAR image reconstruction. Best results are in bold. . . . 173
A.22 Comparison of using the traditional loss versus the proposed majorizer loss for each of the inverse problems. . . . 173
A.23 Comparison of using different $K$ for 60 view parallel beam CT recon. Average runtime for each image is also shown. . . . 174
A.24 Average runtimes of different methods across images in the test dataset for 60 view CT recon. . . . 175
A.25 LPIPS scores for deblurring on LSUN dataset and superresolution on CelebA dataset. . . . 175
A.26 Performance of deblurring on LSUN dataset and superresolution on CelebA dataset with noise level $\sigma = 0.05$. Networks were trained with CelebA images. Best results are in bold. . . . 176
A.27 Results from using DiffPIR to accelerate reconstruction process while applying our network finetuning method for deblurring. . . . 177

# ABSTRACT

We live in a world where imaging systems are ubiquitous. From the cell phones in our pockets to our cars and doorbells and on to telescopes and medical scanners, imaging has changed how we share, document, and understand our world. There is an increasing demand to make these systems more efficient by producing higher-quality images with fewer and fewer resources. When we need an image that is of higher quality than can be directly constructed from its measurements, we have an inverse problem and must rely on reconstruction schemes. These reconstruction algorithms fill the gap between the measured samples and the desired high-quality images by leveraging a model that attempts to recover image contents while reducing artifacts.

Diffusion models are trained on large image datasets to learn an underlying prior. This iterative process, which starts with pure noise and is gradually denoised to eventually obtain a clean image, can be modified to generate or restore the clean image that is also enforced to be consistent with the measurement, thereby solving an inverse problem. However, diffusion models are computationally expensive to train and require large, clean training datasets.

To tackle these challenges, this thesis develops methods of using diffusion models to solve inverse problems. In particular, we develop a novel patch-based diffusion prior that allows for image generation and reconstruction using a memory efficient network trained on small patches of an image. We later improve this diffusion prior by including global image context, allowing for better coherence between patches and accelerated sampling. We also develop a 3D diffusion prior that learns the joint distribution between multiple slices of a 3D volume. These methods allow us to solve 3D inverse problems that would otherwise be computationally infeasible to handle with traditional diffusion models. Furthermore, we demonstrate the efficacy of patch-based diffusion models for solving out-of-distribution inverse problems, reducing the need for a large training dataset. This is done by using a self-supervised loss function that refines the network at reconstruction time, ensuring the network output is consistent with the measurement. Motivated by the relative inaccuracy of trained diffusion models for low noise levels, we propose a novel self-supervised majorizer loss function which is theoretically and empirically shown to learn the score function at low noise levels better. Coupled with a channel concatenation operator that allows a network to accept images with differing numbers of channels, our method enables a single diffusion model to solve inverse problems in multiple modalities. The proposed methods are applied to problems in computed tomography, magnetic resonance imaging, and deblurring, among others, showing their wide generalizability with broad scientific and engineering applications.

# CHAPTER 1

# Introduction

In the real world, many objects cannot be observed directly and must instead be measured through a device or system. A doctor who wishes to view the internal anatomy of a patient can perform a computed tomography (CT) or magnetic resonance imaging (MRI) scan of the patient. A photographer can use a camera to capture the dynamic view of a bustling city. An astronomer studying black holes must use advanced telescopes to capture the desired view. Once the measurement has been obtained by the corresponding imaging system, one must then recover the original desired image from the measurements. This type of problem is known as an *inverse problem*.

In all of these cases, the system that is used to capture the image is inherently noisy. Improving the hardware of the imaging system is one possible solution to mitigate this problem, but is expensive in practice and can lead to slow acquisition times. When we wish to perform fast and cheap imaging, the process will naturally be lossy. Hence, the same measurement will correspond to many possible images. The central problem of computational imaging and this thesis is to develop methods to identify an image close to the ground truth and obtain a high-quality reconstruction.

Traditionally, inverse problems have been solved by reformulating them as an optimization problem consisting of two terms: a *data fidelity* term and a *regularization* term. The data fidelity term ensures that the image is consistent with the measurement, but that term alone is not enough to obtain a high-quality reconstruction due to the lossy nature of the measurement system. Hence, a regularization term, designed to enforce commonly held beliefs about possible clean images, is crucial. This regularization term ranges from simple handcrafted cost functions such as total variation loss or sparsity metrics [29] to complicated data-driven priors that are learned by neural networks [171].

When the measurements are very sparse or the noise level is high, it is especially important to develop a good prior to obtain a high-quality reconstruction. Generative models, and in particular diffusion models, have risen to popularity in recent years, as they are

capable of generating realistic clean images from pure noise. Diffusion models firstly involve adding progressively increasing levels of noise to clean images and training a neural network to denoise them. This process results in a network that has learned the underlying distribution of the training image dataset. Next, to generate a clean image using the trained network, we start with pure noise and use the network to gradually denoise the image, eventually obtaining a clean image belonging to the data distribution. Since diffusion models directly learn the image prior, they are particularly useful for solving inverse problems where the measurements are sparse or noisy, as the prior can be used to fill in missing information. Many methods have been developed for adapting diffusion models from their original purpose of sampling from the prior (generating clean images with no other information) to sampling from the posterior (generating a clean image that is consistent with a measurement, i.e., solving inverse problems), and these methods have shown superior performance over traditional methods.

Despite these successes, diffusion models still face several difficulties when solving inverse problems. Training whole-image diffusion models requires large quantities of clean training data that is often unavailable in many applications. Furthermore, as distributions of images are very high dimensional and complex, deep neural networks with hundreds of millions of parameters are needed to learn the distributions well, making the training process a slow and memory-intensive task. Finally, when solving difficult nonlinear inverse problems, diffusion models exhibit instability as methods of enforcing data fidelity often perturbs the image away from the desired noisy image manifold. Consequently, it is very challenging to apply diffusion models to solve large-scale and 3D imaging problems, applications where training data is limited, and nonlinear inverse problems. This thesis will focus on methods that allow diffusion models to overcome these challenges.

## 1.1 Contributions

In this work, we develop various novel methods targeted at the aforementioned challenges of diffusion models. These include methods of training and applying different types of diffusion models to solve a large variety of inverse problems, ranging from nonlinear inverse problems to 2D and 3D medical and natural imaging tasks. We also explore different problem settings where memory and training data may be limited. The contributions of this work are summarized as follows:

- **Nonlinear inverse problems.** We develop a novel algorithm to solve the nonlinear inverse problem of phase retrieval with the presence of Poisson and Gaussian noise. We show how diffusion models can be used for this problem and apply an acceleration method to reduce the number of diffusion iterations needed, a crucial

addition to an algorithm that would otherwise be very slow due to the complicated Poisson-plus-Gaussian likelihood. Our simulation experiments show that modeling both types of noise yields better performance than only using one type of noise and the diffusion prior is a better choice than other traditional methods [115].

- **Large-scale diffusion models.** We tackle large-scale imaging problems by developing a novel patch-based diffusion model approach for learning image priors. This patch-based prior can learn the prior of an entire image from solely the patches of the image, so that both at training and reconstruction time, it suffices to input patches of the image into the network. Hence, the method reduces the memory requirement and training time. Extensive experiments show that it exhibits superior performance over traditional whole-image diffusion models for a variety of medical and natural inverse problems [82]. We also develop a framework for general patch-based priors that encompasses several other works and derive convergence guarantees. For 3D image reconstruction, we develop a method that learns a 3D image prior by first learning the joint distributions of multiple slices of the 3D volume and then blending these distributions together. We also improve upon the patch-based prior model by incorporating global volumetric information, leading to a trained network that can unconditionally generate high-resolution volumes. We apply our methods to various problems in CT reconstruction where the measurements are very sparse and demonstrate that this 3D prior yields higher-quality reconstructions than previous state-of-the-art (SOTA) methods [162, 199].

- **Limited training data.** We apply our patch-based diffusion models to the mismatched distribution setting via test-time adaptation. In this setting, we are given a pretrained diffusion model, but the test data belongs to a different distribution from the training distribution, so we must refine the network at test time. Simulations show that our patch-based model is more easily adaptable to the mismatched distribution, is less prone to overfitting in the case of very limited data, and yields better performance at solving inverse problems than whole image models [81]. We then generalize this method using a channel operator and novel majorizer test-time adaptation loss function so that we can use a single pretrained network to solve inverse problems in many modalities [79].

- We raise various avenues of exploration for future work, including extending our unified framework for patch-based diffusion models, applying diffusion bridges for faster image reconstruction, and developing other test-time adaptation methods

that are more generalizable to challenging 3D inverse problems. We also discuss the possibility of applying video diffusion models to solve 3D image reconstruction.

## 1.2 Outline

To start, Chapter 2 provides background on the mathematical framework used for modeling inverse problems. It also introduces more specifics of the inverse problems that are covered in the rest of the work, including phase retrieval, CT reconstruction, and image enhancement. Next we introduce the relevant background of diffusion models and their relevance to inverse problems before moving onto mismatched distribution problems.

Chapter 3 details our novel algorithm by which we use diffusion models to solve phase retrieval under realistic system and noise models [115]. We demonstrate with exhaustive experiments that the realistic models and score-based image prior yield better reconstructions.

Chapter 4 provides the patch-based diffusion model framework that will be referenced in the subsequent chapters [82]. We develop a diffusion model prior that can be applied to large images by breaking up an image into patches and learning the prior on the patches separately. We then show that while using less memory and training time, in cases of limited training data, our patch-based approach outperforms traditional whole-image diffusion models in various inverse problems.

Chapter 5 further looks at large-scale diffusion models by tackling the 3D CT reconstruction problem with diffusion models [162]. We develop a method to learn 3D diffusion image priors and use it to solve very sparse-view CT reconstruction problems as well as limited-angle CT reconstruction.

Chapter 6 develops a fully 3D prior capable of unconditionally generating high-resolution 3D CT volumes. This is done by incorporating downsampled whole volumes into the patch-based prior developed in Chapter 4 to provide global information into the prior model. We also empirically demonstrate that this improved prior can be leveraged to obtain higher quality CT reconstructions.

In Chapter 7, we move to problems where the training data and test data have different distributions [81]. Specifically, we cover two settings: the first in which we are only given the measurement without any in-distribution training data and must refine the diffusion network on the fly, and the second in which we are given a very small dataset that can be used to fine-tune the network. We show that, when using patch-based diffusion models, desirable performance qualities are achieved compared to whole-image models.

Chapter 8 generalizes the work of the previous chapter even more by pretraining a single diffusion network on a mixed dataset of different modalities that can then be used to solve inverse problems in different modalities such as MRI and natural imaging. Our novel majorized self-supervised loss function, coupled with a solid theoretical foundation, is the driving force behind this method's flexibility.

Finally, Chapter 9 outlines future work that could be follow-up directions based on this dissertations's research. Broadly, this consists of extending the theory of patch-based diffusion models by running a more systematic set of experiments and developing the theory for more general (and accelerated) sampling methods. Furthermore, we highlight the potential of diffusion bridges to greatly accelerate inverse problem solving while preserving the generalizability of unconditional diffusion model methods, as well as possible methods of leveraging pretrained video diffusion models to solve 3D image reconstruction problems.

The appendix provides additional results, both quantitative and visual, ablation studies, and theoretical derivations. Summarizing, Figure 1.1 provides an overview of the dependencies of the chapters of this thesis and two of the main topics covered, large-scale imaging and distribution shift problems.

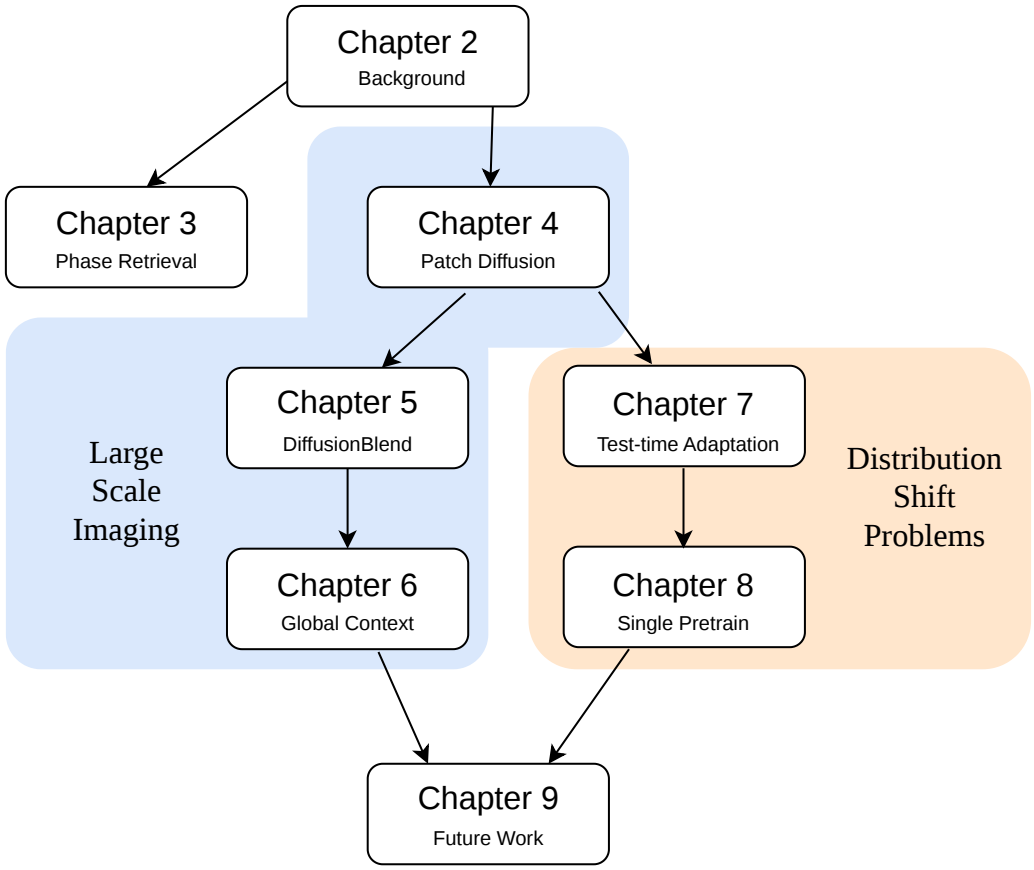



**FIG** 1.1 – Dependency flowchart for the chapters of this thesis.

# CHAPTER 2

# Background

This chapter begins with a mathematical formulation of inverse problems, a broad range of specific examples of inverse problems, and general frameworks that are commonly used to solve these types of problems. Next, we highlight specific inverse problems that are addressed in detail in the rest of this thesis and existing methods that can be used to solve them.

## 2.1 Inverse Problems

Inverse problems are ubiquitous in image processing, and aim to reconstruct an image $\boldsymbol{x}$ from a measurement $\boldsymbol{y}$, where $\boldsymbol{y} = \mathcal{A}(\boldsymbol{x}) + \boldsymbol{\epsilon}$, $\mathcal{A}$ represents a forward operator, and $\boldsymbol{\epsilon}$ represents random unknown noise. In the presence of noise, and particularly when $\mathcal{A}$ is nonlinear or ill-conditioned, the same $\boldsymbol{y}$ can corresponding to multiple possible images $\boldsymbol{x}$. Therefore, to obtain a high-quality reconstruction of $\boldsymbol{x}$, it is necessary to enforce a prior on $\boldsymbol{x}$. More precisely, by Bayes' Rule, $p(\boldsymbol{x}|\boldsymbol{y})$ is proportional to $p(\boldsymbol{x}) \cdot p(\boldsymbol{y}|\boldsymbol{x})$. By making assumptions on the noise $\boldsymbol{\epsilon}$, the distribution $p(\boldsymbol{y}|\boldsymbol{x})$ is readily derived, so it remains to establish a prior $p(\boldsymbol{x})$.

Traditionally, total variation (TV) has been used as a regularizer [35]. This regularizer has a denoising effect on the image while preserving sharp boundaries. Similarly, wavelet based methods [52] transform the image to the wavelet domain where it becomes simpler to denoise the image without distorting its main features. More recently, plug and play methods involve alternating between using a denoiser to denoise the image and a data fidelity update step to ensure that the image conforms with the measurement [34]. Block matching 3D (BM3D) is a fast and effective general purpose denoising algorithm, so its use in inverse problems has been studied [136].

In recent years, data-driven methods have risen in popularity in signal and image processing [122, 118, 80, 194, 196]. In particular, for solving inverse problems, when large

amounts of training data is available, a learned prior can be much stronger than the handcrafted priors used in traditional methods. For instance, plug and play and regularized by denoising methods [172, 197, 123, 83, 121, 33, 156] that involve pretraining a denoiser and applying it at reconstruction time have shown superior performance over using fixed denoisers. These methods have the advantage over supervised deep learning methods such as [89, 104, 169, 187] that the same denoiser may be applied to solve a wide variety of inverse problems.

The next subsections summarize several inverse problems investigated in this thesis.

### 2.1.1 Phase Retrieval

Phase retrieval (PR) is a nonlinear inverse problem, where the goal is to recover a signal from the (square of) magnitude-only measurements that are corrupted by noise [85]. This problem has applications in astronomy [50], X-ray crystallography [137], optical imaging [157], Fourier ptychography [17, 208, 193, 176] and coherent diffractive imaging (CDI) [106]. For example, in holographic CDI, a coherent beam source illuminates a sample of interest and a reference. When the beam hits the sample, it generates secondary electromagnetic waves that propagate until they reach a detector. By measuring the photon flux, the detector can capture and record a diffraction pattern. This pattern is roughly proportional to the square of Fourier transform magnitude of electric field associated with the illuminated objects [13, 14]. Recovering the structure of the sample from the diffraction pattern generated by the mix of sample and reference is a nonlinear inverse problem known as holographic PR. One approach to this problem is maximum a posteriori (MAP) estimation:

$$\hat{\boldsymbol{x}} = \operatorname{argmax}_{\boldsymbol{x}\in\mathbb{R}^N} p(\boldsymbol{x}|\boldsymbol{y}, \bar{\boldsymbol{b}}, \boldsymbol{A}, \boldsymbol{r}) = \operatorname{argmin}_{\boldsymbol{x}\in\mathbb{R}^N} g(\boldsymbol{x}; \boldsymbol{A}, \boldsymbol{y}, \bar{\boldsymbol{b}}, \boldsymbol{r}) + h(\boldsymbol{x}), \tag{2.1}$$

where $\boldsymbol{x}$ denotes a real latent image to recover, $\boldsymbol{y}$ is the recorded measurement vector, $\bar{\boldsymbol{b}}$ denotes the mean of background measurements, and $\boldsymbol{A} \in \mathbb{C}^{M\times N}$ denotes the system matrix in holographic PR, where $M$ denotes the number of measurements and $N$ denotes the dimension of $\boldsymbol{x}$. The known reference image $\boldsymbol{r}$ provides additional information to reduce the ambiguity of $\hat{\boldsymbol{x}}$; using an extended reference is a common technique in holographic CDI [155, 70]. Following Bayes' rule, we denote $g(\boldsymbol{x}) = -\log p(\boldsymbol{y}, \boldsymbol{A}, \boldsymbol{r}|\boldsymbol{x})$ and $h(\boldsymbol{x}) = -\log p(\boldsymbol{x})$ as the data fidelity term and the regularization term, respectively. For simplicity we assume $\boldsymbol{x}$ is real, so $\nabla g(\boldsymbol{x})$ denotes the real component of the gradient of log-likelihood. Hence, all priors are trained with real-valued datasets and are applied to the real components of $\boldsymbol{x}$ as well. The method can be extended to complex images.

The term $g(\boldsymbol{x})$ depends on the noise model, which is generally chosen to be either Gaussian or Poisson. For methods that assume the elements of $\boldsymbol{y}$ follow independent Gaussian distributions $\boldsymbol{y} \sim \mathcal{N}(|\boldsymbol{A}\boldsymbol{x}|^2 + \bar{\boldsymbol{b}}, \boldsymbol{\Sigma})$, where $\boldsymbol{\Sigma} = \sigma^2\boldsymbol{I}$, the data fidelity term $g(\boldsymbol{x})$ in (2.1) becomes $g_{\text{Gau}}(\boldsymbol{x}) \triangleq \|\boldsymbol{y} - \bar{\boldsymbol{b}} - |\boldsymbol{A}\boldsymbol{x}|^2\|_2^2$. To solve the corresponding MAP optimization problem, a popular method is Wirtinger flow (WF) [28, 88, 26, 161] using the Wirtinger gradient: $\nabla g_{\text{Gau}}(\boldsymbol{x}) = 4\boldsymbol{A}' \operatorname{diag}\{|\boldsymbol{A}\boldsymbol{x}|^2 - \boldsymbol{y} + \bar{\boldsymbol{b}}\}\boldsymbol{A}\boldsymbol{x}$. To determine an appropriate step size for the Wirtinger gradient, one can use its Lipschitz constant or methods such as empirical trial and error, backtracking line search, or observed Fisher information [117]. To further accelerate WF, one can use Nesterov's momentum methods [139] or optimized gradient methods [97], leading to the accelerated Wirtinger flow (AWF) [193, 21, 58, 63] that is commonly used in solving PR problems. Apart from WF, other methods such as matrix-lifting [30, 27, 157], error reduction (ER) [64], hybrid input-output (HIO) [62], majorize-minimize (MM) [146] and alternating direction method of multipliers (ADMM) [119] have also been proposed.

On the other hand, the Poisson ML model assumes $\boldsymbol{y} \sim \text{Poisson}(|\boldsymbol{A}\boldsymbol{x}|^2 + \bar{\boldsymbol{b}})$, so that $g(\boldsymbol{x})$ in (2.1) has the form: $g_{\text{Pois}}(\boldsymbol{x}) \triangleq \boldsymbol{1}'(|\boldsymbol{A}\boldsymbol{x}|^2 + \bar{\boldsymbol{b}}) - \boldsymbol{y}' \log\big(|\boldsymbol{A}\boldsymbol{x}|^2 + \bar{\boldsymbol{b}}\big)$ . Similar to the Gaussian case, one can also apply WF [116] with $\nabla g_{\text{Pois}}(\boldsymbol{x}) = 2\boldsymbol{A}\boldsymbol{x} \odot \big(\boldsymbol{1} - \boldsymbol{y} \oslash (|\boldsymbol{A}\boldsymbol{x}|^2 + \bar{\boldsymbol{b}})\big)$ , where $\odot$ and $\oslash$ denote element-wise multiplication and division, respectively.

### 2.1.2 CT Reconstruction

Computed tomography (CT) is a medical imaging technique that allows a 3D object to be imaged by shooting X-rays through it [59]. The measurements consist of a set of 2D projection views obtained from setting up the source and detector at different angles around the object. By definition, $\boldsymbol{y}$ is the (known) set of projection views, $\mathcal{A}$ is the (in most cases assumed to be) linear forward model of the CT measurement system, and $\boldsymbol{x}$ is the unknown image. The CT reconstruction problem then consists of reconstructing $\boldsymbol{x}$ given $\boldsymbol{y}$. To reduce the radiation dose delivered to the patient, sparse-view CT uses a smaller fraction of X-rays compared to the full-view CT [159]. Additionally, limited-angle CT is useful in cases where patients may have mobility issues and cannot use full-angle CT scans [24]. In these cases, $\boldsymbol{y}$ contains far less information than $\boldsymbol{x}$, so this becomes a challenging inverse problem.

Traditional methods for solving this include regularization-based methods that enforce a previously held belief on $\boldsymbol{x}$ and likelihood based methods [59, 175, 192, 37]. Data-driven methods have shown tremendous success in signal and image processing in recent years [122, 118, 80, 194]. In particular, for solving inverse problems, when large amounts of

training data is available, a learned prior can be much stronger than the hand-crafted priors used in traditional methods [197, 123]. For past few years, many deep learning-based method have been proposed for solving the 3D CT reconstruction problem [89, 104, 169, 195]. These methods train a convolutional neural network, such as a U-Net [89], that maps the partial-view filtered backprojection (FBP) reconstructed image to the ground truth image, that is, full-view CT reconstruction. However, these methods often generate blurry images and generalizes poorly for out-of-distribution data [4].

### 2.1.3 MRI Reconstruction

Magnetic resonance imaging (MRI) is another medical imaging technique that uses magnetic fields to obtain images of organs, tissues, and other structures [75]. MRI scanners acquire the Fourier components of the image of interest, called the k-space. However, this acquisition procedure is slow, so acceleration methods such as undersampling the Fourier components have been developed, which can lead to aliasing in reconstructed images. To solve this problem, modern MRI scanners use multiple coils to acquire the Fourier components, providing additional spatial information [145, 67]. Additionally, compressed sensing (CS) MRI [132, 133] improves the quality of the reconstructed images by using suitable sampling patterns. Similar to CT reconstruction and other inverse problems, methods of solving CS MRI reconstruction problems include classical optimization based methods [132], plug and play methods [1], and diffusion model methods [46]. Notably, for deep learning methods, one often represents the complex MRI data using two separate channels consisting of the real and imaginary parts [41].

## 2.2 Diffusion Models

Diffusion models consist of defining a forward stochastic differential equation (SDE) that adds noise to a clean image [168]: for $t \in [0, T]$, $\boldsymbol{x}(t) \in \mathbb{R}^d$, we have

$$\mathrm{d}\boldsymbol{x} = -(\beta(t)/2)\,\boldsymbol{x}\;\mathrm{d}t + \sqrt{\beta(t)}\;\mathrm{d}\boldsymbol{w}, \tag{2.2}$$

where $\beta(t)$ is the noise variance schedule of the process. The distribution of $\boldsymbol{x}(0)$ is the data distribution and the distribution of $\boldsymbol{x}(T)$ is (approximately) a standard Gaussian. Then, image generation is done through the reverse SDE [3]:

$$\mathrm{d}\boldsymbol{x} = (-\beta(t)/2 - \beta(t)\nabla_{\boldsymbol{x}_t} \log p_t(\boldsymbol{x}_t))\;\mathrm{d}t + \sqrt{\beta(t)}d\overline{\boldsymbol{w}}. \tag{2.3}$$

To train a network to approximate the score function $\nabla_{\boldsymbol{x}_t} \log p_t(\boldsymbol{x}_t)$, it is necessary to first discretize the SDE into a sequence of positive noise scales (for white Gaussian $\mathcal{N}(0, \sigma_k^2)$): $\sigma_1 > \sigma_2 > \cdots > \sigma_K$, with $\sigma_K$ being small enough so that noise of this level does not visibly affect the image, and $\sigma_1$ depending on the application. Score matching can be used to train a noise conditional score network (NCSN) [179, 166] as follows:

$$\hat{\boldsymbol{\theta}} = \text{argmin}_{\boldsymbol{\theta}} \sum_{k=1}^{K} \mathbb{E}_{\boldsymbol{x},\tilde{\boldsymbol{x}}} \left[ \left( s_{\boldsymbol{\theta}}(\boldsymbol{x}, \sigma_k) - \frac{\boldsymbol{x} - \tilde{\boldsymbol{x}}}{\sigma_k^2} \right)^2 \right],$$
$$\text{where} \quad \boldsymbol{x} \sim p(\boldsymbol{x}), \quad \tilde{\boldsymbol{x}} \sim \boldsymbol{x} + \mathcal{N}(0, \sigma_k^2 \boldsymbol{I}). \tag{2.4}$$

With enough data, the neural network $s_{\boldsymbol{\theta}}(\boldsymbol{x}, \sigma)$ is expected to learn the distribution $p_\sigma(\boldsymbol{x}) = \int p(\boldsymbol{x}) p_\sigma(\tilde{\boldsymbol{x}}|\boldsymbol{x}) d\boldsymbol{x}$ where $p_\sigma(\tilde{\boldsymbol{x}}|\boldsymbol{x}) = \mathcal{N}(\boldsymbol{x}, \sigma^2 \boldsymbol{I})$. Once the neural network has been trained to approximate the score function, one can start with noise and run the reverse SDE to obtain samples from the learned data distribution.

### 2.2.1 Diffusion Inverse Solvers

Diffusion models serve as a strong image prior as they can generate entire images from pure noise. Most methods that use diffusion models to solve inverse problems involve writing the problem as a conditional generation problem [53, 120, 40] or as a posterior sampling problem [43, 39, 31, 181, 95]. In the former case, the network requires the measurement $\boldsymbol{y}$ (or an appropriate resized transformation of $\boldsymbol{y}$) during training time. Thus, for these task-specific trained models, at reconstruction time, that network is useful only for solving that specific inverse problem. In contrast, for the posterior sampling framework, the network learns an unconditional image prior for $\boldsymbol{x}$ that can help solve any inverse problem related to $\boldsymbol{x}$ without retraining.

More specifically, methods that use an unconditionally trained diffusion model to solve inverse problems generally involve alternating between updating the image using the trained score network and updating the image to be consistent with the measurement. These two steps are generally detached, so many choices of sampling methods and SDE solvers are possible for the former step while changing the method by which data fidelity is enforced. For example, [166] uses Langevin dynamics with a simple gradient descent step toward the measurement. Recognizing that the image lies on a noisy data manifold (as opposed to the clean data manifold) during intermediate timesteps, [39] improves on this method by performing gradient descent with respect to the expectation of the clean image at each iteration. As these methods require a large number of iterations, DDIM [165] was developed as an acceleration method that relies on estimating the clean image

at each iteration to guide the noisy image toward the clean image manifold. Building off this method, [181] and [95] rely on hard constraint enforcement via projection onto the manifold where data consistency is satisfied while using DDIM as the sampling method, greatly reducing the number of sampling steps needed when solving inverse problems. Finally, conjugate gradient descent has proven to be useful for enforcing data fidelity when used in conjunction with DDIM when the measurements are assumed to be compressed but noiseless [41]. However, even when the number of sampling steps is large, these methods all struggle to truly sample from the posterior, as they all rely on approximations to the posterior distribution at intermediate timesteps. More recently, [87] proposed a method using resampling and Monte Carlo methods to estimate the posterior distribution at each timestep which was theoretically shown to sample from the posterior distribution when the number of particles is large.

### 2.2.2 Large-scale Diffusion Models

To reduce the computational burden, latent diffusion models [150] have been proposed, aiming to perform the diffusion process in a much smaller latent space, allowing for faster training and sampling. However, that method requires a pretrained encoder and decoder [57] for a fixed dataset, so it must be retrained for different datasets, and it still requires large amounts of training data. Furthermore, solving inverse problems in the latent space is difficult due to the nonlinearity of the encoder and decoder, so existing methods are slow and unstable, particularly for nonlinear inverse problems [163].

Patch-based diffusion models [183, 55] focus on image generation while training only on patches, reducing the memory requirement by only requiring patch inputs to the score network. However, for high-quality image generation, [183] still requires inputting the entire image into the network at inference time. Furthermore, [55] only allows for unconditional image generation, and it is unclear how to extend this method to solve inverse problems. Supervised patch-based diffusion methods [189, 142] are task specific and do not learn an unconditional image prior that can be applied to all inverse problems. Other patch-based methods [18, 130, 144] learn an unconditional image prior but require the whole image as an input during inference time. Finally, work has been done to perform sampling faster [165, 93, 128], which is unrelated to the training process.

Other strategies for reducing computational costs of diffusion models include multilevel diffusion models that train networks on progressively higher resolution images and employ a similar strategy at inference time to generate high resolution images [69, 9]. Alternative network architectures that reduce the number of parameters have also been

proposed in both the pixel space and latent space [72, 144], but these networks are still difficult to scale up to very large images or 3D images.

### 2.2.3 3D Diffusion Models

Although diffusion-based methods have shown great performance for solving inverse problems for 2D images in different domains, there are few methods that are able to tackle inverse problems for 3D images because of the infeasible computational and data requirements. Training a network that learns the score function of a 3D volume directly would require a prohibitively large amount of memory and training data. Thus, previous works have focused on learning the distribution of 2D slices of a 3D volume and combining the slices in a coherent way at reconstruction time. Specifically, for 3D CT reconstruction, DiffusionMBIR [42] trains a diffusion model on the axial slices of volumes; at reconstruction time, it uses the total variation (TV) regularizer with a posterior sampling approach to encourage consistency between adjacent slices. Similarly, DDS [41] builds on this work by using accelerated methods of sampling and data consistency to greatly reduce the reconstruction time. However, although the TV regularizer has shown some success in maintaining smoothness across slices, it is not a data-driven method and does not properly learn the 3D prior. TPDM [109] addresses this problem by training a separate prior on the coronal slices of volumes with a conditional sampling approach, which serves as a data-driven method of maintaining slice consistency at reconstruction time, but requires that all the volumes have the same cubic shape. In exchange, this method sacrifices the speed gains made by DDS, requiring alternating updates between the two separate priors, and is also twice as computationally expensive at training time.

## 2.3 Distribution Mismatch Problems

Training diffusion models well requires vast amounts of clean training data [168, 73], which is infeasible to collect in many applications such as medical imaging [44, 167, 86], black hole imaging [61, 60], and phase retrieval [115, 188]. In particular, for very challenging inverse problems such as black hole imaging [61] and Fresnel phase retrieval [71], no ground truth images are known and one only has a single measurement $\boldsymbol{y}$ available. In other applications such as dynamic CT reconstruction [147] and single photo emission CT [118], obtaining a high-quality measurement that can lead to a reconstruction that closely approximates the ground truth can be slow or potentially harmful to the patient, so only a very small dataset of clean images are available. Thus, in Chapter 7 we consider

two settings: the *single measurement* setting in which we are given one measurement $\boldsymbol{y}$ whose corresponding $\boldsymbol{x}$ belongs to a different distribution from the training dataset, and the *small dataset* setting in which we are only given a small number of samples $\boldsymbol{x}$ that belong to the same distribution as the test dataset.

When no in-distribution data is available, one approach is to use traditional methods that do not require any training data, such as total variation (TV) [112] or wavelet transform [52] regularizers that encourage image sparsity. More recently, plug-and-play (PnP) methods have risen in popularity [170, 77, 75, 76]; these methods use a denoiser to solve general inverse problems. Although these methods often use a trained denoiser, [154] found that using an off-the-shelf denoiser such as block matching 3D [49] can yield competitive results. Nevertheless, with the rise of deep learning in image processing applications, methods that harness the power of these tools may be desirable.

The deep image prior (DIP) is an extensively studied self-supervised method that is popular when no training data is available and reconstruction from a single meausurement $\boldsymbol{y}$ is desired. The method consists of training a network using the loss function

$$L(\theta) = \|\boldsymbol{y} - \mathcal{A}(f_\theta(\boldsymbol{z}))\|_2^2, \quad \boldsymbol{z} \sim \mathcal{N}(0, \boldsymbol{I}), \tag{2.5}$$

so that $f_\theta(\boldsymbol{z})$ produces the reconstruction. Although the neural network acts as an implicit regularizer whose output tends to lie in the manifold of clean images, DIP is prone to overfitting [177]. Various methods have been proposed involving early stopping, regularization, and network initialization [122, 90, 11]. Nevertheless, the method is very sensitive to parameter selection and implementation and can take a long time to train [90].

Most diffusion inverse solving (DIS) methods learn a prior from a large collection of clean in-distribution training images, but recently [10] and [47] proposed self-supervised diffusion model methods that are based off the DIP framework. These methods involve alternating between the usual reverse diffusion update step to gradually denoise the image and a network refining step in which the score network parameters are updated via the loss function

$$L(\theta) = \|\boldsymbol{y} - \mathcal{A}(\mathrm{CG}(\hat{\boldsymbol{x}}_{0|t}(\boldsymbol{x}_t; \theta)))\|_2^2 \tag{2.6}$$

where conjugate gradient (CG) descent is used to enforce data fidelity. This CG step consists of solving an optimization of the form

$$\mathrm{argmin}_{\boldsymbol{x}} \frac{\gamma}{2}\|\boldsymbol{y} - \mathcal{A}(\boldsymbol{x})\|_2^2 + \frac{1}{2}\|\boldsymbol{x} - \hat{\boldsymbol{x}}_{0|t}\|_2^2, \tag{2.7}$$

where $\gamma$ is a tradeoff parameter controlling the strength of the prior versus the measurement. Crucially, these methods introduce an additional LoRA module [78] to the network and the original network parameters are frozen when backpropagating the loss, which helps to avoid overfitting the whole-image model. Nevertheless, many technical tricks are required [47] involving noisy initializations and early stopping to obtain good results and avoid artifacts. Our patch-based model of Chapter 4 avoids this overfitting issue.

In the small dataset setting, various fine-tuning methods exist to shift the underlying prior learned by a score network away from a mismatched distribution and toward a target distribution. Given a pretrained diffusion network on a mismatched distribution, [138], [207], and [209] among others have studied ways to fine-tune the network to the desired dataset. These methods generally involve freezing certain layers of the original network, appending extra modules that contain relatively few weights, or modifying the loss function to capture details that differ greatly between distributions. However, these methods usually still require thousands of images from the desired distribution and focus on image generation. When solving inverse problems, the reconstructed image should be consistent with the measurement $\boldsymbol{y}$, reducing the number of degrees of the freedom for the image compared to generation, so with proper fine-tuning the data requirement should be lower.

# CHAPTER 3

# Accelerated Wirtinger Flow with Score-based Image Priors for Holographic Phase Retrieval in Poisson-Gaussian Noise Conditions

## 3.1 Motivation

Existing algorithms for Poisson-Gaussian holographic phase retrieval are limited in the literature. Handcrafted regularization-based algorithms such as total variation (TV) and higher order generalizations including total generalized variation (TGV) [22, 23] and the L1-norm of coefficients of wavelet transform [52] have been studied extensively. More recently, deep learning (DL)-integrated algorithms for solving inverse problems in computational imaging have been reported to be the state-of-the-art [141]. A trained network can serve as an object prior for regularizing the reconstructed image so it remains near a learned manifold [20]. Incorporating a trained denoising network as a regularizer $h(\cdot)$ led to methods such as plug-and-play (PnP) [34, 204, 92] and regularization by denoising (RED) [149]. In contrast to training a denoiser using clean images, there is growing interest in self-supervised image denoising approaches that do not require clean data as the training target [110, 15, 182].

In addition to training a denoiser as regularizer, generative model-based priors have also been proposed [7, 186]. Recently, diffusion models have gained significant traction for image generation [166, 73, 54, 168]. These probabilistic image generation models start with a clean image and gradually increase the level of noise added to the image, resulting in white Gaussian noise. Then in the reverse process, a neural network is trained to learn the noise in each step to generate or sample a clean image as in the original data distribution.

This chapter based on [115]. Equal contribution with Zongyu Li.

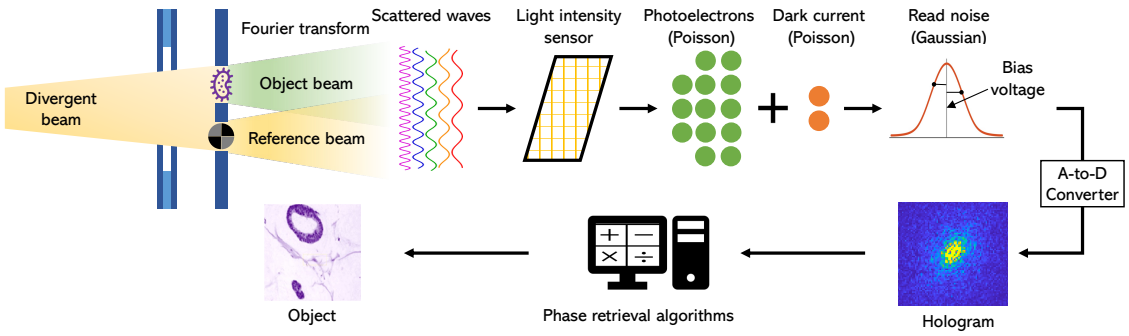



**FIG** 3.1 – Illustration of Poisson and Gaussian noise statistics in Fourier transform holographic phase retrieval.

The score-based diffusion models estimate the gradients of data distribution and can be used as plug-and-play priors for inverse problems [66] such as image deblurring and MRI and CT reconstruction [108, 86, 46, 48, 127, 167].

However, the realm of using score-based models to perform phase retrieval is relatively unexplored; previous relevant works [158, 66] applied denoising diffusion probabilistic modeling (DDPM) to PR but with less realistic system models and under solely Gaussian or Poisson noise statistics.

In summary, our contributions are as follows:

- We present our proposed algorithm known as accelerated Wirtinger flow with a score-based image prior (*i.e.*, $\nabla h(\boldsymbol{x})$ in (2.1)) to address the challenge of holographic phase retrieval (PR) problem in the presence of Poisson and Gaussian (PG) noise statistics.

- Theoretically, we derive a Lipschitz constant for the holographic PR's PG log-likelihood and provide a guarantee that sequences generated by the proposed algorithm converge to critical points of the cost function.

- We perform a series of simulation experiments which demonstrate that: 1) Algorithms using the PG likelihood model yield superior reconstructions compared to those relying solely on either the Poisson or Gaussian likelihood models. 2) With the proposed score-based prior as regularization, the proposed approach generates higher quality reconstructions and is more robust to variations of noise levels (without any parameter tuning) than alternative state-of-the-art methods.

### 3.1.1 Poisson-Gaussian Log-likelihood for PR

Based on the physical model as demonstrated in Fig. 3.1, we model the system matrix $\boldsymbol{A}$ by the (oversampled and scaled) discrete Fourier transform applied to a concatenation of the sample $\boldsymbol{x}$, a blank image (representing the holographic separation condition [107]) and a known reference image $\boldsymbol{r}$, so that $\boldsymbol{y}$ follows the Poisson plus Gaussian distribution:

$$\begin{aligned} &\boldsymbol{y} \sim \mathcal{N}(\text{Poisson}\left(|\boldsymbol{A}(\boldsymbol{x})|^2 + \bar{\boldsymbol{b}}\right), \sigma^2 \boldsymbol{I}), \\ &\boldsymbol{A}(\boldsymbol{x}) \triangleq \alpha \mathcal{F}\{[\boldsymbol{x}, \boldsymbol{0}, \boldsymbol{r}]\}. \end{aligned} \tag{3.1}$$

Here $\sigma^2$ denotes the variance of Gaussian noise, and $\alpha$ denotes a scaling factor (quantum efficiency, conversion gain, etc.) after applying the Fourier transform. Plugging the negative log-likelihood of (3.1) into (2.1) leads to

$$g_{\text{PG}}(\boldsymbol{x}) = \sum_{i=1}^{M} g_i(\boldsymbol{x}), g_i(\boldsymbol{x}) \triangleq -\log \left( \sum_{n=0}^{\infty} \frac{\mathrm{e}^{-\left(|\boldsymbol{a}_i' \boldsymbol{x}|^2 + \bar{b}_i\right)} \cdot \left(|\boldsymbol{a}_i' \boldsymbol{x}|^2 + \bar{b}_i\right)^n}{n!} \cdot \frac{\mathrm{e}^{-\left(\frac{(y_i - n)^2}{\sqrt{2}\sigma}\right)}}{\sqrt{2\pi\sigma^2}} \right).$$

Here $\boldsymbol{a}_i'$ denotes the $i$th row of $\boldsymbol{A}$ (since $\boldsymbol{A}$ is linear). We opt to use WF for estimating $\boldsymbol{x}$ because it is commonly used in practice due to its simplicity and efficiency [28]. The WF algorithm is based on the gradient of (3.2):

$$\begin{aligned} &\nabla g_{\text{PG}}(\boldsymbol{x}) = 2\boldsymbol{A}' \,\text{diag}\{\phi_i(|\boldsymbol{a}_i' \boldsymbol{x}|^2 + b_i; y_i)\} \boldsymbol{A}\boldsymbol{x}, \\ &\phi(u; v) \triangleq 1 - \frac{s(u, v-1)}{s(u, v)}, \quad s(a, b) \triangleq \sum_{n=0}^{\infty} \frac{a^n}{n!} \mathrm{e}^{-\left(\frac{b-n}{\sqrt{2}\sigma}\right)^2}. \end{aligned} \tag{3.2}$$

**Lemma 1.** The function $\phi(u)$ is Lipschitz differentiable and the Lipschitz constant for its derivative $\dot{\phi}(u)$ is:

$$\max\{|\ddot{\phi}(u)|\} \triangleq \mu = \left(1 - \mathrm{e}^{-1/\sigma^2}\right) \mathrm{e}^{\frac{2y_{\max} - 1}{\sigma^2}}, \tag{3.3}$$

where $y_{\max} \triangleq \max_{i \in \{1, \ldots, M\}} y_i$.

The proof of Lemma 1 is given in [38].

To facilitate choosing step sizes, the following Theorem provides a Lipschitz constant for the gradient of the log-likelihood (3.2).

**Theorem 1** *Assume $|x_j|$ is bounded above by $C$ for each $j$, a Lipschitz constant of $\nabla g_{\mathrm{PG}}(\boldsymbol{x})$ is*

$$\begin{aligned}\mathcal{L}(\nabla g_{\mathrm{PG}}) \triangleq\ & 4C^2\|\boldsymbol{A}\|_2^2\,\|\boldsymbol{A}\|_\infty^2\,\left(1-\mathrm{e}^{-\frac{1}{\sigma^2}}\right)\mathrm{e}^{\frac{2y_{max}-1}{\sigma^2}} \\ & + 2\|\boldsymbol{A}\|_2^2\left|1-C^2\,\|\boldsymbol{A}\|_\infty^2\,\left(1-\mathrm{e}^{-\frac{1}{\sigma^2}}\right)\mathrm{e}^{\frac{2y_{max}-1}{\sigma^2}}\right|.\end{aligned} \tag{3.4}$$

*where $y_{max} \triangleq \|\boldsymbol{y}\|_\infty$.*

**Proof:** Let $g_{\mathrm{PG}}(\boldsymbol{x})$ denote a function that maps a vector $\boldsymbol{x}\in\mathbb{R}^N$ to a scalar; it is the sum of each $g_i(\boldsymbol{x}) \triangleq \phi_i(|\boldsymbol{a}_i'\boldsymbol{x}|^2+b_i;y_i)$ over $i=1,\ldots,M$. Let $\boldsymbol{g}(\boldsymbol{x})$ denote a function that maps a vector $\boldsymbol{x}\in\mathbb{R}^N$ to the measurement space $\mathbb{R}^M$; it is the concatenation of each $g_i(\boldsymbol{x})$. So $\nabla g_{\mathrm{PG}}(\boldsymbol{x})\in\mathbb{R}^N$, $\nabla^2 g_{\mathrm{PG}}(\boldsymbol{x})\in\mathbb{R}^{N\times N}$, and $\nabla\boldsymbol{g}(\boldsymbol{x})\in\mathbb{R}^{M\times N}$.

By the chain rule, the Hessian of $g_{\mathrm{PG}}$ is

$$\nabla^2 g_{\mathrm{PG}}(\boldsymbol{x}) = 2\boldsymbol{A}'\left(\mathsf{diag}\{\boldsymbol{A}\boldsymbol{x}\}\nabla\boldsymbol{g}(\boldsymbol{x}) + \mathsf{diag}\{\boldsymbol{g}(\boldsymbol{x})\}\boldsymbol{A}\right). \tag{3.5}$$

Assume $|x_j|\le C$ for each $j$. Then it follows that $\|\,\mathsf{diag}\{\boldsymbol{A}\boldsymbol{x}\}\|_2 \le C\|\boldsymbol{A}\|_\infty$ by the construction of matrix-vector multiplication, leading to a Lipschitz constant for $\nabla g_{\mathrm{PG}}(\boldsymbol{x})$:

$$\mathcal{L}(\nabla g_{\mathrm{PG}}) = 2C\|\boldsymbol{A}\|_2\,\|\boldsymbol{A}\|_\infty\,\|\nabla\boldsymbol{g}(\boldsymbol{x})\|_2 + 2\|\boldsymbol{A}\|_2^2\,\|\,\mathsf{diag}\{\boldsymbol{g}(\boldsymbol{x})\}\|_2. \tag{3.6}$$

Here $\mathcal{L}(\nabla g_{\mathrm{PG}})$ denotes a Lipschitz constant for $\nabla g_{\mathrm{PG}}$, not necessarily the best one. To compute $\|\nabla\boldsymbol{g}(\boldsymbol{x})\|_2$, we substitute the Lipschitz constant of $\dot\phi(u)$ into (3.2) and apply Lemma 1, leading to

$$\|\nabla\boldsymbol{g}(\boldsymbol{x})\|_2 \le 2C\|\boldsymbol{A}\|_2\|\boldsymbol{A}\|_\infty\left(1-\mathrm{e}^{-\frac{1}{\sigma^2}}\right)\mathrm{e}^{\frac{2y_{\max}-1}{\sigma^2}}. \tag{3.7}$$

To compute $\|\,\mathsf{diag}\{\boldsymbol{g}(\boldsymbol{x})\}\|_2$, let

$$t\in[b,\max_i\{|\boldsymbol{a}_i'\boldsymbol{x}|^2\}+b] \subseteq \mathcal{T} \triangleq [b, C^2\|\boldsymbol{A}\|_\infty^2+b]. \tag{3.8}$$

From the fact that $\dot\phi(t)\le 1$ by its construction, one can show:

$$\begin{aligned}\|\,\mathsf{diag}\{\boldsymbol{g}(\boldsymbol{x})\}\|_2 = \|\boldsymbol{g}(\boldsymbol{x})\|_\infty &\le \max_{t\in\mathcal{T}}\{|\dot\phi(t)|\} \\ &\le \left|1-C^2\|\boldsymbol{A}\|_\infty^2\max\{|\ddot\phi(t)|\}\right|.\end{aligned} \tag{3.9}$$

Combining (3.6), (3.7) and (3.9) completes the proof of Theorem 1. ∎

Theorem 1 extends [38] by considering a a nonlinear transformation ($\boldsymbol{A}\boldsymbol{x}\to|\boldsymbol{A}\boldsymbol{x}|^2$) and a different system matrix $\boldsymbol{A}$.

In the practical implementation we approximate (3.2), with a finite sum following [38]:

$$s(a,b) \approx \sum_{n=0}^{n^+} \frac{a^n}{n!} \mathrm{e}^{-\left(\frac{b-n}{\sqrt{2}\sigma}\right)^2}, \quad n^+ = \lceil n^* + \delta\sigma \rceil, \tag{3.10}$$

with $n^*$ given by

$$\begin{aligned} n^* &= \sigma \mathcal{W}\left(\frac{a}{\sigma^2}\mathrm{e}^{b/\sigma^2}\right) \\ &\approx \sigma\left(\frac{b}{\sigma^2}\log\left(\frac{a}{\sigma^2}\right) - \log\left(\frac{b}{\sigma^2}\log\left(\frac{a}{\sigma^2}\right)\right)\right) \\ &= \frac{b}{\sigma}\log\left(\frac{a}{\sigma^2}\right) - \sigma\log\left(\frac{b}{\sigma^2}\log\left(\frac{a}{\sigma^2}\right)\right), \end{aligned} \tag{3.11}$$

where $\mathcal{W}(\cdot)$ denotes the Lambert function. The accuracy of this approximation is controlled by $\delta$. Reference [38] provides a comprehensive analysis on the maximum error value of the truncated sum (3.10), finding that the infinite sum is well-approximated by taking a manageable number of terms. In particular, we found that for our problem specifications, taking 100 terms was sufficient to ensure an approximation with error less than 0.1 percent.

### 3.1.2 Accelerated Wirtinger Flow with Score-based Image Prior

For accelerating the WF algorithm, we followed the implementation of [111] as its convergence guarantee was proved. Assuming that the true score function can be learned properly, when we have a trained score function $s_{\boldsymbol{\theta}}(\boldsymbol{x}, \sigma)$ by applying (2.4), the gradient descent algorithm for MAP estimation (2.1) has the form: $\boldsymbol{x}_{t+1} = \boldsymbol{x}_t - \mu(\nabla g(\boldsymbol{x}_t) + s_{\boldsymbol{\theta}}(\boldsymbol{x}_t, \sigma_k))$. Algorithm 1 summarizes our proposed AWFS algorithm. (Supplement shows the vanilla version without acceleration.) Similar to Langevin dynamics, we choose $\sigma_k$ to be a descending sequence of noise levels. In practice, we generally use each noise level a fixed number of times, with geometrically spaced noise levels between some lower and upper bound. The step size factor $\beta$ in Algorithm 1 can be selected empirically, but we show that the Lipschitz constant of the gradient $\nabla g_{\mathrm{PG}}(\boldsymbol{x}_t) + s_{\boldsymbol{\theta}}(\boldsymbol{x}_t, \sigma_k)$ exists as demonstrated in Theorem 2 (see proof in Supplement); hence with sufficiently small step size $\beta$, the sequence generated by Algorithm 1 will converge to a critical point of the posterior distribution in (2.1).

We assume that the data allows the neural network to learn the score function well, *i.e.*, $s_{\boldsymbol{\theta}}(\boldsymbol{x}, \sigma) \approx \nabla \log(p_\sigma(\boldsymbol{x}))$, where $p_\sigma(\boldsymbol{x}) = p(\boldsymbol{x}) \circledast \mathcal{N}(0, \sigma^2)$, where $\circledast$ denotes (circular) convolution. Supplement shows that $\nabla \log(p_\sigma(\boldsymbol{x}))$ is Lipschitz continuous on $[-C, C]^N$.

**Algorithm 1** Proposed AWFS method for PR: accelerated WF with score-based image prior.

**Require:** Measurement $\boldsymbol{y}$, system matrix $\boldsymbol{A}$, momentum factor $\eta_0 = 1$, step size factor $\beta$, truncation operator $\mathcal{P}_C(\cdot) \to [0, C]$; initial image $\boldsymbol{x}_0$, initial auxiliary variables $\boldsymbol{z}_0 = \boldsymbol{w}_0 = \boldsymbol{v}_0 = \boldsymbol{x}_0$, initialize $\sigma_1 > \sigma_2 > \cdots > \sigma_K$.

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

  **for** $t = 1 : T$ **do**

    Set step size $\mu = \beta\sigma_k^2$.

    Set $\Delta z_{t,k} = \frac{\eta_{t-1,k}}{\eta_{t,k}}(\boldsymbol{z}_{t,k} - \boldsymbol{x}_{t,k})$.

    Set $\Delta x_{t,k} = \frac{\eta_{t-1,k}-1}{\eta_{t,k}}(\boldsymbol{x}_{t,k} - \boldsymbol{x}_{t-1,k})$.

    Set $\boldsymbol{w}_{t,k} = \mathcal{P}_C\left(\boldsymbol{x}_{t,k} + \Delta z_{t,k} + \Delta x_{t,k}\right)$.

    Compute $s_{\boldsymbol{\theta}}(\boldsymbol{x}_{t,k}, \sigma_k)$ and $s_{\boldsymbol{\theta}}(\boldsymbol{w}_{t,k}, \sigma_k)$.

    Set $\boldsymbol{z}_{t+1,k} = \boldsymbol{w}_{t,k} - \mu\left(\nabla g_{\mathrm{PG}}(\boldsymbol{w}_{t,k}) + s_{\boldsymbol{\theta}}(\boldsymbol{w}_{t,k}, \sigma_k)\right)$.

    Set $\boldsymbol{v}_{t+1,k} = \boldsymbol{x}_{t,k} - \mu\left(\nabla g_{\mathrm{PG}}(\boldsymbol{x}_{t,k}) + s_{\boldsymbol{\theta}}(\boldsymbol{x}_{t,k}, \sigma_k)\right)$.

    Set $\eta_{t+1,k} = \frac{1}{2}\left(1 + \sqrt{1 + 4\eta_{t,k}^2}\right)$.

    Choose weight factor $\gamma_{t,k}$ (see Theorem 2).

    Set $\boldsymbol{x}_{t+1,k} = \mathcal{P}_C\left(\gamma_{t,k}\boldsymbol{z}_{t+1,k} + (1 - \gamma_{t,k})\boldsymbol{v}_{t+1,k}\right)$.

  **end for**

**end for**

Return $\boldsymbol{x}_{T,K}$.

Using $p_\sigma(\boldsymbol{x})$, we define the smoothed posterior as

$$p_\sigma(\boldsymbol{x}|\boldsymbol{A}, \boldsymbol{y}, \bar{\boldsymbol{b}}, \boldsymbol{r}) \propto p(\boldsymbol{y}|\boldsymbol{A}, \boldsymbol{x}, \bar{\boldsymbol{b}}, \boldsymbol{r}) p_\sigma(\boldsymbol{x}). \tag{3.12}$$

**Theorem 2** *For a smooth density function $p_{\sigma_k}(\boldsymbol{x})$ that has finite expectation with $\sigma_k > 0$, the Lipschitz constant of $\nabla g_{\mathrm{PG}}(\boldsymbol{x}_{t,k}) + s_{\boldsymbol{\theta}}(\boldsymbol{x}_{t,k}, \sigma_k)$ exists when each element in $\boldsymbol{x}_{t,k}$ satisfies $0 \le |x_j| \le C$ for each j. Furthermore, if the weighting factor $\gamma_{t,k}$ is set to 0 if $p_{\sigma_k}(\boldsymbol{v}_{t+1}|\boldsymbol{y}, \boldsymbol{A}, \bar{\boldsymbol{b}}) \le p_{\sigma_k}(\boldsymbol{z}_{t+1}|\boldsymbol{y}, \boldsymbol{A}, \bar{\boldsymbol{b}})$ and 1 otherwise, then with sufficiently small $\beta$, the inner iteration sequence $\{\boldsymbol{x}_{t,k}\}_{t=1}^T$ generated by Algorithm 1 is bounded, and any accumulation point of that sequence as $t \to \infty$ is a critical point of the smoothed posterior distribution $p_{\sigma_k}(\boldsymbol{x}|\boldsymbol{y}, \boldsymbol{A}, \bar{\boldsymbol{b}}, \boldsymbol{r})$ in* (3.12).

**Sketch of Proof:** By Lipschitz continuity of $\log(p_\sigma(\boldsymbol{x}))$, and from the design of Algorithm 1, $\boldsymbol{x}_{t,k}$ and $\boldsymbol{w}_{t,k}$ are both bounded between $[-C, C]$ for all $t, k$, so the Lipschitz constant $\mathcal{L}^*$ of $\nabla g_{\mathrm{PG}}(\cdot) + s_{\boldsymbol{\theta}}(\cdot)$ exists. With the stepsize $\mu$ satisfying $0 < \mu < \frac{1}{\mathcal{L}^*}$, and the weighting factor $\gamma_{t,k} \in \{0, 1\}$ being chosen according to the higher posterior probability between $p_{\sigma_k}(\boldsymbol{z}|\boldsymbol{A}, \boldsymbol{y}, \bar{\boldsymbol{b}}, \boldsymbol{r})$ and $p_{\sigma_k}(\boldsymbol{v}|\boldsymbol{A}, \boldsymbol{y}, \bar{\boldsymbol{b}}, \boldsymbol{r})$ (see [111]), we satisfy all conditions in

Theorem 1 of [111], which establishes the critical-point convergence of the sequence $\boldsymbol{x}_{t,k}$ generated by Algorithm 1 for any $\sigma_k, k = 1, \ldots, K$. Hence the sequence $\boldsymbol{x}_{t,k}$ generated by Algorithm 1 converges as $t \to \infty$ to a critical-point of the posterior $p_{\sigma_k}(\boldsymbol{x}|\boldsymbol{A}, \boldsymbol{y}, \bar{\boldsymbol{b}}, \boldsymbol{r})$ for any $\sigma_k$. Supplement shows the full proof of Theorem 2.

Theorem 2 states that at each iteration, we should choose $\gamma_{t,k} \in \{0, 1\}$ according to the higher posterior probability to ensure convergence. However, we found empirically that setting $\gamma_{t,k} = 0.5$ also led to convergent sequences (see Fig. 3.9), so we use this practical version for all the results in Section 3.2. An alternative to our score-based image prior approach would be to alternate data fidelity updates with the sampling step of a diffusion model based on the denoising diffusion probabilistic models (DDPM) framework [158]. Here, we implement and present the results of a slightly altered version (details shown in the Supplement). There are two main changes for this version: first, we initialize with the result of a baseline reconstruction method (in this case, the unregularized Poisson method), rather than pure noise; second, we use the PG likelihood to enforce data fidelity as opposed to the Gaussian likelihood. The former change enables the latter change to be feasible, as it reduces the required number of sampling steps by a factor of 30, allowing for the expensive computation of the PG likelihood each step.

## 3.2 Experiment

### 3.2.1 Experiment Settings

**Data**. We tested all algorithms on three datasets: 162 histopathology images related to breast cancer [2] (train/val/test is 122/20/20); 920 images from CelebA dataset [125] (train/val/test is 800/100/20); and 720 images from a homemade CT-density dataset (train/val/test is 600/100/20). The CT-density dataset was generated from single-photon emission computerized tomography (SPECT)/CT images for Yttrium-90 radionuclide therapy after applying the CT-to-density calibration curve [114]. Although the size of training datasets are relatively small compared to typical datasets such as ImageNet or LSUN [73, 168] that have millions of images, we do not require the score functions to learn image priors strong enough to generate realistic images from white Gaussian noise; rather, it suffices for the priors to be able to denoise moderately noisy images.

**System Model**. The system matrix is based on the discrete Fourier transform of the concatenation of the true image $\boldsymbol{x}$, a blank image $\mathbf{0}$ and a reference image $\boldsymbol{r}$ with scaling and oversampling. We set the scaling factor $\alpha$ to be in the range $[0.02, 0.035]$ so that the

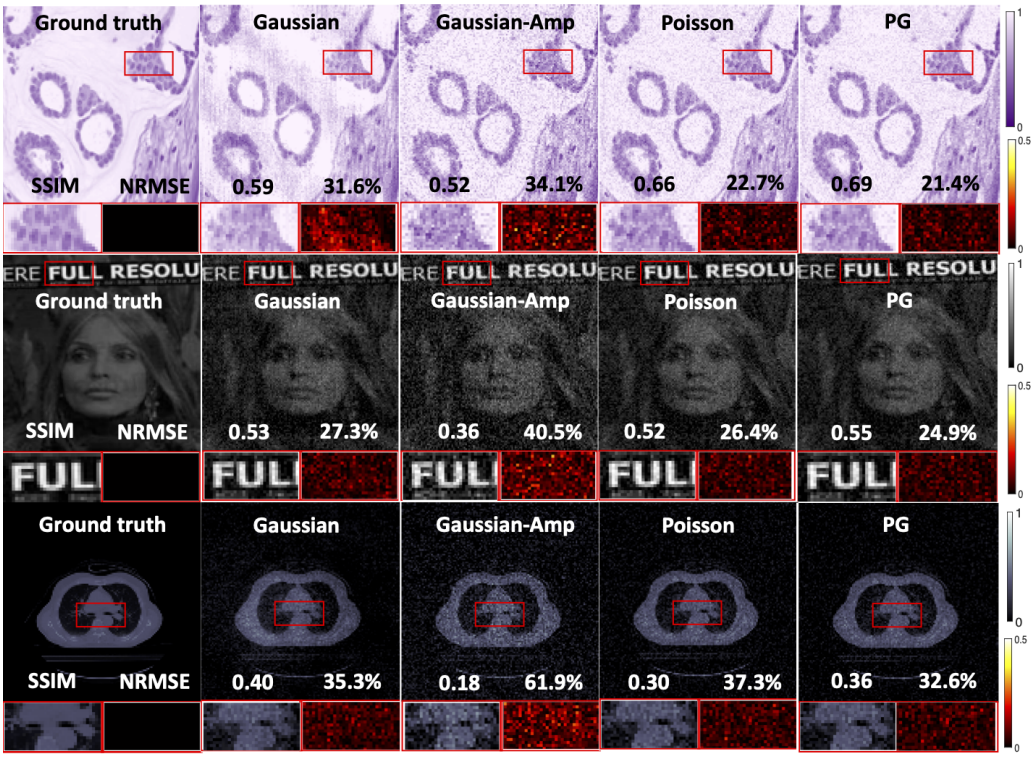



**FIG** 3.2 – Reconstructed images by unregularized methods (Gaussian, Gaussian-Amplitude, Poisson and Poisson-Gaussian) on Histopathology dataset [2], celebA dataset [125] and CT-density dataset. The bottom left/right subfigures correspond to the zoomed in area and the error map for each image. We used $\alpha = 0.035$ and $\sigma = 1$.

average counts per pixel range from 6 to 25; the oversampled ratio is set to 2. We set $\boldsymbol{r}$ to be a binary random image similar to what was used in [107]. The standard deviation of the Gaussian read noise added to the measurements $\boldsymbol{y}$ was set as $\sigma \in [0.5, 1.5]$.

**Algorithms**. For unregularized algorithms, we implemented Gaussian WF [28], Poisson WF [12] and Poisson-Gaussian WF. For regularized algorithms, we implemented smoothed total variation (TV) based on the Huber function [84, p. 184] and PnP/RED methods with the DnCNN denoiser [205]: PnP-ADMM [178], PnP-PGM [91], and RED-SD [149]. We also implemented the RED-SD algorithm with "Noise2Self" zero-shot image denoising network [15] (RED-SD-SELF). For diffusion models, we implemented DOLPH [158] and our proposed AWFS. Supplement shows the implementation details of each algorithm. We used spectral initialization [131] for the Gaussian PR and Poisson PR methods; we then used the output results from Poisson PR to initialize other algorithms. We ran all algorithms until the normalized root mean squared error (NRMSE) between consecutive iterations differed by less than 0.01 percent or reached the maximum number of iterations (*e.g.*, 50).

To evaluate the robustness and limitation of these algorithms, we first tuned the parameters for each algorithm at the noise level when $\alpha = 0.030$ and $\sigma = 1$, and then held them fixed throughout all experiments (Table 3.1, Table 3.2, Fig. 3.7 and Fig. 3.8). In practice the ground truths are unknown, so oracle tuning of test datasets is infeasible (though some form of cross validation may be possible). Though the numbers reported could fluctuate after careful refinement, *e.g.*, by performing grid search on tuning parameters, such techniques would potentially impede the algorithm's practical use.

**Network Training**. For PnP denoising networks, we trained all denoisers on different noise levels $\sigma \in \{9, 11, 13, 15\}$ and found that $\sigma = 15$ worked the best on our data. We also used the denoiser scaling technique from [196] to dynamically adjust the performance of all PnP methods. To perform score matching, we applied 20 geometrically spaced noise levels between 0.005 and 0.1 on each of the training images. All networks were implemented in PyTorch and trained on an NVIDIA Quadro RTX 5000 GPU using the ADAM optimizer [99] for 1000 epochs with the best one being selected based on the mean squared error (MSE) validation loss.

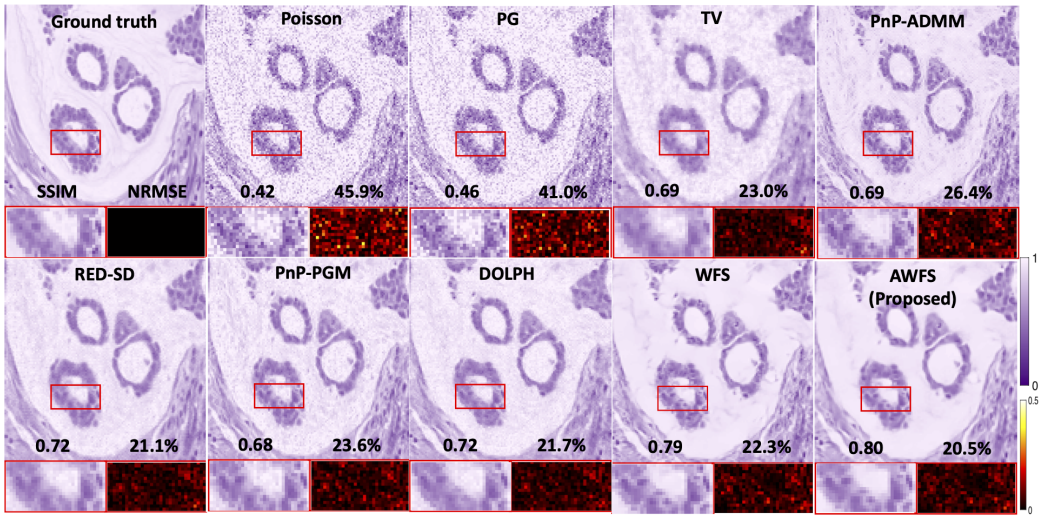



**FIG** 3.3 – Reconstructed images on dataset [2]. The bottom left/right subfigures correspond to the zoomed in area and the error map for each image. We used $\alpha = 0.02$ and $\sigma = 1$.

### 3.2.2 Results

We compared all implemented algorithms both qualitatively, by visualizing the reconstructed images and residual errors, and quantitatively, by computing the NRMSE and

structural similarity index measure (SSIM) [184]. Due to the global phase ambiguity, *i.e.*, all the algorithms can recover the signal only to within a constant phase shift due to the loss of global phase information, we corrected the phase of $\hat{\boldsymbol{x}}$ by $\hat{\boldsymbol{x}}_{\text{corrected}} \triangleq \text{sign}(\langle \hat{\boldsymbol{x}}, \boldsymbol{x}_{\text{true}} \rangle)\, \hat{\boldsymbol{x}}$.

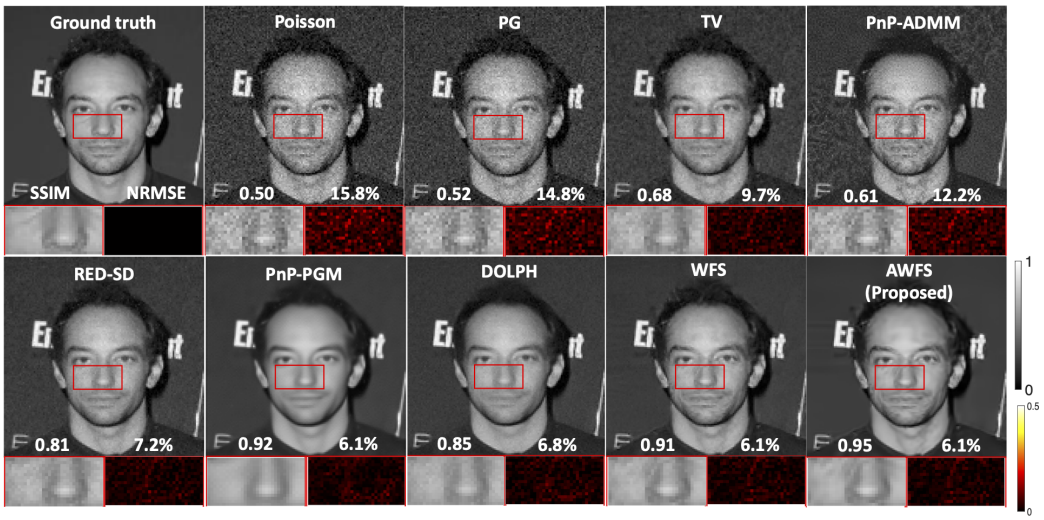



**FIG** 3.4 – Reconstructed images on celebA dataset [125]. The bottom left/right subfigures correspond to the zoomed in area and the error map for each image. We used $\alpha = 0.035$ and $\sigma = 1$.

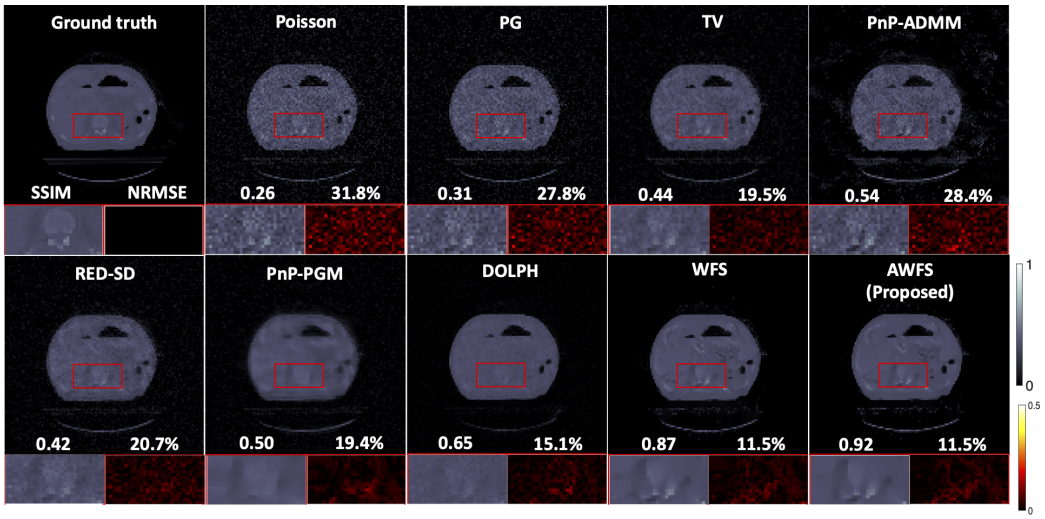



**FIG** 3.5 – Reconstructed images on CT-density dataset. The bottom left/right subfigures correspond to the zoomed in area and the error map for each image. We used $\alpha = 0.035$ and $\sigma = 1$.

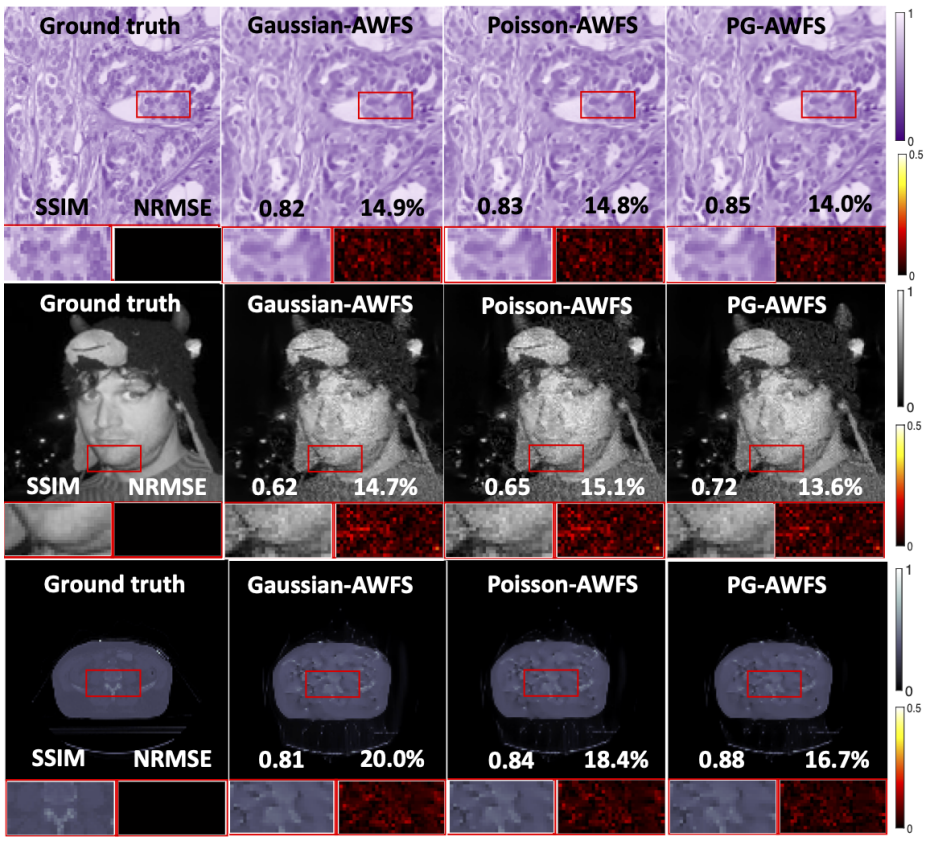


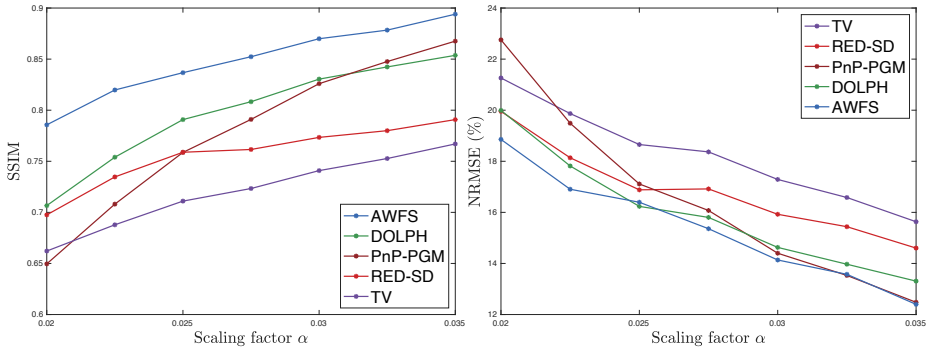

**FIG** 3.6 – Reconstructed images by Gaussian, Poisson and Poisson-Gaussian log-likelihood model with AWFS image prior. Tested on Histopathology dataset [2], celebA dataset [125] and CT-density dataset. The bottom left/right subfigures correspond to the zoomed in area and the error map for each image. $\alpha$ and $\sigma$ were set to 0.025 and 1, respectively.

Fig. 3.2 shows experiments of running unregularized methods based on different noise models on the histopathology, CelebA, and CT density datasets. For comparison, we ran the unregularized methods with a Gaussian only noise model, Poisson only, and PG noise model.

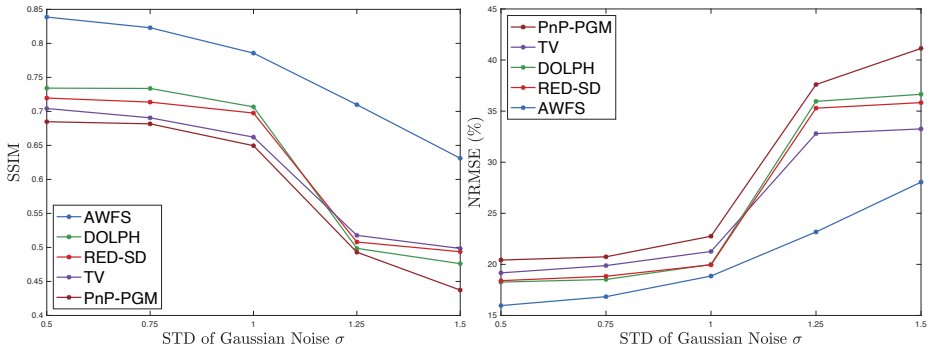
Fig. 3.3, Fig. 3.4 and Fig. 3.5 visualize reconstructed images generated by algorithms mentioned in the previous section. The WF with PG likelihood outperforms WF with Poisson likelihood with a consistently higher SSIM and lower NRMSE. Of the regularized algorithms with PG likelihood, our proposed AWFS had less visual noise and achieved greater detail recovery compared to other methods, as evidenced by the zoomed-in area in these figures. Fig. 3.6 shows that for a variety of datasets, when combined with the AWFS method, while the Poisson only and Gaussian only models lead to reasonable reconstructions, the PG noise model leads to the highest quality image. For all three datasets

**TBL** 3.1 – SSIM and NRMSE for Poisson and Poisson-Gaussian likelihoods. Results were averaged across 7 different noise levels by varying $\alpha \in 0.02 : 0.005 : 0.035$ in (3.1), with $\sigma = 1$.

| Likelihood | Unregularized (SSIM/NRMSE) | | DOLPH (SSIM/NRMSE) | | AWFS (SSIM/NRMSE) | |
|---|---|---|---|---|---|---|
| DataSet: Histopathology [2] | | | | | | |
| Gaussian [28] | 0.52 ± 0.18 | 41.2 ± 25.3 | 0.76 ± 0.07 | 18.0 ± 3.0 | 0.84 ± 0.06 | 16.2 ± 3.7 |
| Poisson [12] | 0.54 ± 0.18 | 31.7 ± 10.2 | 0.72 ± 0.13 | 19.5 ± 6.1 | 0.83 ± 0.06 | 16.2 ± 3.7 |
| Poisson-Gaussian | 0.57 ± 0.18 | 28.9 ± 9.0 | 0.80 ± 0.06 | 16.0 ± 2.9 | **0.85 ± 0.05** | **15.4 ± 3.7** |
| DataSet: CelebA [125] | | | | | | |
| Gaussian [28] | 0.31 ± 0.09 | 55.6 ± 13.9 | 0.70 ± 0.12 | 14.5 ± 17.4 | 0.72 ± 0.16 | 15.3 ± 11.8 |
| Poisson [12] | 0.39 ± 0.10 | 24.5 ± 11.4 | 0.61 ± 0.12 | 15.6 ± 10.6 | 0.72 ± 0.16 | 15.2 ± 11.8 |
| Poisson-Gaussian | 0.42 ± 0.10 | 21.8 ± 9.1 | 0.71 ± 0.11 | **13.7 ± 11.1** | **0.74 ± 0.15** | 14.8 ± 11.9 |
| DataSet: CT-Density | | | | | | |
| Gaussian [28] | 0.29 ± 0.09 | 50.5 ± 8.0 | 0.51 ± 0.12 | 22.4 ± 3.9 | 0.82 ± 0.11 | 19.1 ± 4.8 |
| Poisson [12] | 0.19 ± 0.06 | 48.9 ± 13.1 | 0.38 ± 0.11 | 25.6 ± 7.5 | 0.84 ± 0.08 | 17.8 ± 4.3 |
| Poisson-Gaussian | 0.24 ± 0.06 | 40.8 ± 9.5 | 0.55 ± 0.08 | 20.0 ± 3.3 | **0.88 ± 0.05** | **16.4 ± 3.7** |




(a) SSIM varying $\alpha$. $\sigma$ was set to 1. (b) NRMSE varying $\alpha$. $\sigma$ was set to 1.




(c) SSIM varying $\sigma$; $\alpha$ was set as 0.02. (d) NRMSE varying $\sigma$; $\alpha$ was set as 0.02.

**FIG** 3.7 – Comparison of SSIM and NRMSE varying scaling factor $\alpha \in [0.02, 0.035]$ and STD of Gaussian noise $\sigma \in [0.25, 1.5]$ defined in (3.1).

**TBL** 3.2 – SSIM and NRMSE using Poisson Gaussian likelihood with different regularization/image prior approaches. Results were averaged across 7 different noise levels by varying $\alpha \in 0.02 : 0.005 : 0.035$ in (3.1). WFS$^*$ runs the same number of iterations as AWFS whereas WFS$^\dagger$ runs more iterations until convergence.

| Dataset | Histopathology [2] | | CelebA [125] | | CT-Density | |
|---|---|---|---|---|---|---|
| Methods | SSIM | NRMSE (%) | SSIM | NRMSE (%) | SSIM | NRMSE (%) |
| Unregularized | 0.57 ± 0.18 | 28.9 ± 9.0 | 0.42 ± 0.10 | 21.8 ± 9.1 | 0.24 ± 0.06 | 40.8 ± 9.5 |
| RED-SD-SELF [15] | 0.66 ± 0.13 | 21.9 ± 4.5 | 0.60 ± 0.09 | 15.9 ± 10.6 | 0.34 ± 0.04 | 28.1 ± 4.1 |
| PnP-ADMM [178] | 0.71 ± 0.11 | 20.7 ± 4.2 | 0.56 ± 0.08 | 16.7 ± 8.1 | 0.55 ± 0.03 | 31.2 ± 2.7 |
| TV regularizer | 0.72 ± 0.11 | 18.2 ± 3.9 | 0.64 ± 0.07 | 14.4 ± 8.6 | 0.41 ± 0.03 | 23.7 ± 2.8 |
| RED-SD [149] | 0.76 ± 0.09 | 16.8 ± 3.6 | 0.69 ± 0.11 | 13.9 ± 10.9 | 0.38 ± 0.04 | 25.9 ± 4.0 |
| PnP-PGM [91] | 0.78 ± 0.11 | 16.5 ± 4.5 | **0.74 ± 0.14** | **13.5 ± 11.3** | 0.42 ± 0.07 | 24.6 ± 4.4 |
| DOLPH [158] | 0.80 ± 0.06 | 16.0 ± 2.9 | 0.71 ± 0.11 | 13.7 ± 11.1 | 0.55 ± 0.08 | 20.0 ± 3.3 |
| WFS$^*$ | 0.76 ± 0.12 | 18.2 ± 5.5 | 0.63 ± 0.16 | 16.9 ± 11.8 | 0.53 ± 0.17 | 21.3 ± 7.6 |
| WFS$^\dagger$ | 0.83 ± 0.06 | 16.2 ± 4.0 | 0.70 ± 0.16 | 15.7 ± 11.8 | 0.74 ± 0.13 | 17.3 ± 4.8 |
| AWFS (Proposed) | **0.85 ± 0.05** | **15.4 ± 3.7** | **0.74 ± 0.15** | 14.8 ± 11.9 | **0.88 ± 0.05** | **16.4 ± 3.7** |

shown, when used in conjunction with our AWFS method, including both Poisson and Gaussian likelihoods results in the highest quality reconstruction both in terms of quantitative metrics as well as visually. Thus, although the score function provides a useful prior for recovering an image when the measurement is very noisy, a proper noise model is also crucial to a high quality reconstruction.

For quantitative evaluations, Table 3.1 illustrates the effect of using our proposed PG likelihood as compared to the simpler Poisson likelihoods. We did not run the Gaussian likelihood with DOLPH or AWFS due to the abysmal performance with this likelihood. In all cases, usage of the PG likelihood results in improved image quality in terms of both metrics. Table 3.2 consists of experiments using the PG likelihood and shows the efficacy of the proposed AWFS method over other methods. In particular, our AWFS had superior quantitative performance over all other compared methods on the histopathology and CT-density datasets; in contrast, the PnP-Prox showed the lowest NRMSE on celebA dataset. This is likely due to higher randomness in celebrity faces because the effectiveness of generative models can vary depending on the dataset used. Thus, when provided with a small amount of training data with high randomness, image denoising models (DnCNN) may be more effective than generative models.

We also tested the robustness of the leading algorithms in Table 3.2, by varying both scaling factor $\alpha$ and STD of Gaussian noise $\sigma$. Fig. 3.7 and Fig. 3.8 illustate results, where

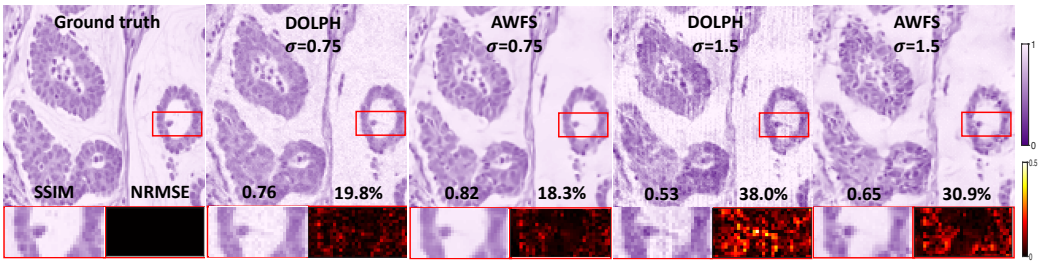
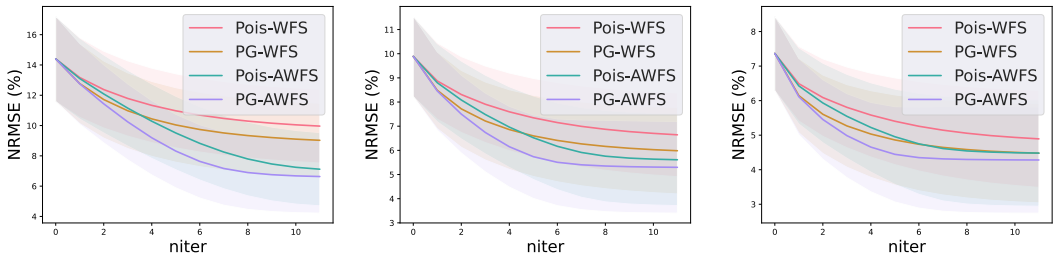


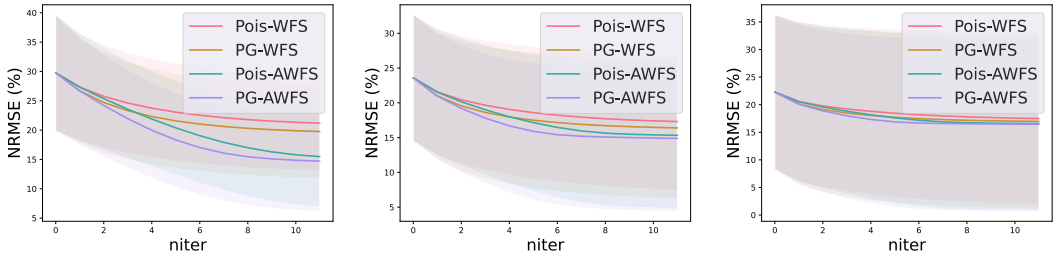

**FIG** 3.8 – Reconstructed images by DOLPH [158] and our proposed AWFS method under different $\sigma$ values, for $\alpha = 0.02$.

our AWFS algorithm had the highest SSIM and lowest NRMSE. In Fig. 3.8, AWFS demonstrated minimal variations in SSIM and NRMSE metrics than DOLPH as evidenced by the smaller discrepancies in SSIM (0.17 vs. 0.23) and NRMSE (12.6% vs. 18.2%) when $\sigma$ varies from 0.75 to 1.5. Fig. 3.9 compares the convergence rate of AWF vs. WF for the Poisson and PG likelihood, respectively. Under a variety of noise levels, AWF consistently converged faster than WF in terms of number of iterations.

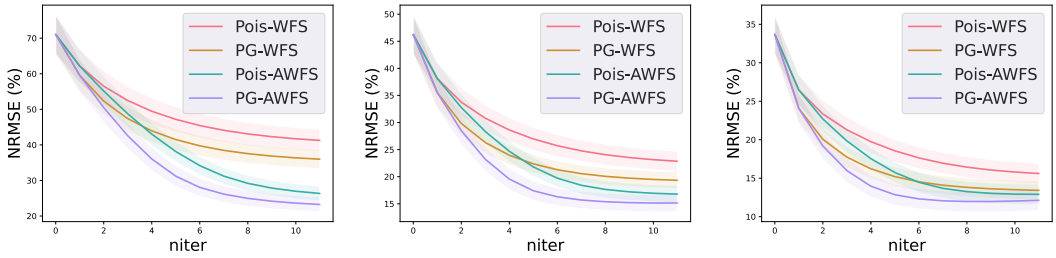
It is a known property of diffusion models that they can produce images with hallucinated features if the measurements are insufficiently informative. In the case of low-count phase retrieval with serious corruptions of both Poisson and Gaussian noise, as is investigated here, the measurement is highly corrupted and contains magnitude-only measurements of the original signal. Thus, it may be difficult for the diffusion models to avoid some otherwise realistic hallucinations if the data consistency is not strong enough to guide the model away from such hallucinations. On the other hand, if the measurements are less corrupted, then the data consistency should be strong enough to avoid such hallucinations. Fig. 3.10 provides examples of this for the CT image dataset via a comparison of the reconstruction quality of the AWFS method over a range of count levels. With the lowest scaling factor, *e.g.*, $\alpha = 0.02$, the measurements were seriously corrupted with noise, and the method may hallucinate some features. However, at higher count level, *e.g.*, $\alpha = 0.05$, there is enough information in the measurement to enforce consistency and avoid noticeable hallucinations. We performed the same experiment twice with different noisy initializations and all other parameters held equal to demonstrate robustness of the method under different initializations.




(a) Histopathology dataset: $\alpha = 0.02$. (b) Histopathology dataset: $\alpha = 0.0275$. (c) Histopathology dataset: $\alpha = 0.035$.




(d) CelebA dataset: $\alpha = 0.02$. (e) CelebA dataset: $\alpha = 0.0275$. (f) CelebA dataset: $\alpha = 0.035$.




(g) CT-density dataset: $\alpha = 0.02$. (h) CT-density dataset: $\alpha = 0.0275$. (i) CT-density dataset: $\alpha = 0.035$.

**FIG** 3.9 – Comparing AWF vs. WF with NRMSE vs. number of iterations under different noise levels. The curves and shadows represent the mean and standard deviation, respectively.

## 3.3 Discussion

PR has a long-standing history in the field of signal processing and imaging. Pioneering works such as the error reduction (ER) and hybrid input-output (HIO) algorithms by Gerchberg Saxton [64] and Fienup [62] have been proposed to address this problem. These iterative algorithms involve constraints imposed on evaluations between the image domain and frequency domain. However, these methods have limitations in terms of the quality of reconstructed images and their convergence remains uncertain [201]. Another approach to solving PR problems is through compressed sensing and optimization techniques like

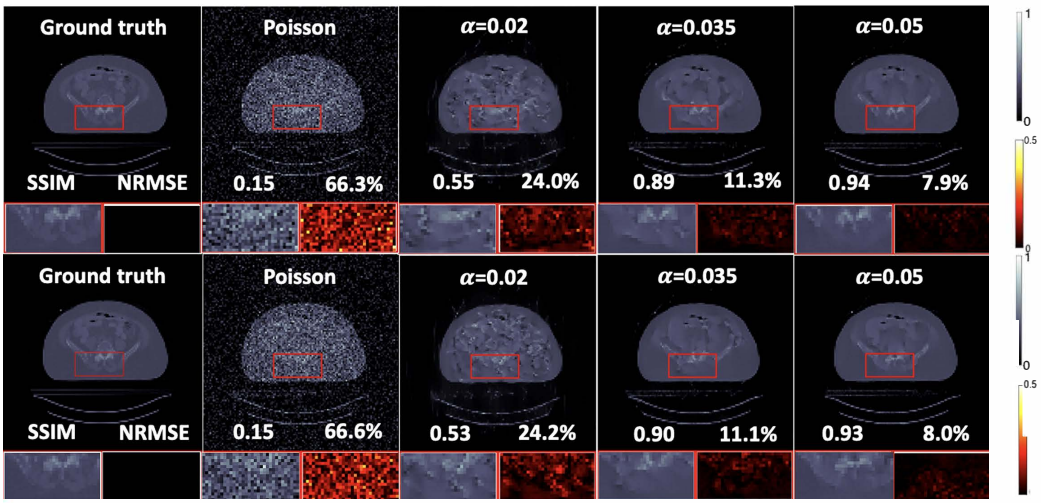



**FIG** 3.10 – Reconstructed images by the unregularized Poisson method (the second column) as well as with the AWFS method for different scaling factors $\alpha$ (third to fifth columns). The top and bottom rows show reconstructions from different measurement realizations.

Wirtinger flow (WF) [28], matrix lifting [30, 27, 157], MM [146] and ADMM [119]. This chapter focuses on the WF algorithm due to being straightforward to incorporate with the DL regularizer for the image prior. The likelihood modelling of the noise statistics existing in the measurement is also critical. Previous studies have primarily focused on modelling either Gaussian or Poisson likelihood only, but in practical scenarios, both types of noise are often encountered. Therefore, this chapter contributes to a more practical perspective of addressing the holographic PR problem by using a PG likelihood and incorporating state-of-the-art deep learning image priors. In the case where the measurement is contaminated with Poisson and Gaussian noise, the speedup in reconstruction is crucial, as the bottleneck of our algorithm is in computing the PG likelihood. Additionally, though it is viable to perform a large number of neural network evaluations to perform image reconstruction, it is unrealistic to compute a similarly large number of PG likelihoods. Thus, we perform acceleration in WF algorithm following [111], which guarantees convergence to a critical point for the Holographic PR problems.

In our evaluation of three datasets, we consistently observed that the use of PG likelihood yielded superior performance compared to using either Poisson or Gaussian likelihood alone, as expected. Additionally, the results obtained from the CT-density dataset were generally of lower quality than those from the other two datasets. This can be attributed to lower average counts per pixel (many zero pixels near the image borders). Using a DL image prior can be considered from two perspectives: training a denoiser

or training to learn the density distribution of images. In our work, we applied both approaches and observed that the effectiveness of these methods differed depending on the dataset tested. Specifically, in the Histopathology dataset [2] and the CT-density dataset, where the images share similar structures, the generative models performs better even when trained with limited data. In the case of the CelebA dataset [125], which includes a wide variety of celebrity faces, generative models did not exhibit as strong performance as denoiser methods when trained on limited data. This is likely due to the fact that generating high-quality images is generally more challenging than removing noise from existing images and may necessitate a larger training dataset. The effectiveness of accelerated WF compared to vanilla WF is due to the non-convexity of the PR problem. Although recent advances in geometric landscape analysis of PR can guarantee that all local minimizers are global even with random initialization [25], in practice the measurements are contaminated by noise so that many more measurements are required for the cost function to have a benign geometric landscape.

Despite the promising results achieved with our proposed AWFS approach, there are several limitations of our work. First, the approximate calculation of the infinite sum in (3.2) is accurate but computationally expensive. Future work should seek ways to accelerate this calculation while maintaining accuracy. Second, we did not implement and test the accelerated WF applied on the diffusion posterior sampling (DPS) method [39], for which the network is fine-tuned from a pretrained state-of-the-art diffusion model. This approach has the potential to advance current methods in PR problem and we will investigate it in the future. Another limitation of the proposed method is that it has been demonstrated on measurements that are based on simulations. To further demonstrate the efficacy of the method in a real-world setting, future work should consist of evaluating the accuracy of the methods when run on real measurement data. Finally, our experiments are limited to real-valued images, however, our method can be extended to handle complex-values images by splitting real and imaginary components into separate reconstruction routines with different pretrained neural networks [211]. Addressing these limitations will be the future direction of this work.

## 3.4 Conclusion

We proposed a novel algorithm based on **A**ccelerated **W**irtinger **F**low and **S**core-based image prior (AWFS) for Poisson-Gaussian holographic phase retrieval. Simulation experiments demonstrate that the proposed AWFS method produced the best reconstruction quality both qualitatively and quantitatively and was more robust to various noise levels,

compared to other state-of-the-art methods. Furthermore, we proved that our proposed algorithm has a critical-point convergence guarantee. Therefore, after the method is extended to accommodate complex-valued images, it should have much promise for real-world PR applications.

# CHAPTER 4

# Learning Image Priors through Patch-based Diffusion Models for Solving Inverse Problems

## 4.1 Motivation

Diffusion models require large quantities of training data and vast computational power to be able to generate high resolution images; Song et al. [168] and Ho et al. [73] used several days to weeks of training on over a million training images in the ImageNet [153] and LSUN [200] datasets to generate $256 \times 256$ images. This high cost motivates the research on improved training efficiency for diffusion models, such as fine-tuning an existing checkpoint on a different dataset [138, 42] to reduce the required training time and data. However, this strategy restricts the range of usable network architectures and requires the existence of a pretrained network, which limits the range of applications. Besides the demanding training data and computational cost, diffusion models also struggle in very large-scale problems, such as very high-resolution or 3D images.

Our proposed method tackles these challenges in a unified way by training diffusion models on patches of images, as opposed to whole images (see Fig. 4.1). We provide the location of the randomly extracted patch to the network to help it learn global image properties. Since each training image contains many patches, the required size of the training

This chapter based on [82]. The related code is at `https://github.com/jasonhu4/PaDIS`.

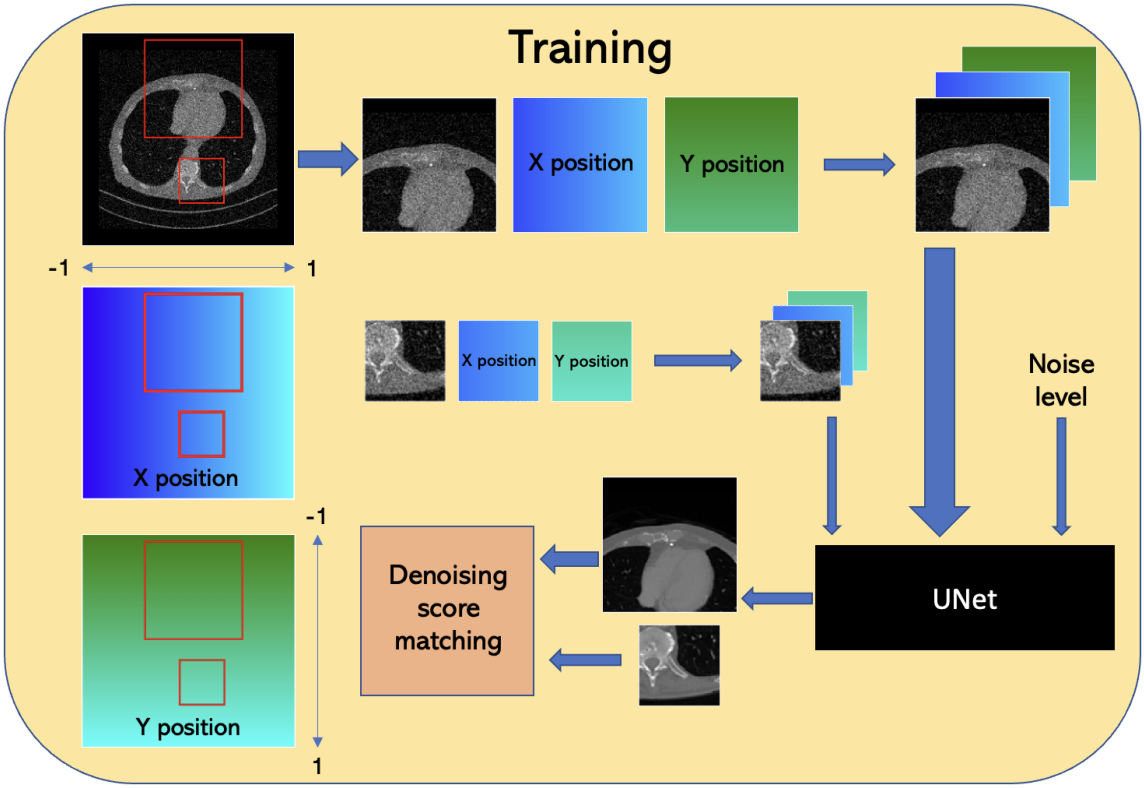



FIG 4.1 – Training the proposed Patch Diffusion Inverse Solver (PaDIS) method. Different sized patches are used in each training iteration. The X and Y position arrays have the same size as the patch and consist of the normalized X and Y coordinates of each pixel of the patch.

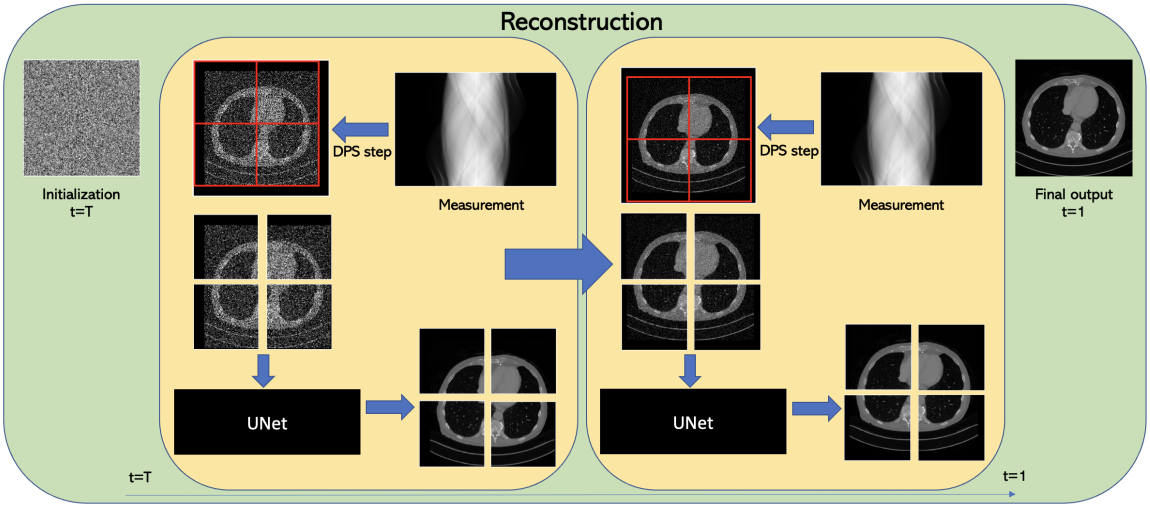



FIG 4.2 – Overview of reconstruction process for the proposed Patch Diffusion Inverse Solver (PaDIS) method. Starting at $t = T$, at each iteration we choose a random partition of the zero-padded image and use the neural network trained on patches to get the score function of the entire image. Due to the shifting patch locations, the output image has no boundary artifacts.

dataset is greatly reduced, from the millions usually needed to generate high-quality images to only a couple thousand or even several hundred. The required memory and training time is also reduced because it is never necessary to backpropagate the whole image through the network. Our proposed method allows for a flexible network architecture and only requires it to accept images of any size, a property true of many UNets [73], so there is much more flexibility in the architecture design than fine-tuning methods.

At inference time (see Fig. 4.2), by first zero padding the image, the proposed approach partitions it into patches in many different ways (see Fig. 4.3), eliminating the boundary artifacts between patches that would appear if non-overlapping patches were used. We develop a method to express the distribution of the whole image in terms of the patch distribution that is learned by the proposed patch-based network. By incorporating positional information of patches, this framework allows us to compute the score function of the whole image without ever inputting the whole image into the network. Unlike previous patch-based works that may be task-specific [190, 191, 142], the prior defined by our approach may be treated in a black box manner and then paired with any other stochastic differential equation (SDE) solver to sample from the prior, or with any general-purpose inverse problem solver to perform image reconstruction. We conduct experiments on multiple datasets and different inverse problems and demonstrate that the proposed method is able to synthesize the patches to produce reasonably realistic images and very accurate reconstructions for inverse problems. Furthermore, PaDIS provides a promising avenue for which generation and inverse problem solving of very large and high-dimensional images may be tackled in the future. (See Ch. 6.)

In summary, our main contributions are as follows:

- We provide a theoretical framework whereby a score function of a high-resolution high-dimensional image is learned purely through the score function of its patches.
- The proposed method greatly reduces the amount of memory and training data needed compared to traditional diffusion models.
- The trained network has great flexibility and can be used with many existing sampling algorithms and is the first patch-based model that can solve inverse problems in an unsupervised manner.
- We perform experiments on a variety of inverse problems to show superior image quality over whole image diffusion model methods while being far less resource heavy.

## 4.2 Methods

To be able to solve large 2D imaging problems as well as 3D and 4D inverse problems, our goal is to develop a model for $p(\boldsymbol{x})$ that does not require inputting the entire image $\boldsymbol{x}$ into any neural network. We would like to simply partition $\boldsymbol{x}$ into nonoverlapping patches, learn the distribution of each of the patches, and then piece them together to obtain the distribution of $\boldsymbol{x}$. However, this would result in boundary artifacts between the patches. Directly using overlapping patches would result in sections of the image covered by multiple patches to be updated multiple times, which is inconsistent with the theory of diffusion models. Ideally, we would like to use nonoverlapping patches to update $\boldsymbol{x}$, but with a variety of different patch tiling schemes so that boundaries between patches do not remain the same through the entire diffusion process.

To accomplish this task, we zero pad $\boldsymbol{x}_0$ and learn the distribution of the resulting zero padded image. More precisely, consider an $N \times N$ image $\boldsymbol{x}_0$ and let $1 \leq P < N$ be an integer denoting the patch size and let $k = \lfloor N/P \rfloor$. Then $\boldsymbol{x}_0$ could be covered with a $(k+1) \times (k+1)$ nonoverlapping patch grid but that would result in $(k+1)P - N$ additional rows and columns for the patches. Hence, we zero pad $\boldsymbol{x}_0$ on all four sides by $M = (k+1)P - N$ to form a new image with $N + 2M$ rows and columns. With slight abuse of notation, we also denote this larger image by $\boldsymbol{x}$. Let $i, j$ be positive integers between 1 and $M$ inclusive. Fig. 4.3 illustrates that we may partition $\boldsymbol{x}$ into $(k+1)^2 + 1$ regions as follows: $(k+1)^2$ of the regions consist of evenly chopping up the square consisting of rows $i$ through $i + N + P$ and columns $j$ through $j + N + P$ into a $k+1$ by $k+1$ grid, with each such partition being $P \times P$, and the last region consists of the remaining bordering part of $\boldsymbol{x}$ that is not included in the first $(k+1)^2$ regions. This last region will always be entirely zeros, and the $(k+1)^2$ square patches fully cover the central $N \times N$ image.

Each pair of integers $i$ and $j$ correspond to one possible partition, so when we let $i$ and $j$ range through all the possible values, our proposal is to model the distribution of $\boldsymbol{x}$ as the following product distribution:

$$p(\boldsymbol{x}) = \prod_{i,j=1}^{i,j=M} p_{i,j,B}(\boldsymbol{x}_{i,j,B}) \prod_{r=1}^{(k+1)^2} p_{i,j,r}(\boldsymbol{x}_{i,j,r})/Z, \tag{4.1}$$

where $\boldsymbol{x}_{i,j,B}$ represents the aforementioned bordering region of $\boldsymbol{x}$ that depends on the specific values of $i$ and $j$, $p_{i,j,B}$ is the probability distribution of that region, $\boldsymbol{x}_{i,j,r}$ is the $r$th $P \times P$ patch when using the partitioning scheme corresponding to the values of $i$ and $j$, $p_{i,j,r}$ is the probability distribution of that region, and $Z$ is an appropriate scaling

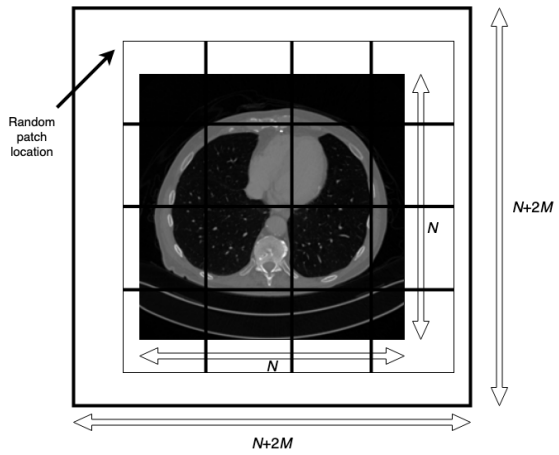



**FIG** 4.3 – Schematic for zero padding and partitioning image into patches

factor. Generative models based on products of patch probabilities have a long history in the Markov random field literature; see ***S*** A.1.5.

The score function of the entire image follows directly from (4.1):

$$\nabla \log p(\boldsymbol{x}) = \sum_{i,j=1}^{i,j=M} \left( \boldsymbol{s}_{i,j,B}(\boldsymbol{x}_{i,j,B}) + \sum_{r=1}^{(k+1)^2} \boldsymbol{s}_{i,j,r}(\boldsymbol{x}_{i,j,r}) \right). \tag{4.2}$$

Thus, we have expressed the score function of the whole image $\boldsymbol{x}$ as sums of scores of patches $\boldsymbol{x}_{i,j,r}$ and the border $\boldsymbol{x}_{i,j,B}$. The former can be learned through score matching as in ***S*** 4.2.1. For the latter, by construction, if $\boldsymbol{x}$ is a zero padded image then $\boldsymbol{x}_{i,j,B}$ is everywhere zero. Thus, for all $\boldsymbol{x}$, $D(\boldsymbol{x}_{i,j,B}) = \mathbb{E}[\boldsymbol{x}_{i,j,B}, t = 0 | \boldsymbol{x}_{i,j,B}, t = T]$ is everywhere zero, where $D$ represents the denoiser function. Then the computation of its score function is trivial by Tweedie's formula [56].

Importantly, unlike previous papers like [183] and [55], our method can compute the score function of the entire image through only inputs of patches to the neural network. This makes it possible to learn a black box function for score functions of large images, where directly training a diffusion model would be infeasible due to memory and time constraints. Furthermore, ***S*** 4.4 shows that in the case of limited data, the large number of patches present in each training image allows the patch-based model to learn a model for the underlying distribution that performs better than whole image models.

### 4.2.1 Patch-wise training

Following the work in [183] and [93], we train a denoiser using the score matching approach. Our neural network $D_\theta(\boldsymbol{x}, \sigma_t)$ accepts the noisy image $\boldsymbol{x}$ and a noise level $\sigma_t$, and

is trained through the following loss function:

$$\mathbb{E}_{t\sim\mathcal{U}(0,T)}\mathbb{E}_{\boldsymbol{x}\sim p(\boldsymbol{x})}\mathbb{E}_{\boldsymbol{\epsilon}\sim\mathcal{N}(0,\sigma_t^2 I)}\|D_\theta(\boldsymbol{x}+\boldsymbol{\epsilon},\sigma_t)-\boldsymbol{x}\|_2^2. \tag{4.3}$$

By Tweedie's formula [56], the score function is given by $s_\theta(\boldsymbol{x},\sigma_t)=(D_\theta(\boldsymbol{x},\sigma_t)-\boldsymbol{x})/\sigma_t^2$. Here, we want to learn the score function of patches $\boldsymbol{x}_{i,j,r}$, so we apply (4.3) to patches of $\boldsymbol{x}$ instead of the entire image. Following [183], we extract patches randomly from the zero-padded image $\boldsymbol{x}$. To incorporate the positional information of the patch, we first define the $x$ positional array as the size $N+2M$ 2D array consisting of the $x$ positions of each pixel of the image scaled between -1 and 1; the $y$ positional array is similarly defined. We then extract the corresponding patches of these two positional arrays and concatenate them along the channel dimension of the noisy image patch as inputs into the network, nothing that noise is not added to the positional arrays.

When directly applying (4.3), it would suffice to fix the patch size $P$ and then train using size $P$ patches exclusively. However, [183] found that by training with patches of varying sizes, training time can be reduced and the neural network learns cross-region dependencies better. Hence, we train both with patches of size $P$ and also patches of *smaller* size, where the size is chosen according to a stochastic scheduling scheme. By using the UNet architecture in [73], the same network can take images of different sizes as input. The Appendix provides details of the experiments.

### 4.2.2 Sampling and reconstruction

The proposed patch-based method for learning $p(\boldsymbol{x})$ may be paired with any method that would otherwise be used for sampling with a full image diffusion model, such as Langevin dynamics [166] and DDPM [73], as well as acceleration methods such as second-order solvers [93] and DDIM [165]. At training time, the network only receives patches of the image as input, along with the positional information of the patch. Nevertheless, we show that when the number of sampling iterations is sufficiently large, the proposed method is still capable of generating reasonably realistic appearing entire images. However, our main goal is solving large-scale inverse problems, not image generation.

Computing $\boldsymbol{s}(\boldsymbol{x})$ via (4.2) would require summing over the score functions of all $(k+1)^2$ patches a total of $M^2$ times (corresponding to the $M^2$ different ways of selecting $i$ and $j$). This method would be prohibitively slow due to the size of $M^2$; instead, for each iteration we randomly choose integers $i$ and $j$ between 1 and $M$ and estimate $\boldsymbol{s}(\boldsymbol{x})$ using just that corresponding term of the outer summation.

**Algorithm 2** Patch Diffusion Inverse Solver (PaDIS)

**Require:** $\sigma_1 < \sigma_2 < \ldots < \sigma_T$, $\epsilon > 0$, $\zeta_i > 0$, $P, M, \boldsymbol{y}$
  Initialize $\boldsymbol{x} \sim \mathcal{N}(0, \sigma_T^2 \boldsymbol{I})$
  **for** $t = T : 1$ **do**
    Sample $z \sim \mathcal{N}(0, \sigma_t^2 \boldsymbol{I})$
    Set $\alpha_t = \epsilon \cdot \sigma_t^2$
    Randomly select integers $i, j \in [1, M]$
    For all $1 \leq r \leq (k+1)^2$, extract patch $\boldsymbol{x}_{i,j,r}$
    Compute $D_{i,j,r} = D_\theta(\boldsymbol{x}_{i,j,r}, \sigma_t)$
    Set $\boldsymbol{s}_{i,j,r} = (D_{i,j,r} - \boldsymbol{x}_{i,j,r})/\sigma_t^2$
    Apply (4.2) to get $\boldsymbol{s} = \boldsymbol{s}(\boldsymbol{x}, \sigma_t)$
    Set $\boldsymbol{x}$ to $\boldsymbol{x} - \zeta_t \nabla_{\boldsymbol{x}_t} \|\boldsymbol{y} - \mathcal{A}(D)\|_2^2$
    Set $\boldsymbol{x}$ to $\boldsymbol{x} + \frac{\alpha_t}{2}\boldsymbol{s} + \sqrt{\alpha_t}\boldsymbol{z}$
  **end for**

For solving inverse problems with diffusion models, there are general algorithms e.g., [46] and [39], as well as more model-specific algorithms, e.g., [95] and [181]. Here, we demonstrate that our method applies to a broad range of inverse problems, and opt to use generalizable methods that do not rely on any special properties (such as the singular value decomposition of the system matrix as in [95], [181]) of the forward operator. We found that DPS [39] in conjunction with Langevin dynamics generally yielded the most stable and high quality results, so we use this approach as our central algorithm. Similar to [39], we chose the stepsize to be $\zeta_i = \zeta / \|\boldsymbol{y} - \mathcal{A}(D(\boldsymbol{x}))\|_2$. To the best of our knowledge, this is the first work that learns a fully patch-based diffusion prior and applies it to solve inverse problems; we call our method **Pa**tch **D**iffusion **I**nverse **S**olver (PaDIS). Computing the score functions of the patches at each iteration is easily parallelized, allowing for fast sampling and reconstruction. Alg. 2 shows the pseudocode for the main image reconstruction algorithm; the appendix shows the pseudocode for the other implemented algorithms.

Finally, we comment on some high-level similarities between our proposed method and [109]; in both cases, the image in question is partitioned into smaller parts in multiple ways. In [109], one of the partitions consists of 2D slices in the x-y direction, and the other partition consists of 2D slices in the x-z direction, whereas for our method, each of the partitions consists of $(k+1)^2$ square patches and the outer border region. For both methods, the score functions of each of the parts are learned independently during training. Then for each sampling iteration, both methods involve choosing one of the partitions, computing the score functions for each of the parts independently, and then

updating the entire image by updating the parts separately. For our approach, the zero-padding method allows for many possible partitions of the image and eliminates boundary artifacts between patches.

## 4.3 A Unified Theory of Patch-based Diffusion Models

### 4.3.1 Motivation

Although patch-based diffusion models have demonstrated success in solving inverse problems [81, 82], certain aspects of the training and reconstruction process lack theoretical justification. Furthermore, multiple other works that appear to use a different framework turn out to share the same framework as these previous two works. This section[1] provides a theoretical justification for much of the training and reconstruction process of several works in the field, as well as a series of experiments that deepen our understanding of the impacts of different types of patch-based diffusion models on inverse problem solving.

### 4.3.2 Methods

We begin by formulating a unified patch framework that encapsulates various methods for combining patch scores into the score of an entire image. To start, let the entire (possibly zero padded) image be denoted by $\boldsymbol{x}$. Then a general patch-based model for a denoiser of $\boldsymbol{x}$ is given by

$$\boldsymbol{D}(\boldsymbol{x}) = \left( \sum_c \boldsymbol{G}_c' \boldsymbol{D}(\boldsymbol{G}_c \boldsymbol{x}) \right) \oslash \left( \sum_c \boldsymbol{G}_c' \boldsymbol{G}_c \mathbf{1} \right), \tag{4.4}$$

where the sums iterate over all (possibly overlapping) patches $c$, $\oslash$ denotes element-wise division, and the $\mathbf{1}$ represents an image of all ones. This model states that to denoise an entire image, we can denoise the patches individually, add together the contributions from each of the patches to the whole image, and then perform pixel-wise division to account for the number of patches that contribute to each pixel. We will show later how PaDIS and [142] fall under this framework.

We start by showing that there is a straightforward equivalence between this equation for denoisers and a similar equation for score functions. For convenience, define $\boldsymbol{M} = \sum_c \boldsymbol{G}_c' \boldsymbol{G}_c \mathbf{1}$. By Tweedie's formula, $\boldsymbol{s}(\boldsymbol{x}) = (\boldsymbol{D}(\boldsymbol{x}) - \boldsymbol{x})/\sigma^2$. Then by using the fact that

[1] This section is not in the paper [82] that forms the basis for the rest of this chapter.

$(\sum_c \boldsymbol{G}_c' \boldsymbol{G}_c \boldsymbol{x}) \oslash \boldsymbol{M} = \boldsymbol{x}$, one can readily use algebra to derive the relationship

$$\boldsymbol{s}(\boldsymbol{x}) = \left( \sum_c \boldsymbol{G}_c' \boldsymbol{s}_V(\boldsymbol{G}_c \boldsymbol{x}) \right) \oslash \boldsymbol{M}, \tag{4.5}$$

where $\boldsymbol{s}_V$ is the score function of a patch.

Following this, if $\boldsymbol{M}$ is a constant $M$ times the matrix of all ones, then the probability distribution of the entire image takes the following form:

$$p(\boldsymbol{x}) = \frac{1}{Z} \left( \prod_c p_c(\boldsymbol{G}_c \boldsymbol{x}) \right)^{1/M}, \tag{4.6}$$

where $Z$ is a scaling constant.

Unfortunately, in the general case where $\boldsymbol{M}$ is not a constant matrix, this "score" function need not correspond to any probability distribution, as illustrated by the following example. Firstly, score functions are the gradient of the (log of a) function, so to simplify things we rewrite (4.5) as

$$\nabla f(\boldsymbol{x}) = \left( \sum_c \boldsymbol{G}_c' \nabla f_c(\boldsymbol{G}_c \boldsymbol{x}) \right) \oslash \boldsymbol{M}, \tag{4.7}$$

for the patch functions $f_c$ and overall function $f$. Now we take the specific example of $\boldsymbol{x}$ containing just three pixels in a row, which we represent as $\boldsymbol{x} = [u\ v\ w]$. Then we define the first patch as the first two pixels and the second patch as the next two pixels. The functions that operate on these two patches are defined as $f_1([u\ v]) = uv$ and $f_2([v\ w]) = v^2 + 2w^2$; their gradients are $\nabla f_1 = (v, u)$ and $\nabla f_2 = (2v, 4w)$. Only the second pixel in $\boldsymbol{x}$ is overlapping, so $\boldsymbol{M} = [1\ 2\ 1]$. Putting it all together gives $\nabla f(\boldsymbol{x}) = (v, \frac{1}{2}u + v, 4w)$. However, the curl of this vector field is readily computed to be $(0, 0, -1/2)$. As this is nonzero, this is not a conservative vector field, meaning it cannot be equal to the gradient of a function. Hence, a "score function" written in the form (4.7) need not have a corresponding probability distribution.

In the general case, this "score" function may not correspond to a probability distribution; nevertheless, we show how the training process can be simplified from the loss from entire image training. By incorporating the positional encoding $c^*$ of patches as an input

to the network, the denoising score matching loss is given by

$$L(\theta) = \mathbb{E}_{t\sim\mathcal{U}(0,T)}\mathbb{E}_{\boldsymbol{x}\sim p(\boldsymbol{x})}\mathbb{E}_{\boldsymbol{\epsilon}\sim\mathcal{N}(0,\sigma_t^2\boldsymbol{I})}\|\boldsymbol{s}_\theta(\boldsymbol{x}+\boldsymbol{\epsilon},\sigma_t)+\boldsymbol{\epsilon}/\sigma_t^2\|_2^2 \tag{4.8}$$

$$= \mathbb{E}_{t\sim\mathcal{U}(0,T)}\mathbb{E}_{\boldsymbol{x}\sim p(\boldsymbol{x})}\mathbb{E}_{\boldsymbol{\epsilon}\sim\mathcal{N}(0,\sigma_t^2\boldsymbol{I})}\|\sum_c \boldsymbol{G}_c'\boldsymbol{s}_V(\boldsymbol{G}_c\boldsymbol{x},c^*;\theta)\oslash\boldsymbol{M}+\boldsymbol{\epsilon}/\sigma_t^2\|_2^2. \tag{4.9}$$

Note that $(\sum_c \boldsymbol{G}_c'\boldsymbol{G}_c\boldsymbol{\epsilon})\oslash\boldsymbol{M}=\boldsymbol{\epsilon}$. So we rewrite the norm inside the expectation as

$$L = \|\sum_c \boldsymbol{G}_c'\boldsymbol{s}_V(\boldsymbol{G}_c\boldsymbol{x},c^*;\theta)\oslash\boldsymbol{M}+\sum_c \boldsymbol{G}_c'\boldsymbol{G}_c\boldsymbol{\epsilon}\oslash\boldsymbol{M}/\sigma_t^2\|_2^2 \tag{4.10}$$

$$= \|(\sum_c \boldsymbol{G}_c'(\boldsymbol{s}_V(\boldsymbol{G}_c\boldsymbol{x},c^*;\theta)+\boldsymbol{G}_c\boldsymbol{\epsilon}/\sigma^2))\oslash\boldsymbol{M}\|_2^2. \tag{4.11}$$

Assuming that the patches collectively cover the entire image, each element of $\boldsymbol{M}$ is at least 1. Thus we have

$$L \le \|\sum_c \boldsymbol{G}_c'(\boldsymbol{s}_V(\boldsymbol{G}_c\boldsymbol{x},c^*;\theta)+\boldsymbol{G}_c\boldsymbol{\epsilon}/\sigma^2)\|_2^2. \tag{4.12}$$

Now we have a term of the form $\|X_1+X_2+\cdots+X_n\|_2^2$, which we can bound using a similar derivation as (6.6) in Ch. 6 (derived in the appendix of Ch. 5):

$$L \le K\sum_c \|\boldsymbol{G}_c'(\boldsymbol{s}_V(\boldsymbol{G}_c\boldsymbol{x},c^*;\theta)+\boldsymbol{G}_c\boldsymbol{\epsilon}/\sigma^2)\|_2^2, \tag{4.13}$$

where $K$ is a constant independent of $\boldsymbol{x}$. Finally, since $\boldsymbol{G}_c'$ is an operator restoring a patch to its location in a full image, it is invariant under the Euclidean norm. Putting everything together, we have

$$L(\theta) \le K\cdot\mathbb{E}_{t\sim\mathcal{U}(0,T)}\mathbb{E}_{\boldsymbol{x}\sim p(\boldsymbol{x})}\mathbb{E}_{\boldsymbol{\epsilon}\sim\mathcal{N}(0,\sigma_t^2\boldsymbol{I})}\mathbb{E}_{\text{random }c}\|\boldsymbol{s}_V(\boldsymbol{G}_c\boldsymbol{x},c^*;\theta)+\boldsymbol{G}_c\boldsymbol{\epsilon}/\sigma_t^2\|_2^2. \tag{4.14}$$

This loss function indicates that (perhaps surprisingly!) regardless of the method by which the patches are used to tile the image, the method of training the patch-based network is the same: extract patches randomly from whole images in the training dataset, and then perform score matching on the resulting patches. This principle extends to overlapping patches and even patches of different sizes. The following subsections demonstrate how various patch-based diffusion models fall under this framework.

**Stochasticity Method Evaluation.** For certain probability models, it is too computationally expensive to evaluate the score functions of all the patches involved, so we instead stochastically select some patches and perform an update according to them. We show next that using stochastic updates leads to convergence to the same distribution as if we used the entire score function.

**Theorem.** Suppose $\boldsymbol{s}(\boldsymbol{x}) = \frac{1}{N}\sum_{i=1}^{N} \boldsymbol{s}_i(\boldsymbol{x})$. When sampling using Langevin dynamics [166], instead of using $\boldsymbol{s}(\boldsymbol{x})$, we use $\boldsymbol{s}_i(\boldsymbol{x})$ for a randomly chosen $i$. Then under certain regularity conditions, the sampling algorithm will lead to an image that is drawn from the distribution $p(\boldsymbol{x})$.

**Proof.** The update step that uses the full score function is

$$\Delta \boldsymbol{x}_t = \frac{\epsilon_t}{2} \cdot \frac{1}{N} \sum_{i=1}^{N} \boldsymbol{s}_i(\boldsymbol{x}_t) + \sqrt{\epsilon_t}\boldsymbol{z}, \tag{4.15}$$

whereas the update step in the stochastic algorithm is

$$\Delta \boldsymbol{x}_t = \frac{\epsilon_t}{2} \cdot \boldsymbol{s}_j(\boldsymbol{x}_t) + \sqrt{\epsilon_t}\boldsymbol{z}. \tag{4.16}$$

We wish to show that the distribution of random vectors in sequence $\{\boldsymbol{x}_t\}$ generated by the stochastic algorithm converge to samples drawn from the same distribution as the algorithm using the full score function.

The crux of the proof is the following idea. We first fix some $\epsilon_0 \in (0, 1)$. By the infinite sum properties on $\epsilon_t$, there is a subsequence $t_1 < t_2 < \cdots$ such that $\sum_{t=t_s+1}^{t_{s+1}} \epsilon_t$ approaches $\epsilon_0$ as $s$ approaches infinity. Now we collectively consider the $t_{i+1} - t_i$ update steps belonging to one term of the subsequence. Across these steps, there are mainly two sources of randomness: the stochasticity of choosing which score function to use, and the added Gaussian noise. We will show that the amount of deviation induced by the stochastic score functions is $O(\epsilon_0)$ while that of the Gaussian noise is $O(\sqrt{\epsilon_0})$. Thus, taking the limit as $\epsilon_0$ approaches o shows that almost all the variance will be due to the Gaussian noise. Hence, these $t_{i+1} - t_i$ update steps are equivalent to a single Langevin update step using the full score function with step size $\epsilon_0$. Thus, the stochastic algorithm converges to the same distribution as the algorithm using the full score function.

To show this, first, define $g(\boldsymbol{x}) = \frac{1}{N}\sum_{i=1}^{N} \boldsymbol{s}_i(\boldsymbol{x})$ and $h_t(\boldsymbol{x}) = \boldsymbol{s}_{jt}(\boldsymbol{x}) - g(\boldsymbol{x})$. Then the stochastic gradient at time $t$ is $g(\boldsymbol{x}) + h_t(\boldsymbol{x})$. The total update due to gradients between steps $t_s$ and $t_{s+1}$ is

$$\sum_{t=t_s+1}^{t_{s+1}} \frac{\epsilon_t}{2}(g(\boldsymbol{x}_t) + h_t(\boldsymbol{x}_t)). \tag{4.17}$$

We analyze the two terms of this sum separately. First,

$$\sum_{t=t_s+1}^{t_{s+1}} \frac{\epsilon_t}{2} g(\boldsymbol{x}_t) = \sum_{t=t_s+1}^{t_{s+1}} \frac{\epsilon_t}{2} g(\boldsymbol{x}_{t_s}) + \sum_{t=t_s+1}^{t_{s+1}} \frac{\epsilon_t}{2} (g(\boldsymbol{x}_t) - g(\boldsymbol{x}_{t_s})). \tag{4.18}$$

The first term here is just $\frac{\epsilon_0}{2} g(\boldsymbol{x}_{t_s})$, which is an update step using the gradient of the full score function. Therefore we need to bound the standard deviation of the error term:

$$E = \left| \sum_{t=t_s+1}^{t_{s+1}} \frac{\epsilon_t}{2} (g(\boldsymbol{x}_t) - g(\boldsymbol{x}_{t_s})) \right| \leq \sum_{t=t_s+1}^{t_{s+1}} \frac{\epsilon_t}{2} |g(\boldsymbol{x}_t) - g(\boldsymbol{x}_{t_s})|. \tag{4.19}$$

Assuming that $g$ has a local Lipschitz constant $L$,

$$E \leq \sum_{t=t_s+1}^{t_{s+1}} \frac{\epsilon_t L}{2} \|\boldsymbol{x}_t - \boldsymbol{x}_{t_s}\|_2 \leq \frac{\epsilon_0 L}{2} \text{max}_{t_s < t \leq t_{s+1}} \|\boldsymbol{x}_t - \boldsymbol{x}_{t_s}\|_2. \tag{4.20}$$

Now it suffices to note that as $\epsilon_0$ goes to $0$, so do all the step sizes within the subsequence. Hence, $\|\boldsymbol{x}_t - \boldsymbol{x}_{t_s}\|_2$ goes to $0$ (at a rate that is unknown but not crucial to the analysis). Therefore, we can write

$$E \leq \frac{\epsilon_0 L}{2} o(1) \leq O(\epsilon_0). \tag{4.21}$$

In summary we have shown that

$$\sum_{t=t_s+1}^{t_{s+1}} \frac{\epsilon_t}{2} g(\boldsymbol{x}_t) = \frac{\epsilon_0}{2} g(\boldsymbol{x}_{t_s}) + O(\epsilon_0). \tag{4.22}$$

Now we analyze the second term which is:

$$\sum_{t=t_s+1}^{t_{s+1}} \frac{\epsilon_t}{2} h_t(\boldsymbol{x}_t) = \sum_{t=t_s+1}^{t_{s+1}} \frac{\epsilon_t}{2} h_t(\boldsymbol{x}_{t_s}) + \sum_{t=t_s+1}^{t_{s+1}} \frac{\epsilon_t}{2} (h_t(\boldsymbol{x}_t) - h_t(\boldsymbol{x}_{t_s})). \tag{4.23}$$

Similar to the derivation for $g$, the second term is readily bounded by $O(\epsilon_0)$ assuming that $h_t$ is Lipschitz. For the first term, note that each $h_t(\boldsymbol{x}_{t_s})$ is an iid random variable. Furthermore, the variance of each such random variable is bounded by the maximum values taken by $s_i(\boldsymbol{x})$ along the trajectory of $\boldsymbol{x}$, which in turn is assumed to be bounded. Hence, we get a weighted average of iid random variables with bounded variance. In particular, the sum of the weights is $O(\epsilon_0)$, so the average deviation due to the entire term is $O(\epsilon_0)$.

Putting everything together,

$$\sum_{t=t_s+1}^{t_{s+1}} \frac{\epsilon_t}{2}(g(\boldsymbol{x}_t) + h_t(\boldsymbol{x}_t)) = \frac{\epsilon_0}{2} g(\boldsymbol{x}_{t_s}) + O(\epsilon_0). \tag{4.24}$$

Now the variance due to the added Gaussian noise is simple: at each step, the noise has variance $\epsilon_t$ and it is independent. Thus the total noise has variance $\epsilon_0$, which means the average deviation due to this noise is $O(\sqrt{\epsilon_0})$. Hence, as $\epsilon_0$ approaches $0$, the random error from the stochastic gradient is negligible compared to the added Gaussian noise. This shows that across the steps $t_s$ to $t_{s+1}$, the expected value of the update is $\frac{\epsilon_0}{2} g(\boldsymbol{x}_{t_s})$ and the variance of the added Gaussian noise is $\epsilon_0$. This variance aligns exactly with a Langevin update step using the full score function with a step size of $\epsilon_0$. Hence, we have shown that the samples generated by the stochastic algorithm converges to the same probability distribution as using the full score function.

### 4.3.3 Examples

Next, we provide specific examples of works that fall under this framework.

**Image restoration in adverse weather conditions.** The work of [142] uses a supervised diffusion model to perform image restoration for images taken in adverse weather conditions. In this paper, the learned patch prior is conditioned on a degraded version of the clean image. We can readily apply this method for learning an unsupervised prior by removing the condition during training. Then the restoration algorithm consists of the step $\Omega_t = \Omega_t + P_d \cdot \boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t^{(d)}, t)$, where $\Omega_t$ is the noise of the whole image at the $t$th timestep, $d$ is an iterator over the overlapping patches, $P_d$ is an operator that maps a patch to its location in the whole image, and $\boldsymbol{x}_t^{(d)}$ is an image patch. This step is followed by the step $\boldsymbol{M} = \boldsymbol{M} + P_d \cdot \mathbf{1}$. At the end of each diffusion timestep, the noise is scaled accordingly with the step $\Omega_t = \Omega_t \oslash \boldsymbol{M}$. This definition of $\boldsymbol{M}$ is consistent with our definition of $\boldsymbol{M}$ and the update steps follow (4.4). Finally, the training process outlined in [142] involves selecting a random patch of a clean image and performing denoising score matching, consistent with (4.14).

**PaDIS.** Our proposed method of using patch shifts with nonoverlapping patches falls under this general framework [82]. We start with the score function model in (4.2) and rewrite it using denoisers as follows:

$$\boldsymbol{D}(\boldsymbol{x}) = \frac{1}{M^2} \sum\nolimits_{i,j=1}^{i,j=M} \left( \boldsymbol{D}_{i,j,B}(\boldsymbol{x}_{i,j,B}) + \sum\nolimits_{r=1}^{(k+1)^2} \boldsymbol{D}_{i,j,r}(\boldsymbol{x}_{i,j,r}) \right), \tag{4.25}$$

where we include an additional $1/M^2$ factor for consistency with later derivations. We drop the $\boldsymbol{G}_c'$ terms that map the denoised patches back to the whole image for convenience. By definition, the denoised border $\boldsymbol{D}_{i,j,B}(\boldsymbol{x}_{i,j,B})$ is everywhere zero. Note that for each of the $M^2$ random patch shifts corresponding to a pair of indices $i$ and $j$, the resulting partition of the zero padded image consists of nonoverlapping patches. Interpreting each such shift as containing $(k+1)^2+1$ patches, the sum in total contains $((k+1)^2+1)M^2$ patches. Furthermore, each pixel is included in exactly $M^2$ of the patches. Hence, we may rewrite (4.25), this time with all the patch extracting operators as

$$\boldsymbol{D}(\boldsymbol{x}) = \left(\sum_c \boldsymbol{G}_c' \boldsymbol{D}(\boldsymbol{G}_c \boldsymbol{x})\right) \oslash (M^2 \cdot \mathbf{1}), \tag{4.26}$$

where $\mathbf{1}$ is the matrix of all ones. Now we are iterating over all $((k+1)^2+1)M^2$ patches using a single iterator $c$. This sum encapsulates both of the sums in (4.25) so that $M^2$ of the terms consists of denoising the image border while the remaining $(k+1)^2M^2$ terms consist of denoising square patches. Finally, with this tiling scheme, it is clear that $\sum_c \boldsymbol{G}_c' \boldsymbol{G}_c \mathbf{1} = M^2 \mathbf{1}$. Hence, we have shown how the score model for PaDIS follows (4.4).

Strictly speaking, when using (4.4) to update the image, it would be necessary to compute the score function of every patch in the sum.

**TPDM.** Used for solving 3D inverse problems, the probability distribution model used in TPDM [109] is given by

$$p(\boldsymbol{x}) = \left(\prod_{i=1}^{N} q_\theta(\boldsymbol{x}[:,:,i])^\alpha\right)\left(\prod_{j=1}^{N} q_\phi(\boldsymbol{x}[j,:,:])^\beta\right)/Z, \tag{4.27}$$

where $\alpha$ and $\beta$ are fixed constants and $Z$ is a scaling constant. This model uses a total of $2N$ patches used to cover the volume, where $N$ of them are along the first indexing dimension and $N$ of them are along the third indexing dimension. Each voxel in the volume is covered by exactly two such patches, so $\boldsymbol{M} = 2 \cdot \mathbf{1}$. Therefore, by setting $\alpha = \beta = 1/2$, we recover the form of (4.6). Furthermore, training is done separately on the 2D slices in the two perpendicular directions of the volumes. Similar to PaDIS, while it would be necessary to use both the distributions $q_\theta$ and $q_\phi$ to update the volume at each step of the diffusion process during reconstruction, the TPDM approach instead takes a stochastic approach of choosing one of the two possible ways.

**DiffusionBlend++.** The probability distribution model used in DiffusionBlend++ (see Ch. 5) is given by

$$\boldsymbol{s}(\boldsymbol{x}) = |S|^{-1} \sum\nolimits_{\mathcal{S}=\mathcal{S}_1 \cup \ldots \cup \mathcal{S}_r} \sum\nolimits_{i=1}^{r} \nabla \log p(\boldsymbol{x}[:,:,\mathcal{S}_i]), \tag{4.28}$$

where the outer sum runs over (possibly a subset of) all possible ways to partition the $N$ slices of the volume into $r$ sets of $N/r$ elements each. This is very similar to PaDIS: $\boldsymbol{M}$ is simply a constant times the matrix of all ones, assuming that each slice is included in the same number of partitions. Furthermore, at reconstruction time, DiffusionBlend++ again uses a stochastic approach to avoid computing the entire score function in (4.28).

In conclusion, we have provided a general framework for patch-based diffusion models and identified several recent works that fall under this framework. Furthermore, we have shown some theoretical justifications and insights that enhance our understanding of these models. In the future, this section would benefit from a series of carefully designed experiments that verify the theoretical results and examine differences in results between possible design choices for patches.

## 4.4 Experiments

**Experimental setup.** For the main CT experiments, we used the AAPM 2016 CT challenge data [135] that consists of 10 3D volumes. We cropped the data in the Z-direction to select 256 slices and then rescaled the images in the XY-plane to have size $256 \times 256$. Finally, we used the XY slices from 9 of the volumes to define 2304 training images, and used 25 of the slices from the tenth volume as test data. For the deblurring and superresolution experiments, we used a 3000 image subset of the CelebA-HQ dataset [126] (with each image being of size $256 \times 256$) for training to demonstrate that the proposed method works well in cases with limited training data. We preprocessed the data by dividing all the RGB values by 255 so that all the pixel values were between 0 and 1. The test data was a randomly selected subset of 25 of the remaining images. In all cases, we report the average metrics across the test images: peak SNR (PSNR) in dB, and structural similarity metric (SSIM) [185]. For the colored images, these metrics were computed in the RGB domain.

For the main patch-based networks, we trained mostly with a patch size of $56 \times 56$ to allow the target image to be completely covered by 5 patches in each direction while minimizing the amount of zero padding needed. We used the network architecture in [93] for both the patch-based networks and whole image networks. All networks were trained

on PyTorch using the Adam optimizer with 2 A40 GPUs. The Appendix provides full details of the hyperparameters.

**Image generation.** Our proposed method is able to learn a reasonable prior for whole images, despite never being trained on any whole images. Fig. 4.4 shows generation results for the CT dataset using three different methods. The top row used the network trained on whole images; the middle row used the method of [183] except that the entire image is never supplied to the network either at training or sampling time; the bottom row used the proposed method. The middle row shows that the positional encoding does ensure reasonably appropriate generated patches at each location. However, simply generating each of the patches independently and then naively assembling them together leads to obvious boundary artifacts due to lack of consistency between patches. Our proposed method solves this problem by using overlapping patches via random patch grid shifts, leading to generated images with continuity throughout. However, some of these images still appear asymmetric or have otherwise unrealistic global structure. This is expected as the network is not trained on regions of the image that are far apart and thus cannot learn long range correlations; in a later chapter, we will develop an improved method that can mitigate this issue.

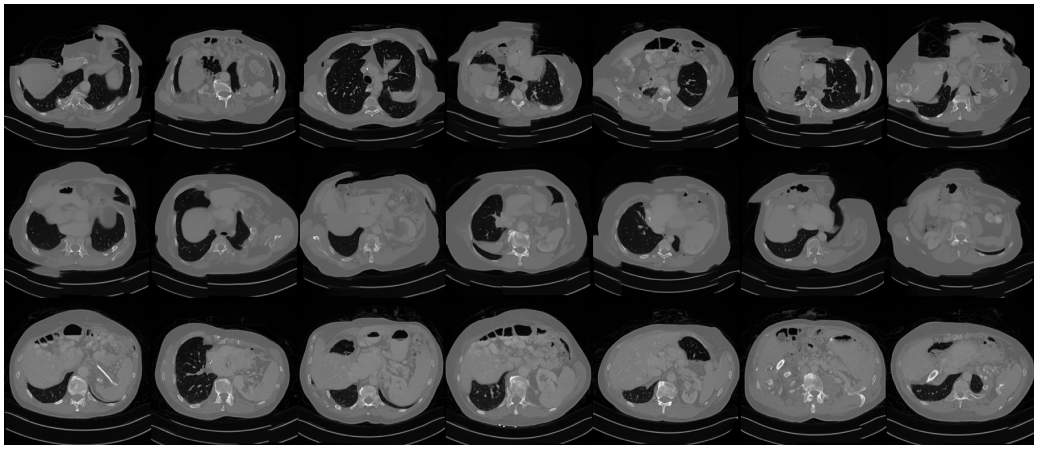


FIG 4.4 – Unconditional generation of CT images. Top row: generation with a network trained on whole image; middle row: patch-only version of [183]; bottom row: proposed PaDIS method.

**Inverse problems.** We tested the proposed method on a variety of different inverse problems: CT reconstruction, deblurring, and superresolution. For the forward and backward projectors in CT reconstruction, we used the implementation provided by [173]. We performed two sparse view CT (SVCT) experiments: one using 8 projection views, and

one using 20 projection views. Both of these were done using a parallel-beam forward projector where the detector size was 512 pixels. For the deblurring experiments, we used a uniform blur kernel of size $9 \times 9$ and added white Gaussian noise with $\sigma = 0.01$ where the clean image was scaled between 0 and 1. For the superresolution experiments, we used a scaling factor of 4 with downsampling by averaging and added white Gaussian noise with $\sigma = 0.01$. DPS has shown to benefit significantly from using a higher number of neural function evaluations (NFEs) [39], so we use 1000 NFEs for all of the diffusion model experiments.

For the comparison methods, we trained a diffusion model on entire images using the same denoising score matching method shown in Section 4.2.1. The inference process was identical to that of the patch-based method, with the exception that the score function of the image at each iteration was computed directly using the neural network, as opposed to needing to break up the zero-padded image into patches. We also compared with two traditional methods: applying a simple baseline and reconstructing via the total variation regularizer (ADMM-TV). For CT, the baseline was obtained by applying the filtered back-projection method to the measurement $\boldsymbol{y}$. For deblurring, the baseline was simply equal to the blurred image. For superresolution, the baseline was obtained by upsampling the low resolution image and using nearest neighbor interpolation. The implementation of ADMM-TV can be found in [74]. We also implemented two plug and play (PnP) methods: PnP-ADMM [196] and regularization by denoising (RED) [83]. We trained denoising CNNs on each of the datasets following [151] and then used them to solve the inverse problems.

For further comparison of diffusion model methods, we implemented different sampling and data consistency algorithms and applied them in conjunction with our patch-based prior. In particular, we compared with Langevin dynamics [166], predictor-corrector sampling [46], and variation exploding DDNM (VE-DDNM) [181]. For all these sampling methods, we used the variation exploding framework for consistency and to avoid retraining the network. We also compared with two other ways of assembling priors of patches to form the prior of an entire image: patch averaging [142] and patch stitching [152].

Table 4.1 shows the main inverse problem solving results. The best results were obtained after training the patch-based models for around 12 hours, while the whole image models needed to be trained for 24-36 hours, demonstrating a significant improvement in training time. Furthermore, when averaging the metrics across the test dataset, our proposed method outperformed the whole image method in terms of PSNR and SSIM for all the inverse problems. The score functions of all the patches can be computed in parallel for each iteration, so the reconstruction times for these methods were very similar (both

around 5 minutes per image). The whole-image results could be more favorable if more training data were used.

We also ran ablation studies examining the effect of various parameters of the proposed method. Namely, we studied the usage of different patch sizes during reconstruction, varying dataset sizes, importance of positional encoding for patches, and different sampling and inverse problem solving algorithms. The results of these studies are in App. A.1.1.

**TBL** 4.1 – Comparison of quantitative results on three different inverse problems. Results are averages across all images in the test dataset. Best results are in bold.

| Method | CT, 20 Views | | CT, 8 Views | | Deblurring | | Superresolution | |
|---|---|---|---|---|---|---|---|---|
| | PSNR↑ | SSIM↑ | PSNR↑ | SSIM↑ | PSNR↑ | SSIM↑ | PSNR↑ | SSIM↑ |
| Baseline | 24.93 | 0.595 | 21.39 | 0.415 | 24.54 | 0.688 | 25.86 | 0.739 |
| ADMM-TV | 26.82 | 0.724 | 23.09 | 0.555 | 28.22 | 0.792 | 25.66 | 0.745 |
| PnP-ADMM [196] | 26.86 | 0.607 | 22.39 | 0.489 | 28.82 | 0.818 | 26.61 | 0.785 |
| PnP-RED [83] | 27.99 | 0.622 | 23.08 | 0.441 | 29.91 | 0.867 | 26.36 | 0.766 |
| Whole image diffusion | 32.84 | 0.835 | 25.74 | 0.706 | 30.19 | 0.853 | 29.17 | 0.827 |
| Langevin dynamics [166] | 33.03 | 0.846 | 27.03 | 0.689 | 30.60 | 0.867 | 26.83 | 0.744 |
| Predictor-corrector [46] | 32.35 | 0.820 | 23.65 | 0.546 | 28.42 | 0.724 | 26.97 | 0.685 |
| VE-DDNM [181] | 31.98 | 0.861 | 27.71 | 0.759 | - | - | 26.01 | 0.727 |
| Patch Averaging [142] | 33.35 | 0.850 | 28.43 | 0.765 | 29.41 | 0.847 | 27.67 | 0.802 |
| Patch Stitching [152] | 32.87 | 0.837 | 26.71 | 0.710 | 29.69 | 0.849 | 27.50 | 0.780 |
| PaDIS (Ours) | **33.57** | **0.854** | **29.48** | **0.767** | **30.80** | **0.870** | **29.47** | **0.846** |

In addition to the main inverse problems shown in Table 4.1, we also ran experiments on three additional inverse problems to demonstrate the effectiveness of our method on a wide variety of inverse problems: 60 view CT reconstruction, fan beam reconstruction using 180 views, and deblurring with a uniform kernel of size $17 \times 17$. For the deblurring experiment, we again added Gaussian noise with level $\sigma = 0.01$ to the measurement. Table 4.2 shows the quantitative results of these experiments and Figure A.6 shows the visual results.

In the bottom of Figure 4.5, some artifacts are present in the reconstructions obtained by the diffusion model methods, although they are more apparent in the whole image model than with PaDIS. The measurements are very compressed in this case, so it is very difficult for any model to obtain diagnostic-quality reconstructions; the baselines perform

TBL 4.2 – Extra inverse problem experiments. Results are averages across all images in the test dataset. Best results are in bold.

| Method | CT, 60 Views | | CT, Fan Beam | | Heavy Deblurring | |
|---|---|---|---|---|---|---|
| | PSNR↑ | SSIM↑ | PSNR↑ | SSIM↑ | PSNR↑ | SSIM↑ |
| Baseline | 25.89 | 0.746 | 20.07 | 0.521 | 21.14 | 0.569 |
| ADMM-TV | 30.93 | 0.833 | 25.78 | 0.719 | 26.03 | 0.724 |
| Whole image diffusion | 35.83 | 0.894 | 26.89 | 0.835 | 28.35 | 0.808 |
| PaDIS (Ours) | **39.28** | **0.941** | **29.91** | **0.932** | **28.91** | **0.818** |

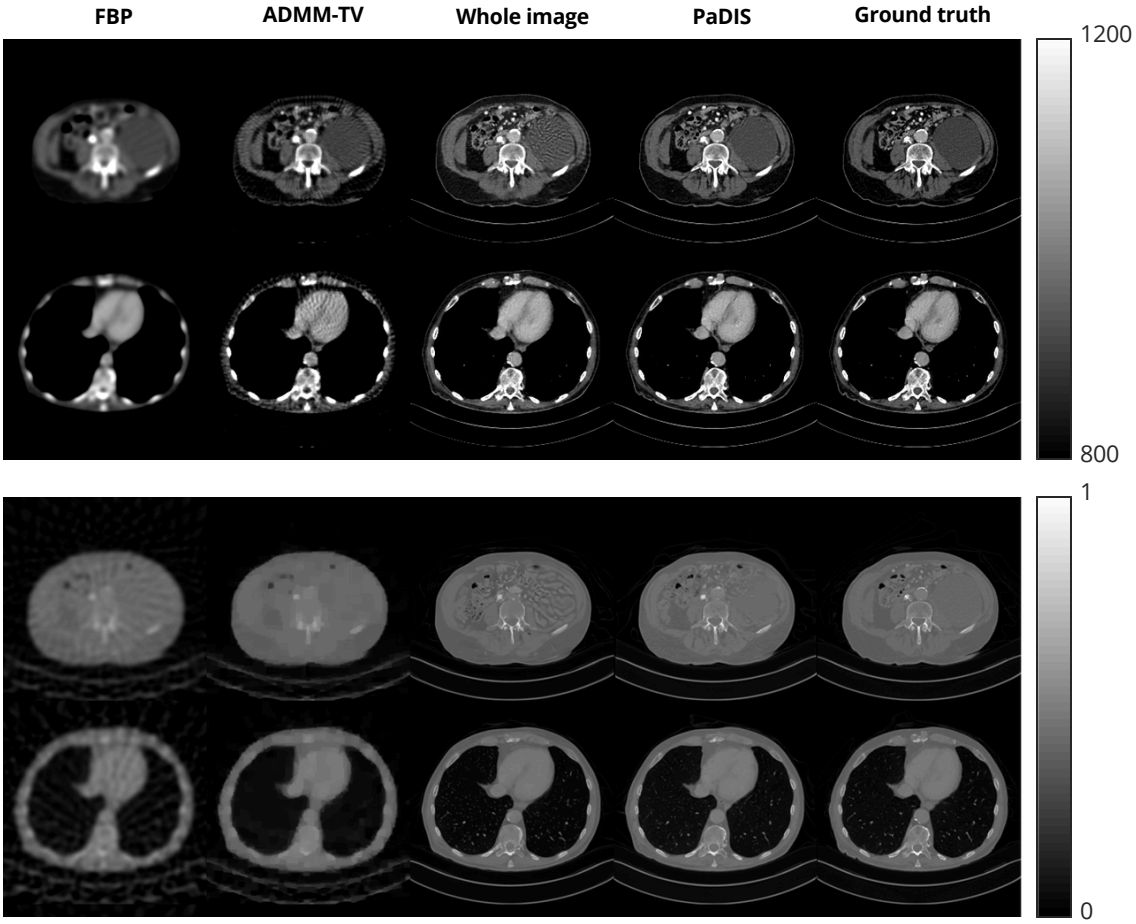



FIG 4.5 – Results of CT reconstruction. 60 views are used for the top two rows, 20 views are used for the bottom two rows. To better show contrast between organs, we use modified Hounsfield units (HU) in the top figure, while we use the same scale the images were trained on in the bottom figure.

significantly worse in terms of quantitative metrics and exhibit severe blurring. In clinical settings, patient diagnosis are typically performed with CT scans consisting of hundreds

of views. The top of Figure 4.5 shows that when 60 views are used, our proposed method yields a much better reconstruction without artifacts. Nevertheless, we show the potential of our proposed methods to reconstruct images from very sparse views with a decent image quality, which could be potentially used for applications such as patient positioning.

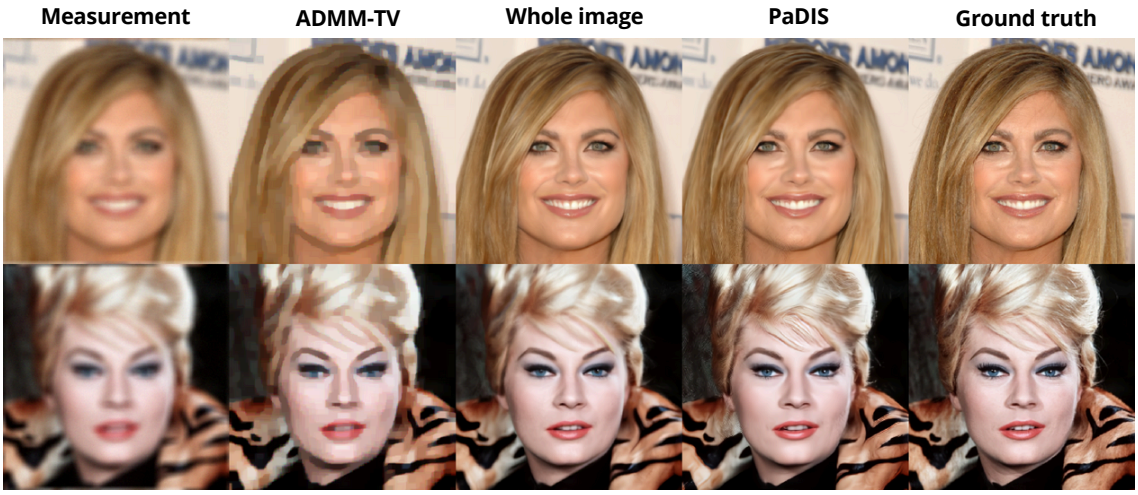



**FIG** 4.6 – Results of deblurring with Gaussian noise ($\sigma = 0.01$).

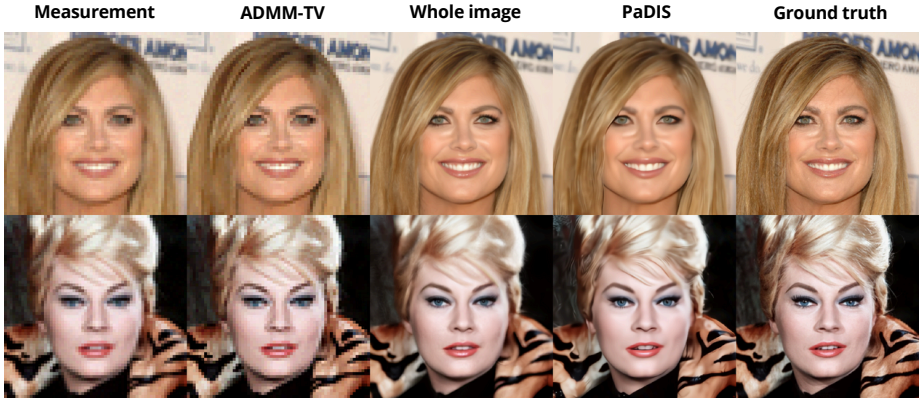



**FIG** 4.7 – Results of superresolution with Gaussian noise ($\sigma = 0.01$).

Finally, since the original AAPM dataset contained CT images of resolution $512 \times 512$, we ran 60 view CT reconstruction experiments with these higher resolution images. Due to the lack of data, we did not train a whole-image model on these higher resolution images. We used a zero padding width of 64 and largest patch size of $64 \times 64$ for training. Figure 4.8 shows the visual results of these experiments. Hence, our proposed methods

can obtain high quality reconstructions for both different inverse problems and also for different image sizes.

**TBL** 4.3 – Results of 60 view CT reconstruction with $512 \times 512$ images. Results are averages across all images in the test dataset. Best results are in bold.

| Method | FBP | ADMM-TV | PaDIS |
|---|---|---|---|
| PSNR ↑ | 28.38 | 29.48 | **36.93** |
| SSIM ↑ | 0.699 | 0.788 | **0.899** |

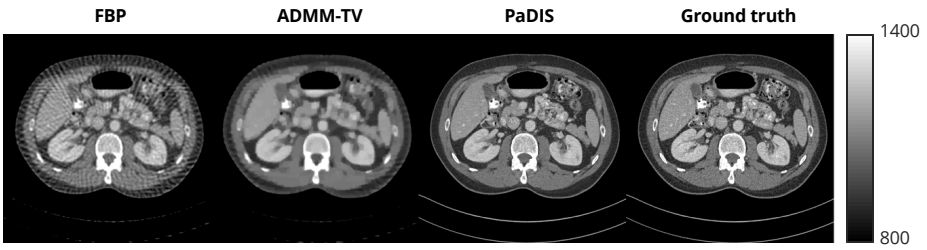



**FIG** 4.8 – Results of 60 view CT reconstruction on $512 \times 512$ images using modified HU units.

## 4.5 Conclusion

In this work, we presented a method of using score-based diffusion models to learn image priors through solely the patches of the image, combined with suitable position encoding. Simulation results demonstrated how the method can be used to solve a variety of inverse problems. Extensive experiments showed that under conditions of limited training data, the proposed method outperforms methods involving whole image diffusion models. In the future, more work could be done on higher quality image generation using a multi-scaled resolution approach [68, 72], using acceleration methods for faster reconstruction, and higher dimensional image reconstruction. Image priors like those proposed in this chapter have the potential to benefit society by reducing X-ray dose in CT scans. Generative models have the risk of inducing hallucinations and being used for disinformation.

# CHAPTER 5

# DiffusionBlend: Learning 3D Image Prior through Position-aware Diffusion Score Blending for 3D Computed Tomography Reconstruction

## 5.1 Motivation

Learning efficient 2D image priors using diffusion models is already computationally expensive, which requires large-scale of training data, training time, and GPU memory. For example, previous works [168, 73] require training for several days to weeks on over a million training images in the ImageNet [153] and LSUN [200] datasets to generate high-quality 2D natural images of size $256 \times 256$. Hence, directly learning a 3D diffusion prior on the entire CT volume would be practically infeasible or prohibitively expensive due to the demanding requirements of training data and computational cost. In addition, real clinical CT data is usually limited and scarce and often has a resolution larger than $256 \times 256 \times 400$, which makes directly training the data prior very challenging. The problem of tackling 3D image inverse problems, especially for medical imaging remains a challenging open research question.

A few recent works [42, 109, 41] have discussed and proposed to solve 3D image reconstruction problems either through employing some hand-crafted regularization to enforce consistency between 2D slices when reconstructing 3D volumetric images [41, 42], or through training several diffusion models for 2D images on each plane (axial, coronal,

This chapter based on [162]. Equal contribution with Bowen Song.

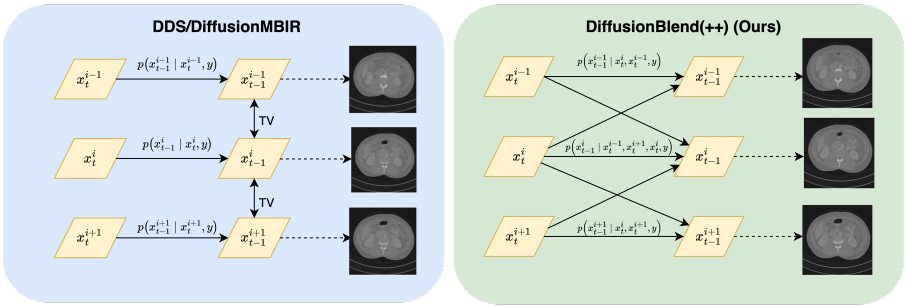



**FIG** 5.1 – Overview of DiffusionBlend++ compared to previous 3D image reconstruction works. Previous work used a hand-crafted TV term to "regularize" adjacent slices, whereas the proposed approach uses learned diffusion score blending between groups of slices. Here $i$ is the slice index, and $t$ is the reconstruction iteration.

and sagittal), and performing reverse sampling with each model alternatively [109]. However, all of these works only learn the distribution of a single 2D slice via the diffusion model, while having not yet explored the dependency between slices that is required to better model the real 3D image prior.

To overcome these limitations, we propose a novel method, called DiffusionBlend, that enables learning the distribution of 3D image patches (a batch of nearby 2D slices), and blends the scores of patches to model the entire 3D volume distribution for image reconstruction. Specifically, we firstly propose to train a diffusion model that models the joint distribution of 3D image patches (nearby 2D slices) in the axial plane conditioning on the slice thickness. Then, we introduce a random blending algorithm that approximates the score function of the entire 3D volume by using the trained 3D-patch score function. Moreover, we can either directly use the trained model to predict the noise of a single 2D slice by taking its corresponding 3D patch as input, or applying a random blending algorithm that firstly randomly partitions the volume into different 3D patches at each time step and then computes the score of each 3D patch during reverse sampling. Through either way, we can output the predicted noise of the entire 3D volume. In this way, our proposed method is able to enforce cross-slice consistency without any hand-crafted regularizer. Our method has the advantage of being fully data-driven and can enforce slice consistency without the TV regularizer as demonstrated in Fig. 5.1. Through exhaustive experiments of ultra-sparse-view and limited-angle 3D CT reconstruction on different datasets, we validate that our proposed method achieves superior reconstruction results for 3D volumetric imaging, outperforming previous state-of-the-art (SOTA) methods. Furthermore, our method achieves better or comparable inference time than SOTA methods, and requires minimum hyperparameter tuning for different tasks and settings.

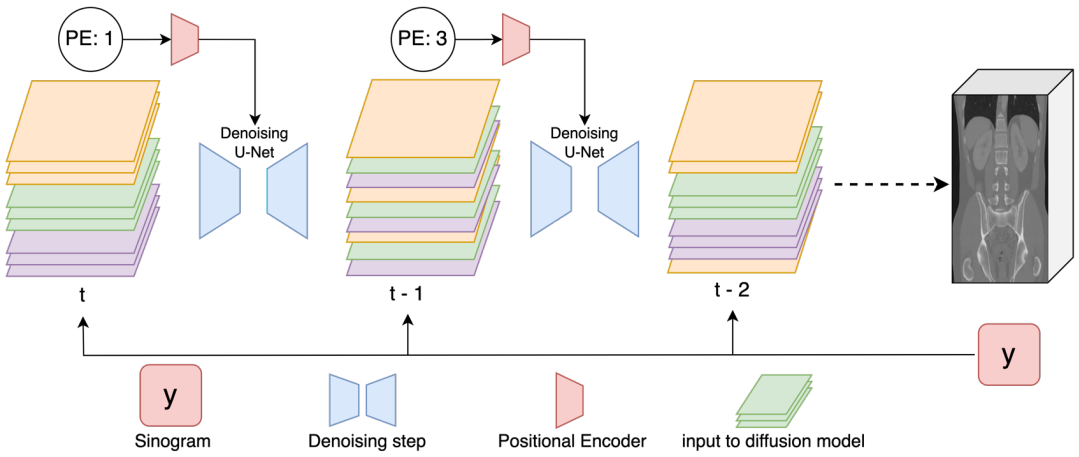



**FIG** 5.2 – Overview of slice blending process during reconstruction for DiffusionBlend++. At each iteration, we partition the slices of the volume in a different way; slices of the same color are inputted into the network independently. Positional encoding (PE) is also inputted to the network as information about the separation between the slices.

In summary, our main contributions are as follows:

- We propose DiffusionBlend(++): a novel method for 3D medical image reconstruction through 3D diffusion priors. To the best of our knowledge, our method is the first diffusion-based method that learns the 3D-patch image prior incorporating the cross-slice dependency, so as to enforce the consistency for the entire 3D volume without any external regularization.
- Specifically, instead of independently training a diffusion model only on separated 2D slices, we propose a novel method that first trains a diffusion model on 3D image patches (a batch of nearby 2D slices) with positional encoding, and at inference time, employs a new approach of random partitioning and diffusion score blending to generate an isotropically smooth 3D volume.
- Extensive experiments validate our proposed method achieves **state-of-the-art** reconstruction results for 3D volumetric imaging for the task of ultra-sparse-view and limited-angle 3D CT reconstruction on different datasets, with improved inference time efficiency and minimal hyperparameter tuning.

## 5.2 Methods

Instead of modeling the 2D slices of the 3D volume as independent data samples during training time, and then applying regularization between slices at reconstruction time, we

propose incorporating information from neighboring slices at training time to enforce consistency between slices. More precisely, our first approach models the data distribution of a 3D volume with $H$ slices in the $z$ dimension as follows:

$$p(\boldsymbol{x}) = \prod_{i=1}^{H} p(\boldsymbol{x}[:,:,i] \mid \boldsymbol{x}[:,:,i-j:i-1], \boldsymbol{x}[:,:,i+1:i+j])/Z, \tag{5.1}$$

where $j$ is a positive integer indicating the number of neighboring slices above and below the target slice that are being used as conditions to predict the target slice, and $Z$ is a normalizing constant. To deal with boundary conditions where the third index may exceed the bounds of the original volume, we apply repetition padding above and below the main volume.

For training, we simply concatenate each of the conditioned slices with the target slice along the channel dimension to serve as an input to the neural network. Then we apply denoising score matching to predict the noise of the target slice as the loss function of the neural network:

$$\mathbb{E}_{t\sim\mathcal{U}(0,T)}\mathbb{E}_{\boldsymbol{x}\sim p(\boldsymbol{x})}\mathbb{E}_{\boldsymbol{y}\sim\mathcal{N}(\boldsymbol{x},\sigma_t^2 I)}\mathbb{E}_{i\in[1,H]}\left\|\frac{D_\theta(\boldsymbol{y}[:,:,i-j:i+j],\sigma_t)-\boldsymbol{x}[:,:,i]}{\sigma_t^2}\right\|_2^2. \tag{5.2}$$

At reconstruction time, the score function of the entire volume decomposes as a sum of score functions of each of the slices:

$$\nabla \log p(\boldsymbol{x}) = \sum_{i=1}^{H} \nabla \log p(\boldsymbol{x}[:,:,i] \mid \boldsymbol{x}[:,:,i-j:i-1], \boldsymbol{x}[:,:,i+1:i+j]). \tag{5.3}$$

In this way, we have rewritten the score of the 3D volume as sums of the scores of the 2D slices learned by the network. This means that we can now apply any algorithm that uses diffusion models to solve inverse problems to solve the 3D CT reconstruction problem. Furthermore, this method of blending together information from different slices allows us to learn a prior for the entire volume that combines information from different slices. We call this method **DiffusionBlend**.

To learn an even better 3D image prior, instead of learning the conditional distribution of individual target slices, we can learn the **joint distribution** of several neighboring slices at once, which we call a 3D patch. Letting $k$ be the number of slices in each patch, we can partition the volume into 3D patches and approximate the distribution of the volume as

$$p(\boldsymbol{x}) = (\prod_{i=1}^{H/k} p(\boldsymbol{x}[:,:,(i-1)k+1:ik]))/Z, \tag{5.4}$$

where $Z$ is a normalizing constant. Comparing this with (5.1), the main difference is instead of conditioning on neighboring slices, we are now incorporating the neighboring slices as a joint distribution. This allows for much faster reconstruction, as $k$ slices are updated simultaneously according to their score function. However, this method faces similar slice consistency issues as in [42], since certain pairs of adjacent slices (namely, pairs whose slice indices are congruent to 0 and 1 modulo $k$) are never updated simultaneously by the network.

To deal with this issue, we propose two additional changes. Firstly, instead of using the same partition (updating the same $k$ slices) at once for each iteration, we can use a different partition so that the previous border slices can be included in another partition. For example, we can randomly sample the end index of the first 3D patch for adjacency slices. Let $m$ be uniformly sampled from $1, 2, ..., k$, we can use the partition

$$\mathcal{S} = \{1, 2, \ldots, H\} = \{1, \ldots, m\} \cup \{m+1, \ldots, m+1+k\} \cup \ldots \cup \{H-k+1, \ldots, H\}, \quad (5.5)$$

instead of $\mathcal{S} = \{1, 2, \ldots, H\} = \{1, \ldots, k\} \cup \{k+1, \ldots, 2k\} \cup \ldots \cup \{H-k+1, \ldots, H\}$, where $m$ is the offset index number in the new partition. We can then compute the score on the new partition. More generally, we can choose an arbitrary partition of $\mathcal{S}$ into $H/k$ sets, each containing $k$ elements for each iteration, updating each slice in the small set simultaneously for that iteration.

Secondly, to better capture information between nonadjacent slices, we apply relative positional encoding as an input to the network. More precisely, if a 3D patch has a slice thickness (the distance between two slices) of $p$, then we let $p$ be input of the positional encoding for that 3D patch. The positional encoding block consists of a sinusoidal encoding module and several dense connection modules, which has the same architecture as the timestamp embedding module of the same diffusion model. In this manner, the network is able to learn how to incorporate information from nonadjacent slices and captures more global information about the entire volume. Recall that for 3D patches of adjacent slices, the border between patches may have inconsistencies. To address this, we can **concatenate each border** as a new 3D patch, and then compute the score from it. If there are $k$ slices in an adjacency-slice 3D patch, then the new 3D patch has the relative positional encoding of $k$, and also has a size of $k$. For instance, if the previous partition is (1,2,3),(4,5,6),(7,8,9), the new partition is (1,4,7),(2,5,8),(3,6,9). Here we are forming a new partition with jumping slices. In practice, since we need a pretrained natural image checkpoint due to scarcity of medical image data, we set $k = 3$ for facilitating fine tuning from natural image checkpoints.

We call the partitioning by 3D patch with adjacent slices as **Adjacency Partition**, and the partitioning by 3D patch with jumping slices as **Cross Partition**. Letting $r = H/k$ be the number of 3D patches, with a random partition, this method is stochastically averaging the different estimations of the $\nabla \log p(\boldsymbol{x})$ by different parititions. Specifically, the estimation of score by a single partition $\mathcal{S}_1 \cup \ldots \cup \mathcal{S}_r$ is given by $\sum_{i=1}^{r} \nabla \log p(\boldsymbol{x}[:, :, \mathcal{S}_i])$. Ideally, we want to compute

$$|S|^{-1} \sum\nolimits_{\mathcal{S}=\mathcal{S}_1 \cup \ldots \cup \mathcal{S}_r} \sum\nolimits_{i=1}^{r} \nabla \log p(\boldsymbol{x}[:, :, \mathcal{S}_i]). \tag{5.6}$$

Similar to [42, 39, 109], we can share the summation in (5.6) across different diffusion steps since the difference between two adjacent iterations $\boldsymbol{x}_i$ and $\boldsymbol{x}_{i+1}$ is minimal.

In summary, we have shown how the score function of the entire volume can be written in terms of scores of the slices of the volume. Hence, similar to DiffusionBlend, this method can be coupled with any inverse problem solving algorithm. The scores of the slices can be approximated using a neural network. Training this network consists of randomly selecting $k$ slices from a volume and concatenating them along the channel dimension to get the input to the network (along with the positional encoding of the slices), and then using denoising score matching as in (5.2) as the loss function.

**Sampling and reconstruction.** With Eq. 5.6, each reconstruction step would require computing the score functions corresponding to each of the partitions of $\mathcal{S}$, and then summing them to get the score function $\boldsymbol{s}(\boldsymbol{x})$. We propose the variable sharing technique for this method, and only need to compute the score of one partition per time step. Hence, each iteration, we instead randomly choose one of the partitions of $\mathcal{S}$ and update the volume of intermediate samples by the score function. Finally, we use repetition padding if $H$ is not a multiple of $k$. This method incorporates a similar slice blending strategy as DiffusionBlend, but allows for significant acceleration at reconstruction time as $k$ slices are updated at once. Furthermore, it allows the network to learn joint information between slices that are farther apart without requiring the increase in computational cost associated with increasing $k$. We call this method **DiffusionBlend++**. The pseudocode of the algorithm can be found in Alg. 3.

In practice, we choose not to select from all possible partitions, but instead select from those where the indices in each $\mathcal{S}_i$ are not too far apart, as the joint information between slices that are very far apart is hard to capture. Table A.11 summarizes the different 3D image prior models. The appendix provides more details about the partition selection scheme.

**Krylov subspace methods.** Following the work of [41], we apply Krylov subspace methods to enforce data consistency with the measurement. At each timestep $t$, by using Tweedie's formula [56], we compute $\hat{\boldsymbol{x}}_t = \mathbb{E}[\boldsymbol{x}_0|\boldsymbol{x}_t]$, and then apply the conjugate gradient method

$$\hat{\boldsymbol{x}}_t' = \text{CG}(\boldsymbol{A}^*\boldsymbol{A}, \boldsymbol{A}^*\boldsymbol{y}, \hat{\boldsymbol{x}}_t, M), \tag{5.7}$$

where in practice, the CG operator involves running $M$ CG steps for the normal equation $\boldsymbol{A}^*\boldsymbol{y} = \boldsymbol{A}^*\boldsymbol{A}\boldsymbol{x}$. We combine this method with the DDIM sampling algorithm [165] to decrease reconstruction time. To summarize, we provide the algorithm for DiffusionBlend++ below. The Appendix provides the training algorithms for our proposed method as well as the reconstruction algorithm for DiffusionBlend.

**Algorithm 3** DiffusionBlend++

**Require:** Forward model $\boldsymbol{A}$, sinogram $\boldsymbol{y}$, hyperparameter $k$
  Initialize $\boldsymbol{x}_T \sim \mathcal{N}(0, \sigma_T^2\boldsymbol{I})$
  **for** $t = T : 1$ **do**
    Randomly select a partition $\mathcal{S} = \mathcal{S}_1 \cup \ldots \cup \mathcal{S}_r$
    **if** $t \mod k = 0$ **then** use cross partition, **else** use random adjacency partitions
    Compute the relative positional encoding $PE_t$
    For each $i$, compute $\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t[:, :, \mathcal{S}_i], PE_t)$
    Compute $\boldsymbol{s} = \nabla \log p(\boldsymbol{x}_t)$ using (5.6)
    Compute $\hat{\boldsymbol{x}}_t = \mathbb{E}[\boldsymbol{x}_0|\boldsymbol{x}_t]$ using Tweedie's formula
    Set $\hat{\boldsymbol{x}}_t' = \text{CG}(\boldsymbol{A}^*\boldsymbol{A}, \boldsymbol{A}^*\boldsymbol{y}, \hat{\boldsymbol{x}}_t)$
    Sample $\boldsymbol{x}_{t-1}$ using $\hat{\boldsymbol{x}}_t'$ and $\boldsymbol{s}$ via DDIM sampling
  **end for**
  **Return** $\boldsymbol{x}$

## 5.3 Experiments

**Experimental setup.** We used the public CT dataset from the AAPM 2016 CT challenge [135] that consists of 10 volumes. We rescaled the images in the XY-plane to have size $256 \times 256$ without altering the data in the Z-direction and used 9 of the volumes for training data and the tenth volume as test data. The training data consisted of approximately 5000 2D slices and the test volume had 500 slices. We also performed experiments on the LIDC-IDRI dataset [6]. For this dataset, we first applied data preprocessing by setting the entire background of the volumes to zero. We rescaled the images in the XY-plane to have size $256 \times 256$, and, to compare with the TPDM method, only took the volumes with at least

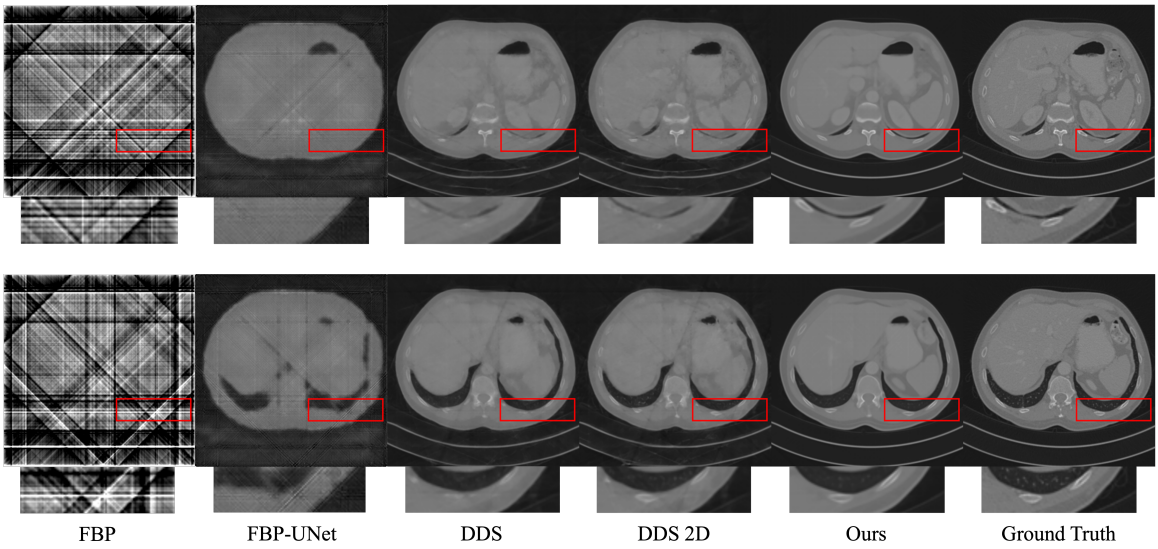



**FIG** 5.3 – Results of CT reconstruction with 4 views on AAPM dataset, axial view.

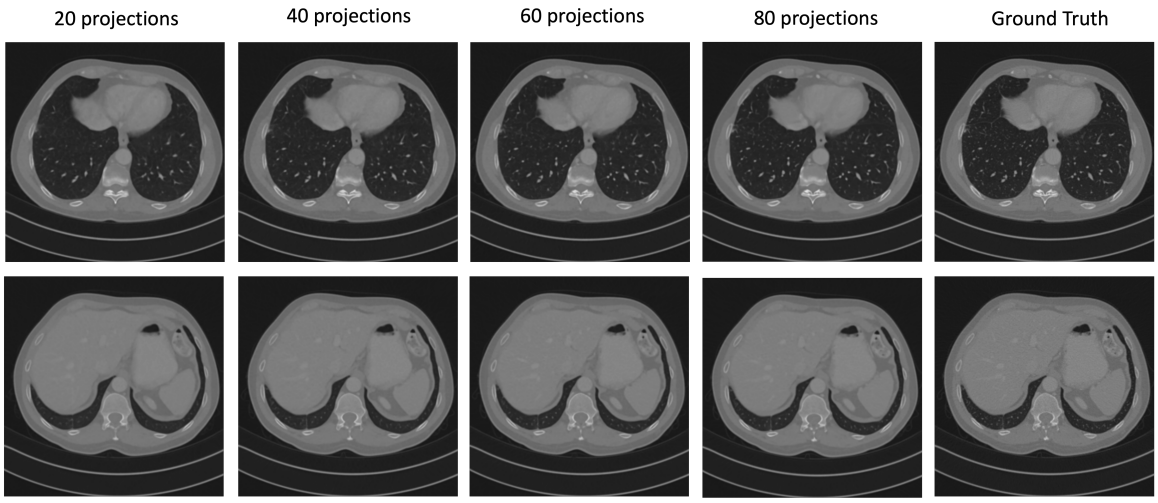



**FIG** 5.4 – Results of DiffusionBlend++ reconstruction with multiple views on AAPM dataset, axial view.

256 slices in the Z-direction, truncating the Z-direction to have exactly 256 slices. This resulted in 357 volumes which we used for training and one volume used for testing.

We performed experiments for sparse view CT (SVCT) and limited angle CT (LACT). The detector size was set to 512 pixels for all cases. For SVCT, we ran experiments on 4, 6, and 8 views. We also ran additional experiments on 20, 40, 60, 80, and 100 views and report the quantitative results in the Appendix. For LACT, we used the full set of views but only spaced around a 90 degree angle. In all cases, implementations of the forward and back projectors can be found in [42].

For a fair comparison between DiffusionBlend and DiffusionBlend++, we selected $j = 1$ for DiffusionBlend and each $\mathcal{S}_i$ to contain 3 elements for DiffusionBlend++. In this manner, both methods involve learning a prior that involves products of joint distributions on 3 slices. To train the score function for DiffusionBlend, we started from scratch using the LIDC dataset. Since this dataset consisted of over 90000 slices, the network was able to properly learn this prior. We then fine tuned this network on the much smaller AAPM dataset. For DiffusionBlend++, the input and output images both had 3 channels from stacking the slices, so we fine-tuned the existing checkpoint from [73]. All networks were trained on PyTorch using the Adam optimizer with A40 GPUs. For reconstruction, we used 200 neural function evaluations (NFEs) for all the results. The appendix provides the full experiment hyperparameters. We observe that DiffusionBlend++ can reconstruct very high quality images that are free of artifacts as demonstrated in Fig.5.4 and Fig.5.3.

**Comparison methods.** We compared our proposed method with baseline methods for CT reconstruction and state of the art 3D diffusion model methods. We used the filtered back projection implementation found in [42]. For the other baseline, we used FBP-UNet [89] which is a supervised method that involves training a network for each specific task mapping the FBP reconstruction to the clean image. Since this is a 2D method, we learned a mapping between 2D slices and then stacked the 2D slices to get the final 3D volume. We also compared with classical CT reconstruction techniques such as SBTV, SIRT, and CGLS [65] to benchmark our algorithm against traditional methods. Results for these methods are reported in the Appendix. For DiffusionMBIR [44], we fine-tuned the score function checkpoints on our data and used the same hyperparameters as the original work. We did the same for TPDM [109]; however, we ran TPDM only on the LIDC dataset because TPDM requires cubic volumes. Finally, we ran two variants of DDS [41]: one in which all the hyperparameters were left unchanged (DDS), and another in which no TV regularizer between slices was enforced (DDS 2D). Both of these methods were run with 200 NFEs. The appendix provides the experiment parameters.

| Method | Sparse-View CT Reconstruction on AAPM | | | | | | Sparse-View CT Reconstruction on LIDC | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 8 views | | 6 views | | 4 views | | 8 Views | | 6 Views | | 4 Views | |
| | PSNR↑ | SSIM↑ | PSNR↑ | SSIM↑ | PSNR↑ | SSIM↑ | PSNR↑ | SSIM↑ | PSNR↑ | SSIM↑ | PSNR↑ | SSIM↑ |
| FBP | 14.66 | 0.359 | 13.65 | 0.293 | 11.94 | 0.222 | 14.79 | 0.217 | 14.11 | 0.191 | 13.18 | 0.169 |
| FBP-UNet | 26.00 | 0.849 | 24.15 | 0.782 | 23.37 | 0.761 | 28.58 | 0.848 | 26.48 | 0.781 | 25.19 | 0.731 |
| DiffusionMBIR | 26.30 | 0.863 | 24.99 | 0.827 | 23.66 | 0.789 | 32.67 | 0.922 | 31.18 | 0.901 | 29.02 | 0.863 |
| TPDM | - | - | - | - | - | - | 27.51 | 0.816 | 25.60 | 0.776 | 21.99 | 0.695 |
| DDS 2D | 32.89 | 0.946 | 31.40 | 0.934 | 28.77 | 0.906 | 30.82 | 0.897 | 29.38 | 0.867 | 27.54 | 0.826 |
| DDS | 33.19 | 0.945 | 31.94 | 0.942 | 29.22 | 0.916 | 31.65 | 0.915 | 30.12 | 0.888 | 27.20 | 0.808 |
| DiffusionBlend (Ours) | 34.29 | 0.955 | 33.26 | 0.949 | 31.84 | 0.944 | 33.34 | 0.933 | 30.94 | 0.905 | 27.96 | 0.849 |
| DiffusionBlend++ (Ours) | **35.69** | **0.966** | **34.68** | **0.960** | **32.93** | **0.952** | **34.46** | **0.947** | **33.03** | **0.932** | **30.98** | **0.912** |

**TBL** 5.1 – Comprehensive comparison of quantitative results on Sparse-View CT Reconstruction on Axial View for AAPM and LIDC datasets. Best results are in bold.

| Method | Sparse-View CT Reconstruction on AAPM | | | | | | Sparse-View CT Reconstruction on LIDC | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 8 views | | 6 views | | 4 views | | 8 Views | | 6 Views | | 4 Views | |
| | PSNR↑ | SSIM↑ | PSNR↑ | SSIM↑ | PSNR↑ | SSIM↑ | PSNR↑ | SSIM↑ | PSNR↑ | SSIM↑ | PSNR↑ | SSIM↑ |
| FBP | 12.30 | 0.345 | 10.14 | 0.277 | 6.78 | 0.204 | 14.88 | 0.234 | 14.30 | 0.207 | 13.43 | 0.187 |
| FBP-UNet | 26.13 | 0.860 | 24.14 | 0.798 | 23.47 | 0.779 | 28.56 | 0.848 | 26.52 | 0.783 | 25.29 | 0.732 |
| DiffusionMBIR | 26.64 | 0.869 | 25.08 | 0.834 | 23.71 | 0.789 | 32.79 | 0.922 | 31.30 | 0.900 | 28.98 | 0.862 |
| TPDM | - | - | - | - | - | - | 27.66 | 0.819 | 25.57 | 0.784 | 21.87 | 0.708 |
| DDS 2D | 33.22 | 0.949 | 31.69 | 0.937 | 29.39 | 0.909 | 30.98 | 0.894 | 29.40 | 0.862 | 27.54 | 0.819 |
| DDS | 33.43 | 0.945 | 32.18 | 0.947 | 29.86 | 0.924 | 31.80 | 0.915 | 30.13 | 0.889 | 27.26 | 0.818 |
| DiffusionBlend (Ours) | 35.09 | 0.958 | 33.97 | 0.952 | 32.38 | 0.943 | 33.73 | 0.934 | 31.16 | 0.907 | 27.93 | 0.855 |
| DiffusionBlend++ (Ours) | **36.48** | **0.968** | **35.38** | **0.963** | **33.22** | **0.954** | **34.86** | **0.946** | **33.20** | **0.932** | **30.97** | **0.913** |

**TBL** 5.2 – Comprehensive comparison of quantitative results on Sparse-View CT Reconstruction on Sagittal View for AAPM and LIDC datasets. Best results are in bold.

**Sparse-view CT.** The results for different numbers of views and across different slices are shown in Tables 5.1, 5.2, and 5.3. DiffusionBlend++ exhibits much better performance over all the previous baseline methods (usually by a few dB) and outperforms DiffusionBlend. The second best method for each experiment is underlined and was, in most cases, DiffusionBlend. The exceptions are when the second best method is DiffusionMBIR, but this method was run with 2000 NFEs and took about 20 hours to run compared to 1-2 hours for both of our methods. The two DDS methods required similar runtime as our methods but in all cases exhibited inferior reconstruction results. Furthermore, DDS 2D generally performed worse than DDS. Thus, DDS failed to properly learn a 3D volume prior and still relied on the TV regularizer. Additionally, although TPDM should learn a 3D prior, the results were very poor compared to the other baselines. Our proposed

| Method | Sparse-View CT Reconstruction on AAPM | | | | | | Sparse-View CT Reconstruction on LIDC | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 8 views | | 6 views | | 4 views | | 8 Views | | 6 Views | | 4 Views | |
| | PSNR↑ | SSIM↑ | PSNR↑ | SSIM↑ | PSNR↑ | SSIM↑ | PSNR↑ | SSIM↑ | PSNR↑ | SSIM↑ | PSNR↑ | SSIM↑ |
| FBP | 14.64 | 0.325 | 13.18 | 0.268 | 11.16 | 0.236 | 14.78 | 0.206 | 14.10 | 0.181 | 13.10 | 0.165 |
| FBP-UNet | 27.34 | 0.878 | 25.12 | 0.827 | 24.10 | 0.810 | 28.87 | 0.858 | 26.59 | 0.793 | 25.37 | 0.744 |
| DiffusionMBIR | 29.86 | 0.908 | 28.12 | 0.875 | 25.68 | 0.843 | 33.29 | 0.922 | 31.69 | 0.903 | 29.21 | 0.868 |
| TPDM | - | - | - | - | - | - | 28.12 | 0.833 | 25.78 | 0.804 | 22.29 | 0.735 |
| DDS 2D | 33.64 | 0.950 | 32.33 | 0.939 | 30.25 | 0.916 | 31.60 | 0.898 | 29.99 | 0.871 | 28.03 | 0.830 |
| DDS | 33.97 | 0.934 | 32.95 | 0.930 | 30.89 | 0.932 | 32.51 | 0.920 | 30.83 | 0.898 | 27.61 | 0.828 |
| DiffusionBlend (Ours) | 36.45 | 0.958 | 35.23 | 0.952 | 33.98 | 0.944 | 34.47 | 0.934 | 31.48 | 0.908 | 28.24 | 0.859 |
| DiffusionBlend++ (Ours) | **37.87** | **0.968** | **36.66** | **0.963** | **34.27** | **0.955** | **35.66** | **0.947** | **33.97** | **0.935** | **31.38** | **0.913** |

**TBL** 5.3 – Comprehensive comparison of quantitative results on Sparse-View CT Reconstruction on Coronal View for AAPM and LIDC datasets. Best results are in bold.

| Method | AAPM Dataset | | | | | | LIDC Dataset | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Axial | | Sagittal | | Coronal | | Axial | | Sagittal | | Coronal | |
| | PSNR↑ | SSIM↑ | PSNR↑ | SSIM↑ | PSNR↑ | SSIM↑ | PSNR↑ | SSIM↑ | PSNR↑ | SSIM↑ | PSNR↑ | SSIM↑ |
| FBP | 16.36 | 0.643 | 16.36 | 0.524 | 15.62 | 0.531 | 18.79 | 0.672 | 19.84 | 0.675 | 20.01 | 0.676 |
| FBP-UNet | 27.38 | 0.910 | 27.81 | 0.918 | 28.44 | 0.930 | 29.42 | 0.885 | 29.50 | 0.884 | 29.54 | 0.887 |
| DiffusionMBIR | 25.98 | 0.872 | 27.14 | 0.877 | 27.74 | 0.903 | 30.52 | 0.906 | 30.57 | 0.906 | 30.68 | 0.907 |
| TPDM | - | - | - | - | - | - | 14.44 | 0.141 | 14.06 | 0.141 | 14.54 | 0.313 |
| DDS 2D | 28.05 | 0.916 | 27.99 | 0.916 | 28.82 | 0.922 | 27.92 | 0.843 | 27.89 | 0.835 | 27.96 | 0.842 |
| DDS | 28.20 | 0.918 | 28.17 | 0.926 | 29.03 | 0.934 | 28.12 | 0.865 | 28.06 | 0.869 | 28.13 | 0.879 |
| DiffusionBlend (Ours) | 35.38 | 0.971 | 35.85 | 0.972 | **37.62** | 0.972 | 30.43 | 0.917 | 31.24 | 0.920 | 31.02 | 0.924 |
| DiffusionBlend++ (Ours) | **35.86** | **0.975** | **36.03** | **0.976** | 37.45 | **0.976** | **34.33** | **0.957** | **34.48** | **0.957** | **34.64** | **0.956** |

**TBL** 5.4 – Comprehensive comparison of quantitative results on Limited-Angle CT Reconstruction on All Views for AAPM and LIDC datasets. Best results are in bold.

method learned a fully 3D prior and achieved the best results in the sagittal and coronal views.

**Limited-angle CT.** Table 5.4 shows all results for limited angle CT reconstruction for both the AAPM and LIDC datasets. Our DiffusionBlend++ method obtains superior performance over all the baseline methods and DiffusionBlend obtains the second best results. Similar to the SVCT experiments, DiffusionMBIR performed the best out of the baseline methods, but took approximately 40 hours to run. FBP-UNet performed reasonably well, but is a supervised method where the network must be retrained for each specific task. DDS is the most directly comparable to our method in runtime and methodology, but performed much worse quantitatively.

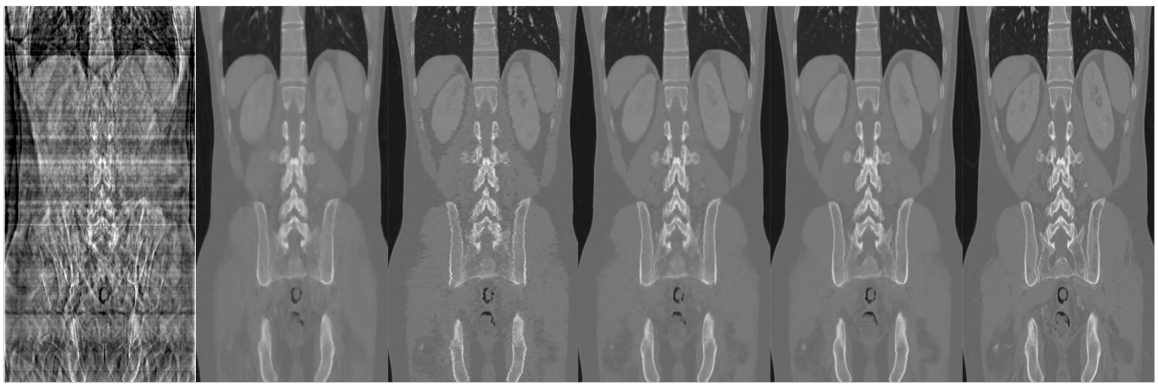



**FIG** 5.5 – Results of CT reconstruction with 8 views on AAPM dataset, coronal view. DiffusionPatch refers to Algorithm 1 with the same partition for every timestsep, and DiffusionBlend+ refers to Algorithm 1 only with partitions of adjacency slices.

| Algorithm | TV value | Difference with gt |
|---|---|---|
| DDS 2D | 0.0104 | 0.0044 |
| DDS | 0.0031 | -0.0034 |
| DiffusionBlend++ (Ours) | 0.0043 | **-0.0022** |
| Ground Truth | 0.0065 | - |

**TBL** 5.5 – TV values of different reconstruction algorithms on the AAPM test set

**Inter-slice smoothness** We demonstrate that DiffusionBlend++ learns the 3D prior internally, and achieves consistency and smoothness between 2D axial-plane slices without any external regularizations. In Table 5.5, we present the total variation (TV) value of the reconstructed images of different reconstruction algorithms on the test set of AAPM dataset, given by $\frac{1}{C\times W\times H}||\mathbf{D}_z(x)||_1$, where $x$ is the image, $\mathbf{D}_z$ is the total variation operator in $z$ direction, and $C$, $W$, $H$ are number of channels, width, and height. We find that both DiffusionBlend++ and DDS have TV less than the ground truth image, which implies that the reconstructed images are smooth in the z direction. However, we observe that DDS over-smooths the images as demonstrated in Fig. 5.5, which is represented by a much lower TV value than the ground truth. On the other hand, DiffusionBlend++ has smoothness level close to the ground truth without sacrificing sharpness of images.

**Effectiveness of adjacency-slice blending and cross-slice blending** We demonstrate that both the adjacency-slice blending and the cross-slice blending module are instrumental to a better reconstruction quality. Table 5.6 demonstrates the effectiveness of adding blending modules to the reverse sampling. Given the pretrained diffusion prior over slice

| Adjacency | Cross | PSNR ↑ | SSIM ↑ |
|---|---|---|---|
| | | 34.85 | 0.954 |
| ✓ | | 36.02 | 0.965 |
| ✓ | ✓ | **36.48** | **0.968** |

**TBL** 5.6 – Effectiveness of Blending Modules, Sagittal view performance on AAPM

patches, we observe that adding the adjacency-slice blending module improves the PSNR over a fixed partition by 1.17dB, and adding an additional cross-slice blending module further improves the PSNR by 1.63dB. Fig. 5.5 demonstrates that adding the cross-slice blending module removes artifacts and recovers sharper edges.

**Ablation Studies** We investigated the performance gain due to individual components. Details can be found in Appendix A.3.

## 5.4 Conclusion

In this work, we proposed two methods of using score-based diffusion models to learn priors of three dimensional volumes and used them to perform CT reconstruction. In both cases, we learn the distributions of multiple slices of a volume at once and blend the distributions together at inference time. Extensive experiments showed that our method substantially outperformed existing methods for 3D CT reconstruction both quantitatively and qualitatively in the sparse view and limited angle settings. In the future, more work could be done on other 3D inverse problems and acceleration through latent diffusion models. Image reconstruction methods like those proposed in this chapter have the potential to benefit society by reducing X-ray dose in CT scans.

# CHAPTER 6

# Local Patches Meet Global Context: Scalable 3D Diffusion Priors for Computed Tomography Reconstruction

## 6.1 Motivation

Diffusion models have emerged as a powerful class of generative model for image generation [168, 140, 144, 150], modeling the underlying image distribution $p(\boldsymbol{x})$ through learned denoising processes across progressively varied noise levels. The trained network approximates the score function of image distribution $\boldsymbol{s}(\boldsymbol{x}) = \nabla \log p(\boldsymbol{x})$. At inference time, starting from the initialization of Gaussian noise image, the noise signal is iteratively removed using the trained denoiser network, ultimately resulting in clean image samples drawn from the learned distribution prior $p(\boldsymbol{x})$.

Inverse problems can be formulated as $\boldsymbol{y} = \mathcal{A}(\boldsymbol{x}) + \boldsymbol{n}$, where $\boldsymbol{y}$ is a known measurement, $\mathcal{A}$ is the forward operator, $\boldsymbol{x}$ is the unknown image to be recovered, and $\boldsymbol{n}$ is random noise. Specifically, computed tomography (CT) reconstruction can also be formulated as an inverse problem that aims to reconstruct a 3D volume $\boldsymbol{x}$ from the measurement $\boldsymbol{y}$, containing projection views of $\boldsymbol{x}$ acquired from different angles [59]. Furthermore, to minimize patient radiation exposure, it is desirable to reduce the number of acquired projection views, typically on the order of 100-1000 views, to substantially fewer [162, 42], which is called *sparse-view CT reconstruction*. In this case, the inverse problem becomes ill-posed, as there are many potential images $\boldsymbol{x}$ that could correspond to a measurement $\boldsymbol{y}$. Hence, it is necessary to impose prior knowledge on $\boldsymbol{x}$ to obtain a reasonable reconstruction, such as [101, 104] in previous works.

This chapter based on [199]. Equal contribution with Taewon Yang.

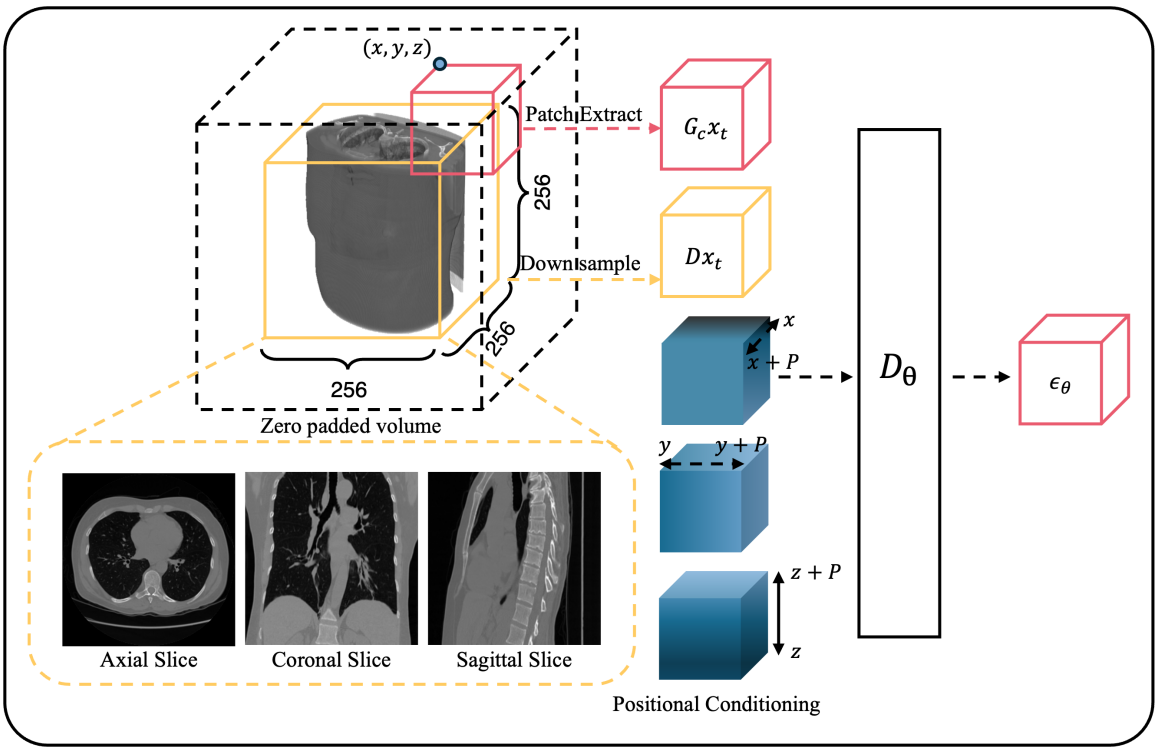



**FIG** 6.1 – Training the 3D patch diffusion model. Noisy 3D patch $\boldsymbol{G}_c\boldsymbol{x}_t$ (local details), downsampled volume $\boldsymbol{D}\boldsymbol{x}_t$ (global context), and positional encoding are concatenated as inputs to the denoiser network $D_\theta$ in each training iteration, to predict the noise $\boldsymbol{\epsilon}_\theta$ in the input noisy 3D patch. Positional encoding consists of voxel-based coordinates in $x, y, z$ that are normalized to $[-1, 1]$.

Given that diffusion models can learn powerful generative priors over image data, it is natural to leverage these priors to address sparse-view CT reconstruction problem. Previous works [39, 95, 181, 115, 45] have proposed various approaches leveraging diffusion models to solve different inverse problems, including 2D CT reconstruction. These approaches generally consist of, at inference time, interweaving the original diffusion model steps with data fidelity enforcement steps that push the image to be consistent with the measurements.

However, challenges emerge when directly applying these existing approaches to real-world sparse-view CT tasks, where the images are inherently three-dimensional volumes. Standard clinical CT images have the full size of $512 \times 512 \times 512$, making it almost infeasible to fit the entire image into the network training due to memory constraints. Furthermore, training diffusion models typically requires large-scale datasets [203], often comprising hundreds of thousands of images such as ImageNet [153] or LSUN [200]. For CT reconstruction, there is a lack of high-quality volumetric CT data in such scale. Finally,

training time of diffusion models on 3D images would be prohibitively expensive. Thus, the high computational costs in memory, time, and data pose significant challenges to directly train diffusion models on 3D CT images.

To address these challenges, prior works have proposed leveraging 2D image priors, either by combining them with hand-crafted external regularization terms [42] or by training an additional 2D image prior along a perpendicular plane [109]. However, these approaches tend to produce blurry artifacts due to sensitivity to the hand-crafted regularization term, or require long inference times due to the use of two perpendicular 2D CT priors. Recently, [162] introduced a new solution to learn the joint distribution over a small set of 2D slices simultaneously, but the underlying network remains a 2D architecture with multiple channels, needing significant reconstruction time comparable to that of [41]. However, these methods do not fully realize and exploit the generative capacity of diffusion models for high-resolution 3D volumetric images. Recent work by Teng et al. [174] trained a 3D convolutional UNet with $256 \times 256 \times 30$ slabs and used DPS to reconstruct $256 \times 256 \times 120$ CT volumes. No 3D image generation was shown.

To overcome these limitations, we propose a novel approach that learns scalable 3D diffusion priors by representing 3D volumes as **position-encoded local patches**. To ensure high-quality 3D image generation, our study emphasizes the importance of incorporating global context. Specifically, by jointly modeling the distribution of local patches alongside a downsampled version of the entire volume, our method effectively unifies global contextual information with local image statistics, to obtain improved **global-aware patch priors**. Beyond image generation, the resulting fully 3D prior demonstrates superior performance in large-scale inverse problem solving, such as high-resolution 3D CT reconstruction. Overall, our contributions are three-fold.

- We propose a novel 3D patch-based diffusion model that learns a fully 3D diffusion prior, enabling the scalable generation of high-resolution 3D images, and offering an unprecedentedly efficient and accurate solution to address high-resolution 3D inverse problems.

- Our key technical contribution is a framework that models the joint distribution of position-aware 3D local patches and a downsampled 3D volume, enabling scalable learning of a global-contextual 3D patch-based prior by effectively integrating local and global information to achieve high-quality 3D image generation.

- Extensive experiments on different datasets and different-sized images show that our proposed method achieves SOTA performance for 3D sparse-view CT reconstruction in various settings.

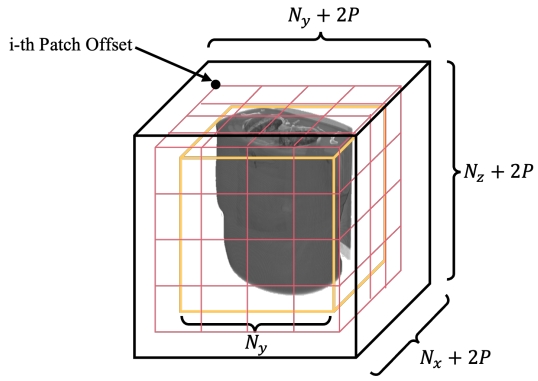



**FIG** 6.2 – Schematic illustration for zero padding and partitioning image into 3D patches. Each index $i$ represents one of $P^3$ possible ways to choose a patch offset tuple.

## 6.2 Methods

### 6.2.1 Global-Aware Patch Prior

We begin by following the formulation of learning patch-based priors as described in [82], but generalized for 3D images. In patch-based diffusion models, one needs first to zero pad each $N_x \times N_y \times N_z$ image by an amount $P$ on each side. Slightly recycling notation, we let $\boldsymbol{x}$ denote the resulting padded image. We partition $\boldsymbol{x}$ into many square patches, with one bordering region of zeros by first choosing the $i$th patch offset tuple $(o_1, o_2, o_3) \in \{0, \ldots, P-1\}^3$ according to Figure 6.2. The number of patches needed in the $x$ direction to perfectly tile the image is $k_1 = N_x/P$, and define $k_2 = N_y/P$ and $k_3 = N_z/P$ as the number of patches to perfectly tile the image in the $y$ and $z$ directions, respectively. Thus, the total number of patches needed to cover the image with an offset is $l = (k_1 + 1)(k_2 + 1)(k_3 + 1)$. Hence, our model for the data distribution has the form

$$p(\boldsymbol{x}) = \frac{1}{Z}\left(\prod_{i=1}^{P^3} p_{i,B}(\boldsymbol{x}_{i,B}) \prod_{r=1}^{l} p_{i,r}(\boldsymbol{x}_{i,r})\right)^{1/P^3}, \tag{6.1}$$

where $\boldsymbol{x}_{i,B}$ represents the bordering region of $\boldsymbol{x}$ that depends on the specific value of $i$, $p_{i,B}$ is the probability distribution of that region, $\boldsymbol{x}_{i,r}$ is the $r$th $P \times P \times P$ patch when using the partitioning scheme corresponding to the $i$th patch offset, $p_{i,r}$ is the probability distribution of that region, and $Z$ is a normalizing factor. This model takes a product over all possible patch offsets, eliminating boundary artifacts that would occur if a fixed patch offset was used.

Crucially, however, the model (6.1) does not account for global structure within the whole image or correlation between two distant patches. This can lead to reasonable local structure within each generated local patches, but a lack of coherent and realistic longer range details. At the same time, memory constraints in 3D preclude inputting the entire image into the network that approximates any part of the score function. Motivated by these considerations, we develop an improved **patch-based prior with global context** as follows:

$$p(\boldsymbol{x}) = \left( \prod_{i=1}^{P^3} p_{i,B}(\boldsymbol{x}_{i,B}) \prod_{r=1}^{l} p_{i,r}(\boldsymbol{x}_{i,r}, \boldsymbol{D}\boldsymbol{x}) \right)^{1/P^3} /Z, \tag{6.2}$$

so that now $p_{i,r}$ represents a joint distribution between the local patch $\boldsymbol{x}_{i,r}$ *and* the downsampled image $\boldsymbol{D}\boldsymbol{x}$, with $\boldsymbol{D}$ representing an operator that downsamples the whole image to a smaller image size. (Note that in practice, $\boldsymbol{D}$ may require pixel interpolation if the whole image dimension is not a multiple of the downsampled image.) Then

$$\log p(\boldsymbol{x}) = \frac{1}{P^3} \Big( \sum_{i=1}^{P^3} (\log p_{i,B}(\boldsymbol{x}_{i,B}) + \sum_{c} \log p_c(\boldsymbol{G}_c \boldsymbol{x}, \boldsymbol{D}\boldsymbol{x})) \Big), \tag{6.3}$$

where for convenience we reindex the patches with $c$ and $\boldsymbol{G}_c$ denotes the same patch grabbing operator as aforementioned. To take the gradient of this model, first define $\boldsymbol{u} = \boldsymbol{G}_c \boldsymbol{x}$ and $\boldsymbol{v} = \boldsymbol{D}\boldsymbol{x}$. Then

$$\nabla \log p(\boldsymbol{x}) = \frac{1}{P^3} \left( \sum_{i=1}^{P^3} \sum_{c} (\boldsymbol{G}_c' \nabla_{\boldsymbol{u}} \log p_c(\boldsymbol{G}_c \boldsymbol{x}, \boldsymbol{D}\boldsymbol{x}) \; + \; \boldsymbol{D}' \nabla_{\boldsymbol{v}} \log p_c(\boldsymbol{G}_c \boldsymbol{x}, \boldsymbol{D}\boldsymbol{x})) \right), \tag{6.4}$$

where we dropped the gradient of the bordering term since it is known to be zero, and we applied the chain rule. Hence, we can approximate the score functions $\nabla_{\boldsymbol{u}} \log p_c(\boldsymbol{G}_c \boldsymbol{x}, \boldsymbol{D}\boldsymbol{x})$ and $\nabla_{\boldsymbol{v}} \log p_c(\boldsymbol{G}_c \boldsymbol{x}, \boldsymbol{D}\boldsymbol{x})$ with the neural networks $\boldsymbol{s}_u$ and $\boldsymbol{s}_v$ respectively. The denoising score matching loss is then

$$L(\boldsymbol{\theta}) = \frac{1}{P^3} \mathbb{E}_{\boldsymbol{\epsilon} \sim \mathcal{N}(0, \sigma_t^2 \boldsymbol{I})} \left\| \sum_{i=1}^{P^3} \sum_{c} (\boldsymbol{G}_c' \boldsymbol{s}_u(\boldsymbol{u}, \boldsymbol{v}; \boldsymbol{\theta}) + \boldsymbol{D}' \boldsymbol{s}_v(\boldsymbol{u}, \boldsymbol{v}; \boldsymbol{\theta})) - P^3 \boldsymbol{\epsilon} / \sigma_t^2 \right\|_2^2. \tag{6.5}$$

Note that $\hat{\boldsymbol{x}} = \boldsymbol{x} + \boldsymbol{\epsilon}$ and now $\boldsymbol{u} = \boldsymbol{G}_c \hat{\boldsymbol{x}}$, $\boldsymbol{v} = \boldsymbol{D}\hat{\boldsymbol{x}}$, and $\boldsymbol{x}$ is sampled from the clean image distribution but we dropped these terms from the expectation for simplicity.

Unfortunately, due to the double summation inside of the norm, it is computationally infeasible to train the network using this loss function. Hence, we first use the inequality

$$\|X_1 + \ldots + X_n\|_2^2 \leq n \cdot (\|X_1\|_2^2 + \ldots + \|X_n\|_2^2), \tag{6.6}$$

to upper bound the loss (6.5) without the expectation to get

$$L \leq C \sum_{i=1}^{P^3} \| \sum_c \boldsymbol{G}_c' \nabla_{\boldsymbol{u}} \log p_c(\boldsymbol{u}, \boldsymbol{v}) + \boldsymbol{D}' \nabla_{\boldsymbol{v}} \log p_c(\boldsymbol{u}, \boldsymbol{v}) - \boldsymbol{\epsilon}/\sigma_t^2 \|_2^2, \tag{6.7}$$

for a constant $C > 0$. Therefore, during training we can choose a random $i$ and then minimize the inner sum. For fixed $i$, we usually choose each of the patches $c$ in a non-overlapping way. Unfortunately, the $\boldsymbol{D}'$ operator upscales the $\nabla_{\boldsymbol{v}}$ patch-sized term into a whole image sized term, so the sum over $c$ will contain contributions from each patch for every pixel. However, the part that is outside the patch $c$ is difficult to learn as the only contribution is from the upscaled part involving $\boldsymbol{D}'$. Thus we instead focus on learning the patch-specific part inside of the loss. Rewriting the norm part of the loss, this is

$$L_c \approx \| \sum_c \boldsymbol{G}_c \boldsymbol{G}_c' \nabla_{\boldsymbol{u}} \log p_c(\boldsymbol{u}, \boldsymbol{v}) + \boldsymbol{G}_c \boldsymbol{D}' \nabla_{\boldsymbol{v}} \log p_c(\boldsymbol{u}, \boldsymbol{v}) - \boldsymbol{G}_c \boldsymbol{\epsilon}/\sigma_t^2 \|_2^2. \tag{6.8}$$

When factoring out the $\boldsymbol{G}_c$ term, we observe that each term in this sum independently contributes to one of the non-overlapping patches of the entire image. Hence, when training the network, we do not have to perform denoising score matching across the entire image: we need only add noise to the entire image, provide one of the patches and the downsampled image to the network, and have the network learn to denoise just the patch $\boldsymbol{G}_c \boldsymbol{\epsilon}$. In practice, to represent the joint score functions of $\boldsymbol{G}_c \boldsymbol{x}$ and $\boldsymbol{D}\boldsymbol{x}$, we input $\boldsymbol{G}_c \boldsymbol{x}$ and $\boldsymbol{D}\boldsymbol{x}$ to the network via concatenation along the channel dimension (where $\boldsymbol{D}$ is chosen appropriately such that $\boldsymbol{G}_c \boldsymbol{x}$ and $\boldsymbol{D}\boldsymbol{x}$ are the same size), and represent $\nabla_{\boldsymbol{u}} \log p_c(\boldsymbol{G}_c \boldsymbol{x}, \boldsymbol{D}\boldsymbol{x}) + \nabla_{\boldsymbol{v}} \log p_c(\boldsymbol{G}_c \boldsymbol{x}, \boldsymbol{D}\boldsymbol{x})$ using single channels of the network output.

To facilitate the network to learn different prior distributions of 3D local patches at different locations, it is desirable to incorporate the positional information of the 3D patches. We do this by defining the $x$ positional array as a 3D array representing the $x$ coordinates of each pixel in the image, and normalizing to be scaled between -1 and 1. Similarly, $y$ and $z$ positional arrays are defined for $y$ and $z$ coordinates of the pixels, respectively. To enable the network to learn patch distributions that vary based on location within the image, we extract patches from these positional arrays corresponding to the image patches.

As shown in Figure 6.1, these positional patches are also concatenated along the channel dimension with $\boldsymbol{G}_c\boldsymbol{x}$ and $\boldsymbol{D}\boldsymbol{x}$.

In summary, we use a 3D UNet with five input channels: the noisy patch, the downsampled (noisy) image, and the three positional arrays. The network has one output channel, used to approximate $\nabla_{\boldsymbol{u}} \log p_c(\boldsymbol{u}, \boldsymbol{v})$ and $\nabla_{\boldsymbol{v}} \log p_c(\boldsymbol{u}, \boldsymbol{v})$, and is trained according to the loss (6.8). Most importantly, the spatial dimensions of the input are only in the 3D patch size of $P \times P \times P$ as shown in Figure 6.1 Therefore, we never need to input the whole 3D image into the network. The appendix shows the pseudocode for the training algorithm.

### 6.2.2 Sampling and Reconstruction Algorithm

Diffusion models are inevitably slow, as they typically require hundreds or thousands of steps to sample one image. Prior works have explored accelerating the diffusion sampling stage [165, 93, 128], For example, to expedite the sampling process, DDIM [165] proposed the following update rule

$$\boldsymbol{x}_{t-1} = \sqrt{\bar{\alpha}_{t-1}}\,\hat{\boldsymbol{x}}_t + \sqrt{1 - \bar{\alpha}_{t-1} - \sigma_t^2}\,\hat{\epsilon}_t + \sigma_t \boldsymbol{\epsilon}_t \tag{6.9}$$

where $\boldsymbol{\epsilon}_t \sim \mathcal{N}(0, I)$, $\bar{\alpha}_{t-1}$ is the noise schedule at timestep $t-1$, $\hat{\boldsymbol{\epsilon}}_t$ is the predicted noise and $\hat{\boldsymbol{x}}_t$ is the denoised estimate

$$\hat{\boldsymbol{x}}_t = \frac{1}{\sqrt{\bar{\alpha}_t}}(\boldsymbol{x}_t - \sqrt{1 - \bar{\alpha}_t}\,\hat{\boldsymbol{\epsilon}}_t). \tag{6.10}$$

In [165], the stochasticity is controlled by $\sigma_t$ that is defined as

$$\sigma_t = \eta\sqrt{(1 - \bar{\alpha}_{t-1})/(1 - \bar{\alpha}_t)}\sqrt{1 - \bar{\alpha}_t/\bar{\alpha}_{t-1}} \tag{6.11}$$

where $\eta \in [0, 1]$ so that $\eta = 0$ leads to fully deterministic sampling. Alternatively, [41] considers

$$\sigma_t = \eta\sqrt{1 - \bar{\alpha}_{t-1}}, \tag{6.12}$$

for which (6.9) becomes

$$\boldsymbol{x}_{t-1} = \sqrt{\bar{\alpha}_{t-1}}\hat{\boldsymbol{x}}_t + \sqrt{1 - \bar{\alpha}_{t-1}}(\sqrt{1 - \eta^2} \cdot \hat{\boldsymbol{\epsilon}}_t + \eta\boldsymbol{\epsilon}_t). \tag{6.13}$$

Assuming the estimated $\hat{\boldsymbol{\epsilon}}_t$ and the stochastic $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \boldsymbol{I})$ are independent Gaussians, $\tilde{\boldsymbol{\epsilon}} = \sqrt{1 - \eta^2} \cdot \hat{\boldsymbol{\epsilon}}_t + \eta\boldsymbol{\epsilon}_t$ is also a Gaussian. Hence, $\tilde{\boldsymbol{\epsilon}}$ can equivalently be represented as

a sample from $\tilde{\boldsymbol{\epsilon}} \sim \mathcal{N}(0, \boldsymbol{I})$. This preservation of noise variance ensures that the model samples $\boldsymbol{x}_{t-1}$ from the marginal distribution $q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_0)$.

For the underlying sampling algorithm, we choose DDIM [165] as a fast and reliable sampler that many other works [181, 95, 47] have also used as the backbone for solving inverse problems. During the sampling step, each update uses (6.13).

The score function model (6.4) allows for estimation of the score of the whole image from the trained network. However, it requires averaging over $P^3$ terms in the outer loop, which would be prohibitively expensive. To address this issue, [82] proposed a method that, instead of taking the average, chooses one of the terms at random and uses that as the approximation of the score.

However, when the number of sampling steps is limited, this approach becomes inapplicable, introducing boundary artifacts between patches. Also, the sampling process on one initial point does not consider the joint distribution of pixels across different patches at the same time step. Here, we propose a novel recurrent noising strategy to further mitigate the boundary artifacts when reducing the sampling steps for improving sampling efficiency. Specifically, at each iteration of DDIM, we denoise and renoise the image for $K$ times, where in each time a different randomly chosen patch location, and finally average the results over $K$ times to obtain the sample. The sampling equation is

$$\boldsymbol{x}_{t-1} = \sqrt{\bar{\alpha}_{t-1}}\hat{\boldsymbol{x}}_t^{avg} + \sqrt{1 - \bar{\alpha}_{t-1} - \sigma_t^2} \cdot \hat{\boldsymbol{\epsilon}}_t + \sigma_t \boldsymbol{\epsilon}_t \tag{6.14}$$

where $\sigma_t$ is equal to (6.12), $\hat{\boldsymbol{x}}_t^{avg}$ is the averages of $\{\hat{\boldsymbol{x}}_{i^w,t}\}_{w=1}^K$ and $\hat{\boldsymbol{\epsilon}}_t$ is the scaled sum of $\{\boldsymbol{\epsilon}_{i^w,t}\}_{w=1}^K$, computed over $K$ randomly chosen patch locations $i^w$. Here, noise variance is preserved similar to (6.13), allowing this method to sample from $q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_0)$.

Previous works [41, 162] showed that conjugate gradient method allows the model to apply data consistency with measurement much faster. At each timestep $t$, using the denoised estimate (6.10) we apply conjugate gradient method on the following forward operator $\boldsymbol{A}$:

$$\hat{\boldsymbol{x}}_t' = \text{CG}(\boldsymbol{A}^*\boldsymbol{A}, \boldsymbol{A}^*\boldsymbol{y}, \hat{\boldsymbol{x}}_t, M) \tag{6.15}$$

where the CG operator runs $M$ CG steps for the equation $\boldsymbol{A}^*\boldsymbol{y} = \boldsymbol{A}^*\boldsymbol{A}\boldsymbol{x}$. We apply this data consistency term after $K$ iterations and then apply DDIM sampling to solve the inverse problem. To summarize, our proposed sampling algorithm is shown in 4.

**Algorithm 4** 3D Patch Diffusion Sampling with Recurrent Noising

**Require:** Recurrent noising number $K$, $\boldsymbol{A}$, $\boldsymbol{y}$, $P$

Initialize $\boldsymbol{x}_T \sim \mathcal{N}(0, \boldsymbol{I})$
**for** $t = N : 1$ **do**
  **for** $w = 1 : K$ **do**
    Randomly select integer $i^w \in [1, P^3]$
    For $1 \leq r \leq l$, extract 3D patches $\boldsymbol{x}_{i,r,t}$
    **for** $r = 1 : l$ **do**
      $\hat{\boldsymbol{x}}_{i^w,r,t}, \boldsymbol{\epsilon}_{i^w,r,t} \leftarrow \boldsymbol{D}_\theta(\boldsymbol{x}_{i^w,r,t}, \sigma_t)$
    **end for**
    Concatenate all $\hat{\boldsymbol{x}}_{i^w,r,t}$ to entire volume $\hat{\boldsymbol{x}}_{i^w,t}$
    Concatenate all $\boldsymbol{\epsilon}_{i^w,r,t}$ to entire volume $\boldsymbol{\epsilon}_{i^w,t}$
    $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \boldsymbol{I})$
    $\boldsymbol{x}_t \leftarrow \sqrt{\bar{\alpha}_t}\hat{\boldsymbol{x}}_{i^w,t} + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}$
  **end for**
  $\hat{\boldsymbol{x}}_t^{\text{avg}} = \sum_{w=1}^{K} \hat{\boldsymbol{x}}_{i^w,t}/K \ , \quad \hat{\boldsymbol{\epsilon}}_t = \sum_{w=1}^{K} \boldsymbol{\epsilon}_{i^w,t}/\sqrt{K}$
  Set $\hat{\boldsymbol{x}}_t' = \text{CG}(\boldsymbol{A}^*\boldsymbol{A}, \boldsymbol{A}^*\boldsymbol{y}, \hat{\boldsymbol{x}}_t^{\text{avg}})$
  Sample $x_{t-1}$ via DDIM using (6.9)
**end for**

## 6.3 Results

### 6.3.1 Experimental Setup

**Dataset preparation.** We conducted experiments using the LIDC-IDRI dataset [6]. The original dataset has a spatial resolution of $512 \times 512$ in the XY-plane, with varying sizes along the Z-axis. In our experiments, we selected the first 256 slices along the Z-axis and filtered out samples with inconsistent shapes. To ensure comparability with contemporary comparison methods, we initially rescaled each XY-plane to $256 \times 256$. This preprocessing resulted in 90 volumes for training and 1 volume for testing. We also performed experiments on the AAPM dataset [135] that contains only 10 CT volumes, to evaluate our method's performance with a limited amount of data. This dataset was preprocessed in the same manner as the LIDC-IDRI dataset. To demonstrate that our approach can be extended to larger volumes, we additionally prepared a version of the LIDC-IDRI dataset with clinically relevant dimensions of $512 \times 512 \times 256$.

**Baseline setting.** For the LIDC-IDRI dataset, the model was trained for approximately 8 days with a batch size of 64 and a patch size of $32 \times 32 \times 32$. For the AAPM dataset, which contains only 9 volumes, we trained the model from scratch for 5 days using the

same batch and patch sizes. For the larger LIDC-IDRI dataset, the model was trained for 28 days with a batch size of 16 and a patch size of $64 \times 64 \times 32$. All training was done on a single NVIDIA A100 GPU. We applied our proposed sampling method using $K = 2$ and $\eta = 0.8$ for both $256 \times 256 \times 256$ and $512 \times 512 \times 256$ volumes. The total number of model parameters is 68.59 million.

### 6.3.2 Unconditional 3D Image Generation

Our proposed method is able to learn a fully 3D prior on $256 \times 256 \times 256$ sized CT volumes, allowing it to generate high quality volumes. We used (6.13) with $K = 1$ and $\eta = 0.4$ and 200 steps for unconditional sampling. Figure 6.3 shows unconditional volume samples from LIDC-IDRI prior, visualized across axial, coronal, and sagittal slices. The top row shows slices from an unconditionally sampled volume, while the bottom row shows the nearest-neighbor volume from the training dataset. These results indicate that the proposed prior does not simply memorize from the training dataset and is capable of generating realistic high-resolution 3D CT volumes with fine-grained details of anatomic structure. Further generation results on different prior are in the appendix.

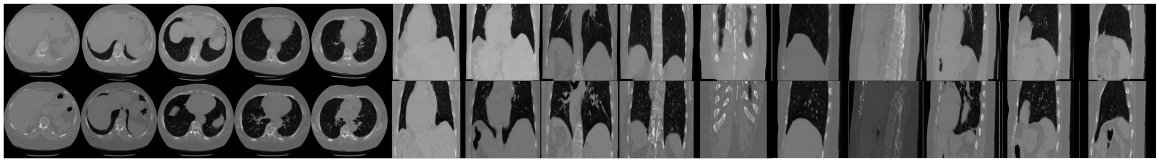


**FIG** 6.3 – Unconditional 3D image generation results using the LIDC-IDRI prior. The top row shows axial, coronal, and sagittal slices from a generated volume, and the bottom row shows the corresponding slices from its nearest-neighbor volume in the training dataset. The slice indices for the axial, coronal, and sagittal views are $[30, 80, 130, 180, 230]$, $[70, 100, 130, 160, 190]$, and $[60, 130, 160, 190, 210]$, respectively

### 6.3.3 Solving Inverse Problems

We conduct experiments of sparse-view CT (SVCT) reconstruction from 8, 20, and 60 views on one of the test volumes. The results in Table 6.1 and Figure 6.4 demonstrate that our method is able to reconstruct high-quality images. Furthermore, we compared our proposed method with various classical methods and diffusion model methods for 3D CT reconstruction. We used the filtered back projection method with a ramp filter, whose implementation is found in [42]. We also used the total variation (TV) regularizer and

TBL 6.1 – PSNR (dB) comparison of various methods for CT reconstruction on different datasets and resolutions with best results in bold.

| Dataset / Method | LIDC ($256 \times 256 \times 256$) | | | LIDC ($512 \times 512 \times 256$) | | | AAPM ($256 \times 256 \times 256$) | | |
|---|---|---|---|---|---|---|---|---|---|
| Views | 8 | 20 | 60 | 8 | 20 | 60 | 8 | 20 | 60 |
| FBP | 15.08 | 20.21 | 29.37 | 14.46 | 20.05 | 28.02 | 13.81 | 18.39 | 27.71 |
| ADMM-TV | 21.15 | 24.12 | 29.43 | 22.47 | 25.39 | 29.38 | 23.71 | 26.75 | 31.28 |
| FBP-UNet [89] | 22.60 | 27.77 | 32.78 | 26.09 | 30.32 | 35.34 | 25.93 | 30.06 | 36.93 |
| DDS [41] | 23.23 | 29.92 | 35.79 | 24.37 | 27.33 | 28.61 | 30.18 | 34.94 | 38.87 |
| DiffusionBlend [162] | 30.43 | 35.89 | 40.87 | 31.69 | 35.94 | 39.32 | – | – | – |
| Blend + FT [162] | 32.94 | 37.05 | 42.72 | 33.01 | 36.44 | 39.31 | 32.25 | 37.21 | 42.21 |
| **Proposed** | **33.06** | **38.56** | **43.70** | **33.17** | **37.33** | **40.16** | **32.96** | **38.47** | **42.66** |

solved the optimization problem using ADMM (ADMM-TV) with the implementation in [74]. For the other baseline, we implemented FBP-UNet [89] which is a supervised method that involves training a UNet that maps FBP reconstruction to the clean image. Since FBP-UNet is a 2D method, we learned a mapping between 2D slices and then stacked the 2D slices to get the final 3D volume.

For diffusion model methods, we ran DDS [41] and DiffusionBlend [162]. To obtain the best possible results for DiffusionBlend, we used the trick in the original work that involved taking a network that was pretrained on ImageNet and then finetuning it on the relevant CT dataset. Thus, we specifically ran DiffusionBlend++ with the prior trained on groups of 3 slices. To illustrate the effect of using the pretrained network, we show experiments both with the finetuning method (Blend+FT) and with networks that were trained from scratch on the CT datasets. Since our proposed method does not rely on a pretrained network, the most fair comparison is with [162] without a pretrained network; nevertheless we showed both results to illustrate how our proposed method can obtain superior results. Both of these methods were run with 200 sampling steps. The appendix provides the experiment parameters.

### 6.3.4 Ablation studies

We conducted ablation studies to analyze the factors that may influence the performance of our 3D patch diffusion model.

TBL 6.2 – Per volume runtimes of different methods for 8 view CT recon.

| Method | Runtime (minutes) ↓ |
|---|---|
| FBP | 0.1 |
| ADMM-TV | 6 |
| FBP-UNet | 1 |
| DDS | 64 |
| DiffusionBlend | 55 |
| Proposed | 20 |

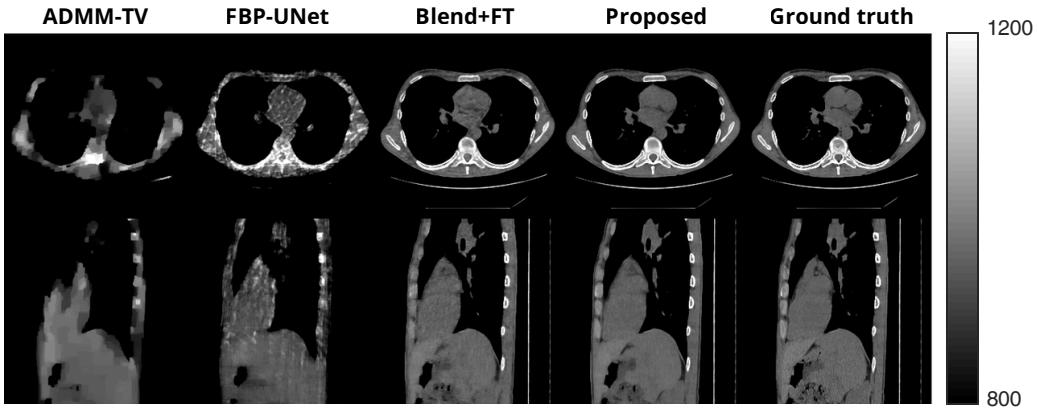



FIG 6.4 – Results of our proposed method and comparison methods for 20 view CT recon on LIDC $256 \times 256 \times 256$ dataset. Images are shown in modified Hounsfield units. The top row shows the axial slice and the bottom row shows the sagittal slice from the reconstructed volume.

**Impact of downsample channel.** Downsample channel plays a critical role in our method, allowing the model to learn the global structure. Without the downsample channel the model will only learn the local relationship between patches, failing to generate realistic CT images. To validate this, we compared the generation quality of the 3D patch-based diffusion model with and without the downsample channel using the Fréchet Inception Distance (FID), as shown in Fig. 6.3.

**Impact of patch size.** Consider an extreme case where the patch size is $1 \times 2$. According to our modeled prior distribution in (6.1), using such small patches is theoretically valid but unlikely to fully capture image statistics. To better understand this behavior, we examined how patch size affects the model's performance on inverse problem tasks. We trained

TBL 6.3 – FID comparison with and without downsample channel

| Method | w. downsample | w.o. downsample |
|---|---|---|
| FID ↓ | 40.80 | 112.12 |

models with varying patch sizes, which is feasible due to the input-size-agnostic property of U-Net architectures. Patch sizes were chosen by dividing the full volume size by a scaling factor. For example, a scaling factor of 8 on a $256 \times 256 \times 256$ volume corresponds to patches of size $32 \times 32 \times 32$. Since the data are volumetric, doubling the patch size in each dimension reduced the batch size by a factor of 8. As a result, smaller patches train faster in wall-clock time: the factor-16 setting converges within 3 days, whereas the factor-8 setting requires approximately 8 days on $256 \times 256 \times 256$ volumes.

Experimental results indicate that as the patch size decreases, the model failed to capture meaningful semantic structure, leading to reduced PSNR on the sparse-view CT (SVCT) task. As shown in Table 6.4, model performance degrades as the scaling factor increases. This observation suggests that when patches become too small relative to the full image, they behave more like individual pixels. Consequently, the model can no longer learn a coherent image-level prior, causing (6.1) to describe a less effective model.

TBL 6.4 – PSNR in dB of different patch and volume size

| Factor | $256 \times 256 \times 256$ | $512 \times 512 \times 256$ |
|---|---|---|
| 8 | **33.06** | **33.17** |
| 16 | 27.87 | 31.71 |

**Impact of recurrent noising during sampling process.** This section explores how recurrent noising affects the reconstruction quality. To understand how the number of iterations $K$ affects the reconstructed images, we tested on the 8-view SVCT task with 200 sampling steps. Table 6.5 shows that recurrent noising during sampling ($K > 1$) yielded higher PSNR than reconstruction without any recurrent noising ($K = 1$). The PSNR peaked at $K = 2$ and gradually decreased as $K$ increased. We conjecture that this

TBL 6.5 – PSNR in dB of different $K$, with best results in bold

| $K$ | 8 view | 20 view | 60 view |
|---|---|---|---|
| 1 | 32.53 | 38.18 | 43.46 |
| 2 | **33.06** | **38.56** | **43.70** |
| 3 | 32.97 | 38.43 | 43.52 |
| 4 | 32.74 | 38.15 | 43.44 |

is because the estimated $\hat{\epsilon}_t$ is not strictly sampled from a Gaussian distribution, causing error accumulation as $K$ increases.

## 6.4 Conclusion

In this chapter, we propose a novel 3D patch-based diffusion model that is conditioned with downsampled volume and coordinates to enable learning the 3D CT image prior in an efficient way. By learning from the local patches coupled with global context, our model is able to generate high-resolution 3D images. Experiments across different volume sizes and CT datasets showed that our method achieve SOTA results on SVCT reconstruction while reducing inference time by more than $2\times$. One of the limitations is that our approach assumes the data is well aligned to a consistent coordinate, which may reduce applicability to less structured image domains. In the future, we plan to investigate alternative methods that allow the model to learn the positional information of the patch, thereby improving robustness in less structured data.

# CHAPTER 7

# Test-Time Adaptation Improves Inverse Problem Solving with Patch-Based Diffusion Models

## 7.1 Motivation

In imaging, inverse problems are important and consist of reconstructing an image $\boldsymbol{x}$ from a measurement $\boldsymbol{y} = \mathcal{A}(\boldsymbol{x}) + \boldsymbol{\epsilon}$. Here, $\mathcal{A}$ represents a forward operator and $\boldsymbol{\epsilon}$ represents random unknown noise. By Bayes' rule, $\log p(\boldsymbol{x}|\boldsymbol{y})$ is proportional to $\log p(\boldsymbol{x}) + \log p(\boldsymbol{y}|\boldsymbol{x})$, so obtaining a good prior $p(\boldsymbol{x})$ is crucial for recovering $\boldsymbol{x}$ when $\boldsymbol{y}$ contains far less information than $\boldsymbol{x}$. Diffusion models obtain state-of-the-art results for learning a strong prior and sampling from it, so competitive results can be obtained when using them to solve inverse problems [43, 39, 162, 181, 96, 115].

However, these diffusion inverse solvers (DIS) require well-trained diffusion models, which in turn require large amounts of clean training data [168, 73]. It may be infeasible to collect large training data sets in many applications such as medical imaging [43, 167, 86], black hole imaging [61, 60], and phase retrieval [115, 188]. For example, in practical applications such as dynamic CT reconstruction [147] and single photon emission CT [118], obtaining high quality measurements, which can lead to reconstructions closely approximating the ground truth, can be slow or potentially harmful to patients, so only very small datasets of clean images are available. Moreover, for very challenging inverse problems such as black hole imaging [61] and Fresnel phase retrieval [71], no ground truth images are known, so one must obtain a reconstruction from only a single measurement $\boldsymbol{y}$.

This chapter based on [81].

In these practical applications with extremely limited or even nonexistent available data (measurement-only), it can be difficult or even impossible to train a diffusion model to approximate the underlying distribution well. Therefore, it is desirable to develop a method that can be applied regardless of dataset limitations. In this paper, we investigate applying a self-supervised loss at reconstruction time that allows the network to be adapted to the test data. Specifically, we investigate three different settings with various data availability ranging from nonexistent, limited, and abundant training data: (1) the *dataless* setting in which no training data is available and we are only given measurement $\boldsymbol{y}$, (2) the *small dataset* setting in which we are only given a small number of samples $\boldsymbol{x}$ that belong to the same distribution as the test dataset, (3) and the *in-distribution* setting in which we have sufficient training data of the same distribution as the test dataset. Our goal is to develop methods that can be used to adapt the trained model to testing data in these settings with varying training data availability.

Recently, some previous works have aimed to address these problems by demonstrating that diffusion models have a stronger generalization ability than other deep learning methods [86], and that slight distribution mismatches between the training data and test data may not significantly degrade the reconstructed image quality. However, in cases of particularly sparse or noisy measurements, as well as when the test data is severely out of distribution (OOD) with a significant domain shift, an improper choice of training data leads to an incorrect prior that causes substantial image degradation and hallucinations [61, 10]. To address these challenges in the dataless setting, recent works first train a network on a large sample of synthetic data that may differ greatly from the test data. Then, at reconstruction time, the trained diffusion network is updated based on the measurement $\boldsymbol{y}$ [10, 47]. This adaptation aims to shift the underlying prior learned by the network towards the appropriate prior corresponding to the test data distribution. However, with the large number of network parameters, a parameter-sensitive network adaptation approach is required at test time to avoid overfitting to the measurement. Furthermore, these methods have not been tested in the small dataset setting, and in the in-distribution setting, excessive network refining yields inconsistent improvement and degradation in the reconstructed image [10, 47].

Patch-based diffusion models have shown success both for image generation [183, 55] and for inverse problem solving [82]. In particular, the method of [82] involves training networks that take in only patches of images at training and reconstruction time, learning priors of the entire images from only image patches based on positional encoding. In cases of limited training data, [82] shows that patch-based diffusion models outperform whole image models for solving certain inverse problems. These works motivate our key insight

that patch-based diffusion priors potentially obtain stronger generalizability than whole-image diffusion priors for both the dataless setting and the small dataset setting, given a severe lack of data. Inspired by this, we propose to use patch-based diffusion models to tackle the challenges arising from data limitations and mismatched distributions in a unified way. We first develop a test-time adaptation method to take a network trained on patches from a mismatched distribution and adapt it on the fly at reconstruction time. We then show that this method can improve image quality consistently *in all three settings* with varying data availability ranging from abundant, to limited and nonexistent.

In summary, our contributions are as follows:

- We integrate the patch-based diffusion model with the deep image prior (DIP) framework to take a network trained on a mismatched distribution and adapt it at reconstruction time towards the test-time distribution using a self-supervised loss.
- Experimentally, we find this approach of test-time diffusion adaptation leads to improved image quality both in terms of quantitative and qualitative metrics, outperforming using whole-image diffusion models under the same setting. In particular, even when the diffusion model was trained in-distribution, test-time adaptation offers an additional improvement.
- Theoretically, we show that in certain cases, using the self-supervised loss with a patch-based prior leads to a beneficial form of data augmentation compared to whole-image diffusion prior, providing further justification for the superior generalizability of adapting the patch-based prior.

## 7.2 Methods

This section extends the patch-based diffusion model framework of [82] so that it can be applied with the self-supervised loss for test-time adaptation.

### 7.2.1 Patch-based prior

We first zero pad the $N \times N$ image by an amount $P$ on each side and model the resulting padded image $\boldsymbol{x}$. When choosing the $i$th patch offset tuple $(o_1, o_2) \in \{0, \ldots, P-1\}^2$ in Figure 7.1, we partition $\boldsymbol{x}$ into many square patches and one bordering region consisting of all zeros. Since $k = N/P$ patches are needed in one direction to perfectly cover the

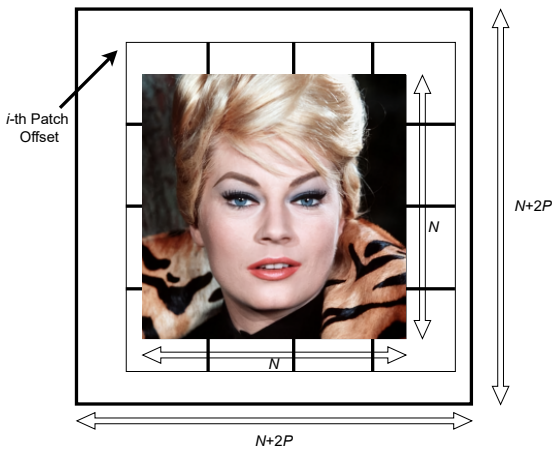



**FIG** 7.1 – Schematic for zero padding and partitioning image into patches. Each index $i$ represents one of $P^2$ possible ways to choose a patch offset tuple.

image, our model for the data distribution has the form

$$p(\boldsymbol{x}) = \frac{1}{Z} \prod_{i=1}^{P^2} p_{i,B}(\boldsymbol{x}_{i,B}) \prod_{r=1}^{(k+1)^2} p_{i,r}(\boldsymbol{x}_{i,r}), \tag{7.1}$$

where $\boldsymbol{x}_{i,B}$ represents the bordering region of $\boldsymbol{x}$ that depends on the specific value of $i$, $p_{i,B}$ is the probability distribution of that region, $\boldsymbol{x}_{i,r}$ is the $r$th $P \times P$ patch when using the partitioning scheme corresponding to the value of $i$, $p_{i,r}$ is the probability distribution of that region, and $Z$ is a normalizing factor. This model uses many possible tilings of the image, eliminating boundary artifacts that would occur if only one tiling was used.

For training, we use a neural network $D_\theta(\boldsymbol{x}, \sigma_t)$ that accepts a noisy image $\boldsymbol{x}$ (or a patch of that image) and the noise level $\sigma_t$. For each patch, we define the $x$ positional array as the 2D array consisting of the $x$ positions of each pixel of the image, scaled between -1 and 1, and the $y$ positional array is similarly defined for the $y$ positions. To allow the network to learn different patch distributions at different locations in the image, we extract the corresponding patches of these positional arrays and concatenate them along the channel dimension of the noisy image patch and treat the entire array as the network input. Since we are using a patch-based prior, we perform denoising score matching on patches of an image instead of the whole image. Hence, the training loss is given by

$$\arg\min_{\boldsymbol{\theta}} \mathbb{E}_{t\sim\mathcal{U}(0,T)} \mathbb{E}_{\boldsymbol{x}\sim p(\boldsymbol{x})} \mathbb{E}_{\boldsymbol{\epsilon}\sim\mathcal{N}(0,\sigma_t^2 I)} \| D_\theta(\boldsymbol{x} + \boldsymbol{\epsilon}, \sigma_t) - \boldsymbol{x} \|_2^2, \tag{7.2}$$

where here $\boldsymbol{x} \sim p(\boldsymbol{x})$ represents a patch drawn from a sample of the training dataset, $\sigma_t$ is a predetermined noise schedule, and $\mathcal{U}$ denotes the uniform distribution. See [82] and Appendix A.4.3 for more details about the patch-based training process.

### 7.2.2 Diffusion self-supervision

We first review how to apply the self-supervised loss in conjunction with diffusion models. For each specific measurement $\boldsymbol{y}$, the original DIP framework optimizes the network parameters $\boldsymbol{\theta}$ via the self-supervised loss (2.5) from the predicted reconstructed image. Diffusion models provide a prediction of the reconstructed image at each timestep: namely, the expectation of the clean image $\mathbb{E}[\boldsymbol{x}_0|\boldsymbol{x}_t]$ is approximated by the denoiser $D_\theta(\boldsymbol{x}_t)$ via Tweedie's formula. Then the expectation conditioned on the measurement $\mathbb{E}[\boldsymbol{x}_0|\boldsymbol{x}_t, \boldsymbol{y}]$ can be obtained through one of many methods of enforcing the data fidelity constraint. That conditional expectation should be used in test-time adaptation of diffusion models.

We begin with the unconditional expectation by leveraging the patch-based prior. Following (7.1), we apply Tweedie's formula to express the denoiser of $\boldsymbol{x}$ solely in terms of denoisers of the patches of $\boldsymbol{x}$. Because the outermost product is computationally very expensive, in practice we approximate $D_\theta(\boldsymbol{x})$ using only a single randomly selected value of offset index $i$ for each denoiser evaluation:

$$D_\theta(\boldsymbol{x}) \approx D_{i,B}(\boldsymbol{x}_{i,B}) + \sum_{r=1}^{(k+1)^2} D_{i,r}(\boldsymbol{x}_{i,r}). \tag{7.3}$$

By definition, $D_{i,B}(\boldsymbol{x}_{i,B}) = 0$ and we compute each $D_{i,r}(\boldsymbol{x}_{i,r})$ with the network. Note that (7.3) provides an *unconditional* estimate of the clean image; to obtain an approximate conditional estimate $D_\theta(\boldsymbol{x}_t|\boldsymbol{y})$ of the clean image, we run $C$ iterations of the conjugate gradient descent algorithm for minimizing $\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_2$, initialized with the unconditional estimate [41].

The image that is being reconstructed might not come from the distribution of the training images. Hence, the estimate $D_\theta(\boldsymbol{x}_t|\boldsymbol{y})$ may be far from the true denoised image. Thus, we use $\boldsymbol{y}$ to update the parameters of the network such that $D_\theta(\boldsymbol{x}_t|\boldsymbol{y})$ becomes more consistent with the measurement:

$$\boldsymbol{\theta} \leftarrow \arg\min_{\boldsymbol{\theta}} \|\boldsymbol{y} - \boldsymbol{A}\, D_\theta(\boldsymbol{x}_t|\boldsymbol{y})\|_2^2. \tag{7.4}$$

Previously, additional LoRA parameters [78] were used as an injection to the network to leave the original parameters unchanged during this process [10, 47]. However, the

**Algorithm 5** Test-time Adapted Diffusion Inverse Solver

**Require:** $\sigma_1 < \sigma_2 < \ldots < \sigma_T$, $\epsilon > 0$, $P, C, \boldsymbol{y}, K$
  Initialize $\boldsymbol{x} \sim \mathcal{N}(0, \sigma_T^2 \boldsymbol{I})$
  **for** $t = T : 1$ **do**
    **if** $t$ mod $K = 0$ **then**
      Compute $D_\theta(\boldsymbol{x}_t)$ using (7.3) with a random index $i$
      Run $C$ iterations of CG initialized with $D_\theta(\boldsymbol{x}_t)$ to obtain $D_\theta(\boldsymbol{x}_t|\boldsymbol{y})$
      Define $L(\boldsymbol{\theta}) = \|\boldsymbol{y} - \boldsymbol{A} D_\theta(\boldsymbol{x}_t|\boldsymbol{y})\|_2^2$
      Update $\boldsymbol{\theta}$ by backpropagating $L(\boldsymbol{\theta})$
    **end if**
    Sample $\boldsymbol{z} \sim \mathcal{N}(0, \sigma_t^2 \boldsymbol{I})$
    Set $\alpha_t = \epsilon \cdot \sigma_t^2$
    Compute $D(\boldsymbol{x}_t)$ using (7.3) with a random index $i$
    Run $C$ iterations of CG for (2.7) initialized with $D(\boldsymbol{x}_t)$
    Set $\boldsymbol{s}_t = (D - \boldsymbol{x}_t)/\sigma_t^2$
    Set $\boldsymbol{x}_{t-1}$ to $\boldsymbol{x}_t + \frac{\alpha_t}{2} \boldsymbol{s}_t + \sqrt{\alpha_t} \boldsymbol{z}$
  **end for**

effect of using different ranks for LoRA versus other methods of network fine-tuning on DIS has not been studied extensively, so we opt to update all the weights of the network in this step. Appendix A.4.1 shows results from using the LoRA module.

Crucially, iterative usage of CG for computing the conditional denoiser allows for simple and efficient backpropagation through this loss function, a task that would be much more computationally challenging if another DIS such as [39] or [181] were used. Furthermore, because the number of diffusion steps is large and the change in $\boldsymbol{x}_t$ is small between consecutive timesteps, we apply this network refining step only for certain iterations of the diffusion process, reducing the computational burden.

After this step, we apply the refined network to compute a new estimate of the score of $\boldsymbol{x}_t$ and then use it to update $\boldsymbol{x}_t$. Similar to the network refining step, we use the stochastic version of the denoiser given by (7.3) rather than the full version. Ref. [82] showed that for patch-based priors, Langevin dynamics [166] worked particularly well as a sampling algorithm, so we used it here in conjunction with CG steps to enforce data fidelity. Algorithm 5 summarizes the entire test-time adapted method for solving inverse problems.

### 7.2.3 Patch-based training

In the small dataset and in-distribution settings, we are provided clean training images that can be used to train the patch-based network. In the small dataset setting, the dataset

is too small to directly train a diffusion model from scratch, so we initialize the training process with a checkpoint trained on a different large dataset and then fine-tune this checkpoint on the small dataset. Works such as [138] and [207] found that this fine-tuning process allows diffusion models to be trained with a much smaller dataset than would otherwise be required. On the other hand, for the in-distribution setting, we initialize the weights of the network randomly and train on the large in-distribution dataset.

Ref. [183] found that training with varying patch sizes improved image generation performance compared to fixing the patch size to that used during inference. Here, we also applied a varying patch size scheme during fine-tuning as a method of data augmentation. We used the UNet architecture in [73] that can accept images of different sizes. Hence, the loss becomes

$$\arg\min_{\boldsymbol{\theta}} \mathbb{E}_{t\sim\mathcal{U}(0,T)}\mathbb{E}_{\boldsymbol{x}\sim p_d(\boldsymbol{x})}\mathbb{E}_{\boldsymbol{\epsilon}\sim\mathcal{N}(0,\sigma_t^2 I)}\|D_\theta(\boldsymbol{x}+\boldsymbol{\epsilon},\sigma_t)-\boldsymbol{x}\|_2^2, \tag{7.5}$$

where $\boldsymbol{x}\sim p_d(\boldsymbol{x})$ represents drawing a patch of random size and location from an image belonging to the fine-tuning dataset. Appendix A.4.3 provides full details of the training process.

### 7.2.4 Theoretical Analysis

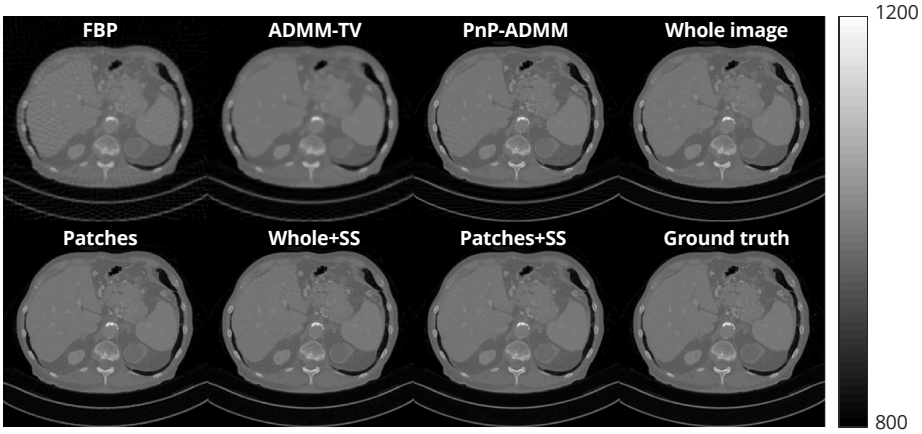



**FIG** 7.2 – Dataless setting: Results of 60 view CT reconstruction using self supervised (SS) loss. The display uses modified HU units to show more contrast between organs.

This section provides a rough sketch of why Algorithm 5 should work better for the patch-based model compared to a whole-image model. Firstly, rewrite (7.3) as follows:

$$D(\boldsymbol{x}) = \sum_c \boldsymbol{G}_c' D_\theta(\boldsymbol{G}_c \boldsymbol{x}, c), \tag{7.6}$$

where $c$ denotes the patch location, $\boldsymbol{G}_c$ denotes a patch extracting operator that extracts the patch corresponding to location $c$ from the whole image $\boldsymbol{x}$. Thus $\boldsymbol{G}_c'$ is an operator that takes a patch and returns a whole image with the corresponding patch filled in (and the rest of the entries are zeros). Note that $c$ is input to the patch-based network $D_\theta$ through positional encoding.

Now we analyze (7.4) using this framework. The loss function for self-supervision is given by:

$$L(\boldsymbol{\theta}) = \left\| \boldsymbol{y} - \boldsymbol{A} \sum_c \boldsymbol{G}_c' D_\theta(\boldsymbol{G}_c \boldsymbol{x}, c|\boldsymbol{y}) \right\|_2^2. \tag{7.7}$$

For inverse problems such as superresolution and CT reconstruction, $\boldsymbol{A}$ is a wide matrix that has full row rank. Hence, even when the measurement $\boldsymbol{y}$ is noisy, there exists some $\boldsymbol{x}_0$ with $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}_0$. Then we have

$$L(\boldsymbol{\theta}) = \left\| \boldsymbol{A}(\boldsymbol{x}_0 - \sum_c \boldsymbol{G}_c' D_\theta(\boldsymbol{G}_c \boldsymbol{x}, c|\boldsymbol{y})) \right\|_2^2 \tag{7.8}$$

$$L(\boldsymbol{\theta}) \leq \|\boldsymbol{A}\|_2^2 \left\| \boldsymbol{x}_0 - \sum_c \boldsymbol{G}_c' D_\theta(\boldsymbol{G}_c \boldsymbol{x}, c|\boldsymbol{y}) \right\|_2^2 \tag{7.9}$$

$$= \|\boldsymbol{A}\|_2^2 \left\| \sum_c \boldsymbol{G}_c' \boldsymbol{G}_c \boldsymbol{x}_0 - \sum_c \boldsymbol{G}_c' D_\theta(\boldsymbol{G}_c \boldsymbol{x}, c|\boldsymbol{y}) \right\|_2^2, \tag{7.10}$$

where in the last step, we have used the fact that at each diffusion iteration, the patches are nonoverlapping. Thus,

$$L(\boldsymbol{\theta}) \leq \|\boldsymbol{A}\|_2^2 \left\| \sum_c \boldsymbol{G}_c' (\boldsymbol{G}_c \boldsymbol{x}_0 - D_\theta(\boldsymbol{G}_c \boldsymbol{x}, c|\boldsymbol{y})) \right\|_2^2. \tag{7.11}$$

Now we have a sum of the form $\|\boldsymbol{z}_1 + \ldots + \boldsymbol{z}_n\|_2^2$, which [162] showed to be upper bounded by $K(\|\boldsymbol{z}_1\|_2 + \ldots + \|\boldsymbol{z}_n\|_2^2)$ for a fixed constant $K$. Applying this inequality and absorbing

$\|\boldsymbol{A}\|_2^2$ into the constant, we have

$$L(\boldsymbol{\theta}) \leq K \sum_c \|\boldsymbol{G}_c'(\boldsymbol{G}_c\boldsymbol{x}_0 - D_\theta(\boldsymbol{G}_c\boldsymbol{x}, c|\boldsymbol{y}))\|_2^2 \tag{7.12}$$

$$= K \sum_c \|\boldsymbol{G}_c\boldsymbol{x}_0 - D_\theta(\boldsymbol{G}_c\boldsymbol{x}, c|\boldsymbol{y})\|_2^2 \tag{7.13}$$

A similar derivation for the whole image model shows that the loss in that case is bounded by

$$L_w(\boldsymbol{\theta}) \leq K\|\boldsymbol{x}_0 - D_\theta(\boldsymbol{x}|\boldsymbol{y})\|_2^2. \tag{7.14}$$

Table A.18 shows that performance is improved by using more backpropagation iterations for the loss function; hence, although in practice we only perform a fixed number of iterations for speed, optimally we should aim to reduce the loss $L(\boldsymbol{\theta})$ to zero. Observe that (7.13) has the same form as the loss that would be used for refining the network with a whole image model (7.14). However, now instead of a loss of a single image, we now have individual losses of many patches of an image. For example, the experiments of Table 7.1 used 25 patches to tile each image for each diffusion iteration, so we had 25 losses. This method of data augmentation helps explain why the patch-based model obtains better performance than the whole-image model when performing test-time adaptation. We additionally note that although the positional encoding input into the network is different for each patch, the network does not separately learn a distribution for each position, as the weights are shared across these different positions. This is analogous to the analysis of [54], where a single diffusion model was trained on the 1000 classes of ImageNet with the class label of the image being included as an additional input to the network. Since each class only had around 1000 images, it would have been very difficult to train a diffusion model on only one of the classes, but by training across all the classes at once, a much better network can be trained.

## 7.3 Experiments

This section reports empirical results on sparse-view CT reconstruction, image deblurring, and image super-resolution. For each computational imaging application, we illustrate the benefits of using the self supervised (SS) loss in all three of the settings described in the introduction: *dataless* (zero-shot SS), *small dataset* (fine tuning followed by SS), and *in-distribution* (where one might expect SS to be counter-productive).




**FIG** 7.3 – Dataless setting: Results of deblurring using test-time adaptation via the self supervised (SS) loss and several comparison methods.

**Experimental setup.** For the CT experiments, we used the AAPM 2016 CT challenge data from [135]. We applied the same data processing methods as in [82] with the exception that we used all the XY (transaxial) slices of size $256 \times 256$ from the 9 training volumes to train the in distribution networks, yielding a total of 5936 slices. For the deblurring and superresolution experiments, we used the CelebA-HQ dataset [126] with each image having size $256 \times 256$. The test data was a randomly selected subset of 10 of the images not used for training. In all cases, we report the average metrics across the test images: peak SNR (PSNR) in dB, and structural similarity metric (SSIM) [185].

In the dataless and small dataset settings, since there is insufficient data from the test distribution to train the diffusion model well, we first trained the network using generated synthetic data. This data consisted of phantom images consisting of randomly placed ellipses of different shapes and sizes. See Fig. A.37 for examples. These phantoms can be generated on the fly in large quantities. We used networks trained on grayscale phantoms for the CT experiments and networks trained on RGB phantoms for the deblurring and superresolution experiments. Appendix A.4.2 contains precise specifications of the phantoms. Then in the small dataset setting, we fine-tuned the network (originally trained on the phantom images) on the small dataset. The small dataset consisted of 10 images randomly selected from the in-distribution training set; we also ran ablation studies using different quantities of in-distribution data in Appendix A.4.1. In contrast, for the in-distribution setting, there was sufficient data to directly train a diffusion model on the in-distribution data, so we initialized the network weights randomly and trained only on the in-distribution dataset.

We trained the patch-based networks with $64 \times 64$ patches and used a zero padding value of 64, so that 5 patches in both directions were used to cover the target image. We used the network architecture in [93] for both the patch-based networks and whole-image networks. All networks were trained on PyTorch using the Adam optimizer with 2 A40 GPUs.

**Diffusion test-time adaptation**. To evaluate the effectiveness of test-time adaptation, we applied Algorithm 5 to solve each of the inverse problems. For the forward and backward projectors in CT reconstruction, we used the implementation provided by the ODL [173]. We performed two sparse-view CT (SVCT) experiments: one using 20 projection views, and one using 60 projection views. Both of these used a parallel-beam forward projector where the detector size was 512 pixels. For the deblurring experiments, we used a uniform blur kernel of size $9 \times 9$ and added white Gaussian noise with $\sigma = 0.01$ where the clean image was scaled between 0 and 1. For the superresolution experiments, we used a scaling factor of 4 with downsampling by averaging and added white Gaussian noise with $\sigma = 0.01$.

For the comparison methods, we ran experiments that directly used the diffusion model without test-time adaptation. In particular, we used the network trained only on phantoms for the dataless setting, the fine-tuned network in the small dataset setting, and the in-distribution network in the last setting. We used the same sampling algorithm (Langevin dynamics) and inverse problem solving method (conjugate gradient descent) as Algorithm 5, but removed the test-time adaptation step. Additionally, for these diffusion model methods, we implemented both the patch-based version as well as the whole-image version. The whole-image networks were trained with the loss function in (4.3) and used the same network architecture as the patch-based models, but the input of the network was the entire image and did not contain positional encoding information.

We also compared with more traditional methods: applying a simple baseline, reconstructing via the total variation regularizer (ADMM-TV), and two plug and play (PnP) methods: PnP-ADMM [196] and PnP-RED [83]. For CT, the baseline was obtained by applying the filtered back-projection method to the measurement $\boldsymbol{y}$. For deblurring, the baseline was simply equal to the blurred image $\boldsymbol{y}$. For superresolution, the baseline was obtained by upsampling the low resolution image $\boldsymbol{y}$ using nearest-neighbor interpolation. The implementation of ADMM-TV is in [74]. Finally, since we assume we do not have access to a large sample of clean training data, we used the off-the-shelf denoiser BM3D [49]. Appendix A.4.3 contains the values of all the parameters of the algorithms.

Tables 7.1, 7.3, and 7.4 show the main results for the dataless, small dataset, and in-distribution settings, respectively. In the latter two settings, we chose to redisplay only

TBL 7.1 – Comparison of quantitative results on four different inverse problems in the dataless setting with the self-supervised loss (SS). Results are averages across all images in the test dataset. Best results for practical use are in bold.

| Method | CT, 20 Views | | CT, 60 Views | | Deblurring | | Superresolution | |
|---|---|---|---|---|---|---|---|---|
| | PSNR↑ | SSIM↑ | PSNR↑ | SSIM↑ | PSNR↑ | SSIM↑ | PSNR↑ | SSIM↑ |
| Baseline | 24.93 | 0.613 | 30.15 | 0.784 | 23.93 | 0.666 | 25.42 | 0.724 |
| ADMM-TV | 26.81 | 0.750 | 31.14 | 0.862 | 27.58 | 0.773 | 25.22 | 0.729 |
| PnP-ADMM [196] | 30.20 | 0.838 | 36.75 | 0.932 | 28.98 | 0.815 | 27.29 | 0.796 |
| PnP-RED [83] | 27.12 | 0.682 | 32.68 | 0.876 | 28.37 | 0.793 | 27.73 | 0.809 |
| Whole image | 28.11 | 0.800 | 33.10 | 0.911 | 25.85 | 0.742 | 25.65 | 0.742 |
| Patches [82] | 27.44 | 0.719 | 33.97 | 0.934 | 26.77 | 0.782 | 26.12 | 0.759 |
| Whole+SS [10] | 33.19 | 0.861 | 40.47 | 0.957 | 29.50 | 0.831 | 27.07 | 0.701 |
| Patches+SS (Ours) | **33.77** | **0.874** | **41.45** | **0.969** | **30.34** | **0.860** | **28.10** | **0.827** |

the results of the best baseline out of the baseline, ADMM-TV, PnP-ADMM, and PnP-RED shown in Table 7.1 and labeled that row "Best baseline." Particularly for the dataless setting, applying the test-time adaptation method yields much higher quantitative results when averaged across the test dataset than simply using the pretrained diffusion model in all the inverse problems. However, we observe that even when the pretrained diffusion model was trained on the large in-distribution dataset, including the test-time adaptation step still resulted in further improvement in image quality. Thus, in contrast with many other self-supervised methods such as DIP, our method can avoid overfitting to the measurement and even benefit from test-time refinement. Appendix A.4.1 further analyzes overfitting and shows that by increasing the number of network refining iterations done per diffusion iteration, the image quality does not drop, indicating that overfitting is avoided. We further note that in all three settings, when using the self-supervised loss, the patch-based prior outperformed the whole image prior, which is consistent with our theoretical analysis of Algorithm 5. Lastly, Fig. 7.2 shows that some artifacts appear in the whole-image SS method that are absent in our patch SS method.

We also ran ablation studies to examine the effect of various parameters on the proposed method. [10] and [47] used the LoRA module for solving single-measurement inverse problems with diffusion models. We tested this method for CT reconstruction and

deblurring with different rank adjustments and found this method to be inferior to modifying the weights of the entire network. We also ran experiments using networks with different numbers of weights. Appendix A.4.1 shows the results of these experiments.

The initial motivation of using patch-based diffusion models was partially to solve high resolution imaging problems [82]. To show that our method scales to larger images, we ran experiments on 60 view CT reconstruction and deblurring with $512 \times 512$ images in the dataless setting. For the CT experiments, we still used the AAPM dataset [135] processed in the same way as for Table 7.1, but kept the slices in their original size of $512 \times 512$. For deblurring, we used the FFHQ dataset [94] which contains images of size $512 \times 512$. We scaled each of the RGB channels to between 0 and 1. We used a uniform blur kernel of size $17 \times 17$ and added noise with $\sigma = 0.01$. We used the same patch-based networks trained for Table 7.1 as initializations for these out of distribution experiments. Since the patch size at reconstruction time was kept to be $64 \times 64$ as before, we used 9 patches in both directions (for 81 total) to tile each image. Table 7.2 shows results of these experiments, where our method obtained the highest quality reconstruction. Although the improvement is modest, note that we trained the patch-based model on phantoms images of size $256 \times 256$, which is extremely far out of distribution from the test dataset.

**TBL** 7.2 – Results of inverse problem solving in dataless setting for $512 \times 512$ images.

| Method | CT, 60 views | | Deblur | |
|---|---|---|---|---|
| | PSNR↑ | SSIM ↑ | PSNR↑ | SSIM ↑ |
| Baseline | 28.33 | 0.700 | 24.11 | 0.649 |
| ADMM-TV | 29.36 | 0.788 | 28.14 | 0.760 |
| PnP-ADMM | 37.48 | 0.910 | 29.77 | 0.812 |
| Patch, naive | 29.32 | 0.793 | 26.58 | 0.749 |
| Patch, SS | **37.82** | **0.919** | **30.35** | **0.825** |

**Small dataset fine-tuning.** We further examined the effects of fine-tuning the networks using the small dataset with the patch-based model and the whole image model. Figures 7.6 and 7.7 further investigate the effect of overfitting. For different amounts of training time using the small in-distribution dataset, we ran the reconstruction algorithm for 60-view CT. While the whole-image model exhibited substantial image degradation when the network was fine-tuned for too long, the patch-based model retained relatively

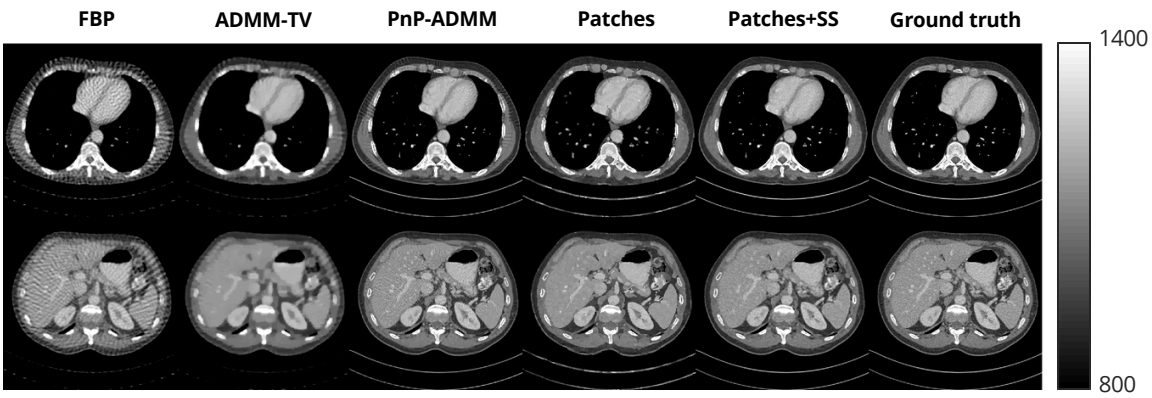



**FIG** 7.4 – Results of 60 view reconstruction on $512 \times 512$ images in dataless setting displayed in Hounsfield units.

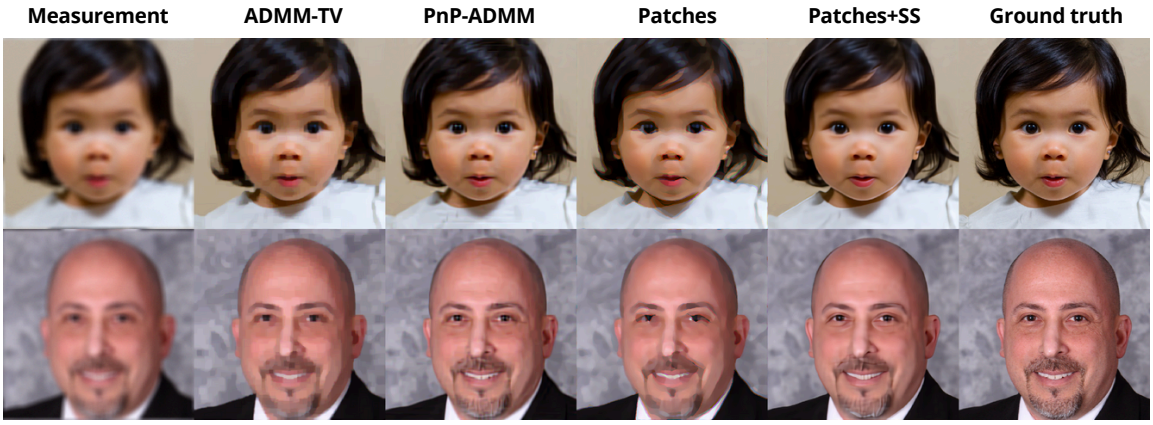



**FIG** 7.5 – Results of deblurring on $512 \times 512$ images in dataless setting.

stable performance throughout the entire training process. This illustrates that whole-image diffusion models exhibits severe overfitting problems when only a small amount of training data is unavailable, similar to the original DIP method. Furthermore, patch-based diffusion models assist greatly with this problem and the results are evident for solving inverse problems.

To look at the priors learned by the different models from fine-tuning, we unconditionally generated images from the checkpoints obtained by fine-tuning on the 10 image CT dataset. Figure 7.9 shows a subset of the generated images where we used the checkpoints obtained after 4 hours of training. The top two rows consist of images generated by the whole-image model and the bottom two rows consist of images generated by the patch

TBL 7.3 – Comparison of results for using self-supervised (SS) diffusion models in small dataset setting. The diffusion model is first trained on ellipse phantoms and then fine-tuned with the small dataset. Best results are in bold.

| Method | CT, 20 Views | | CT, 60 Views | | Deblurring | | Superresolution | |
|---|---|---|---|---|---|---|---|---|
| | PSNR↑ | SSIM↑ | PSNR↑ | SSIM↑ | PSNR↑ | SSIM↑ | PSNR↑ | SSIM↑ |
| Best baseline | 30.20 | 0.838 | 36.75 | 0.932 | 28.98 | 0.815 | 27.73 | 0.809 |
| Whole image | 33.09 | 0.875 | 40.54 | 0.964 | 28.41 | 0.812 | 27.29 | 0.775 |
| Patches | 33.44 | 0.875 | 41.21 | 0.965 | 29.25 | 0.840 | 28.10 | 0.827 |
| Whole image+SS | 34.57 | 0.881 | 41.34 | 0.962 | 30.16 | 0.852 | 27.72 | 0.814 |
| Patches+SS | **36.43** | **0.914** | **42.42** | **0.971** | **30.56** | **0.867** | **28.60** | **0.834** |

TBL 7.4 – Comparison of results for using diffusion models trained on thousands of in-distribution images to solve inverse problems. Best results are in bold. Self-supervised refinement of network weights is beneficial even in this setting.

| Method | CT, 20 Views | | CT, 60 Views | | Deblurring | | Superresolution | |
|---|---|---|---|---|---|---|---|---|
| | PSNR↑ | SSIM↑ | PSNR↑ | SSIM↑ | PSNR↑ | SSIM↑ | PSNR↑ | SSIM↑ |
| Best baseline | 30.20 | 0.838 | 36.75 | 0.932 | 28.98 | 0.815 | 27.73 | 0.809 |
| Whole image | 33.99 | 0.886 | 41.67 | 0.969 | 29.87 | 0.851 | 28.33 | 0.801 |
| Patches | 34.02 | 0.889 | 41.70 | 0.967 | 30.12 | 0.865 | 28.49 | 0.835 |
| Whole image+SS | 35.38 | 0.897 | 41.68 | 0.966 | 30.31 | 0.854 | 27.96 | 0.797 |
| Patches+SS | **36.82** | **0.923** | **42.33** | **0.970** | **30.78** | **0.875** | **28.72** | **0.842** |

diffusion model. To emphasize the memorization point, we grouped together similar looking images in the top two rows: it can be seen that the images in each group look virtually identical, despite the fact that the pure white noise initializations for each sample was different. On the other hand, while the samples generated by the patch diffusion model also show some unrealistic features, they all show some distinct features, which implies that this model has much better generalization ability.

Finally, to demonstrate that our method also works well even when the mismatched distribution is closer to the true distribution, we also ran an experiment where the networks were initially trained on the LIDC-IDRI dataset of CT scans [6]. We extracted 10000 2D slices from the 3D volumes and rescaled all the images so that the pixel values were between 0 and 1. We then ran Algorithm 1 to perform CT reconstruction where the test

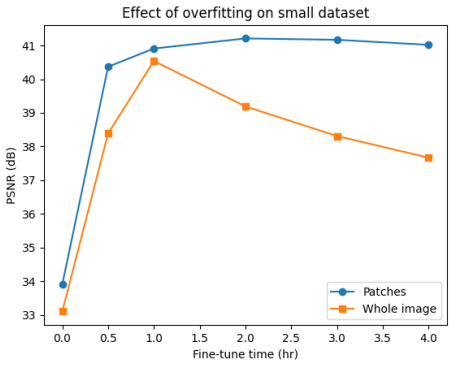



FIG 7.6 – Comparison of PSNR between patch-based model and whole-image model for overfitting in small dataset setting.

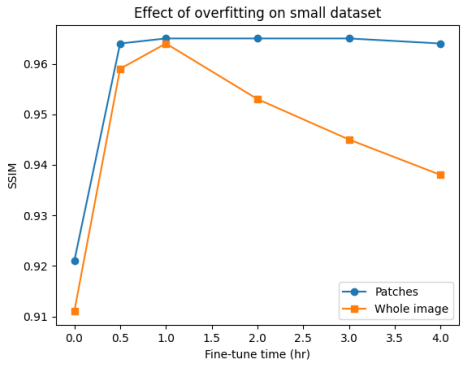



FIG 7.7 – Comparison of SSIM between patch-based model and whole-image model for overfitting in small dataset setting.




FIG 7.8 – **Small dataset setting**: Results of inverse problem solving. Top row is 60 view CT recon, middle row is deblurring, and bottom row is superresolution.

dataset was the same as the one used in Table 7.1. Table 7.5 shows the results of this experiment. Our method achieved better quantitative results than the whole-image method and even outperformed the reconstructions using the in distribution network but without any test-time adaptation.

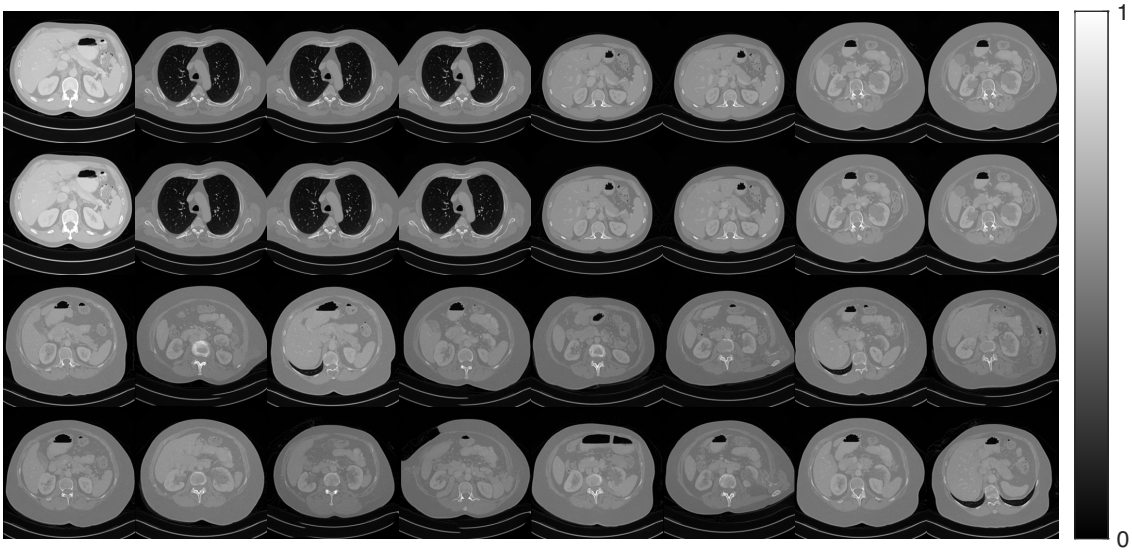



**FIG** 7.9 – Unconditional generation of CT images from networks fine-tuned in the small dataset setting. Top two rows were generated with the whole image model; bottom two rows were generated with the patch-based model.

## 7.4 Conclusion

This paper presented a method of using patch-based diffusion models with test-time adaptation to solve inverse problems when the data distribution might be mismatched from the trained network. In particular, we conducted experiments in setting when no data is available, when a small dataset of training images is available, and when a large in-distribution dataset is available. In all the settings, applying the self-supervised loss improved image quality, even for a well-trained network. Furthermore, the patch-based method outperformed whole-image methods in a variety of inverse problems and we provided theoretical justifications to explain this improvement. In the future, more work could be done on using acceleration methods for faster reconstruction, exploring other less computationally expensive methods of fine-tuning the network geared toward inverse problem solving, and methods of refining the prior when a set of measurements are available [198]. Limitations of the work include a slow runtime for the test-time adapted algorithm and a lack of theoretical guarantees for dataset size requirements. Providing uncertainty quantification is also an open problem for such self-supervised methods.

TBL 7.5 – CT reconstruction results where the initial checkpoint was trained on LIDC dataset and refined on the fly with the AAPM measurement.

| Dataset size | CT, 20 views | | CT, 60 views | |
|---|---|---|---|---|
| | PSNR↑ | SSIM ↑ | PSNR↑ | SSIM ↑ |
| Whole image | 33.04 | 0.874 | 40.43 | 0.949 |
| Patches | 33.88 | 0.886 | 40.96 | 0.955 |
| Whole image+SS | 35.01 | 0.894 | 41.95 | 0.967 |
| Patches+SS (Ours) | 36.34 | 0.918 | 42.32 | 0.972 |

# CHAPTER 8

# SPAR: Refine a Single Pretrained Diffusion Model to Solve Inverse Problems in Many Modalities

## 8.1 Motivation

Inverse problems arise in real-world settings where an object cannot be observed directly and indirect measurements are obtained. Such inverse problems commonly occur in many imaging applications, for example, deblurring and super-resolution with natural images; as well as in medical imaging, such as computed tomography (CT) and magnetic resonance imaging (MRI) reconstruction. Many real-world inverse problems are highly ill-posed and difficult to solve because the solution may not be unique and the measurement noise can further complicate the reconstruction process. Furthermore, in practical applications, imaging modalities and resolutions can vary significantly, complicating the reconstruction process. Generally, the measurement acquisition process is represented as $\boldsymbol{y} = \mathcal{A}(\boldsymbol{x}) + \boldsymbol{\epsilon}$, where $\boldsymbol{y}$ is the unknown measurement, $\boldsymbol{x}$ is the unknown clean image to be recovered, $\mathcal{A}$ is an operator representing the measurement system, and $\boldsymbol{\epsilon}$ is unknown random noise.

A common approach to solving inverse problems is to enforce prior knowledge about $\boldsymbol{x}$ to obtain a high-quality reconstruction, such as sparsity in compressed sensing [148]. Recently, diffusion models have become state-of-the-art methods for generating high-quality image data from random noise samples [73, 166] and, as such, can be integrated as a learned prior into many inverse problem solvers. Since diffusion models were proposed to be

This chapter is based on a paper currently under review [79]. Code at `https://github.com/jasonhu4/SPAR`.

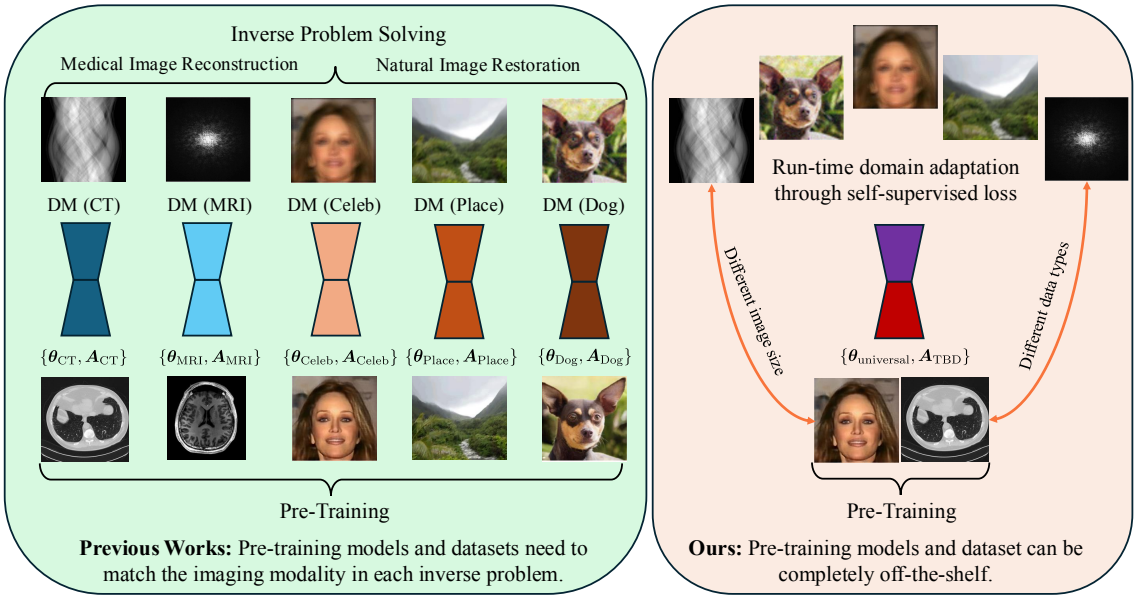



**FIG** 8.1 – Comparison of traditional domain-specific diffusion models (DM) versus the proposed SPAR method. Left: previous works require separate pre-training for each domain/modality (CT, MRI, CelebA, Places, Dogs). Right: The proposed method enables domain adaptation at testing time through self-supervised learning, avoiding the need for domain-specific pre-training.

trained unconditionally on natural images [73, 168], the key challenge in many diffusion inverse solvers (DIS) [181, 95, 39, 163, 115] is enforcing consistency with observed measurements while simultaneously leveraging the image prior learned by the unconditional diffusion model. Owing to the strong learned prior, diffusion models provide excellent performance for various inverse problems, especially when the measurements are very compressed or noisy [39, 42, 41, 115, 51, 5, 143].

Nevertheless, separate diffusion models are usually trained for different imaging applications, including different modalities or for various anatomic sites *e.g.*, contrast CT, low-dose CT, brain MRI, knee MRI, natural RGB images, etc. Training diffusion models from scratch is very computationally expensive and typically requires large-scale high-quality training datasets and substantial GPU resources [73, 168], which may be infeasible in many practical scientific or medical application domains having only scarce data. Therefore, on-the-fly network fine-tuning methods have been developed to tackle this challenge. Specifically, [47] and [10] first train a diffusion model on a dataset with constrained distribution, then perform network refining on the fly to solve an inverse problem where the clean image corresponds to a different data distribution but the same modality (*e.g.*,

training with brain MRI and then reconstructing knee MRI). This network refining relies on a self-supervised loss that enforces the clean image (and by extension, the output of the denoiser learned by the diffusion network) to be consistent with the measurement. By enforcing measurement consistency as a constraint, the pretrained diffusion model weights are updated as part of the reconstruction process, adapting the diffusion prior to out-of-distribution (OOD) test samples that were unseen during the original diffusion training phase. A patch-based prior [81] enhances these methods, improving robustness to overfitting to the measurement $\boldsymbol{y}$. However, these methods still exhibit instability at the tail ends of the noise scale [47], which requires technical tricks such as truncating the diffusion process to resolve. Furthermore, although existing methods have explored how to generalize across different distributions within the same imaging modalities (such as different CT datasets), they still require training multiple networks for applications of different modalities (*e.g.*, CT and MRI), as the image contents, pixel sizes and data types (*e.g.*, real and complex valued) differ across these modalities.

Our proposed ***Single Pretrain Always Refine (SPAR)*** method tackles these challenges as follows. First, we developed *a unified model architecture* based on a channel-wise conditioning mechanism that accommodates images of different modalities, including grayscale CT images, complex-valued MRI images, and RGB colored images. This versatile framework enables the diffusion network to be pretrained on a mixture of images from diverse modalities, thereby ensuring that the learned distribution captures a broad range of image types that can be reconstructed with test-time refinement. To further enhance the model flexibility, we adopt a *patch-based diffusion prior* that handles images of different sizes. Furthermore, motivated by the observation that diffusion sampling behavior varies across timesteps (i.e., noise levels), we propose a *novel majorizer-based self-supervised loss* function for a more effective test-time adaptation, especially at lower noise levels, while mitigating the risk of overfitting. Overall, our proposed method provides a unified framework that enables a single versatile pretrained diffusion model to address a wide range of inverse problems across different imaging modalities through test-time domain adaptation, as illustrated in Fig. 8.1. Our key contributions are as follows:

- We introduce ***SPAR***, a novel test-time adaptation framework for solving diverse inverse problems using a single, pretrained patch-based diffusion model. A core innovation is a channel-wise conditioning mechanism that allows the pretrained diffusion network to process different image modalities (e.g., grayscale, RGB, complex-valued) within a unified architecture.

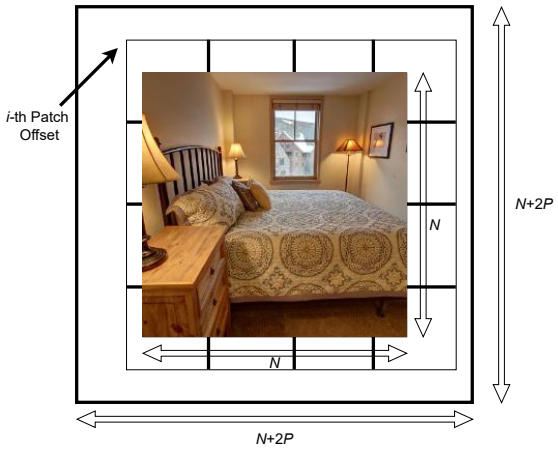



**FIG** 8.2 – Schematic illustration for zero padding and partitioning image into patches. Each index $i$ represents one of $P^2$ possible ways to choose a patch offset tuple.

- To address the distributional mismatch between training and test data, we propose a novel self-supervised loss function to continuously refine the model during reconstruction. We provide theoretical justification to show that, at lower noise levels, this loss function is less prone to errors that arise from imperfect denoising score matching at training time.
- With the proposed modality- and resolution-agnostic framework, we experimentally validate that SPAR achieves improved image qualitiy across a wide spectrum of inverse problems across imaging modalities and resolutions, even with only a single pretrained model.

## 8.2 Methods

### 8.2.1 Patch-based Prior

Previous works show that patch-based diffusion models achieve consistent performance in a wide variety of inverse problems where the amount of training data varies significantly [82, 81]. The patch-based diffusion models exhibit generalization capabilities especially under data-constrained conditions, which motivates our adoption of this framework as the foundation for our method.

In patch-based diffusion models, one firsts zero pads each $N \times N$ training image by $P$ pixels on each side. Slightly recycling notation, we let $\boldsymbol{x}$ denote the resulting padded image. We partition $\boldsymbol{x}$ into many square patches, with one bordering region of zeros, by first choosing the $i$th patch offset tuple $(o_1, o_2) \in \{0, \ldots, P-1\}^2$ according to Figure 8.2.

Since $k = N/P$ patches are needed in one direction to perfectly tile the image, our model for the data distribution has the form

$$p(\boldsymbol{x}) = \frac{1}{Z} \prod_{i=1}^{P^2} p_{i,B}(\boldsymbol{x}_{i,B}) \prod_{r=1}^{(k+1)^2} p_{i,r}(\boldsymbol{x}_{i,r}), \tag{8.1}$$

where $\boldsymbol{x}_{i,B}$ represents the bordering region of $\boldsymbol{x}$ that depends on the specific value of $i$, $p_{i,B}$ is the probability distribution of that region, $\boldsymbol{x}_{i,r}$ is the $r$th $P \times P$ patch when using the partitioning scheme corresponding to the $i$th patch offset, $p_{i,r}$ is the probability distribution of that region, and $Z$ is a normalizing factor. This model takes a product over all possible patch offsets, eliminating boundary artifacts that would occur if a fixed patch offset was used. However, when computing the denoiser of $\boldsymbol{x}$ using Tweedie's formula [56] in practice, we use a stochastic version of (8.1) corresponding to selecting a random index $i$ for each denoiser evaluation:

$$D_\theta(\boldsymbol{x}) \approx D_{i,B}(\boldsymbol{x}_{i,B}) + \sum_{r=1}^{(k+1)^2} D_{i,r}(\boldsymbol{x}_{i,r}). \tag{8.2}$$

### 8.2.2 Model Pretraining

For pretraining, we use a neural network $D_\theta(\boldsymbol{x}, \sigma_t)$ that takes as input a noisy image $\boldsymbol{x}$ (or a patch of that image) along with the noise level $\sigma_t$. To incorporate positional information, we define the $x$ positional array as a 2D array representing the $x$ coordinates of each pixel in the image, scaled between -1 and 1. Similarly, the $y$ positional array is defined for the $y$ coordinates of the pixels. To enable the network to learn patch distributions that vary based on location within the image, we extract patches from these positional arrays corresponding to the image patches. These positional patches are concatenated along the channel dimension of the noisy image patch, forming a unified input array for the network. (The network output is just the image patch, not the position channels.) As our approach is based on patch-based priors, we apply denoising score matching on image patches rather than the whole image. The training loss is:

$$\arg\min_{\boldsymbol{\theta}} \mathbb{E}_{t\sim\mathcal{U}(0,T)} \mathbb{E}_{\boldsymbol{x}\sim p(\boldsymbol{x})} \mathbb{E}_{\boldsymbol{\epsilon}\sim\mathcal{N}(0,\sigma_t^2 I)} \| D_\theta(\boldsymbol{x} + \boldsymbol{\epsilon}, \sigma_t) - \boldsymbol{x} \|_2^2, \tag{8.3}$$

where $\boldsymbol{x} \sim p(\boldsymbol{x})$ represents a patch drawn from an image in the training dataset, $\sigma_t$ is a predetermined noise schedule, and $\mathcal{U}$ denotes the uniform distribution. Training on patches of varying sizes can reduce training time and improve the network's generation

capability [183]. Thus, we use a UNet architecture as the backbone [73] to take in patches of different sizes, which is trained with varying patch sizes [183, 82].

**Channel-wise conditioning mechanism.** Real-world images span a wide range of modalities and types, so our aim is to train a diffusion model that can be adapted to the different possible images that may be required at reconstruction time. However, grayscale images only have 1 channel, whereas natural images have 3 channels. We handle both types of images with a single network that inputs and outputs 3-channel images by concatenating the image with itself. Specifically, we duplicate each grayscale image three times and concatenate it along the channel dimension so it becomes a 3-channel image that can fit as inputs to the network. We add the *same* random Gaussian noise to each channel; Appendix A.5.2 justifies this process over the alternative approach of using different noises across channels. Thus, the network should learn to denoise all 3 identical channels of this image.

Finally, we include an additional class label [93] as a network input to inform the network whether the image input was a grayscale or a colored image. References [140] and [93], among others, have shown that including class labels to delineate between different image distributions can improve training and inference results. Appendix A.5.2 provides a theoretical derivation that shows this method of performing denoising score matching for grayscale images is equivalent to training diffusion models for grayscale images in the traditional way (with single-channel networks) under certain regularity assumptions. Appendix A.5.4 provides the pseudocode for the pretraining stage of SPAR.

### 8.2.3 Self-Supervised Model Finetuning

To refine a diffusion model for OOD testing samples at reconstruction time, we use the measurement-consistency constraints to update the model weights (in a similar manner as [47]) by

$$\boldsymbol{\theta} \leftarrow \arg\min_{\boldsymbol{\theta}} L(\boldsymbol{\theta}; \boldsymbol{x}_t), \quad L(\boldsymbol{\theta}; \boldsymbol{x}_t) \triangleq \Psi(D_\theta(\boldsymbol{x}_t|\boldsymbol{y})) \tag{8.4}$$

$$\Psi(\boldsymbol{x}) \triangleq \frac{1}{2}\|\boldsymbol{y} - \boldsymbol{A}\,\boldsymbol{x}\|_2^2. \tag{8.5}$$

We replace (8.5) with a quadratic majorizer of the form:

$$\phi(\boldsymbol{x}; \tilde{\boldsymbol{x}}) = \Psi(\tilde{\boldsymbol{x}}) + \langle \nabla\Psi(\tilde{\boldsymbol{x}}), \boldsymbol{x} - \tilde{\boldsymbol{x}} \rangle + \frac{\mathcal{L}}{2}\|\boldsymbol{x} - \tilde{\boldsymbol{x}}\|_2^2, \tag{8.6}$$

where $\mathcal{L}$ denotes a Lipschitz constant of $\nabla\Psi$ and $\tilde{\boldsymbol{x}}$ can be any point. Appendix 8.2.4 shows that $\phi(\boldsymbol{x};\tilde{\boldsymbol{x}}) \geq \Psi(\boldsymbol{x})$. When applying this majorizer with the self-supervised loss $L(\boldsymbol{\theta},\tilde{\boldsymbol{x}})$, the majorizer of the network refining loss is

$$\check{L}(\boldsymbol{\theta};\tilde{\boldsymbol{x}}) \triangleq \langle \boldsymbol{A}'(\boldsymbol{A}\tilde{\boldsymbol{x}} - \boldsymbol{y}), \boldsymbol{x}_{\boldsymbol{\theta}}\rangle + \frac{\mathcal{L}}{2}\|\boldsymbol{x}_{\boldsymbol{\theta}} - \tilde{\boldsymbol{x}}\|_2^2, \tag{8.7}$$

where $\boldsymbol{x}_{\boldsymbol{\theta}} = D_{\boldsymbol{\theta}}(\boldsymbol{x}_t|\boldsymbol{y})$, $\boldsymbol{A}'$ denotes the (Hermitian) transpose of $\boldsymbol{A}$, and we set $\tilde{\boldsymbol{x}}$ to be the denoised image from the previous diffusion iteration with a detached gradient. We drop constants that do not depend on $\boldsymbol{x}_{\boldsymbol{\theta}}$. In practice, we treat $\mathcal{L}$ as a tunable parameter.

The benefit of using this surrogate loss lies in the observation of [98] that diffusion models tend to have problems learning the score function at low levels of added noise. Section 8.2.4 shows that for low noise levels, (8.7) is less sensitive to inaccuracies in the learned score function than (8.4). However, for high noise levels, (7.4) is more accurate because the majorizer is an upper bound for the true loss function, with $\phi(\boldsymbol{x};\tilde{\boldsymbol{x}}) = \Psi(\boldsymbol{x})$ only when $\boldsymbol{x} = \tilde{\boldsymbol{x}}$. At high noise levels, $\tilde{\boldsymbol{x}}$ is likely to change significantly from one iteration to the next, because each diffusion iteration could substantially alter the image with significant added noise. Hence, the gap between the true loss (7.4) and the majorizer loss will be large. Motivated by such observations, we propose a novel noise-varying self-supervised loss function for test-time diffusion model adaptation, which is designed to use (7.4) for the higher noise levels while using (8.7) for lower noise levels. Section 8.2.4 provides a more rigorous theoretical analysis to justify the benefit of this new proposed self-supervised loss. In the implementation, we used a parameter search to find the optimal threshold $\sigma'$ in Algorithm 6; Appendix A.5.3 compares the results of using different values of $\sigma'$.

Finally, though the original network is trained on 3-channel images, we also wish to also perform reconstruction on 1-channel images (grayscale images) and 2-channel images (MRI images with the real and imaginary parts separated and concatenated along the channel dimension). Therefore, when using (8.2) to denoise these images, we need to first expand the image to 3 channels, e.g., by repeating one of the channels. We then process the resulting 3-channel output by using appropriate channel averaging to get a denoised 1- or 2-channel image. Mathematically, this is defined as follows:

- When $\boldsymbol{x}$ has 1 channel, we define the operator $\boldsymbol{H} : \mathbb{R}^{1\times M\times M} \to \mathbb{R}^{3\times M\times M}$ that concatenates $\boldsymbol{x}$ with itself along the channel dimension 3 times. In essence: $\boldsymbol{H} = \begin{bmatrix}1\\1\\1\end{bmatrix} \otimes \boldsymbol{I}_{M^2}$.

- When $\boldsymbol{x}$ has 2 channels, we define the operator $\boldsymbol{H} : \mathbb{R}^{2\times M\times M} \to \mathbb{R}^{3\times M\times M}$ that concatenates an additional copy of the first channel of $\boldsymbol{x}$ with itself before the second channel. In essence: $\boldsymbol{H} = \left[\begin{smallmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \end{smallmatrix}\right] \otimes \boldsymbol{I}_{M^2}$.
- Denoting by $\tilde{D}_{\boldsymbol{\theta}}$ the denoising network trained on 3-channel images, the denoiser of $\boldsymbol{x}$ is given by

$$D(\boldsymbol{x}) = \begin{cases} \frac{1}{3}\boldsymbol{H}'\tilde{D}_{\boldsymbol{\theta}}(\boldsymbol{H}\boldsymbol{x}), & \boldsymbol{x} \in \mathbb{R}^{1\times M\times M}, \\ \boldsymbol{H}'\tilde{D}_{\boldsymbol{\theta}}(\boldsymbol{H}\boldsymbol{x}) \oslash [2\ 1], & \boldsymbol{x} \in \mathbb{R}^{2\times M\times M}, \end{cases} \tag{8.8}$$

  where $\oslash$ denotes channel-wise division.

We can express (8.8) succinctly as $D(\boldsymbol{x}) = \boldsymbol{H}^\dagger \tilde{D}_{\boldsymbol{\theta}}(\boldsymbol{H}x)$, where $\boldsymbol{H}^\dagger$ denotes the pseudoinverse of the operator $\boldsymbol{H}$. In general, we have $\boldsymbol{H}^\dagger = (\boldsymbol{K} \otimes \boldsymbol{I})^\dagger = \boldsymbol{K}^\dagger \otimes \boldsymbol{I}^\dagger = \boldsymbol{K}^\dagger \otimes \boldsymbol{I}$. In particular:

$$\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}^\dagger = \frac{1}{3} \cdot [1\ 1\ 1], \quad \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}^\dagger = \begin{bmatrix} 0.5 & 0.5 & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{8.9}$$

The first case corresponds to taking an average across all 3 channels to obtain a single-channel image, which is consistent with the first part of (8.8). The second case corresponds to averaging the first 2 channels to obtain the first output channel and setting the third channel to be the second output channel, which is consistent with the second part of (8.8). Hence, we have shown that we convert a 3-channel denoiser $\tilde{D}_{\boldsymbol{\theta}}$ to a 1- or 2-channel denoiser using $D(\boldsymbol{x}) = \boldsymbol{H}^\dagger \tilde{D}_{\boldsymbol{\theta}}(\boldsymbol{H}\boldsymbol{x})$.

Algorithm 6 summarizes the entire proposed algorithm with self-supervised diffusion model refining at reconstruction time. Because [47] observed irregularities at the high noise levels when using similar self-supervised diffusion algorithms, we adopt a similar approach as [45] where we initialize the image using a simple baseline $\boldsymbol{A}^\dagger \boldsymbol{y}$ and choose a much lower value for $\sigma_T$ than would otherwise be optimal.

### 8.2.4 Theoretical Analysis of Majorizer Loss

This section shows that the majorizer loss is less sensitive to inaccuracies in the estimated score function for low noise levels. We return to the specific majorizer given by (8.6). Previous works analyzed theoretical properties of diffusion models assuming that the prior is a mixture of Gaussians [180, 113]. Each component of the mixture may represent one

**Algorithm 6** SPAR Image Reconstruction

**Input:** $\sigma_1 < \sigma_2 < \ldots < \sigma_T$, $\epsilon > 0$, $T$, $\boldsymbol{y}$, $K$, $\sigma'$
Initialize $\boldsymbol{x} \sim \mathcal{N}(\boldsymbol{A}^\dagger \boldsymbol{y}, \sigma_T^2 \boldsymbol{I})$
Initialize $\tilde{\boldsymbol{x}} \sim \mathcal{N}(0, \boldsymbol{I})$
**for** $t = T : 1$ **do**
  **if** $t \bmod K = 0$ **then**
    Compute $D_\theta(\boldsymbol{x}_t)$ using (8.2) with a random index $i$
    and apply (8.8) if $\boldsymbol{x}$ has fewer than 3 channels.
    Compute $D_\theta(\boldsymbol{x}_t|\boldsymbol{y})$ using $D_\theta(\boldsymbol{x}_t)$ and (2.7).
    **if** $\sigma_t > \sigma'$ **then**
      Define $L(\boldsymbol{\theta}) = \|\boldsymbol{y} - \boldsymbol{A} D_\theta(\boldsymbol{x}_t|\boldsymbol{y})\|_2^2$.
    **else**
      Define $L(\boldsymbol{\theta}) = \langle \boldsymbol{A}'(\boldsymbol{A}\tilde{\boldsymbol{x}} - \boldsymbol{y}), \boldsymbol{x}_{\boldsymbol{\theta}} \rangle + \frac{\mathcal{L}}{2} \|\boldsymbol{x}_{\boldsymbol{\theta}} - \tilde{\boldsymbol{x}}\|_2^2$,
      where $\boldsymbol{x}_{\boldsymbol{\theta}} = D_\theta(\boldsymbol{x}_t|\boldsymbol{y})$.
    **end if**
    Update $\boldsymbol{\theta}$ by backpropagating $L(\boldsymbol{\theta})$.
  **end if**
  Sample $\boldsymbol{z} \sim \mathcal{N}(0, \sigma_t^2 \boldsymbol{I})$.
  Set $\alpha_t = \epsilon \cdot \sigma_t^2$.
  Compute $D(\boldsymbol{x}_t)$ using (8.2) with a random index $i$
  and apply (8.8) if $\boldsymbol{x}$ has fewer than 3 channels.
  Solve (2.7) for $D(\boldsymbol{x}_t|\boldsymbol{y})$ and set it to $\tilde{\boldsymbol{x}}$.
  Set $\boldsymbol{s}_t = (D(\boldsymbol{x}_t|\boldsymbol{y}) - \boldsymbol{x}_t)/\sigma_t^2$.
  Set $\boldsymbol{x}_{t-1}$ to $\boldsymbol{x}_t + \frac{\alpha_t}{2} \boldsymbol{s}_t + \sqrt{\alpha_t} \boldsymbol{z}$.
**end for**

image class in the dataset. When solving inverse problems, the measurements lack fine details but provide global information about the ground truth image structure that can help identify the image class. Thus, the mixture of Gaussians approximately reduces to a single Gaussian from which the ground truth belongs with high probability. As such, the following analysis assumes that the clean image distribution $p(\boldsymbol{x})$ follows a Gaussian with mean $\mu$ and covariance $\boldsymbol{\Sigma}$. Furthermore, Algorithm 6 uses the majorizer loss only for small $\sigma_t$, so we assume $\sigma_t \ll 1$ for the remainder of the analysis. Finally, since the number of diffusion iterations is large, $\sigma_t$ does not change significantly between consecutive sampling steps.

We begin by analyzing the following problem setup: suppose $\boldsymbol{X} \sim \mathcal{N}(\mu, \boldsymbol{\Sigma})$ and $\boldsymbol{Y} = \boldsymbol{X} + \sigma \boldsymbol{z}$, where $\boldsymbol{z} \sim \mathcal{N}(0, \boldsymbol{I})$. Because $\boldsymbol{X}$ and $\boldsymbol{Y}$ are jointly Gaussian:

$$\begin{aligned}
\mathbb{E}[\boldsymbol{X}|\boldsymbol{Y}] &= \mu + \operatorname{Cov}(\boldsymbol{X}, \boldsymbol{Y}) \operatorname{Cov}(\boldsymbol{Y})^{-1}(\boldsymbol{Y} - \mu) \\
\operatorname{Cov}(\boldsymbol{X}, \boldsymbol{Y}) &= \operatorname{Cov}(\boldsymbol{X}, \boldsymbol{X} + \sigma \boldsymbol{z}) = \operatorname{Cov}(\boldsymbol{X}) = \boldsymbol{\Sigma} \\
\operatorname{Cov}(\boldsymbol{Y}) &= \operatorname{Cov}(\boldsymbol{X}) + \operatorname{Cov}(\sigma \boldsymbol{z}) = \boldsymbol{\Sigma} + \sigma^2 \boldsymbol{I} \\
\mathbb{E}[\boldsymbol{X}|\boldsymbol{Y}] &= \mu + \boldsymbol{\Sigma}(\boldsymbol{\Sigma} + \sigma^2 \boldsymbol{I})^{-1}(\boldsymbol{Y} - \mu).
\end{aligned}$$

Expanding out the inverse with a geometric series and assuming $\sigma$ is small, this simplifies to

$$\mathbb{E}[\boldsymbol{X}|\boldsymbol{Y}] \approx \boldsymbol{Y} - \sigma^2 \boldsymbol{\Sigma}^{-1}(\boldsymbol{Y} - \mu). \tag{8.10}$$

Returning to the problem at hand, assume the measurement is noiseless so that $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}_0$. For small $\sigma_t$, $\mathbb{E}[\boldsymbol{x}_0|\boldsymbol{x}_t, \boldsymbol{y}]$ is approximately a projection of $\mathbb{E}[\boldsymbol{x}_0|\boldsymbol{x}_t]$ onto the affine subspace defined by $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}_0$; past diffusion inverse solvers [41, 181] have made similar assumptions. Using $\boldsymbol{X} \triangleq \boldsymbol{x}_0$ and $\boldsymbol{Y} \triangleq \boldsymbol{x}_t = \boldsymbol{x}_0 + \sigma_t \boldsymbol{z}$ in (8.10) yields $\mathbb{E}[\boldsymbol{x}_0|\boldsymbol{x}_t] = \boldsymbol{x}_t + O(\sigma^2)$. Thus,

$$\mathbb{E}[\boldsymbol{x}_0|\boldsymbol{x}_t, \boldsymbol{y}] - \mathbb{E}[\boldsymbol{x}_0|\boldsymbol{x}_t] \approx \boldsymbol{P}\left(\boldsymbol{x}_t + O(\sigma_t^2) - \boldsymbol{x}_0\right), \tag{8.11}$$

where $\boldsymbol{P}$ is a linear operator that represents orthogonally projecting onto the row space of $\boldsymbol{A}$. Since $\boldsymbol{x}_t - \boldsymbol{x}_0 = \sigma_t \boldsymbol{z} = O(\sigma_t)$, we have $\mathbb{E}[\boldsymbol{x}_0|\boldsymbol{x}_t, \boldsymbol{y}] - \mathbb{E}[\boldsymbol{x}_0|\boldsymbol{x}_t] = O(\sigma_t)$.

Further assume that the trained denoiser satisfies $D_\theta(\boldsymbol{x}_t) = \mathbb{E}[\boldsymbol{x}_0|\boldsymbol{x}_t] + \boldsymbol{\Delta}$, where $\boldsymbol{\Delta}$ is a small error term. If this denoising neural network were simply the identity function, the error $\boldsymbol{\Delta}$ would be $\boldsymbol{x}_t - \boldsymbol{x}_0$, which is $O(\sigma_t)$; hence, a well trained denoiser should have $\boldsymbol{\Delta} < O(\sigma_t)$. In conclusion, the difference between $D_\theta(\boldsymbol{x}_t)$ and $\mathbb{E}[\boldsymbol{x}_0|\boldsymbol{x}_t, \boldsymbol{y}]$ is $O(\sigma_t)$.

Next we analyze the majorizer loss. From the setup above, we have $\boldsymbol{X} \triangleq \boldsymbol{x}_0$, $\boldsymbol{Y} \triangleq \boldsymbol{x}_0 + \sigma_t \boldsymbol{z}$, $\tilde{\boldsymbol{x}} = D_\theta(\boldsymbol{x}_{t+1}|\boldsymbol{y}) = \mathbb{E}[\boldsymbol{X}|\boldsymbol{Y}] + O(\sigma_t)$, using the conclusion from the previous paragraph in the final equality as well as assuming the noise scale is changing consecutively in small steps $\sigma_t \approx \sigma_{t+1}$. Note that $\boldsymbol{x}_t = \boldsymbol{x}_0 + \sigma_t \boldsymbol{z}$. Then $\tilde{\boldsymbol{x}} - \boldsymbol{x}_0 = \sigma_t \boldsymbol{z} - \sigma_t^2 \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}_0 + \sigma_t^2 \boldsymbol{z} - \mu) + O(\sigma_t)$. This shows that $\tilde{\boldsymbol{x}} - \boldsymbol{x}_0$ is approximately $O(\sigma_t)$, when $\sigma_t$ is small. Now we rewrite (8.7) as

$$L(\boldsymbol{\theta}) = (\tilde{\boldsymbol{x}} - \boldsymbol{x}_0)' \boldsymbol{A}' \boldsymbol{A} \boldsymbol{x} + \frac{L}{2} \|(\boldsymbol{x} - \boldsymbol{x}_0) + (\boldsymbol{x}_0 - \tilde{\boldsymbol{x}})\|_2^2. \tag{8.12}$$

Note that $\boldsymbol{x} = D_\theta(\boldsymbol{x}_t|\boldsymbol{y})$ (before any network refining) is the current prediction of $\boldsymbol{x}_0$ from a noisy version of $\boldsymbol{x}_0$. We assume the noise level $\sigma_t$ does not change significantly between consecutive sampling steps, so we can follow the same derivation above to get

that $\boldsymbol{x} - \boldsymbol{x}_0$ is $O(\sigma_t)$. This shows that the entire term inside of the squared Euclidean norm in (8.12) is $O(\sigma_t)$, so the term itself is $O(L\sigma_t^2)$. In contrast, the linear term in this loss is only $O(\|\boldsymbol{A}\|_2^2\sigma_t)$, where typically $L$ is chosen to be on the same order as $\|\boldsymbol{A}\|_2^2$. Hence, when $\sigma_t$ is small, the linear term dominates the loss.

Next we look at the gradients of the loss functions. For convenience, set $\boldsymbol{x}_{\boldsymbol{\theta}} = D_\theta(\boldsymbol{x}_t|\boldsymbol{y})$ so that (7.4) becomes $L(\boldsymbol{\theta}) = (1/2)\|\boldsymbol{A}\boldsymbol{x}_0 - \boldsymbol{A}\boldsymbol{x}_{\boldsymbol{\theta}}\|_2^2$. Hence, the backpropagation step through the network uses the gradients

$$\frac{\partial L}{\partial \boldsymbol{\theta}} = \frac{\partial L}{\partial \boldsymbol{x}}\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{\theta}} = \boldsymbol{A}'\boldsymbol{A}(\boldsymbol{x}_{\boldsymbol{\theta}} - \boldsymbol{x}_0)\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{\theta}}. \tag{8.13}$$

The gradient for the quadratic majorizer loss is

$$\frac{\partial L}{\partial \boldsymbol{\theta}} = (\boldsymbol{A}'\boldsymbol{A}(\tilde{\boldsymbol{x}} - \boldsymbol{x}_0) + L\,(\boldsymbol{x}_{\boldsymbol{\theta}} - \tilde{\boldsymbol{x}}))\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{\theta}}. \tag{8.14}$$

The inaccuracies in the approximate score function arise in $\boldsymbol{x}_{\boldsymbol{\theta}}$, and hence $\boldsymbol{x}_{\boldsymbol{\theta}} - \boldsymbol{x}_0$ in (8.13) and $\boldsymbol{x}_{\boldsymbol{\theta}} - \tilde{\boldsymbol{x}}$ in (8.14). Crucially, we showed earlier that the dominant term in (8.14) is $\boldsymbol{A}'\boldsymbol{A}(\tilde{\boldsymbol{x}} - \boldsymbol{x}_0)$. Therefore, while errors in the score function fully propagate themselves to the gradient when using the original loss (8.13), those errors have much less effect on the gradient of the majorizer (8.14). This analysis provides a theoretical justification why the majorizer loss is less sensitive to errors in the score function than the original loss when $\sigma$ is small, and guided the design of Alg. 6.

## 8.3 Experiments

The pretraining used a small set of CT images and human face images only. For the CT images, we used the AAPM 2016 CT challenge data from [135]. We preprocessed the data [82] to get a total of 2304 training slices of size $256 \times 256$. For the natural images, we used the CelebA-HQ dataset [126] with each image having size $256 \times 256$. We then randomly selected a small subset of 3000 CelebA images for pre-training. We trained the patch-based network with patch sizes from $16 \times 16$ to $64 \times 64$, and used a zero padding value of 64. The network architecture follows from [93] for both the patch-based networks and whole-image networks. All networks were trained on PyTorch using the Adam optimizer with two NVIDIA A40 GPUs.

TBL 8.1 – Comparison of quantitative results on four different medical imaging inverse problems. Results are averages across all images in the test dataset. Best results for practical use are in bold.

| Method | PBCT, 60 Views | | FBCT, 40 Views | | $512 \times 512$ CT | | CS-MRI, $7\times$ | |
|---|---|---|---|---|---|---|---|---|
| | PSNR↑ | SSIM↑ | PSNR↑ | SSIM↑ | PSNR↑ | SSIM↑ | PSNR↑ | SSIM↑ |
| Baseline | 30.15 | 0.784 | 17.86 | 0.381 | 28.33 | 0.700 | 33.94 | 0.894 |
| ADMM-TV | 31.14 | 0.862 | 24.20 | 0.628 | 29.36 | 0.788 | 36.74 | 0.924 |
| PnP-ADMM [196] | 36.75 | 0.932 | 28.86 | 0.747 | 37.48 | 0.910 | 35.77 | 0.907 |
| PaDIS+FC [82] | 39.16 | 0.942 | 27.91 | 0.796 | 33.11 | 0.831 | 35.17 | 0.904 |
| SCD [10] | 41.16 | 0.962 | 21.28 | 0.463 | – | – | – | – |
| Ours (SPAR) | **42.72** | **0.972** | **36.11** | **0.918** | **38.81** | **0.929** | **39.15** | **0.949** |
| Ideal* | 42.82 | 0.973 | 36.34 | 0.923 | 38.94 | 0.930 | 39.42 | 0.953 |

*not available in practice with a single diffusion model

TBL 8.2 – Comparison of quantitative results on different natural imaging inverse problems: deblurring and super-resolution. The second column indicates whether the method can handle images from different modalities. Best results for practical use are in bold.

| Method | Diff. Modality | Deblur, LSUN | | Deblur, FFHQ | | Superresolution | |
|---|---|---|---|---|---|---|---|
| | | PSNR↑ | SSIM↑ | PSNR↑ | SSIM↑ | PSNR↑ | SSIM↑ |
| Baseline | ✓ | 22.03 | 0.602 | 24.10 | 0.649 | 25.42 | 0.742 |
| ADMM-TV | ✓ | 27.01 | 0.801 | 28.44 | 0.772 | 25.22 | 0.729 |
| PnP-ADMM [196] | ✓ | 28.28 | 0.847 | 30.79 | 0.829 | 27.29 | 0.796 |
| DPIR [204] | × | 28.96 | 0.857 | – | – | 28.98 | 0.825 |
| DiffPIR [210] | × | 28.11 | 0.798 | – | – | 27.68 | 0.796 |
| DDRM [95] | × | 27.19 | 0.803 | – | – | 28.60 | 0.811 |
| PaDIS+FC [82] | ✓ | 27.17 | 0.809 | 28.91 | 0.774 | 27.23 | 0.782 |
| SCD [10] | × | 28.57 | 0.842 | – | – | 26.15 | 0.635 |
| Ours (SPAR) | ✓ | **29.49** | **0.877** | **31.43** | **0.843** | **29.10** | **0.842** |
| Ideal* | ✓ | 29.57 | 0.879 | 31.55 | 0.848 | 29.26 | 0.848 |

*not available in practice with a single diffusion model.

To evaluate our proposed SPAR method, we ran experiments on a wide variety of datasets with many different imaging applications, particularly on images that are out

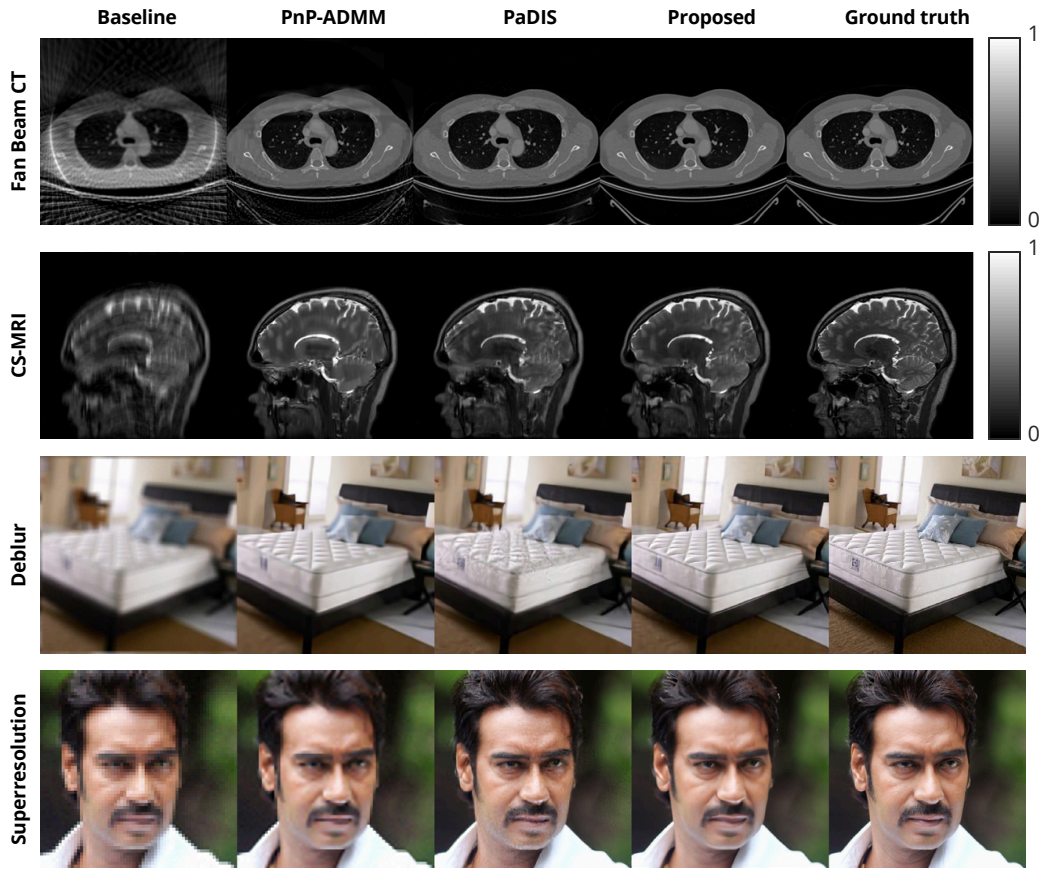



**FIG** 8.3 – Results of our proposed method and comparison methods for various inverse problems. MRI images are complex-valued so we show the pixel magnitudes.

of the training data distribution. We benchmarked the algorithm performance on: LIDC-IDRI dataset [6], fastMRI brain dataset from [100], the LSUN dataset [200], the FFHQ-512 dataset [94], in addition to the AAPM and CelebA datasets that we used for pre-training. For the LIDC-IDRI dataset, we first applied data preprocessing by setting the entire background of the volumes to zero. We rescaled the images in the XY-plane to have size $256 \times 256$. For the fastMRI brain dataset, we took the complex-valued image space data and rescaled it so that the real parts had a maximum value of 1, and applied the same scale factor to the imaginary parts. We also resized the images and coil maps (provided in the dataset) to $256 \times 256$. We also ran $512 \times 512$ CT experiments on the AAPM dataset by taking the original data from [135] without downsampling. For all experiments, we set aside 10 images in each dataset to use as test images and reported quantitative metrics that were averaged across these test images.

For CT imaging, we performed sparse-view reconstruction with both a parallel-beam projector (PBCT) and a fan-beam projector (FBCT), using 60 projection views in the former and 40 views in the latter. In both cases, we used a detector with size 512 pixels, with the forward and backward projectors implemented by the ODL package [173]. We also performed PBCT reconstruction experiments on the $512 \times 512$ sized CT images with 60 projection views. The PBCT experiments were performed with the AAPM dataset and the FBCT experiments were performed with the LIDC dataset.

For the MRI experiments, we performed compressed sensing MRI (CS-MRI) reconstruction. The sampling mask was chosen in a Cartesian pattern where each horizontal line within the center $10\%$ of the image width was sampled, and all other horizontal lines were sampled with probability 0.03. Hence, the acceleration factor was $6.9\times$. We used 12 coils with the implementation for the forward and back projectors being the same as those in [75].

For the natural images, we ran deblurring and superresolution experiments. For the $256 \times 256$ images, we used a uniform blur kernel of size $9 \times 9$ and added white Gaussian noise with $\sigma = 0.01$ where the clean image was scaled between 0 and 1. We used the same settings but with a blur kernel of size $17 \times 17$ for the $512 \times 512$ images. For the superresolution experiments (only done on $256 \times 256$ images), we used a scaling factor of 4 with downsampling by averaging and added white Gaussian noise with $\sigma = 0.01$. Appendix A.5.5 contains additional experiments with noise level $\sigma = 0.05$.

For comparison, we applied the same patch-based diffusion network trained on the mixed dataset to solve the inverse problems, but without self-supervision. Hence, the checkpoint weights were held constant throughout the reconstruction process, similar to PaDIS [82]. We denote this method as PaDIS+FC (flexible channel) to clarify that in addition to [82], we applied (8.8) to handle images of different modalities. Previous work [81] showed that using conjugate gradient descent to enforce data fidelity yielded better results than DIS methods such as DDNM [181] and DPS [39]. Thus, we ran experiments using steerable controlled diffusion (SCD) [10] which also uses conjugate gradient descent; since this method does not support using a single network to reconstruct images of different modalities, we separately trained networks for it on the AAPM and CelebA datasets. For a more thorough comparison, we also compared with other SOTA methods for natural imaging: DPIR [204], DiffPIR [210], and DDRM [95]. Although these methods usually use a diffusion model trained for each dataset, for fair comparisons with SPAR we trained the networks on the CelebA dataset only.

We also compared our method with traditional image reconstruction methods: applying a non-iterative baseline, reconstructing using the total variation (TV) regularizer

TBL 8.3 – Comparison between diffusion inverse solvers using the same network for CT reconstruction and image deblurring.

| Solver | FBCT (40 Views) | | Deblur (LSUN) | |
|---|---|---|---|---|
| | PSNR↑ | SSIM↑ | PSNR↑ | SSIM↑ |
| Langevin [166] | 26.11 | 0.798 | 26.67 | 0.790 |
| VE-DDNM [181] | 18.05 | 0.424 | 26.40 | 0.777 |
| VE-DPS [39] | 23.03 | 0.714 | 28.36 | 0.847 |
| Proposed (SPAR) | **36.11** | **0.918** | **29.49** | **0.877** |

and solving the optimization problem using ADMM (ADMM-TV), and plug and play via ADMM (PnP-ADMM) [196]. The baseline methods for each of the inverse problems is as follows: applying the filtered back-projection (FBP) method to the measurement $\boldsymbol{y}$ for CT, taking $\boldsymbol{A}'\boldsymbol{y}$ for CS-MRI, taking the blurred image $\boldsymbol{y}$ for deblurring, and upsampling the low resolution image $\boldsymbol{y}$ using bilinear interpolation for superresolution. The implementation of ADMM-TV is in [74]. Finally, since our proposed method does not require a clean dataset for each type of image, we used the off-the-shelf denoiser BM3D [49] for PnP-ADMM. Appendix A.5.6 reports the values of all the parameters of the algorithms.

Tables 8.1 and 8.2 show the quantitative results of the medical and natural image experiments, respectively. In Table 8.1, the second column indicates whether or not the method directly supports using a single network to handle images of different modalities. For reference, we also show the results of the ideal case, where we trained a patch-based diffusion model on a large clean dataset of the same distribution as the test images and still applied the majorizer loss during reconstruction. Our proposed method outperformed the state-of-the-art benchmarks that do have access to a large training dataset in all the inverse problems across different modalities. Notably, our SPAR method performed well in CS-MRI reconstruction even though the diffusion network was not trained on any complex-valued or 2-channel images. Figure 8.3 shows the visual results of the experiments. Appendix A.5.5 reports LPIPS scores and runtimes for the comparison methods.

We further investigated the benefit of using the majorizer loss function (8.7). We ran a "simplified" version of Algorithm 6 with $\sigma' = 0$ (i.e., always using the loss in (7.4)) and evaluated the PSNR of the intermediate image throughout the CS-MRI reconstruction process. Figure 8.4 shows the results of this experiment compared with using the proposed Algorithm 6 that uses the majorizer loss (8.7). Both reconstruction algorithm versions consist of running the diffusion process in reverse, so the image quality improves

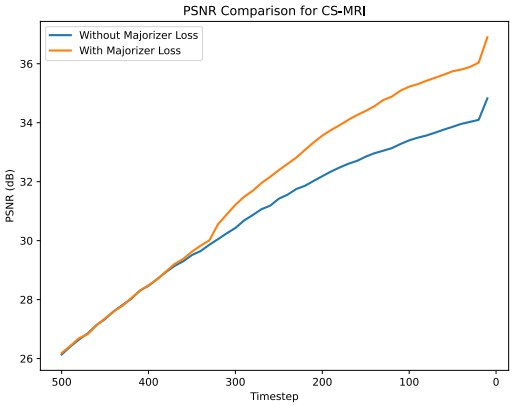



**FIG** 8.4 – PSNR over time for different loss types in CS-MRI.

as the timestep decreases. Although the diffusion process has 1000 steps, we truncated the graph at $t = 500$ to better illustrate the divergence in PSNR between the two different self-supervised loss functions. When using the simpler loss (7.4), due to inaccuracies in the learned score function, when the timestep (and hence the noise level of the image) becomes small, the improvement in image quality is lower compared to using the proposed majorizer loss (8.7). This experimentally reinforces the theoretical findings shown in Section 8.2.4 that the majorizer loss is less sensitive to inaccuracies in the learned score function.

We also investigated various other diffusion inverse solvers that leverage an unconditional prior. These methods assume that a diffusion model has been trained properly on a large sample of training images that belong to the same distribution as the test data. Therefore, subpar performance is expected when the test images belong to a different distribution from the training data. Since we trained our patch-based diffusion model on a mixture of AAPM CT images and CelebA images, we performed experiments on LIDC CT images and LSUN images to illustrate this point. In particular, we ran fan-beam CT with 40 projection views using the LIDC dataset and deblurring on the LSUN dataset.

Table 8.3 shows the results of these experiments. We compared Langevin dynamics with gradient descent for data fidelity [166], variance exploding DDNM [181] (as opposed to variance preserving, which would have required a differently trained network), and variance exploding DPS [39]. These results show that since the other inverse solvers do not account for a distribution shift, the results are substantially worse than our proposed method. Appendix A.5.3 contains more details as well as visualizations of these experiments. Finally, Appendix A.5.5 contains a discussion with some results about applying acceleration methods in conjunction with our method.

## 8.4 Conclusion

This chapter proposes SPAR (Single Pretrain Always Refine), a method that significantly reduces the pre-training overhead by leveraging adaptive self-supervision to refine the network at reconstruction time in many modalities. This flexibility allows SPAR to tackle inverse problems even in modalities absent from the initial training dataset. Extensive experiments demonstrated that using a single pretrained network, our SPAR method achieved consistently better image quality than existing methods. Limitations of this work include a lack of demonstration on nonlinear inverse problems as well as 3D and video inverse problems. The memory efficiency of patch-based modeling should facilitate such extensions. Additionally, for inverse problems such as impainting with a large missing area, the proposed method may not be effective, as the pretrained image prior does not have any information from the measurement to guide the reconstruction in the missing area. This is in contrast to impainting with random missing pixels, for which the proposed method should be more effective. Image priors such as the ones used in this paper have the potential to benefit society by reducing the scan time of MRI scans and the dose of CT scans.

# CHAPTER 9

# Future Work

Diffusion models have shown outstanding performance for various inverse problems spanning several modalities, but are limited by their computational and data requirements. This thesis covered several works that improve the flexibility of diffusion models and allow them to overcome these shortcomings. This chapter outlines some future directions and potential projects that build off this thesis. These projects advance our overall goal of developing diffusion models that can be used to solve large-scale image reconstruction problems, possibly with limited training data.

## 9.1 Improving Existing Works

### 9.1.1 Generalizing Patch-Based Diffusion Models

Chapter 4 described a general framework for patch-based diffusion models via (4.4) and proved that under certain conditions, a stochastic version of Langevin dynamics can be applied to sample from the prior. We also identified some previous works, as well as our own works which use prior models that fall under this framework. However, a few questions remain to be answered which could be the subject of future research.

- It was shown that regardless of the method by which patches are used to tile the whole image, provided the size of the patch is unchanging, the same method of training the network via performing denoising score matching on individual patches can be applied. However, different patch tiling methods do affect reconstruction results even when the same network is used, as shown in Chapter 4. A natural question then arises: which patch tiling methods are optimal and how does the application as well as diffusion inverse solver affect the results?

- The stochastic sampling theorem applies only for Langevin dynamics as the number of sampling steps tends to infinity. Are there analogous results for other sampling methods such as DDPM, DDIM, or other acceleration methods?
- Even when global information is utilized as in Chapter 6, for complex datasets such as natural imaging datasets, unconditional image generation can often lead to unrealistic results. This suggests the existence of a gap between the true whole-image prior and the patch-based image prior. How can the patch-based prior be improved to better reflect real-world image priors?
- More generally, one can not only learn the unconditional priors of patches but a prior of patches conditioned on other randomly chosen patches. In this case, the training process would consist of choosing a patch at random to denoise, while conditioning on some number of other randomly selected patches. This process could also help the model learn global context, reduce overfitting, and improve robustness.

### 9.1.2 Test-time Adaptation for 3D Problems

Chapters 7 and 8 tackled the problem of refining a diffusion network according to a measurement at reconstruction time for 2D images. Chung et al. [47] applied a similar method as Chapter 7 to perform test-time adaptation for 3D CT reconstruction. Since parallel-beam CT was used, different z-slices of the volume do not interfere with each other, and it is possible to choose slices of the volume (instead of the entire volume) and their corresponding CT projections to perform test-time adaptation. However, for more complicated forward models, such as cone-beam CT or spatiotemporal deblurring in dynamic imaging, it would be necessary to perform network backpropagation with the entire volume, which is computationally infeasible using either the loss of [47] or Chapter 8. Future work could consist of developing an alternative way to refine the network during reconstruction time which does not require backpropagating through the entire forward model and volume.

Separately, a natural question that arises during test-time adaptation is the characteristics of the network after reconstruction and hence finetuning. In the 2D setting, we have observed that for CT reconstruction, this finetuned network does not have the ability to unconditionally generate meaningful images. It may not be reasonable to expect the network to generate a fully diverse set of images from the measurement of only a single in-distribution image; nevertheless, [164] shows that under deterministic sampling via DDIM, small perturbations to the initial noise result in small perturbations in the generated image which still result in a realistic image. Hence, it may be possible to obtain a refined network that can regenerate the clean image and local variants of it.

### 9.1.3 Diffusion Bridge Network Refining

Diffusion bridges are a generalization of the traditional diffusion model framework. While the traditional diffusion model framework involves learning a neural network that can transform an image from the pure white Gaussian noise distribution to some target data distribution [73, 166], diffusion bridges use a network to transform an image between two arbitrary distributions. We use $p_0$ to denote the distribution of images degraded in some fashion (i.e., blurry images, low resolution images, etc.) and $p_1$ to denote the clean image distribution [120, 40]. Hence, the forward process consists of adding noise in a controlled fashion to go from the clean image to the degraded image, while the reverse process consists of recovering the clean image from the degraded image.

When using diffusion bridges to solve inverse problems, a separate network must be trained for each inverse problem. For example, for deblurring, the degraded image distribution consists of blurry images, and a network trained for this cannot then be applied to perform image superresolution. This is a disadvantage of diffusion bridge methods compared to traditional diffusion model approaches, where the unconditionally trained image prior can be used to solve any inverse problem on that image dataset. The advantage of diffusion bridge methods is that since they start from a distribution that is closer to the clean image than pure noise, the number of NFEs (neural function evaluations) needed at reconstruction time is much lower than traditional diffusion models. In practice, competitive performance has been observed using only around 10 NFEs with diffusion bridges [120] compared to 100-1000 for traditional diffusion models. Furthermore, the number of NFEs used can be freely changed at reconstruction time, and leads to an observable tradeoff between FID and PSNR that pushes the Pareto boundary [40].

Future work could consist of one possible solution that achieves the best of both worlds: solving many different inverse problems using one pretrained neural network in a small number of sampling steps. We achieve this by starting with a diffusion bridge network that is trained specifically for some inverse problem. Next, in a similar way as Chapter 7, if the inverse problem or the dataset differs from that of the pretrained network, we include a data consistency loss function to adapt the network to the different dataset and inverse problem. Table 9.1 summarizes properties of different DIS methods. Notably, this method would only require training a network for each modality. If we were to combine the ideas of patch-based diffusion models and channel operators from SPAR, it could even be possible to use a single neural network in conjunction with a diffusion bridge. Furthermore, the number of required diffusion iterations would be far smaller than previous unconditional DIS methods, significantly improving the reconstruction speed.

| | Training | Patches? | Sampling speed |
|---|---|---|---|
| DPS/DDNM/standard DIS | For each dataset | Optional | Slow (2 min) |
| SPAR | Once | Yes | Very slow (3 min) |
| Diffusion bridge: I2SB, CDDB | For each inverse problem | No | Fast (5 sec) |
| Bridge + refine | For each modality | Optional | Somewhat fast? |

TBL 9.1 – Comparison of different diffusion-based methods

### 9.1.4 Provable Posterior Sampling

There are various diffusion inverse solvers that leverage an unconditionally trained prior model; Chapter 4 uses DPS [39] and Chapters 5 and 6 use the conjugate gradient descent method of [41]. However, all of these methods only *approximately* sample from the posterior $p(\boldsymbol{y}|\boldsymbol{x})$ and in particular, approximate the intractable $\nabla \log p(\boldsymbol{y}|\boldsymbol{x}_t)$ during the reconstruction process. Methods that can perform exact posterior sampling have various limitations: Monte Carlo methods [32, 36] are only exact when the number of particles used tends to infinity, and are thus computationally expensive. Diffusion bridges [120, 40] provably sample from the posterior in fewer steps than traditional diffusion models, but require training a network specific to each inverse problem, limiting their flexibility. Finally, variational methods [60, 8] directly train a second network (in addition to the unconditional prior network) to approximate the posterior distribution and use this second network to quickly sample from the posterior, but the additional training step is costly. Future work could involve developing a method that still only uses a single unconditional diffusion model, yet can provably sample from the posterior reasonably quickly.

## 9.2 Leveraging Video Diffusion Models

Video diffusion models have recently risen to popularity, showing the ability to generate high quality videos [129, 124]. By treating the temporal dimension of videos as the third spatial dimension for 3D imaging problems, one can hope to use pretrained video diffusion models (possibly with finetuning) to solve 3D inverse problems. However, there are two primary difficulties with this approach. Firstly, video diffusion models generally use a network that consists of two modules: a spatial module that batchifies the temporal dimension and processes the spatial dimensions using a typical diffusion UNet or transformer, and a temporal module that batchifies the two spatial dimensions but applies an

attention layer in the temporal dimension [134, 202]. As a result, the network must process the entire video at once, which constrains the number of frames in the videos, usually to fewer than 40 even with modern GPUs. This is prohibitively limiting for CT and MRI reconstruction, where the third spatial dimension can often have size 256 or greater. Secondly, video generation papers frequently do not have a simple black box network whose input is solely the noisy video and whose output is the denoised video or its score function [160, 19], instead relying on a conditioning mechanism on lower resolution videos or previous frames to guide the generation process. This makes it less straightforward to leverage these models in conjunction with diffusion inverse solvers.

Existing works on video inverse problems often instead leverage a 2D diffusion model, but add similar noise across different frames during the reconstruction process so as to preserve temporal continuity [102, 103]. These methods generally outperform 3D image reconstruction methods, as the pretrained 2D diffusion models are expressive and powerful, capturing a diverse range of modes in natural videos. However, these methods still do not learn a data driven prior that captures the relationship between different frames. Bai et al. [8] uses a pretrained video prior and trains a network approximating the posterior distribution for a specific inverse problem using a variational approach. Furthermore, the frames are processed in batches with a shifting window, which allows the network to learn a prior across multiple frames while also bypassing the memory requirements of processing the entire video at once. Although this method is able to perform reconstruction of videos of 30 frames in only a second, it requires training separate posterior networks for each inverse problem. Future work would consist of developing a method that can achieve the following objectives:

- Leverage a pretrained video diffusion model, such as stable video diffusion [19], with only a relatively cheap network finetuning step for each dataset
- Learn correlations between different slices in a data-driven way
- Process volumes of varying numbers of slices in the z direction
- Have a direct black box representation of the score of the volume which can then be applied in conjunction with a wide range of diffusion inverse solvers