

LEARNING TO GENERATE DATA BY ESTIMATING GRADIENTS OF THE
DATA DISTRIBUTION

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Yang Song
August 2022

© 2022 by Yang Song. All Rights Reserved.
Re-distributed by Stanford University under license with the author.



This work is licensed under a Creative Commons Attribution-
Noncommercial 3.0 United States License.
<http://creativecommons.org/licenses/by-nc/3.0/us/>

This dissertation is online at: <https://purl.stanford.edu/zy983tp3399>

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Stefano Ermon, Primary Adviser

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Tatsunori Hashimoto

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Tengyu Ma

Approved for the Stanford University Committee on Graduate Studies.

Stacey F. Bent, Vice Provost for Graduate Education

This signature page was generated electronically upon submission of this dissertation in electronic format.

Abstract

Generating realistic data with complex patterns, such as images, audio, or molecular structures, often relies on expressive probabilistic models to represent and estimate high-dimensional data distributions. However, even with the power of deep neural networks, building powerful probabilistic models is non-trivial. One major challenge is the need to normalize probability distributions; that is, to ensure the total probability equals one. This necessitates summing over all possible model outputs, which quickly becomes impractical in high-dimensional spaces. In this dissertation, I propose to address this difficulty by working with data distributions through their score functions. These functions, defined as gradients of log data densities, capture information about the corresponding data distributions without requiring normalization, hence can be modeled with highly flexible deep neural networks.

This dissertation is organized into three parts. In Part I, I show how to estimate the score function from a finite dataset with expressive deep neural networks and efficient statistical methods. In Part II, I discuss several ways to generate new data samples from models of score functions, building upon ideas from homotopy methods, Markov chain Monte Carlo, diffusion processes, and differential equations. The resulting *score-based generative models* (also known as *diffusion models*) achieved record-breaking generation performance for numerous data modalities, challenging the long-standing dominance of generative adversarial networks on many tasks. Importantly, the sampling procedure of score-based generative models can be flexibly controlled for solving inverse problems, demonstrated by their superior performance on multiple tasks in medical image reconstruction. In Part III, I show how to evaluate probability values accurately with models of score functions. Taken together, score-based generative models provide a flexible, powerful and versatile solution for data generation in machine learning and many other disciplines of science and engineering.

Acknowledgments

I am incredibly grateful to my advisor, Stefano Ermon, who provided constant support throughout my six-year PhD journey. His guidance was invaluable in every aspect of my academic career, from identifying research directions and crafting rigorous papers to securing funding, delivering clear presentations, teaching, and mentoring. I appreciate Stefano's encouragement to pursue my research interests, even when they seemed risky, as well as his flexibility and willingness to help at any time. He consistently prioritized my career success, and his advice was crucial during my job search, particularly when it came to my job talk, research statement, and interviews. Beyond being an excellent scientist, Stefano is an exceptional person whom I am fortunate to have as a mentor. From him, I've learned about the importance of optimism and perseverance, as well as how to become an independent researcher. I hope to one day be just half as good an advisor as Stefano is—something that I know will take decades of learning and practice.

My life at Stanford has been greatly enriched by colleagues in Ermon group. I was lucky to have Aditya Grover sit next to me as a role model; his unmatched work ethics, calm demeanor, and witty comments were always a source of inspiration for me. Neal Jean was like an elder brother who was never reluctant to share his wisdom and life experiences. Kristy Choi was a reliable collaborator and a patient teacher for my written English. Rui Shu never failed to bring up new ideas and fresh perspectives in our chats. Jonathan Kuck was my roommate at several overseas conferences where we had a blast exploring foreign cities together. I enjoyed casual afternoon chats with Jiaming Song and I will never forget our joint experiences on sailing, golfing, and video-gaming. I have learned a lot from Shengjia Zhao about developing research ideas, pushing on research projects, navigating the job market and even investing in stocks and options. Daniel Levy left the group early on but continued to be a close friend; we had a lot of fun taking challenging classes and playing Yo-Yos together. Chenlin Meng, Lantao Yu, and Kuno Kim were excellent research collaborators epitomizing

diligence, ingenuity and creativity. I will dearly miss the fantastic atmosphere in Ermon group, which would not have existed without many other (former) group members: Chris Cundy, Tri Dao, Volodymyr Kuleshov, Gengchen Mai, Charlie Marx, Willie Neiswanger, Andy Shih, and Yusuke Tashiro.

I am deeply grateful for having had the chance to work with so many great minds over the years. I would not have become who I am today if not for the help of my undergraduate research mentors Jun Zhu, Raquel Urtasun, Richard Zemel, and Alexander Schwing, who guided me into the exciting field of machine learning, supported me in pursuing a Ph.D. overseas, and advised me on life and research before and throughout my time at Stanford. I was fortunate to have broadened my horizons in summer internships at Microsoft, Uber, and Google, under the excellent mentorship of Nate Kushman, Sebastian Nowozin, Raquel Urtasun, Ben Poole, Durk Kingma, Jascha Sohl-Dickstein, and Abhishek Kumar. These fun experiences have inspired and shaped my research directions significantly. I am grateful to have had the chance to work with Percy Liang and Surya Ganguli during rotations at Stanford. I am also much obliged to Tatsunori Hashimoto, Percy Liang, Tengyu Ma, and Lei Xing for holding my oral exam and offering advice on my career plans. It is always a blessing to spend time with junior students and watch them grow into experienced researchers, for which I thank Sahaj Garg, Wenzhe Li, Chenlin Meng, Chenhao Niu, Winnie Xu, and Yilun Xu for putting their trust in me during their internships in Ermon group. In addition to people already mentioned, I also had the privilege of collaborating with Conor Durkan, Kelly He, Iain Murray, Liyue Shen, Jiaxin Shi, Jiajun Wu, and Jun-Yan Zhu, without whom this dissertation would not have been complete.

While on the job market, I have received generous help from David Carlson, David Duvenaud, Will Grathwohl, Renjie Liao, Eero Simoncelli, Arash Vahdat, and Shenlong Wang, who informed me of job openings and supported my job applications. I would like to express my special thanks to Alex Dimakis, Durk Kingma, and Lei Xing for endorsing my applications with their reference letters. My job talk would not have been the same without the honest and insightful feedback from Aditya Grover, Tatsunori Hashimoto, Daniel Levy, Renjie Liao, Tengyu Ma, Jiaxin Shi, Shenlong Wang, Zhaoyou Wang, Keith Winstein, and Jun-Yan Zhu. It was extremely lucky for me to have my dissertation research supported by the National Science Foundation, Toyota Research Institute, Google TPU Research Cloud, a J.P. Morgan Ph.D. Fellowship, and an Apple Ph.D. Fellowship in AI/ML. I am also very thankful to Oncel Tuzel, my fellowship mentor at Apple, for providing generous advice in

our regular check-ins.

I am incredibly fortunate to have had the company of many friends both in and outside of Stanford during my time in sunny California. My long-time friend, Zhaoyou Wang, has been an amazing support, always there to lift me up in low points and celebrate with me in high points. Together we've experienced so much, from road trips and hiking to skiing and even my wedding ceremony. Friends like Pengyu Li, Weichen Wang, Yuguang Wu, and Jie Zhang have provided great conversation, while Yidan Chen and Tong Yang have joined me on many memorable road trips and hikes.

Finally, I would like to thank my family, to whom this dissertation is dedicated. My wife, Zhi Bie, has been a constant source of love and support, even during my most difficult moments. Our dog, Duoduo, has kept me company through many late nights of paper writing, and my parents, Qinggang Song and Xia Meng, have shown me unconditional love and unwavering support. Words cannot fully express how thankful I am to you all.

To my family.

Contents

Abstract	iv
Acknowledgments	v
1 Introduction	1
1.1 Approach: Score-Based Generative Modeling	2
1.1.1 Learning Score Models from Data	3
1.1.2 Generating Data Samples with Score Models	4
1.1.3 Evaluating Probability Values with Score Models	7
1.2 Dissertation Structure	8
2 Background	10
2.1 Statistical Foundations of Generative Models	10
2.1.1 Problem Settings	10
2.1.2 Model Training	11
2.1.3 The Challenge of Normalization	12
2.2 Existing Methods	13
2.2.1 Approximating the Normalizing Constant	13
2.2.2 Restricting Model Architectures	15
2.2.3 Modeling Only the Data Generating Process	18
2.3 Score Functions and Score Models	19
I Learning Score Models from Data	22
3 Learning Score Models with Random Projections	23

3.1	Introduction	24
3.2	Background	25
3.2.1	Score Matching	25
3.2.2	Score Estimation for Implicit Distributions	27
3.3	Density & Score Estimation with Sliced Score Matching	28
3.3.1	Sliced Score Matching	28
3.3.2	Sliced Score Estimation	30
3.4	Theoretical Analysis	30
3.4.1	Consistency	30
3.4.2	Asymptotic Normality	31
3.4.3	Connection with Other Methods	32
3.5	Related Work	33
3.5.1	Scalable Score Matching	33
3.5.2	Kernel Score Estimators	35
3.6	Experiments	35
3.6.1	Density Estimation	35
3.6.2	Score Estimation	38
3.7	Conclusion	42
4	Learning High Order Score Models with Denoising	43
4.1	Introduction	44
4.2	Background	45
4.2.1	Scores of a Distribution	45
4.2.2	Denoising Score Matching	46
4.2.3	Tweedie’s Formula	46
4.3	Estimating Higher Order Scores by Denoising	47
4.3.1	DSM in the View of Tweedie’s Formula	47
4.3.2	Second Order Denoising Score Matching	48
4.3.3	High Order Denoising Score Matching	49
4.4	Learning Second Order Score Models	50
4.4.1	Instantiating Objectives for Second Order Score Models	50
4.4.2	Parameterizing Second Order Score Models	51
4.4.3	Antithetic Sampling for Variance Reduction	51

4.4.4	The Accuracy and Efficiency of Learning Second Order Scores	52
4.5	Uncertainty Quantification with Second Order Score Models	53
4.6	Sampling with Second Order Score Models	55
4.6.1	Background on the Sampling Methods	56
4.6.2	Synthetic Datasets	57
4.6.3	Image Datasets	58
4.7	Conclusion	59
II	Generating Data Samples with Score Models	60
5	Data Generation via Annealed Langevin Dynamics	61
5.1	Introduction	61
5.2	Score-Based Generative Modeling	63
5.2.1	Score Matching for Score Estimation	64
5.2.2	Sampling with Langevin Dynamics	65
5.3	Challenges of Score-Based Generative Modeling	65
5.3.1	The Manifold Hypothesis	66
5.3.2	Low Data Density Regions	66
5.4	Noise Conditional Score Networks: Learning and Inference	69
5.4.1	Noise Conditional Score Networks	69
5.4.2	Learning NCSNs via Score Matching	70
5.4.3	NCSN Inference via Annealed Langevin Dynamics	71
5.5	Experiments	72
5.6	Related Work	76
5.7	Conclusion	77
6	Improved Techniques for Data Generation	78
6.1	Introduction	78
6.2	Background	79
6.2.1	Langevin Dynamics	79
6.2.2	Score-Based Generative Modeling	80
6.3	Choosing Noise Scales	82
6.3.1	The Initial Noise Scale	83

6.3.2	Other Noise Scales	85
6.3.3	Incorporating the Noise Information	86
6.4	Configuring Annealed Langevin Dynamics	87
6.5	Improving Stability with Moving Average	88
6.6	Combining All Techniques Together	89
6.7	Conclusion	92
7	Data Generation with Differential Equations	93
7.1	Introduction	94
7.2	Background	96
7.2.1	Denoising Score Matching with Langevin Dynamics (SMLD)	96
7.2.2	Denoising Diffusion Probabilistic Models (DDPM)	97
7.3	Score-Based Generative Modeling with SDEs	97
7.3.1	Perturbing Data with SDEs	98
7.3.2	Generating Samples by Reversing the SDE	99
7.3.3	Estimating Scores for the SDE	99
7.3.4	Examples: VE, VP SDEs and Beyond	100
7.4	Solving the Reverse SDE	101
7.4.1	General-Purpose Numerical SDE Solvers	101
7.4.2	Predictor-Corrector Samplers	102
7.4.3	Probability Flow and Connection to Neural ODEs	103
7.4.4	Architecture Improvements	105
7.5	Controllable Generation	106
7.6	Conclusion	107
8	Medical Image Reconstruction with Score Models	108
8.1	Introduction	109
8.2	Background	110
8.2.1	Linear Inverse Problems	110
8.2.2	Score-Based Generative Models	111
8.3	Solving Inverse Problems with Score-Based Generative Models	114
8.3.1	A Convenient Form of the Linear Measurement Process	115
8.3.2	Incorporating Observations into the Sampling Process	116
8.4	Experiments	119

8.5 Conclusion	124
III Evaluating Probability Values with Score Models	125
9 Computing, Bounding and Optimizing Likelihoods	126
9.1 Introduction	127
9.2 Score-Based Diffusion Models	128
9.2.1 Diffusing Data to Noise with an SDE	128
9.2.2 Generating Samples with the Reverse SDE	129
9.3 Likelihood of Score-Based Diffusion Models	131
9.4 Bounding the Likelihood of Score-Based Diffusion Models	132
9.4.1 Bounding the KL Divergence with Likelihood Weighting	133
9.4.2 Bounding the Log-Likelihood on Individual data points	135
9.4.3 Numerical Stability	137
9.4.4 Related Work	137
9.5 Improving the Likelihood of Score-Based Diffusion Models	138
9.5.1 Variance Reduction via Importance Sampling	138
9.5.2 Variational Dequantization	140
9.5.3 Experiments	140
9.6 Conclusion	143
10 Conclusion	144
A Additional Proofs	147
A.1 Proofs for Chapter 3	147
A.1.1 Notations	147
A.1.2 Basic Properties	147
A.1.3 Consistency	150
A.1.4 Asymptotic Normality	156
A.1.5 Noise Contrastive Estimation	163
A.2 Proofs for Chapter 4	164
A.3 Proofs for Chapter 6	169
A.4 Proofs for Chapter 8	172
A.5 Proofs for Chapter 9	174

B Additional Details and Results	184
B.1 Chapter 3	184
B.1.1 Samples	184
B.1.2 Additional Details of Experiments	184
B.1.3 Pesudocode	195
B.2 Chapter 4	195
B.2.1 Analysis on Second Order Score Models	195
B.2.2 Uncertainty Quantification	198
B.2.3 Ozaki Sampling	200
B.3 Chapter 5	202
B.3.1 Architectures	202
B.3.2 Additional Experimental Details	204
B.3.3 Samples	206
B.4 Chapter 6	215
B.4.1 Experimental Details	215
B.4.2 Additional Experimental Results	220
B.5 Chapter 7	239
B.5.1 The Framework for More General SDEs	239
B.5.2 VE, VP and sub-VP SDEs	240
B.5.3 SDEs in the Wild	243
B.5.4 Probability Flow ODE	245
B.5.5 Reverse Diffusion Sampling	251
B.5.6 Ancestral Sampling for SMLD Models	252
B.5.7 Predictor-Corrector Samplers	253
B.5.8 Architecture Improvements	257
B.5.9 Controllable Generation	263
B.6 Chapter 8	268
B.6.1 Additional Results	268
B.6.2 The Task of Metal Artifact Removal	268
B.6.3 Details of datasets	274
B.6.4 Details of Score-Based Generative Models	275
B.6.5 Training Details of Baseline Models	276
B.7 Chapter 9	280

B.7.1	Experimental Details	280
B.7.2	Numerical Stability	281
C	Code	286

List of Tables

3.1	Score matching losses and log-likelihoods for NICE models on MNIST. $\sigma = 0.1$ is by grid search and $\sigma = 1.74$ is from the heuristic of [Sar+18].	38
3.2	Negative log-likelihoods on MNIST, estimated by AIS. [†] The result of SSM with $M = 100$, in which case SSM matches the computational cost of kernel methods, which used 100 samples for each data point.	39
3.3	FID scores of different methods versus number of training iterations on CelebA dataset.	39
4.1	Mean squared error between the estimated <i>second order scores</i> and the ground truth on 10^5 test samples. Each setup is trained with three random seeds and multiple noise scales σ	53
4.2	Speed analysis: direct modeling vs. autodiff.	56
4.3	ESS on synthetic datasets. Datasets are shown in Fig. 4.5. We use 32 chains, each with length 10000 and 1000 burn-in steps.	56
5.1	Inception and FID scores for CIFAR-10	73
6.1	Inception and FID scores.	90
7.1	Comparing different reverse-time SDE solvers on CIFAR-10. Shaded regions are obtained with the same computation (number of score function evaluations). Mean and standard deviation are reported over five sampling runs. “P1000” or “P2000”: predictor-only samplers using 1000 or 2000 steps. “C2000”: corrector-only samplers using 2000 steps. “PC1000”: Predictor-Corrector (PC) samplers using 1000 predictor and 1000 corrector steps. . .	102
7.2	CIFAR-10 sample quality.	104

8.1	Results for undersampled MRI reconstruction on BraTS. First two methods are supervised learning techniques trained with $8\times$ acceleration. The others are unsupervised techniques.	121
8.2	Results for sparse-view CT reconstruction on LIDC and LDCT. FISTA-TV is a standard iterative reconstruction method that does not need training. cGAN, Neumann, and SIN-4c-PRN are supervised learning techniques trained with 23 projection angles.	121
8.3	MAR results on LIDC.	122
9.1	SDEs and their corresponding weightings for score matching losses.	134
9.2	Negative log-likelihood (bits/dim) and sample quality (FID scores) on CIFAR-10 and ImageNet 32×32. Abbreviations: “NLL” for “negative log-likelihood”; “Uni. deq.” for “Uniform dequantization”; “Var. deq.” for “Variational dequantization”; “LW” for “likelihood weighting”; and “IS” for “importance sampling”. Bold indicates best result in the corresponding column. Shaded rows represent models trained with both likelihood weighting and importance sampling.	141
9.3	NLLs on CIFAR-10 and ImageNet 32x32.	142
B.1	VAE samples on MNIST.	185
B.2	VAE samples on CelebA.	186
B.3	WAE samples on MNIST.	187
B.4	WAE samples on CelebA.	188
B.5	Architectures on MNIST. In our models, $D_\epsilon = D_z$. D_z takes the values 8 and 32 in different experiments.	193
B.6	Architectures on CelebA. In our models, $D_\epsilon = D_z$. D_z takes the values 8 or 32 in different experiments.	194
B.7	The architectures of NCSN for images of various resolutions.	216
B.8	The architectures of NCSNv2 for images of various resolutions.	217
B.9	Hyperparameters of NCSN/NCSNv2. The latter is configured according to Techniques 6.1 to 6.4. σ_1 and L determine the set of noise levels. T and ϵ are parameters of annealed Langevin dynamics.	218
B.10	$HYPE_\infty$ scores on CelebA 64 × 64. *With truncation tricks.	221
B.11	Training and sampling speed of NCSNv2 on various datasets.	222

B.12 Comparing different samplers on CIFAR-10, where “P2000” uses the rounding interpolation between noise scales. Shaded regions are obtained with the same computation (number of score function evaluations). Mean and standard deviation are reported over five sampling runs.	257
B.13 Optimal signal-to-noise ratios of different samplers. “P1000” or “P2000”: predictor-only samplers using 1000 or 2000 steps. “C2000”: corrector-only samplers using 2000 steps. “PC1000”: PC samplers using 1000 predictor and 1000 corrector steps.	257

List of Figures

1.1	An illustration of the score function (vectors) vs. the density function (contours) for a mixture of two Gaussians.	3
1.2	<i>Overview of score-based generative modeling.</i> By gradually introducing noise into the data, we create a sequence of noisy data densities. We can then reverse this process, removing noise step-by-step, in order to generate novel samples. Using a noise-conditional score model, we estimate the score function for each noise scale, which helps us determine how to sample in a way that leads us towards higher data density regions with less noise (see the illustrative figure on the right).	5
1.3	Random portraits ($> 10^6$ pixels) sampled from our score-based generative model [Son+21b]. These people do not exist.	6
1.4	Inverse problems in medical image reconstruction.	7
3.1	SM loss after training DKEF models on UCI datasets with different loss functions; lower is better. Results for approximate backpropagation are not shown because losses were larger than 10^9	35
3.2	SM performance degrades linearly with the data dimension, while efficient approaches have relatively similar performance.	36
4.1	From left to right: (a) D_2 SM loss without variance reduction ($\sigma = 10^{-3}$). (b) D_2 SM loss with variance reduction ($\sigma = 10^{-3}$). (c) Estimated $\tilde{\mathbf{s}}_1$. (d) Estimated $\tilde{\mathbf{s}}_2$, where the estimation for D_2 SM (0.001) is too far from the ground truth to appear on the plot.	52
4.2	Denoising 2-d synthetic data. The incorporation of $\tilde{\mathbf{s}}_2$ improves uncertainty quantification.	54

4.3	Visualizations of the estimated covariance matrix diagonals on MNIST and CIFAR-10. For CIFAR-10 images, we visualize the diagonal for R, G, B channels separately. Images corrupted with more noise tend to have larger covariance values, indicating larger uncertainty in denoising. Pixels in background have smaller values than pixels near edges, indicating more confident denoising.	55
4.4	Eigenvectors of the estimated covariance matrix on MNIST. The first column shows the noisy images ($\sigma = 0.5$) and the second column shows clean images. The remaining columns show the first 19, plus the 30, 80 and 200-th eigenvectors of the matrix. We can see digit 7 and 9 in the eigenvectors corresponding to the noisy 7, and digit 4 and 9 in the second row, which implies that the estimated covariance matrix can capture different possibilities of the denoising results.	55
4.5	Sampling with Ozaki and Langevin dynamics. We tune the optimal step size separately for both methods. The number in the parenthesis (Column 2 and 4) stands for the iterations used for sampling. We observe that Ozaki obtains more reasonable samples than Langevin dynamics using 1/6 or 1/3 iterations. Column 3 and 5 show samples within a single chain with length 31000 and 1000 burn-in steps.	56
4.6	Sampling a two mode distribution. We use the same step size $\epsilon = 0.01$ for both methods. We observe that Ozaki sampling converges faster than Langevin sampling.	57
4.7	Sampling on MNIST. We observe that Ozaki sampling converges faster than Langevin dynamics. We use step size $\sigma = 0.02$ and initialize the chain with Gaussian noise for both methods.	58
4.8	Sample diversity analysis. The number in the parenthesis in Fig. 4.8a denotes the step size. We initialize the chain with MNIST test images and report the percentage of images that have changed class labels from the initialization w.r.t. sampling iterations. We observe that Ozaki sampling has more diverse samples.	58
5.1	Left: Sliced score matching (SSM) loss w.r.t. iterations. No noise is added to data. Right: Same but data are perturbed with $\mathcal{N}(0, 0.0001)$.	66

5.2	Left: $\nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x})$; Right: $s_{\theta}(\mathbf{x})$. The data density $p_{\text{data}}(\mathbf{x})$ is encoded using an orange colormap: darker color implies higher density. Red rectangles highlight regions where $\nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x}) \approx s_{\theta}(\mathbf{x})$.	67
5.3	Samples from a mixture of Gaussian with different methods. (a) Exact sampling. (b) Sampling using Langevin dynamics with the exact scores. (c) Sampling using annealed Langevin dynamics with the exact scores. Clearly Langevin dynamics estimate the relative weights between the two modes incorrectly, while annealed Langevin dynamics recover the relative weights faithfully.	68
5.4	Intermediate samples of annealed Langevin dynamics.	74
5.5	Uncurated samples on MNIST, CelebA, and CIFAR-10 datasets.	74
5.6	Image inpainting on CelebA (left) and CIFAR-10 (right). The leftmost column of each figure shows the occluded images, while the rightmost column shows the original images.	75
6.1	Generated samples on datasets of decreasing resolutions. From left to right: FFHQ 256×256 , LSUN bedroom 128×128 , LSUN tower 128×128 , LSUN church_outdoor 96×96 , and CelebA 64×64 .	80
6.2	Running annealed Langevin dynamics to sample from a mixture of Gaussian centered at images in the CIFAR-10 test set.	84
6.3	Training loss curves of two noise conditioning methods.	86
6.4	FIDs and color artifacts over the course of training (best viewed in color). The FIDs of NCSN have much higher volatility compared to NCSN with EMA. Samples from the vanilla NCSN often have obvious color shifts. All FIDs are computed with the denoising step.	89
6.5	FIDs for different groups of techniques. Subscripts of “NCSN” are IDs of techniques in effect. “NCSNv2” uses all techniques. Results are computed with the denoising step.	90
6.6	From top to bottom: FFHQ 256^2 , LSUN bedroom 128^2 , LSUN tower 128^2 , and LSUN church_outdoor 96^2 . Within each group of images: the first row shows uncurated samples from NCSNv2, and the second shows the interpolation results between the leftmost and rightmost samples with NCSNv2. You may zoom in to view more details.	91

6.7 NCSN vs. NCSNv2 samples on LSUN church_outdoor (a)(b) and LSUN bedroom (c)(d)	91
7.1 Solving a reverse-time SDE yields a score-based generative model. Transforming data to a simple noise distribution can be accomplished with a continuous-time SDE. This SDE can be reversed if we know the score of the distribution at each intermediate time step, $\nabla_{\mathbf{x}} \log p_t(\mathbf{x})$	95
7.2 Overview of score-based generative modeling through SDEs. We can map data to a noise distribution (the prior) with an SDE (Section 7.3.1), and reverse this SDE for generative modeling (Section 7.3.2). We can also reverse the associated probability flow ODE (Section 7.4.3), which yields a deterministic process that samples from the same distribution as the SDE. Both the reverse-time SDE and probability flow ODE can be obtained by estimating the score $\nabla_{\mathbf{x}} \log p_t(\mathbf{x})$ (Section 7.3.3).	98
7.3 Probability flow ODE enables fast sampling with adaptive stepsizes as the numerical precision is varied (<i>left</i>), and reduces the number of score function evaluations (NFE) without harming quality (<i>middle</i>). The invertible mapping from latents to images allows for interpolations (<i>right</i>).	105
7.4 <i>Left:</i> Class-conditional samples on 32×32 CIFAR-10. Top four rows are automobiles and bottom four rows are horses. <i>Right:</i> Inpainting (top two rows) and colorization (bottom two rows) results on 256×256 LSUN. First column is the original image, second column is the masked/gray-scale image, remaining columns are sampled image completions or colorizations.	107
8.1 We can smoothly perturb images to noise by following the trajectory of an SDE. By estimating the score function $\nabla_{\mathbf{x}} \log p_t(\mathbf{x})$ with neural networks (called score models), it is possible to approximate the reverse SDE and then solve it to generate image samples from noise.	112
8.2 Linear measurement processes for sparse-view CT (<i>left</i>) and undersampled MRI (<i>right</i>).	115
8.3 (<i>Left</i>) An overview of our method for solving inverse problems with score-based generative models. (<i>Right</i>) An illustration about how to combine $\hat{\mathbf{x}}_{t_i}$ and \mathbf{y} to form $\hat{\mathbf{x}}'_{t_i}$	118

8.4 Examples of sparse-view CT reconstruction results on LIDC 320×320 (<i>Top row</i>) and LDCT 512×512 (<i>Bottom row</i>), all with 23 projections. You may zoom in to view more details.	120
8.5 Performance vs. numbers of measurements. Shaded areas represent standard deviation. (<i>Left</i>) MRI on BraTS. (<i>Center</i>) CT on LIDC. (<i>Right</i>) Comparing score-based generative models for undersampled MRI reconstruction on BraTS.	124
9.1 We can use an SDE to diffuse data to a simple noise distribution. This SDE can be reversed once we know the score of the marginal distribution at each intermediate time step, $\nabla_{\mathbf{x}} \log p_t(\mathbf{x})$	130
9.2 Learning curves with the likelihood weighting on the CIFAR-10 dataset (smoothed with exponential moving average). Importance sampling significantly reduces the loss variance.	139
B.1 Log-likelihoods after training DKEF models on UCI datasets with different loss functions; higher is better. Results for approximate backpropagation are not shown because log-likelihoods were smaller than -10^6	191
B.2 Eigenvectors (sorted by eigenvalues) of $\text{Cov}[\mathbf{x} \tilde{\mathbf{x}}]$ estimated by $\tilde{\mathbf{s}}_2(\tilde{\mathbf{x}}; \boldsymbol{\theta})$ on MNIST (more details in Section 4.5).	199
B.3 Eigenvectors (sorted by eigenvalues) of $\text{Cov}[\mathbf{x} \tilde{\mathbf{x}}]$ estimated by $\tilde{\mathbf{s}}_2(\tilde{\mathbf{x}}; \boldsymbol{\theta})$ on MNIST (more details in Section 4.5).	200
B.4 Sampling 2-D synthetic data with score functions. The number in the parenthesis stands for the number of iterations used for sampling. We observe that Ozaki obtains more reasonable samples using 1/6 or 1/3 iterations compared to Langevin dynamics. The second column uses the optimal step size for Ozaki, and the third column uses the same step size and setting for Langevin dynamics. The fourth column uses the optimal step size for Langevin dynamics.	201
B.5 Uncurated samples on MNIST, CelebA, and CIFAR-10 datasets from the baseline model.	206
B.6 Intermediate samples from Langevin dynamics for the baseline model.	207
B.7 Nearest neighbors measured by the ℓ_2 distance between images. Images on the left of the red vertical line are samples from NCSN. Images on the right are nearest neighbors in the training dataset.	208

B.8	Nearest neighbors measured by the ℓ_2 distance in the feature space of an Inception V3 network pretrained on ImageNet. Images on the left of the red vertical line are samples from NCSN. Images on the right are nearest neighbors in the training dataset.	208
B.9	Extended MNIST samples	209
B.10	Extended CelebA samples	210
B.11	Extended CIFAR-10 samples	211
B.12	Extended intermediate samples from annealed Langevin dynamics for CelebA.	212
B.13	Extended intermediate samples from annealed Langevin dynamics for CelebA.	212
B.14	Extended image inpainting results for CelebA. The leftmost column of each figure shows the occluded images, while the rightmost column shows the original images.	213
B.15	Extended image inpainting results for CIFAR-10. The leftmost column of each figure shows the occluded images, while the rightmost column shows the original images.	214
B.16	FIDs and color artifacts over the course of training (best viewed in color). The FIDs of NCSN have much higher volatility compared to NCSN with EMA. Samples from the vanilla NCSN often have obvious color shifts. All FIDs are computed without the denoising step.	220
B.17	FIDs for different groups of techniques. Subscripts of “NCSN” are IDs of techniques in effect. “NCSNv2” uses all techniques. Results are computed without the denoising step.	221
B.18	Uncurated samples from NCSN (a) and NCSNv2 (b) on CelebA 64×64 . . .	221
B.19	EMA reduces undesirable color shifts on CIFAR-10. We show samples from NCSN and NCSN with EMA at the 50k/100k/200k-th training iteration. . .	223
B.20	EMA reduces undesirable color shifts on CelebA-10. We show samples from NCSN and NCSN with EMA at the 50k/100k/150k-th training iteration. . .	224

B.21 Samples from models with different groups of techniques applied. NCSN is the original model in Chapter 5 and does not use any of the newly proposed techniques. Subscripts of “NCSN” denote the IDs of techniques in effect. NCSN ₅ only applies EMA. NCSN _{1,2,4,5} applies both EMA and technique group (ii). NCSNv2 is the result of all techniques combined. Checkpoints are selected according to the lowest FID (with denoising) over the course of training.	225
B.22 Samples from models with different groups of techniques applied. NCSN is the original model in Chapter 5 and does not use any of the newly proposed techniques. Subscripts of “NCSN” denote the IDs of techniques in effect. NCSN ₅ only applies EMA. NCSN _{1,2,4,5} applies both EMA and technique group (ii). NCSNv2 is the result of all techniques combined. Checkpoints are selected according to the lowest FID (without denoising) over the course of training.	226
B.23 Training vs. test loss curves of NCSNv2.	227
B.24 Nearest neighbors on CIFAR-10. NCSNv2 samples are on the left side of the red vertical line. Corresponding nearest neighbors are on the right side in the same row.	228
B.25 Nearest neighbors on CelebA 64 × 64.	228
B.26 Nearest neighbors on LSUN church_outdoor 96 × 96.	229
B.27 Nearest neighbors on FFHQ 256 × 256.	229
B.28 NCSNv2 interpolation results on CelebA 64 × 64.	230
B.29 NCSNv2 interpolation results on LSUN church_outdoor 96 × 96.	230
B.30 NCSNv2 interpolation results on LSUN bedroom 128 × 128.	231
B.31 NCSNv2 interpolation results on LSUN tower 128 × 128.	231
B.32 NCSNv2 interpolation results on FFHQ 256 × 256.	232
B.33 Uncurated CIFAR-10 32 × 32 samples from NCSNv2.	233
B.34 Uncurated CelebA 64 × 64 samples from NCSNv2.	234
B.35 Uncurated LSUN church_outdoor 96 × 96 samples from NCSNv2.	235
B.36 Uncurated LSUN bedroom 128 × 128 samples from NCSNv2.	236
B.37 Uncurated LSUN tower 128 × 128 samples from NCSNv2.	237
B.38 Uncurated FFHQ 256 × 256 samples from NCSNv2.	238

B.39 Discrete-time perturbation kernels and our continuous generalizations match each other almost exactly. (a) compares the variance of perturbation kernels for SMLD and VE SDE; (b) compares the scaling factors of means of perturbation kernels for DDPM and VP SDE; and (c) compares the variance of perturbation kernels for DDPM and VP SDE.	245
B.40 Samples from the probability flow ODE for VP SDE on 256×256 CelebA-HQ. Top: spherical interpolations between random samples. Bottom: temperature rescaling (reducing norm of embedding).	249
B.41 Comparing the first 100 dimensions of the latent code obtained for a random CIFAR-10 image. “Model A” and “Model B” are separately trained with different architectures.	250
B.42 <i>Left:</i> The dimension-wise difference between encodings obtained by Model A and B. As a baseline, we also report the difference between shuffled representations of these two models. <i>Right:</i> The dimension-wise correlation coefficients of encodings obtained by Model A and Model B.	250
B.43 PC sampling for LSUN bedroom and church. The vertical axis corresponds to the total computation, and the horizontal axis represents the amount of computation allocated to the corrector. Samples are the best when computation is split between the predictor and corrector.	256
B.44 The effects of different architecture components for score-based models trained with VE perturbations.	258
B.45 unconditional cifar-10 samples from ncsn++ cont. (deep, ve).	260
B.46 samples on 1024×1024 celeba-hq from a modified ncsn++ model trained with the ve sde.	261
B.47 Class-conditional image generation by solving the conditional reverse-time SDE with PC. The curve shows the accuracy of our noise-conditional classifier over different noise scales.	265
B.48 Extended inpainting results for 256×256 bedroom images.	269
B.49 Extended inpainting results for 256×256 church images.	270
B.50 Extended colorization results for 256×256 bedroom images.	271
B.51 Extended colorization results for 256×256 church images.	272

B.52 SSIM vs. numbers of measurements. Shaded areas represent standard deviation. (<i>Left</i>) MRI on BraTS. (<i>Center</i>) CT on LIDC. (<i>Right</i>) Comparing score-based generative models for undersampled MRI reconstruction on BraTS.	273
B.53 Examples of metal artifact removal on LIDC. You may zoom in to view more details.	273
B.54 The linear measurement process of metal artifact removal.	274
B.55 Samples on CIFAR-10. (a) Model with the best FID. (b) ScoreFlow trained with likelihood weighting + importance sampling + VP SDE. Samples of both models are generated with the same random seed.	281
B.56 Samples on ImageNet 32×32 . (a) Model with the best FID. (b) ScoreFlow trained with likelihood weighting + importance sampling + VP SDE. Samples of both models are generated with the same random seed.	282

Chapter 1

Introduction

What I cannot create, I do not understand.

— Richard Feynman, 1988

Modeling and generating high-dimensional data is an essential component of many important tasks in machine learning. For example, machine translation [SVL14; BCB15; Vas+17; Bro+20], text-to-image generation [Ram+21; Nic+21; Ram+22; Sah+22], and code completion [Che+21a; Nij+22] all rely on models that can output complex data (documents, pictures, code) with thousands to millions of attributes or dimensions (tokens, pixels, and characters). These models, often called *(probabilistic) generative models* in machine learning, operate under the assumption that data points come from an unknown data distribution [Sai96; CB21]. Utilizing expressive deep neural networks, generative models can parameterize complex probability distributions to estimate the underlying data distribution of a given dataset. We can then compare the quality of various data points by computing their probabilities, or create new data points by sampling from the estimated distribution.

However, a significant challenge arises when applying generative models to data with high dimensionality. Since generative models estimate a probability distribution, they must adhere to the normalization constraint, which mandates that the total probability of all possible data configurations equals one [LeC+06; SK21]. This is relatively easy to enforce for small output spaces, such as image classification and linear regression, which have relatively simple outputs with just a few dimensions. However, enforcing this constraint with high-dimensional data is generally impractical, as the number of possible data configurations grows exponentially with each dimension, making the computation of total probability

intractable [Erm+13].

To approach this difficulty, existing generative models use one of three methods (more detail in Section 2.2): (i) Markov chain Monte Carlo (MCMC) for approximate normalization [AHS85; LeC+06; Ngi+11; DM19a; SK21], (ii) models with exactly normalized architectures [BB99; VKK16; DKB15; DSB17; KW14; RMW14a; KD18], or (iii) generative adversarial networks (GANs [Goo+14]) that bypass normalization. However, each of these approaches has its own trade-offs, such as imprecise probability calculations (i, iii), reduced model flexibility (ii), or low-diversity samples (iii) [Met+16; Rot+17]. This leaves an important question open: **can we address the normalization challenge without sacrificing model flexibility, sample quality, or probability evaluation?**

1.1 Approach: Score-Based Generative Modeling

This dissertation presents an affirmative answer to the above question based on working with the gradient function of the logarithmic probability density, known as the (*Stein*) *score function* (or score for brevity) [GM15; LLJ16]. As illustrated in Fig. 1.1, the score function is a vector field that illustrates directions of maximum probability density growth. Clearly, it can be derived from the density function via differentiation, or used to recover the density function through integration. Score functions uniquely determine probability distributions and, importantly, do not require normalization (see Section 2.3). This allows for modeling with flexible deep neural networks without additional constraints.

By working with the data distribution through score functions, we explore *score-based generative modeling* [Son+19; SE19b; SE20; Son+21b; Son+21a], a new technique that allows for **flexible model architectures, high sample quality, and accurate probability evaluation**. Our empirical findings have shown that this method can generate realistic data that surpasses the state-of-the-art on a variety of datasets [SE19b; Son+21b; Son+21a; Tas+21], **challenging the dominance of Generative Adversarial Networks (GANs [Goo+14]) in many image generation tasks**. Following our original work, a growing body of research¹ has emerged and developed this approach to successfully generate realistic data in domains such as images [DN21], audio [Che+21b], speech [TKK21], time series [Tas+21], molecules [Xu+22], material structures [Xie+22] and point clouds [Cai+20a].

We organize the dissertation into three thematic parts to focus on different advantages

¹See <https://scorebasedgenerativemodeling.github.io/> for a partial list.

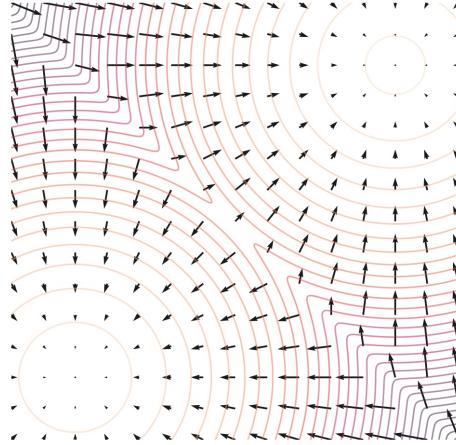


Figure 1.1: An illustration of the score function (vectors) vs. the density function (contours) for a mixture of two Gaussians.

of score-based generative models and discuss corresponding technical details: First, models of score functions, called *score models*, can be parameterized by very flexible deep neural networks, and estimated from data with efficient and principled statistical approaches (Section 1.1.1); Second, we can use score models to generate high-quality data samples, while also providing a means to tackle Bayesian reasoning tasks and ill-posed inverse problems (Section 1.1.2); Third, we can compute probability values accurately with score models (Section 1.1.3) as in traditional probabilistic models. We provide more detailed discussions for each of these topics in the sections below.

1.1.1 Learning Score Models from Data

Since score functions do not require normalization, score models can be parameterized by very flexible deep neural networks with minimal constraints on their architectures. This allows us to take full advantage of the deep learning revolution [LBH15] and employ deep neural network architectures that have been proven successful in other relevant tasks to estimate complex score functions of high-dimensional data by building powerful score models. However, for improved performance on generative modeling, the flexibility of model architectures must be achieved in conjunction with effective model training. How can we efficiently learn complex score models from large-scale datasets?

To answer this question, we make our first contribution in this dissertation by improving upon score matching [HD05], a classic statistical method for learning score models from

data examples (the task of which is called *score estimation*). However, standard score matching becomes costly when dealing with high-dimensional data and deep neural network models [MSS12; KL10; Son+19]. To overcome this issue, we developed *sliced score matching* [Son+19], which draws inspiration from Monte Carlo estimation and sketching techniques [Woo+14]. It applies score matching to random one-dimensional slices of score functions, which is much more computationally efficient than standard score matching. We have proven that this approach yields **consistent and normally distributed parameters**, and it has performed comparably to standard score matching in multiple tests **with computational and memory costs that are tens to thousands of times lower**.

While score models capture the first order gradients of data distributions, it is insufficient for many important applications that require high order gradients. Building upon the connection between denoising and score functions, as unveiled by Tweedie’s formula [Efr11] and denoising score matching [Vin11; RS11], we extend the definition of score functions to high order cases, and then propose an efficient training objective to estimate high order score functions via denoising [Men+21]. Our models can accurately approximate high order gradients of the log data density, enabling new applications in uncertainty quantification and accelerated MCMC sampling.

1.1.2 Generating Data Samples with Score Models

To generate data samples from a trained score model, we can use various established MCMC techniques such as Langevin dynamics [Par81; GM94] and Hybrid Monte Carlo [Dua+87; Nea+11]. These techniques are able to sample from a probability distribution based solely on the score function. However, when dealing with high-dimensional data, the convergence rates of these methods decrease significantly [DM19b; LZT22], limiting their efficacy in generating complex data types such as images, audio, and text.

We tackle this curse of dimensionality by employing a **divide-and-conquer approach**² that breaks down the challenge of generating high-dimensional data into a series of more manageable denoising tasks [Tia+20]. As illustrated in Fig. 1.2, this approach, which we refer to as *score-based generative modeling* [Son+19; SE19b; SE20; Son+21b; Son+21a], involves gradually adding and then removing noise from the data in order to generate novel

²Similar ideas appeared before in, for example, simulated annealing [KGV83], parallel tempering [SW86], annealed importance sampling [Nea01], path sampling [GM98], numerical continuation methods [AG03], variational walkback [Goy+17], infusion training [BHV17], and most related to ours, diffusion probabilistic models [Soh+15].

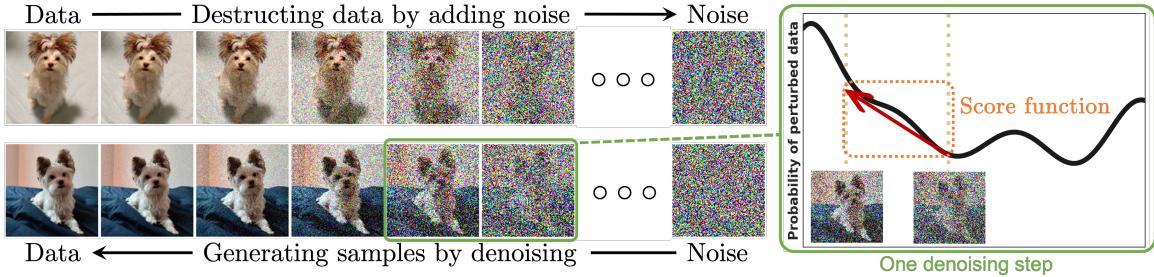


Figure 1.2: *Overview of score-based generative modeling.* By gradually introducing noise into the data, we create a sequence of noisy data densities. We can then reverse this process, removing noise step-by-step, in order to generate novel samples. Using a noise-conditional score model, we estimate the score function for each noise scale, which helps us determine how to sample in a way that leads us towards higher data density regions with less noise (see the illustrative figure on the right).

samples. The key to successfully denoising data lies in estimating the score function of noise-perturbed data distributions; in order to do so, we train a *noise-conditional score model* [SE19b] that predicts the score function for each noise scale. Intuitively, this model allows us to identify regions of higher data density as we denoise the data.

To implement the reverse process in Fig. 1.2, we can work backwards from the highest noise level, then sequentially use Langevin dynamics with the estimated score functions to decrease the noise level until we reach our desired data density. In the limit of infinitely many noise levels, this approach, referred to as annealed Langevin dynamics [SE19b; SE20], can be characterized as the solution to a stochastic differential equation (SDE) determined by the score functions at all noise levels [Son+21b]. By utilizing numerical SDE solvers, we are able to produce a variety of sampling methods that can trade-off between generation quality and computational efficiency. This SDE formulation also enables us to link score-based generative models with diffusion probabilistic models [Soh+15], unifying the two approaches within a common framework.

Our empirical results demonstrate that score-based generative modeling can generate realistic high-dimensional data that surpasses the state-of-the-art in terms of quality [SE19b; Son+21b; Son+21a; Tas+21]. **This approach has emerged as a challenger to the long-held dominance of Generative Adversarial Networks (GANs) in the realm of image generation tasks.** For example, our initial implementation [SE19b] achieved the state-of-the-art Inception Score on the CIFAR-10 dataset, which is a common benchmark



Figure 1.3: Random portraits ($> 10^6$ pixels) sampled from our score-based generative model [Son+21b]. These people do not exist.

within generative modeling. Our subsequent work with score-based generative models [Son+21b] further solidified our lead over other methods and allowed us to generate **high-fidelity images up to 1024×1024 resolution** (see Fig. 1.3).

Generative models are most useful when they can be flexibly controlled to satisfy user constraints. Score-based generative models, in particular, **excel at generating high-quality data samples unconditionally while also allowing for user control over the sampling process**. These models are powerful enough to solve various ill-posed inverse problems even without being specifically trained for them. We have demonstrated their success in many popular inverse problems encountered in computer vision, such as image inpainting, image colorization, and class-conditional image generation [Son+21b]. Additionally, in comparison to task-specific deep learning approaches, score-based generative models perform as well or better on a variety of inverse problems in medical imaging (*e.g.*, computed tomography and magnetic resonance imaging), while maintaining better robustness in situations where measurement settings change [Son+22].

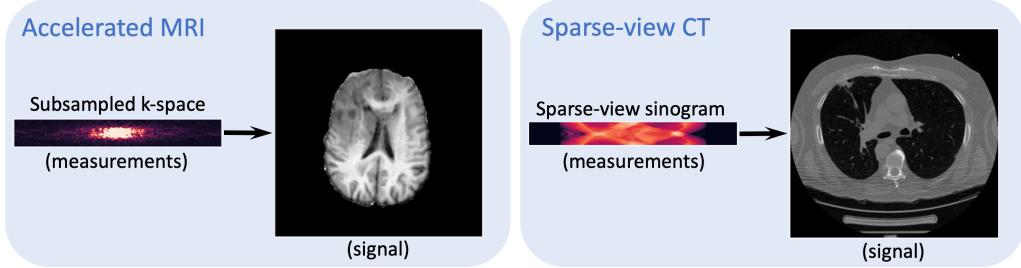


Figure 1.4: Inverse problems in medical image reconstruction.

1.1.3 Evaluating Probability Values with Score Models

Score-based generative models **not only generate realistic data samples, but also accurately compute probability values (also known as likelihoods) for a given data point**. These likelihoods have a variety of applications, such as outlier detection [Son+18a], model comparison [TOB16a], and data compression [TBB19; Kin+21]. In order to compute these likelihoods, we convert the stochastic differential equation (SDE) mentioned before into an ordinary differential equation (ODE) called the *probability flow ODE* [Son+21b], which we then solve with established numerical ODE solvers. By doing so, we can simultaneously generate samples and compute the accurate log-likelihood for the probabilistic model associated with the sampling process. Furthermore, we have proven that a particular score matching objective lower bounds the log-likelihood on data, meaning we can use this objective to **efficiently optimize likelihood prediction of score-based generative models** [Son+21a].

Our empirical findings demonstrate that the likelihoods of score-based generative models are comparable or superior to other state-of-the-art methods when applied to realistic image datasets [Son+21a]. A related work from Google Brain took this even further [Kin+21]; by utilizing improved model architectures and training techniques, they were able to surpass previous likelihood records and achieve state-of-the-art performance in lossless image compression. These recent results **challenge the longstanding dominance of autoregressive generative models** [VKK16] and **variational autoencoders** [KW14; RMW14a] in terms of likelihood performance.

1.2 Dissertation Structure

This dissertation is comprised of three thematic parts.

Part I focuses on efficient and scalable techniques for training various types of score models parameterized by deep neural networks, with the goal to improve score estimation for high-dimensional datasets.

- In Chapter 3, we leverage random projections to significantly reduce the training cost of score models, boosting the efficiency of score estimation for large-scale modern datasets. This chapter is based on [Son+19].
- In Chapter 4, we consider the problem of estimating higher order gradients of the data distribution, or higher order score functions, for novel applications in uncertainty quantification and accelerated MCMC sampling. Our method trains high order score models efficiently through a generalized connection between denoising and score estimation. This chapter is built upon [Men+21].

Part II investigates the challenges of vanilla MCMC approaches for generating high-dimensional data samples from trained score models. To combat the curse of dimensionality, we leverage ideas from homotopy methods, MCMC, and differential equations to devise sampling methods that are effective for high-dimensional data while also being flexibly controllable.

- In Chapter 5, we propose to remediate the difficulties of vanilla MCMC sampling by training noise-conditional score models on datasets perturbed with various magnitudes of Gaussian noise. We then generate samples by sequentially performing Langevin MCMC with annealed noise levels, demonstrating significant improvements in image generation. This chapter was previously published as [SE19b].
- In Chapter 6, we present a principled analysis on the learning and sampling of score-based generative models in high-dimensional spaces, giving rise to a set of widely applicable techniques that significantly improve the stability and quality of the sampling process. This chapter is based on [SE20].
- In Chapter 7, we investigate score-based generative modeling in the limit of infinitely many noise levels with tools from differential equations. Our analysis yields a new

framework that includes a stochastic differential equation (SDE) and an associated ordinary differential equation (ODE) for sample generation, which deepens our theoretical understanding and improves the quality, efficiency & controllability of sampling. This chapter is mostly based on [Son+21b].

- In Chapter 8, we propose new methods for controlling the sampling process of score-based generative models to solve a broad family of linear inverse problems in medical image reconstruction, including magnetic resonance imaging (MRI), and computed tomography (CT). This chapter is based on [Son+22].

Part III is about evaluating and training to optimize the probability values of arbitrary data points with score-based generative models.

- In Chapter 9, we show that the ODE introduced in Chapter 7 implies a new way to evaluate probability values or likelihoods exactly through iterative numerical ODE solvers. In addition, we show that score functions provide a lower bound on the log-likelihood, which allows for efficient maximum likelihood training without iterative likelihood evaluation. This chapter is based on [Son+21b] and [Son+21a].

We conclude and discuss future research avenues in Chapter 10.

Chapter 2

Background

We set out to provide some background knowledge on the foundations of generative models. We first setup the problem of generative modeling from a statistical perspective, then analyze the challenges of training large-scale generative models on high-dimensional datasets. Thereafter, we introduce three predominant ways in current literature to get around these challenges, and discuss their pros and cons. Finally, we introduce score functions and score models, the central research topic of this dissertation, and discuss their potential to overcome the drawbacks of existing methods.

2.1 Statistical Foundations of Generative Models

2.1.1 Problem Settings

Let $\{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\}$ denote a dataset of size N . A widely accepted assumption in machine learning and statistics is that all data points are independent and identically distributed (i.i.d.) samples from an underlying data distribution $p_{\text{data}}(\mathbf{x})$. That is,

$$\{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\} \stackrel{\text{i.i.d.}}{\sim} p_{\text{data}}(\mathbf{x}).$$

For ease of notation, we slightly abuse the symbol p to denote either a probability distribution or its probability density/mass function depending on the context. Clearly, once this data distribution $p_{\text{data}}(\mathbf{x})$ becomes available, we can generate an arbitrary number of new data points by sampling from $p_{\text{data}}(\mathbf{x})$, and compute the probability value or likelihood of an arbitrary data sample \mathbf{x}' by evaluating $p_{\text{data}}(\mathbf{x}')$.

The goal of generative modeling is thus to model and estimate this unknown data distribution $p_{\text{data}}(\mathbf{x})$ from a given dataset, so that we can generate new data samples at will and, ideally, also query the probability values of arbitrary data points. To this end, we build a statistical model $p_{\boldsymbol{\theta}}(\mathbf{x})$, where $\boldsymbol{\theta} \in \Theta$ denotes model parameters, and Θ denotes the set of allowable parameter values. We seek to find the optimal parameter $\boldsymbol{\theta}^* \in \Theta$ such that

$$p_{\boldsymbol{\theta}^*}(\mathbf{x}) \approx p_{\text{data}}(\mathbf{x}).$$

When the statistical model $p_{\boldsymbol{\theta}^*}(\mathbf{x})$ is very close to the data distribution $p_{\text{data}}(\mathbf{x})$, we can generate new samples and evaluate probability values using $p_{\boldsymbol{\theta}^*}(\mathbf{x})$ as a proxy for $p_{\text{data}}(\mathbf{x})$. The statistical model $p_{\boldsymbol{\theta}}(\mathbf{x})$ is often called a *generative model* in the machine learning community.

2.1.2 Model Training

Given a distance metric $D(\cdot \parallel \cdot)$ that measures the difference between $p_{\boldsymbol{\theta}}(\mathbf{x})$ and $p_{\text{data}}(\mathbf{x})$, we can obtain the optimal parameter $\boldsymbol{\theta}^*$ by solving

$$\boldsymbol{\theta}^* = \arg \min_{\boldsymbol{\theta} \in \Theta} D(p_{\text{data}} \parallel p_{\boldsymbol{\theta}})$$

Clearly, we have $p_{\boldsymbol{\theta}^*}(\mathbf{x}) \approx p_{\text{data}}(\mathbf{x})$ if Θ is sufficiently large, and the model family $\{p_{\boldsymbol{\theta}}\}_{\boldsymbol{\theta} \in \Theta}$ is sufficiently expressive. Popular instances of $D(\cdot \parallel \cdot)$ include f -divergences [Csi64; AS66] and integral probability metrics [Mül97]. In general, given a convex and semi-continuous function $f : [0, \infty) \rightarrow \mathbb{R}$ with $f(0) = 1$, we can define an f -divergence as below:

$$D_f(p_{\text{data}} \parallel p_{\boldsymbol{\theta}}) := \int p_{\boldsymbol{\theta}}(\mathbf{x}) f\left(\frac{p_{\text{data}}(\mathbf{x})}{p_{\boldsymbol{\theta}}(\mathbf{x})}\right) d\mathbf{x}, \quad (2.1)$$

assuming $p_{\text{data}}(\mathbf{x})$ is absolutely continuous with respect to $p_{\boldsymbol{\theta}}(\mathbf{x})$. For integral probability metrics, we define them in reference to an appropriate function family \mathcal{F} :

$$D_{\mathcal{F}}(p_{\text{data}} \parallel p_{\boldsymbol{\theta}}) := \sup_{f \in \mathcal{F}} \mathbb{E}_{p_{\text{data}}(\mathbf{x})}[f(\mathbf{x})] - \mathbb{E}_{p_{\boldsymbol{\theta}}(\mathbf{x})}[f(\mathbf{x})]. \quad (2.2)$$

For training generative models on a given dataset $\{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\} \stackrel{\text{i.i.d.}}{\sim} p_{\text{data}}(\mathbf{x})$, it is necessary to obtain sample-based estimates of the metric $D(\cdot \parallel \cdot)$. As an example, plugging

$f(x) = x \log x$ into the f -divergence in Eq. (2.1), we obtain the Kullback-Leibler (KL, [KL51]) divergence

$$\begin{aligned} D_{KL}(p_{\text{data}} \parallel p_{\boldsymbol{\theta}}) &:= \int p_{\text{data}}(\mathbf{x}) \log \left(\frac{p_{\text{data}}(\mathbf{x})}{p_{\boldsymbol{\theta}}(\mathbf{x})} \right) d\mathbf{x} \\ &= \mathbb{E}_{p_{\text{data}}(\mathbf{x})} \left[\log \left(\frac{p_{\text{data}}(\mathbf{x})}{p_{\boldsymbol{\theta}}(\mathbf{x})} \right) \right], \end{aligned} \quad (2.3)$$

wherein the expectation can be estimated using the empirical mean over samples in the training dataset:

$$\begin{aligned} D_{KL}(p_{\text{data}} \parallel p_{\boldsymbol{\theta}}) &\approx \frac{1}{N} \sum_{i=1}^N \log \left(\frac{p_{\text{data}}(\mathbf{x}_i)}{p_{\boldsymbol{\theta}}(\mathbf{x}_i)} \right) \\ &= \frac{1}{N} \sum_{i=1}^N \log p_{\text{data}}(\mathbf{x}_i) - \frac{1}{N} \sum_{i=1}^N \log p_{\boldsymbol{\theta}}(\mathbf{x}_i) \\ &\stackrel{(i)}{=} -\frac{1}{N} \sum_{i=1}^N \log p_{\boldsymbol{\theta}}(\mathbf{x}_i) + \text{const.} \end{aligned} \quad (2.4)$$

Here (i) holds because $\frac{1}{N} \sum_{i=1}^N \log p_{\text{data}}(\mathbf{x}_i)$ does not depend on the model parameter $\boldsymbol{\theta}$ and is therefore a constant for the sake of model training. Eq. (2.4) implies that we can train a generative model $p_{\boldsymbol{\theta}}(\mathbf{x})$ on the dataset $\{\mathbf{x}_1, \mathbf{x}_2, \dots, \dots\} \stackrel{\text{i.i.d.}}{\sim} p_{\text{data}}(\mathbf{x})$ by maximizing its averaged log-likelihood over training data points, *i.e.*,

$$\boldsymbol{\theta}^* = \arg \max_{\boldsymbol{\theta}} \frac{1}{N} \sum_{i=1}^N \log p_{\boldsymbol{\theta}}(\mathbf{x}_i), \quad (2.5)$$

which is precisely the classic approach of *maximum likelihood estimation* (MLE) in statistics, a standard method for learning probabilistic models from data.

2.1.3 The Challenge of Normalization

To model valid probability distributions, $p_{\boldsymbol{\theta}}(\mathbf{x})$ must satisfy two natural constraints for all possible values of $\boldsymbol{\theta}$:

1. (Non-negativity) $\forall \mathbf{x} : p_{\boldsymbol{\theta}}(\mathbf{x}) \geq 0$.
2. (Normalization) $\int p_{\boldsymbol{\theta}}(\mathbf{x}) d\mathbf{x} = 1$.

Compared to non-negativity, enforcing normalization is much more challenging because it involves integrating/summing the values of $p_{\theta}(\mathbf{x})$ over all \mathbf{x} , which is typically an intractable counting problem for complex $p_{\theta}(\mathbf{x})$ in high-dimensional data domains. Although we are in the deep learning era and have very powerful deep neural networks at our disposal to form expressive function approximators, building a properly normalized statistical model is still non-trivial and has become one crucial challenge for creating powerful generative models.

2.2 Existing Methods

Below we give an introduction to several predominant families of generative models in the literature. We classify them into three categories based on how they address the challenge of normalization.

2.2.1 Approximating the Normalizing Constant

Since exact normalization is generally intractable, we often have to resort to approximation. *Energy-based models* (EBMs, [AHS85; LeC+06; SK21]) are one such family of generative models that perform approximate normalization using Monte Carlo sampling. Inspired by Boltzmann distributions in statistical mechanics, the generative model $p_{\theta}(\mathbf{x})$ is parameterized in the form of

$$p_{\theta}(\mathbf{x}) = \frac{\exp(-E_{\theta}(\mathbf{x}))}{Z_{\theta}}, \quad (2.6)$$

where $E_{\theta}(\mathbf{x})$ is a scalar-valued function (called the *energy function*) with parameter θ , and Z_{θ} is a constant (called the *normalizing constant* or *partition function*) defined by

$$Z_{\theta} := \int \exp(-E_{\theta}(\mathbf{x})) d\mathbf{x}, \quad (2.7)$$

which ensures the proper normalization of $p_{\theta}(\mathbf{x})$. The energy function $E_{\theta}(\mathbf{x})$ does not need to be normalized and hence can be parameterized with flexible deep neural networks. The partition function Z_{θ} , in contrast, is responsible for all the heavy lifting of normalization. Due to the difficulty of computing high-dimensional integrals for complicated functions, evaluating Z_{θ} is non-trivial and therefore approximation techniques are essential for tractable learning and inference of EBMs.

We first focus on how approximation methods enable tractable training of EBMs. With

maximum likelihood estimation, we can learn an EBM by solving the following optimization problem:

$$\begin{aligned}\boldsymbol{\theta}^* &= \arg \max_{\boldsymbol{\theta}} \frac{1}{N} \sum_{i=1}^N \log p_{\boldsymbol{\theta}}(\mathbf{x}_i) \\ &= \arg \max_{\boldsymbol{\theta}} -\frac{1}{N} \sum_{i=1}^N E_{\boldsymbol{\theta}}(\mathbf{x}_i) - \log Z_{\boldsymbol{\theta}} \\ &= \arg \min_{\boldsymbol{\theta}} \underbrace{\frac{1}{N} \sum_{i=1}^N E_{\boldsymbol{\theta}}(\mathbf{x}_i)}_{:=\mathcal{L}(\boldsymbol{\theta})} + \log Z_{\boldsymbol{\theta}}.\end{aligned}$$

When $E_{\boldsymbol{\theta}}(\mathbf{x})$ is a deep neural network, as is often the case, we typically find the optimal parameter $\boldsymbol{\theta}^*$ by minimizing $\mathcal{L}(\boldsymbol{\theta})$ with *gradient descent*. Importantly, the gradient of $\mathcal{L}(\boldsymbol{\theta})$ is given by

$$\nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}) = \frac{1}{N} \sum_{i=1}^N \nabla_{\boldsymbol{\theta}} E_{\boldsymbol{\theta}}(\mathbf{x}_i) + \nabla_{\boldsymbol{\theta}} \log Z_{\boldsymbol{\theta}},$$

where $\nabla_{\boldsymbol{\theta}} E_{\boldsymbol{\theta}}(\mathbf{x})$ can be efficiently evaluated with automatic differentiation or backpropagation [RHW86], but $\nabla_{\boldsymbol{\theta}} \log Z_{\boldsymbol{\theta}}$ involves the intractable partition function and is thus hard to obtain. Luckily, the latter can be approximated with contrastive divergence [Hin02] and Markov chain Monte Carlo (MCMC), a family of sampling techniques that do not require evaluating the normalizing constant. In particular, we have

$$\begin{aligned}\nabla_{\boldsymbol{\theta}} \log Z_{\boldsymbol{\theta}} &= \nabla_{\boldsymbol{\theta}} \log \int \exp(-E_{\boldsymbol{\theta}}(\mathbf{x})) d\mathbf{x} = \frac{\int \nabla_{\boldsymbol{\theta}} \exp(-E_{\boldsymbol{\theta}}(\mathbf{x})) d\mathbf{x}}{\int \exp(-E_{\boldsymbol{\theta}}(\mathbf{x})) d\mathbf{x}} \\ &= -\frac{\int \exp(-E_{\boldsymbol{\theta}}(\mathbf{x})) \nabla_{\boldsymbol{\theta}} E_{\boldsymbol{\theta}}(\mathbf{x}) d\mathbf{x}}{\int \exp(-E_{\boldsymbol{\theta}}(\mathbf{x})) d\mathbf{x}} = -\int p_{\boldsymbol{\theta}}(\mathbf{x}) \nabla_{\boldsymbol{\theta}} E_{\boldsymbol{\theta}}(\mathbf{x}) d\mathbf{x} \\ &= -\mathbb{E}_{p_{\boldsymbol{\theta}}(\mathbf{x})}[\nabla_{\boldsymbol{\theta}} E_{\boldsymbol{\theta}}(\mathbf{x})].\end{aligned}\tag{2.8}$$

As a result, one can first generate a sample $\hat{\mathbf{x}}$ from $p_{\boldsymbol{\theta}}(\mathbf{x})$ with MCMC (which notably does not require computing $Z_{\boldsymbol{\theta}}$), then approximate $\nabla_{\boldsymbol{\theta}} \log Z_{\boldsymbol{\theta}}$ with the following one-sample Monte Carlo estimator:

$$\nabla_{\boldsymbol{\theta}} \log Z_{\boldsymbol{\theta}} \approx -\nabla_{\boldsymbol{\theta}} E_{\boldsymbol{\theta}}(\hat{\mathbf{x}}).$$

This enables approximate maximum likelihood training for EBMs.

Sampling from EBMs is straightforward as we can employ MCMC to generate new data samples from $p_{\theta}(\mathbf{x})$ without requiring its partition function Z_{θ} . For evaluating probability values, however, an estimate of Z_{θ} is unavoidable since it is on the denominator of $p_{\theta}(\mathbf{x})$. Although there exist methods such as annealed importance sampling [Nea01] for estimating Z_{θ} , EBMs cannot compute accurate probability values due to inevitable estimation errors. On the flip side, EBMs enjoy greater flexibility of model architectures since $E_{\theta}(\mathbf{x})$ can be parameterized by expressive deep neural networks without explicitly enforcing normalization.

2.2.2 Restricting Model Architectures

It is possible to enforce normalization exactly by designing model architectures in special ways. Below we introduce three representative generative models as examples.

Autoregressive Models

In probability theory, the *chain rule* dictates that any high-dimensional probability distribution can be factorized into the product of a chain of one-dimensional conditional distributions. Building on this fact, a family of generative models called *autoregressive models* [BB99; Ger+15; Uri+16; VKK16] construct high-dimensional statistical models by composing a plethora of univariate conditional distributions that are much easier to parameterize and normalize. In particular, autoregressive models define $p_{\theta}(\mathbf{x})$ in the form of

$$p_{\theta}(\mathbf{x}) = \prod_{i=1}^d p_{\theta}(x_i \mid \mathbf{x}_{<i}), \quad (2.9)$$

where d is the dimensionality of \mathbf{x} , x_i denotes the i -th element of \mathbf{x} , and $\mathbf{x}_{<i}$ denotes the set $\{x_1, x_2, \dots, x_{i-1}\}$ (where we let $\mathbf{x}_{<1} := \emptyset$). This autoregressive structure can guarantee the exact normalization of $p_{\theta}(\mathbf{x})$ once each conditional distribution $p_{\theta}(x_i \mid \mathbf{x}_{<i})$ is properly normalized, since

$$\begin{aligned} \int p_{\theta}(\mathbf{x}) d\mathbf{x} &= \int \prod_{i=1}^d p_{\theta}(x_i \mid \mathbf{x}_{<i}) d\mathbf{x} \\ &= \int p(x_1) \int p_{\theta}(x_2 \mid x_1) \int \cdots \int p_{\theta}(x_d \mid \mathbf{x}_{<d}) dx_d \cdots dx_2 dx_1. \end{aligned}$$

Autoregressive models parameterize $p_{\theta}(x_i \mid \mathbf{x}_{<i})$ with normalized parametric statistical

distributions (like exponential families) on top of deep neural network backbones to enforce the exact normalization of $p_{\theta}(\mathbf{x})$. As such, they can accurately evaluate the probability of any data point and learn from data with maximum likelihood training. For data generation, autoregressive models produce each sample $\hat{\mathbf{x}}$ by sequentially generating $\hat{x}_1, \hat{x}_2, \dots, \hat{x}_d$ from $p_{\theta}(\hat{x}_1), p_{\theta}(\hat{x}_2 | \hat{x}_1), \dots, p_{\theta}(\hat{x}_d | \hat{\mathbf{x}}_{<d})$ with ancestral sampling [KF09]. However, autoregressive factorization taxes the flexibility of model architectures in two important ways: First, data dimensions have to be put in a particular order; yet not all data domains possess a natural ordering across data dimensions. For example, it is unclear what would a natural ordering be for pixels in image generation. Second, enforcing a strict ordering of data dimensions in autoregressive modeling limits our choice of model architectures to some special constructs, such as masked convolutions [VKK16].

Normalizing Flows

Normalizing flows are another family of generative models that have exact normalization. The key idea is based on the *change of variable formula*. Suppose $\mathbf{z} \in \mathbb{R}^d$ is a continuous random variable obeying a prior distribution $\pi(\mathbf{z})$, which is a tractable distribution allowing efficient probability evaluation and fast sample generation. Normalizing flows parameterize a *continuously differentiable invertible function* $\mathbf{f}_{\theta} : \mathbb{R}^d \rightarrow \mathbb{R}^d$ with a deep neural network, then use it to transform \mathbf{z} to another random variable $\mathbf{x} = \mathbf{f}_{\theta}(\mathbf{z})$. This invertible transformation also morphs $\pi(\mathbf{z})$ to a distribution over \mathbf{x} , which we denote as $p_{\theta}(\mathbf{x})$. In view of the change of variable formula, $p_{\theta}(\mathbf{x})$ can be derived from $\pi(\mathbf{z})$ and $\mathbf{f}_{\theta}(\mathbf{z})$ as follows:

$$p_{\theta}(\mathbf{x}) = \pi(\mathbf{f}_{\theta}^{-1}(\mathbf{x})) |\det \mathbf{J}_{\mathbf{f}_{\theta}^{-1}}(\mathbf{x})|, \quad (2.10)$$

where we use $\mathbf{J}_{\mathbf{f}}$ to represent the Jacobian of function \mathbf{f} .

Since $\pi(\mathbf{z})$ is a normalized distribution, Eq. (2.10) ensures that $p_{\theta}(\mathbf{x})$ is also properly normalized. Assuming $|\det \mathbf{J}_{\mathbf{f}_{\theta}^{-1}}(\mathbf{x})|$ is tractable, normalizing flows can evaluate the exact probability of any data point as in Eq. (2.10), and consequently allow efficient maximum likelihood training. For data generation, one can first draw a sample $\hat{\mathbf{z}} \sim \pi(\mathbf{z})$, then feed it into the invertible function \mathbf{f}_{θ} to directly produce a sample $\hat{\mathbf{x}} = \mathbf{f}_{\theta}(\hat{\mathbf{z}}) \sim p_{\theta}(\mathbf{x})$. However, parameterizing an invertible function with a tractable determinant of the Jacobian, as required in normalizing flows, is highly non-trivial for deep neural networks. As a result, we have to resort to special neural network structures, such as coupling and autoregressive layers

[DKB15; DSB17; Kin+16; PPM17; Pap+19], to build powerful normalizing flow models.

Variational Autoencoders

Similar to autoregressive models and normalizing flows, variational autoencoders (VAEs [KW14; RMW14a]) enjoy exact normalization by constructing the probability distribution in a restricted way. A VAE introduces an auxiliary random variable \mathbf{z} , called the latent variable, to facilitate modeling the distribution of \mathbf{x} . It is comprised of three components: a prior distribution $p(\mathbf{z})$ that allows fast sampling and efficient probability evaluation, an “encoder” distribution $q_\phi(\mathbf{z} \mid \mathbf{x})$ with parameter ϕ , and a “decoder” distribution $p_\theta(\mathbf{x} \mid \mathbf{z})$, with parameter θ . VAEs construct a normalized probability distribution $p_\theta(\mathbf{x})$ as follows:

$$p_\theta(\mathbf{x}) = \int p(\mathbf{z})p_\theta(\mathbf{x} \mid \mathbf{z}) d\mathbf{z}.$$

Intuitively, $p_\theta(\mathbf{x})$ is an infinite mixture of distributions, where $p(\mathbf{z})$ is the mixture coefficient, and $p_\theta(\mathbf{x} \mid \mathbf{z})$ is the mixture component. Clearly, $p_\theta(\mathbf{x})$ is automatically normalized as long as both $p(\mathbf{z})$ and $p_\theta(\mathbf{x} \mid \mathbf{z})$ are normalized. Since $p(\mathbf{z})$ is already a (normalized) prior distribution, VAEs parameterize $p_\theta(\mathbf{x} \mid \mathbf{z})$ and $q_\phi(\mathbf{z} \mid \mathbf{x})$ as special parametric statistical distributions, such as exponential families, with deep neural network backbones to ensure the normalization of $p_\theta(\mathbf{x})$. The encoder $q_\phi(\mathbf{z} \mid \mathbf{x})$ is particularly useful for maximum likelihood training and probability evaluation. In fact, the log-likelihood of VAEs can be bounded from below, as given by

$$\begin{aligned} \log p_\theta(\mathbf{x}) &= \log \int p(\mathbf{z})p_\theta(\mathbf{x} \mid \mathbf{z}) d\mathbf{z} = \log \int q_\phi(\mathbf{z} \mid \mathbf{x}) \frac{p(\mathbf{z})p_\theta(\mathbf{x} \mid \mathbf{z})}{q_\phi(\mathbf{z} \mid \mathbf{x})} d\mathbf{z} \\ &\stackrel{(i)}{\geq} \int q_\phi(\mathbf{z} \mid \mathbf{x}) \log \frac{p(\mathbf{z})p_\theta(\mathbf{x} \mid \mathbf{z})}{q_\phi(\mathbf{z} \mid \mathbf{x})} d\mathbf{z} \\ &= \mathbb{E}_{q_\phi(\mathbf{z} \mid \mathbf{x})} [\log p_\theta(\mathbf{x} \mid \mathbf{z})] - \text{KL}(q_\phi(\mathbf{z} \mid \mathbf{x}) \parallel p(\mathbf{z})), \end{aligned} \quad (2.11)$$

where (i) is due to Jensen’s inequality. Here the expectation over $q_\phi(\mathbf{z} \mid \mathbf{x})$ can be estimated with Monte Carlo, and the KL divergence term can often be evaluated in closed form. We can then evaluate the lower bound of $p_\theta(\mathbf{x})$ by plugging $p(\mathbf{z})$, $p_\theta(\mathbf{x} \mid \mathbf{z})$ and $q_\phi(\mathbf{z} \mid \mathbf{x})$ into Eq. (2.11). For maximum likelihood training, we optimize for both θ and ϕ by maximizing the lower bound in Eq. (2.11) on every data instance $\{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\} \stackrel{\text{i.i.d.}}{\sim} p_{\text{data}}(\mathbf{x})$, giving

rise to the following objective¹:

$$\max_{\phi, \theta} \frac{1}{N} \sum_{i=1}^N \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x}_i)} [\log p_\theta(\mathbf{x}_i | \mathbf{z})] - \text{KL}(q_\phi(\mathbf{z} | \mathbf{x}_i) \| p(\mathbf{z})).$$

We use ancestral sampling to generate samples from $p_\theta(\mathbf{x})$. Specifically, we first draw a sample $\hat{\mathbf{z}} \sim p(\mathbf{z})$, then feed it into the decoder to obtain $\hat{\mathbf{x}} \sim p_\theta(\mathbf{x} | \hat{\mathbf{z}})$. The resulting $\hat{\mathbf{x}}$ is thus the desired sample from $p_\theta(\mathbf{x})$.

In summary, VAEs construct an exactly normalized distribution $p_\theta(\mathbf{x})$ at the cost of architectural constraints imposed on the encoder distribution $q_\phi(\mathbf{z} | \mathbf{x})$ and decoder distribution $p_\theta(\mathbf{x} | \mathbf{z})$, and only give approximate probability values.

2.2.3 Modeling Only the Data Generating Process

The challenge of normalization originates from modeling probability density/mass functions, thus may be avoided if we take alternative ways to represent the probability distribution. In this section, we introduce an example called generative adversarial networks (GANs [Goo+14]), a family of generative models that directly model the sampling process of a probability distribution, sidestepping the difficulty of normalization completely.

In GANs, the data generating process is modeled as a two-step procedure. First, we sample a latent vector $\mathbf{z} \in \mathbb{R}^m$ from a prior distribution $p(\mathbf{z})$ which typically has a simple analytical form (*e.g.*, standard Gaussian distributions or uniform distributions) and allows efficient sample generation. Next, we transform this latent vector through a deterministic function $f_\theta : \mathbb{R}^m \rightarrow \mathbb{R}^n$ to obtain the sample $\mathbf{x} = f_\theta(\mathbf{z}) \in \mathbb{R}^n$, where θ denotes model parameters. The distribution of \mathbf{x} , denoted as $p_\theta(\mathbf{x})$, can be implicitly defined from $p(\mathbf{z})$ and f_θ , but there is no need to directly work with it and therefore normalization is not a concern for GANs.

For training the deterministic mapping f_θ , GANs make use of an auxiliary model $g_\phi : \mathbb{R}^n \rightarrow [0, 1]$ called the critic, where ϕ denotes corresponding model parameters. The auxiliary model is tasked with discerning real data samples in the dataset from synthetic, “fake” samples produced by the deterministic mapping, whereas the mapping strives to produce samples that are hard for the critic to tell apart from true data samples. The critic and the deterministic mapping therefore compete with each other during training, resulting

¹The optimization procedure is more complicated than described here, involving reparameterization tricks and variance reduction techniques, for which we refer readers to [KW14; RMW14a; MG14].

in a two-player adversarial game. In particular, the training objective is given by

$$\min_{\theta} \max_{\phi} \mathbb{E}_{p_{\text{data}}(\mathbf{x})} [\log q_{\phi}(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim p(\mathbf{z})} [\log(1 - q_{\phi}(f_{\theta}(\mathbf{z})))], \quad (2.12)$$

where the expectation over $p_{\text{data}}(\mathbf{x})$ can be approximated with the empirical mean over the training dataset, and the expectation over $p(\mathbf{z})$ can be easily estimated with Monte Carlo. In fact, Eq. (2.12) is an approximation to a certain type of f -divergences (*cf.*, Section 2.1.2) between $p_{\text{data}}(\mathbf{x})$ and $p_{\theta}(\mathbf{x})$. Suppose q_{ϕ} is sufficiently powerful and the inner optimization of Eq. (2.12) can be done exactly, Goodfellow et al. [Goo+14] shows that Eq. (2.12) is equivalent to

$$\min_{\theta} 2D_{\text{JS}}(p_{\text{data}}(\mathbf{x}) \parallel p_{\theta}(\mathbf{x})) - \log 4,$$

where $D_{\text{JS}}(\cdot \parallel \cdot)$ denotes the Jensen-Shannon divergence (which is one type of f -divergences). As a result, the training objective Eq. (2.12) encourages the implicit model distribution $p_{\theta}(\mathbf{x})$ to be maximally close to the data distribution $p_{\text{data}}(\mathbf{x})$. It is also possible to train GANs with integral probability metrics (*cf.*, Section 2.1.2) like the Wasserstein distance [ACB17].

By construction, the data generating procedure of GANs is very efficient, involving just one evaluation of the deterministic mapping. The models of f_{θ} and g_{ϕ} do not require special structures to parameterize and hence can be built with flexible deep neural networks. However, unlike other families of generative models, GANs are unable to provide probability values since they do not directly model any probability distribution. This makes GANs unsuitable for certain applications that require probability evaluation, such as lossless compression [TBB19; Kin+21] and generative classification [Gra+20; Zim+21]. Moreover, the training procedure in Eq. (2.12) involves solving an adversarial game, which requires advanced optimization techniques that often suffer from stability issues [Met+16].

2.3 Score Functions and Score Models

A *Stein score function* [Ste72; HD05; LLJ16] is defined as the gradient of the log density of a probability distribution. Specifically, for a probability density function $p(\mathbf{x})$, the

corresponding Stein score function $\mathbf{s}(\mathbf{x})$ is given by

$$\mathbf{s}(\mathbf{x}) := \nabla_{\mathbf{x}} \log p(\mathbf{x}).$$

Beware the difference from a Fisher score function, which is defined as the gradient of the log density with respect to *model parameters*. By contrast, A Stein score function takes the derivative with respect to the *random variable*. For brevity, we henceforth refer to “Stein score function” as “score” or “score function” for short.

Given a probability density function, one can uniquely determine its score function by taking derivatives according to the definition. Conversely, given a score function, one can obtain the corresponding density function by noting

$$\log p(\mathbf{x}) = \log p(\mathbf{x}_0) + \int_0^1 \mathbf{s}(\mathbf{x}_0 + t(\mathbf{x} - \mathbf{x}_0))^T (\mathbf{x} - \mathbf{x}_0) dt,$$

where $\log p(\mathbf{x}_0)$ can be determined from the normalization constraint:

$$\int p(\mathbf{x}) d\mathbf{x} = 1.$$

In consequence, score functions preserve the same amount of information as probability density functions do when used to represent probability distributions.

Being oblivious to normalization is the key advantage of score functions over density functions. Specifically, suppose $\tilde{p}(\mathbf{x})$ is an unnormalized probability density, *i.e.*, $\int \tilde{p}(\mathbf{x}) d\mathbf{x} = Z \neq 1$. The score function is given by

$$\mathbf{s}(\mathbf{x}) = \nabla_{\mathbf{x}} \log \frac{\tilde{p}(\mathbf{x})}{Z} = \nabla_{\mathbf{x}} \log \tilde{p}(\mathbf{x}) - \nabla_{\mathbf{x}} \log Z = \nabla_{\mathbf{x}} \log \tilde{p}(\mathbf{x}),$$

which is oblivious to the normalizing constant Z . Since there is no need of calculating the normalizing constant for evaluating score functions, it is thus much easier to model them with flexible deep neural networks.

From now on, we call the model of a score function the *score model*, which is denoted by $\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x})$, where $\boldsymbol{\theta}$ represents model parameters. The only constraint for a score model is that it must be the gradient of a scalar-valued function, or in other words, it must represent a conservative vector field. This can be enforced by parameterizing an energy function $E_{\boldsymbol{\theta}}(\mathbf{x})$ (*cf.*, Section 2.2.1) first with a deep neural network, then construct the score model with

$s_{\theta}(\mathbf{x}) = -\nabla_{\mathbf{x}} E_{\theta}(\mathbf{x})$. Computing the gradient of a deep neural network is often efficient with backpropagation [Hin02] or automatic differentiation [Mar+15; Pas+17; Bra+18]. Therefore, a score model constructed in this way is fast to evaluate. Moreover, we shall see in later parts of this dissertation that the requirement of modeling conservative vector fields can be lifted without affecting model training, sample generation, or probability evaluation. In this case, the score model can be parameterized directly with a flexible deep neural network without additional constraints.

In the following chapters, we will present *score-based generative models*, a family of generative techniques based on modeling the score function of the data distribution. As discussed already, score models completely sidestep the challenge of normalization, and thus can leverage flexible deep neural networks to build powerful generative models. Specifically, we will show how to train score models with stable training algorithms, how to produce high quality samples from score models, and how to evaluate probability values accurately with score models. This demonstrates that score-based generative models can overcome various drawbacks faced by existing generative models, and hence represent a promising research avenue towards better data generation.

Part I

Learning Score Models from Data

Chapter 3

Learning Score Models with Random Projections

How can we train flexible score models from high-dimensional data? The standard technique is score matching [HD05]. Score matching is a well-established method for estimating unnormalized statistical models, and more generally, score models. However, it has been so far limited to simple, shallow models or low-dimensional data, due to the difficulty of computing the Hessian diagonals of log-density functions. We show this difficulty can be mitigated by projecting the scores onto random vectors before comparing their differences. This objective, called *sliced score matching*, only involves Hessian-vector products, which can be easily implemented using reverse-mode automatic differentiation. Therefore, sliced score matching is amenable to more complex models and higher dimensional data than score matching. Theoretically, we prove the consistency and asymptotic normality of sliced score matching estimators. Moreover, we demonstrate that sliced score matching can be used to learn deep score estimators for implicit distributions. In our experiments, we show sliced score matching can learn deep energy-based models and score models effectively, and can produce accurate score estimates for applications such as variational inference with implicit distributions and training Wasserstein Auto-Encoders [Tol+18].

This chapter was previously published as [Son+19].

3.1 Introduction

Score matching [HD05] is particularly suitable for learning unnormalized statistical models (a.k.a., energy based models). It is based on minimizing the distance between the derivatives of the log-density functions (a.k.a., *scores*) of the data and model distributions. Unlike maximum likelihood estimation (MLE), the objective of score matching only depends on the scores, which are oblivious to the (usually) intractable partition functions. However, score matching requires the computation of the diagonal elements of the Hessian of the model’s log-density function. This Hessian trace computation is generally expensive [MSS12], requiring a number of forward and backward propagations proportional to the data dimension. This severely limits its applicability to complex models parameterized by deep neural networks, such as deep energy-based models [LeC+06; Wen+19].

Several approaches have been proposed to alleviate this difficulty: Kingma and LeCun [KL10] propose approximate backpropagation for computing the trace of the Hessian; Martens et al. [MSS12] develop curvature propagation, a fast stochastic estimator for the trace in score matching; and Vincent [Vin11] transforms score matching to a denoising problem which avoids second-order derivatives. These methods have achieved some success, but may suffer from one or more of the following problems: inconsistent parameter estimation, large estimation variance, and cumbersome implementation.

To alleviate these problems, we propose sliced score matching, a variant of score matching that can scale to deep unnormalized models and high dimensional data. The key intuition is that instead of directly matching the high-dimensional scores, we match their projections along random directions. Theoretically, we show that under some regularity conditions, sliced score matching is a well-defined statistical estimation criterion that yields consistent and asymptotically normal parameter estimates. Moreover, compared to the methods of Kingma and LeCUN [KL10] and Martens et al. [MSS12], whose implementations require customized backpropagation for deep networks, sliced score matching only involves Hessian-vector products, thus can be easily and efficiently implemented in frameworks such as TensorFlow [Mar+15] and PyTorch [Pas+17].

Beyond training unnormalized models, sliced score matching can also be naturally adapted as an objective for estimating the score function of a data generating distribution [SHS14; Str+15] by training a score function model parameterized by deep neural networks. This observation enables many new applications of sliced score matching. For example,

we show that it can be used to provide accurate score estimates needed for variational inference with implicit distributions [Hus17] and learning Wasserstein Auto-Encoders (WAE, [Tol+18]).

Finally, we evaluate the performance of sliced score matching on learning unnormalized statistical models (density estimation) and estimating score functions of a data generating process (score estimation). For density estimation, experiments on deep kernel exponential families [Wen+19] and NICE flow models [DKB15] show that our method is either more scalable or more accurate than existing score matching variants.

For score estimation, our method improves the performance of variational auto-encoders (VAE) with implicit encoders, and can train WAEs without a discriminator or MMD loss by directly optimizing the KL divergence between aggregated posteriors and the prior. In both situations we outperformed kernel-based score estimators [LT18; SSZ18] by achieving better test likelihoods and better sample quality in image generation.

3.2 Background

Given i.i.d. samples $\{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\} \subset \mathbb{R}^D$ from a data distribution $p_d(\mathbf{x})$, our task is to learn an unnormalized density, $\tilde{p}_m(\mathbf{x}; \boldsymbol{\theta})$, where $\boldsymbol{\theta}$ is from some parameter space Θ . The model's partition function is denoted as $Z_{\boldsymbol{\theta}}$, which is assumed to be existent but intractable. Let $p_m(\mathbf{x}; \boldsymbol{\theta})$ be the normalized density determined by our model, we have

$$p_m(\mathbf{x}; \boldsymbol{\theta}) = \frac{\tilde{p}_m(\mathbf{x}; \boldsymbol{\theta})}{Z_{\boldsymbol{\theta}}}, \quad Z_{\boldsymbol{\theta}} = \int \tilde{p}_m(\mathbf{x}; \boldsymbol{\theta}) d\mathbf{x}.$$

For convenience, we denote the score functions of p_m and p_d as $\mathbf{s}_m(\mathbf{x}; \boldsymbol{\theta}) := \nabla_{\mathbf{x}} \log p_m(\mathbf{x}; \boldsymbol{\theta})$ and $\mathbf{s}_d(\mathbf{x}) := \nabla_{\mathbf{x}} \log p_d(\mathbf{x})$ respectively. Note that since $\log p_m(\mathbf{x}; \boldsymbol{\theta}) = \log \tilde{p}_m(\mathbf{x}; \boldsymbol{\theta}) - \log Z_{\boldsymbol{\theta}}$, we immediately conclude that $\mathbf{s}_m(\mathbf{x}; \boldsymbol{\theta})$ does not depend on the typically intractable partition function $Z_{\boldsymbol{\theta}}$.

3.2.1 Score Matching

Learning unnormalized models with maximum likelihood estimation (MLE) can be difficult due to the intractable partition function $Z_{\boldsymbol{\theta}}$. To avoid this, score matching [HD05] minimizes

the Fisher divergence between p_d and $p_m(\cdot, \boldsymbol{\theta})$, which is defined as

$$L(\boldsymbol{\theta}) := \frac{1}{2} \mathbb{E}_{p_d} [\|\mathbf{s}_m(\mathbf{x}; \boldsymbol{\theta}) - \mathbf{s}_d(\mathbf{x})\|_2^2]. \quad (3.1)$$

Since $\mathbf{s}_m(\mathbf{x}; \boldsymbol{\theta})$ does not involve $Z_{\boldsymbol{\theta}}$, the Fisher divergence does not depend on the intractable partition function. However, Eq. (3.1) is still not readily usable for learning unnormalized models, as we only have samples and do not have access to the score function of the data $\mathbf{s}_d(\mathbf{x})$.

By applying integration by parts, Hyvärinen [HD05] shows that $L(\boldsymbol{\theta})$ can be written as $L(\boldsymbol{\theta}) = J(\boldsymbol{\theta}) + C$ (*cf.*, Theorem 1 in [HD05]), where

$$J(\boldsymbol{\theta}) := \mathbb{E}_{p_d} \left[\text{tr}(\nabla_{\mathbf{x}} \mathbf{s}_m(\mathbf{x}; \boldsymbol{\theta})) + \frac{1}{2} \|\mathbf{s}_m(\mathbf{x}; \boldsymbol{\theta})\|_2^2 \right], \quad (3.2)$$

C is a constant that does not depend on $\boldsymbol{\theta}$, $\text{tr}(\cdot)$ denotes the trace of a matrix, and

$$\nabla_{\mathbf{x}} \mathbf{s}_m(\mathbf{x}; \boldsymbol{\theta}) = \nabla_{\mathbf{x}}^2 \log \tilde{p}_m(\mathbf{x}; \boldsymbol{\theta}) \quad (3.3)$$

is the Hessian of the log-density function. The constant can be ignored and the following unbiased estimator of the remaining terms is used to train $\tilde{p}_m(\mathbf{x}; \boldsymbol{\theta})$:

$$\hat{J}(\boldsymbol{\theta}; \mathbf{x}_1^N) := \frac{1}{N} \sum_{i=1}^N \left[\text{tr}(\nabla_{\mathbf{x}} \mathbf{s}_m(\mathbf{x}_i; \boldsymbol{\theta})) + \frac{1}{2} \|\mathbf{s}_m(\mathbf{x}_i; \boldsymbol{\theta})\|_2^2 \right]$$

where \mathbf{x}_1^N is a shorthand used throughout the paper for a collection of N data points $\{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\}$ sampled i.i.d. from p_d , and $\nabla_{\mathbf{x}} \mathbf{s}_m(\mathbf{x}_i; \boldsymbol{\theta})$ denotes the Hessian of $\log \tilde{p}_m(\mathbf{x}; \boldsymbol{\theta})$ evaluated at \mathbf{x}_i .

Computational difficulty While the score matching objective $\hat{J}(\boldsymbol{\theta}; \mathbf{x}_1^N)$ avoids the computation of $Z_{\boldsymbol{\theta}}$ for unnormalized models, it introduces a new computational difficulty: computing the trace of the Hessian of a log-density function, $\nabla_{\mathbf{x}}^2 \log \tilde{p}_m$. A naïve approach of computing the trace of the Hessian requires D times more backward passes than computing the gradient $\mathbf{s}_m = \nabla_{\mathbf{x}} \log \tilde{p}_m$ (see Algorithm B.1 in Appendix B.1). For example, the trace could be computed by applying backpropagation D times to \mathbf{s}_m to get each diagonal term of $\nabla_{\mathbf{x}}^2 \log \tilde{p}_m$ sequentially. In practice, D can be many thousands, which can render score matching too slow for practical purposes. Moreover, Martens et al. [MSS12] argue from

a theoretical perspective that it is unlikely that there exists an algorithm for computing the diagonal of the Hessian defined by an arbitrary computation graph within a constant number of forward and backward passes.

3.2.2 Score Estimation for Implicit Distributions

Besides parameter estimation in unnormalized models, score matching can also be used to estimate scores of *implicit distributions*, which are distributions that have a tractable sampling process but without a tractable density. For example, the distribution of random samples from the generator of a GAN [Goo+14] is an implicit distribution. Implicit distributions can arise in many more situations such as the marginal distribution of a non-conjugate model [Sun+19], and models defined by complex simulation processes [TRB17]. In many cases learning and inference become intractable due to the need of optimizing an objective that involves the intractable density.

In these cases, directly estimating the score function $s_q(\mathbf{x}) = \nabla_{\mathbf{x}} \log q_{\boldsymbol{\theta}}(\mathbf{x})$ based on i.i.d. samples from an implicit density $q_{\boldsymbol{\theta}}(\mathbf{x})$ can be useful. For example, suppose our learning problem involves optimizing the entropy $H(q_{\boldsymbol{\theta}}(\mathbf{x}))$ of an implicit distribution. This situation is common when dealing with variational free energies [KW14]. Suppose $\mathbf{x} \sim q_{\boldsymbol{\theta}}$ can be reparameterized as $\mathbf{x} = g_{\boldsymbol{\theta}}(\boldsymbol{\epsilon})$, where $\boldsymbol{\epsilon}$ is a simple random variable independent of $\boldsymbol{\theta}$, such as a standard normal, and $g_{\boldsymbol{\theta}}$ is a deterministic mapping. We can write the gradient of the entropy with respect to $\boldsymbol{\theta}$ as

$$\begin{aligned}\nabla_{\boldsymbol{\theta}} H(q_{\boldsymbol{\theta}}) &:= -\nabla_{\boldsymbol{\theta}} \mathbb{E}_{q_{\boldsymbol{\theta}}(\mathbf{x})} [\log q_{\boldsymbol{\theta}}(\mathbf{x})] \\ &= -\nabla_{\boldsymbol{\theta}} \mathbb{E}_{p(\boldsymbol{\epsilon})} [\log q_{\boldsymbol{\theta}}(g_{\boldsymbol{\theta}}(\boldsymbol{\epsilon}))] \\ &= -\mathbb{E}_{p(\boldsymbol{\epsilon})} [\nabla_{\mathbf{x}} \log q_{\boldsymbol{\theta}}(\mathbf{x})|_{\mathbf{x}=g_{\boldsymbol{\theta}}(\boldsymbol{\epsilon})} \nabla_{\boldsymbol{\theta}} g_{\boldsymbol{\theta}}(\boldsymbol{\epsilon})],\end{aligned}$$

where $\nabla_{\boldsymbol{\theta}} g_{\boldsymbol{\theta}}(\boldsymbol{\epsilon})$ is usually easy to compute. The score $\nabla_{\mathbf{x}} \log q_{\boldsymbol{\theta}}(\mathbf{x})$ is intractable but can be approximated by score estimation.

Score matching is an attractive solution for score estimation Eq. (3.1) naturally serves as an objective to measure the difference between the a trainable score function and the score of a data generating process. We will discuss this in more detail in Section 3.3.2 and mention some other approaches of score estimation in Section 3.5.2.

3.3 Density & Score Estimation with Sliced Score Matching

3.3.1 Sliced Score Matching

We observe that one dimensional problems are usually much easier to solve than high dimensional ones. Inspired by the idea of Sliced Wasserstein distance [Rab+11], we consider projecting $s_d(\mathbf{x})$ and $s_m(\mathbf{x}; \boldsymbol{\theta})$ onto some random direction \mathbf{v} and propose to compare their average difference along that random direction. More specifically, we consider the following objective as a replacement of the Fisher divergence $L(\boldsymbol{\theta})$ in Eq. (3.1):

$$L(\boldsymbol{\theta}; p_{\mathbf{v}}) := \frac{1}{2} \mathbb{E}_{p_{\mathbf{v}}} \mathbb{E}_{p_d} \left[(\mathbf{v}^\top s_m(\mathbf{x}; \boldsymbol{\theta}) - \mathbf{v}^\top s_d(\mathbf{x}))^2 \right]. \quad (3.4)$$

Here $\mathbf{v} \sim p_{\mathbf{v}}$ and $\mathbf{x} \sim p_d$ are independent, and we require $\mathbb{E}_{p_{\mathbf{v}}}[\mathbf{v}\mathbf{v}^\top] > 0$ and $\mathbb{E}_{p_{\mathbf{v}}}[\|\mathbf{v}\|_2^2] < \infty$. Many examples of $p_{\mathbf{v}}$ satisfy these requirements. For instance, $p_{\mathbf{v}}$ can be a multivariate standard normal ($\mathcal{N}(0, I_D)$), a multivariate Rademacher distribution (the uniform distribution over $\{\pm 1\}^D$), or a uniform distribution on the hypersphere \mathbb{S}^{D-1} (recall that D refers to the dimension of \mathbf{x}).

To eliminate the dependence of $L(\boldsymbol{\theta}; p_{\mathbf{v}})$ on $s_d(\mathbf{x})$, we use integration by parts as in score matching (cf., the equivalence between Eq. (3.1) and Eq. (3.2)). Defining

$$J(\boldsymbol{\theta}; p_{\mathbf{v}}) := \mathbb{E}_{p_{\mathbf{v}}} \mathbb{E}_{p_d} [\mathbf{v}^\top \nabla_{\mathbf{x}} s_m(\mathbf{x}; \boldsymbol{\theta}) \mathbf{v} + \frac{1}{2} (\mathbf{v}^\top s_m(\mathbf{x}; \boldsymbol{\theta}))^2], \quad (3.5)$$

the equivalence is summarized in the following theorem.

Theorem 3.1. *Under some regularity conditions (Assumption A.1–Assumption A.3 in Appendix A.1.2), we have*

$$L(\boldsymbol{\theta}; p_{\mathbf{v}}) = J(\boldsymbol{\theta}; p_{\mathbf{v}}) + C, \quad (3.6)$$

where C is a constant w.r.t. $\boldsymbol{\theta}$.

Other than our requirements on $p_{\mathbf{v}}$, the assumptions are exactly the same as in Theorem 1 of [HD05]. We advise the interested readers to read Appendix A.1.2 for technical statements of the assumptions and a rigorous proof of the theorem.

In practice, given a dataset \mathbf{x}_1^N , we draw M projection vectors independently for each point \mathbf{x}_i from $p_{\mathbf{v}}$. The collection of all such vectors $\{\mathbf{v}_{ij}\}_{1 \leq i \leq N, 1 \leq j \leq M}$ are abbreviated as

\mathbf{v}_{11}^{NM} . We then use the following unbiased estimator of $J(\boldsymbol{\theta}; p_{\mathbf{v}})$

$$\hat{J}(\boldsymbol{\theta}; \mathbf{x}_1^N, \mathbf{v}_{11}^{NM}) := \frac{1}{N} \frac{1}{M} \sum_{i=1}^N \sum_{j=1}^M \mathbf{v}_{ij}^T \nabla_{\mathbf{x}} s_m(\mathbf{x}_i; \boldsymbol{\theta}) \mathbf{v}_{ij} + \frac{1}{2} \left(\mathbf{v}_{ij}^T s_m(\mathbf{x}_i; \boldsymbol{\theta}) \right)^2. \quad (3.7)$$

Note that when $p_{\mathbf{v}}$ is a multivariate standard normal or multivariate Rademacher distribution, we have $\mathbb{E}_{p_{\mathbf{v}}}[(\mathbf{v}^T s_m(\mathbf{x}; \boldsymbol{\theta}))^2] = \|s_m(\mathbf{x}; \boldsymbol{\theta})\|_2^2$, in which case the second term of $J(\boldsymbol{\theta}; p_{\mathbf{v}})$ can be integrated analytically, and may lead to an estimator with reduced variance:

$$\hat{J}_{\text{vr}}(\boldsymbol{\theta}; \mathbf{x}_1^N, \mathbf{v}_{11}^{NM}) := \frac{1}{N} \frac{1}{M} \sum_{i=1}^N \sum_{j=1}^M \mathbf{v}_{ij}^T \nabla_{\mathbf{x}} s_m(\mathbf{x}_i; \boldsymbol{\theta}) \mathbf{v}_{ij} + \frac{1}{2} \|s_m(\mathbf{x}_i; \boldsymbol{\theta})\|_2^2. \quad (3.8)$$

Empirically, we do find that \hat{J}_{vr} has better performance than \hat{J} . We refer to this version as sliced score matching with variance reduction (SSM-VR). In fact, we can leverage $\mathbb{E}_{p_{\mathbf{v}}}[(\mathbf{v}^T s_m(\mathbf{x}; \boldsymbol{\theta}))^2] = \|s_m(\mathbf{x}; \boldsymbol{\theta})\|_2^2$ to create a control variate for guaranteed reduction of variance (Appendix B.1.2). \hat{J}_{vr} is also closely related to Hutchinson's trace estimator [Hut89], which we will analyze later in Section 3.4.3.

For sliced score matching, the second derivative term $\mathbf{v}^T \nabla_{\mathbf{x}} s_m(\mathbf{x}; \boldsymbol{\theta}) \mathbf{v}$ is much less computationally expensive than $\text{tr}(\nabla_{\mathbf{x}} s_m(\mathbf{x}; \boldsymbol{\theta}))$. Using auto-differentiation systems that support higher order gradients, we can compute it using two gradient operations for a single \mathbf{v} , as shown in Algorithm 3.1. Similarly, when there are M \mathbf{v} 's, the total number of gradient operations is $M + 1$. In contrast, assuming the dimension of \mathbf{x} is D and $D \gg M$, we typically need $D + 1$ gradient operations to compute $\text{tr}(\nabla_{\mathbf{x}} s_m(\mathbf{x}; \boldsymbol{\theta}))$ because each diagonal entry of $\nabla_{\mathbf{x}} s_m(\mathbf{x}; \boldsymbol{\theta})$ needs to be computed separately (see Algorithm B.1 in Appendix B.1).

Algorithm 3.1 Sliced Score Matching

Require: $\tilde{p}_m(\cdot; \boldsymbol{\theta}), \mathbf{x}, \mathbf{v}$

- 1: $s_m(\mathbf{x}; \boldsymbol{\theta}) \leftarrow \text{grad}(\log \tilde{p}_m(\mathbf{x}; \boldsymbol{\theta}), \mathbf{x})$
- 2: $\mathbf{v}^T \nabla_{\mathbf{x}} s_m(\mathbf{x}; \boldsymbol{\theta}) \leftarrow \text{grad}(\mathbf{v}^T s_m(\mathbf{x}; \boldsymbol{\theta}), \mathbf{x})$
- 3: $J \leftarrow \frac{1}{2} (\mathbf{v}^T s_m(\mathbf{x}; \boldsymbol{\theta}))^2$ (or $J \leftarrow \frac{1}{2} \|s_m(\mathbf{x}; \boldsymbol{\theta})\|_2^2$)
- 4: $J \leftarrow J + \mathbf{v}^T \nabla_{\mathbf{x}} s_m(\mathbf{x}; \boldsymbol{\theta}) \mathbf{v}$

return J

In practice, we can tune M to trade off variance and computational cost. In our experiments, we find that oftentimes $M = 1$ is already a good choice.

3.3.2 Sliced Score Estimation

As mentioned in Section 3.2.2, the task of score estimation is to estimate $\nabla_{\mathbf{x}} \log q(\mathbf{x})$ at some test point \mathbf{x} , given a set of samples $\mathbf{x}_1^N \stackrel{\text{i.i.d.}}{\sim} q(\mathbf{x})$. In what follows, we show how our sliced score matching objective $\hat{J}(\boldsymbol{\theta}; \mathbf{x}_1^N, \mathbf{v}_{11}^{NM})$ can be straightforwardly adapted for this task.

We propose to use a vector-valued deep neural network $\mathbf{h}(\mathbf{x}; \boldsymbol{\theta}) : \mathbb{R}^D \rightarrow \mathbb{R}^D$ as our score model. Then, substituting \mathbf{h} into $\hat{J}(\boldsymbol{\theta}; \mathbf{x}_1^N, \mathbf{v}_{11}^{NM})$ for s_m , we get

$$\frac{1}{N} \frac{1}{M} \sum_{i=1}^N \sum_{j=1}^M \mathbf{v}_{ij}^\top \nabla_{\mathbf{x}} \mathbf{h}(\mathbf{x}_i; \boldsymbol{\theta}) \mathbf{v}_{ij} + \frac{1}{2} (\mathbf{v}_{ij}^\top \mathbf{h}(\mathbf{x}_i; \boldsymbol{\theta}))^2,$$

and we can optimize the objective to get $\hat{\boldsymbol{\theta}}$. Afterwards, $\mathbf{h}(\mathbf{x}; \hat{\boldsymbol{\theta}})$ can be used as an approximation to $\nabla_{\mathbf{x}} \log q(\mathbf{x})$.¹

Using a similar argument of integration by parts (*cf.*, Eqs. (3.4) to (3.6)), we have

$$\mathbb{E}_{p_{\mathbf{v}}} \mathbb{E}_{p_d} \left[\mathbf{v}^\top \nabla_{\mathbf{x}} \mathbf{h}(\mathbf{x}; \boldsymbol{\theta}) \mathbf{v} + \frac{1}{2} (\mathbf{v}^\top \mathbf{h}(\mathbf{x}; \boldsymbol{\theta}))^2 \right] = \frac{1}{2} \mathbb{E}_{p_{\mathbf{v}}} \mathbb{E}_{p_d} [(\mathbf{v}^\top \mathbf{h}(\mathbf{x}; \boldsymbol{\theta}) - \mathbf{v}^\top \nabla_{\mathbf{x}} \log q(\mathbf{x}))^2] + C,$$

which implies that minimizing $\hat{J}(\boldsymbol{\theta}; \mathbf{x}_1^N, \mathbf{v}_{11}^{NM})$ with $s_m(\mathbf{x}; \boldsymbol{\theta})$ replaced by $\mathbf{h}(\mathbf{x}; \boldsymbol{\theta})$ is approximately minimizing the average projected difference between $\mathbf{h}(\mathbf{x}; \boldsymbol{\theta})$ and $\nabla_{\mathbf{x}} \log q(\mathbf{x})$. Hence, $\mathbf{h}(\mathbf{x}; \hat{\boldsymbol{\theta}})$ should be close to $\nabla_{\mathbf{x}} \log q(\mathbf{x})$ and can serve as a score estimator.

3.4 Theoretical Analysis

In this section, we present several theoretical results to justify sliced score matching as a principled objective. We also discuss the connection of sliced score matching to other methods. For readability, we will defer rigorous statements of assumptions and theorems to Appendix A.1.

3.4.1 Consistency

One important question to ask for any statistical estimation objective is whether the estimated parameter is consistent under reasonable assumptions. Our results confirm that

¹Note that $\mathbf{h}(\mathbf{x}; \boldsymbol{\theta})$ may not correspond to the gradient of any scalar function. For $\mathbf{h}(\mathbf{x}; \boldsymbol{\theta})$ to represent a gradient, one necessary condition is $\nabla_{\mathbf{x}} \times \mathbf{h}(\mathbf{x}; \boldsymbol{\theta}) = \mathbf{0}$ for all \mathbf{x} , which may not be satisfied by general networks. However, this is oblivious to the fact that $\mathbf{h}(\mathbf{x}; \hat{\boldsymbol{\theta}})$ will be close to $\nabla_{\mathbf{x}} \log q_{\boldsymbol{\theta}}(\mathbf{x})$ in ℓ_2 norm. As will be shown later, our argument based on integration by parts does not require $\mathbf{h}(\mathbf{x}; \boldsymbol{\theta})$ to be a gradient.

for any M , the objective $\hat{J}(\boldsymbol{\theta}; \mathbf{x}_1^N, \mathbf{v}_{11}^{NM})$ is consistent under suitable assumptions as $N \rightarrow \infty$.

We need several standard assumptions to prove the results rigorously. Let p_m be the normalized density induced by our unnormalized model \tilde{p}_m , which is assumed to be normalizable. First, we assume Θ is compact (Assumption A.6), and our model family $\{p_m(\mathbf{x}; \boldsymbol{\theta}) : \boldsymbol{\theta} \in \Theta\}$ is well-specified and identifiable (Assumption A.4). These are standard assumptions used for proving the consistency of MLE [Vaa98]. We also adopt the assumption in [HD05] that all densities are strictly positive (Assumption A.5). Finally, we assume that $p_m(\mathbf{x}; \boldsymbol{\theta})$ has some Lipschitz properties (Assumption A.7), and $p_{\mathbf{v}}$ has bounded higher-order moments (Assumption A.2, true for all $p_{\mathbf{v}}$ considered in the experiments). Then, we can prove the consistency of $\hat{\boldsymbol{\theta}}_{N,M} := \arg \min_{\boldsymbol{\theta} \in \Theta} \hat{J}(\boldsymbol{\theta}; \mathbf{x}_1^N, \mathbf{v}_{11}^{NM})$:

Theorem 3.2 (Consistency). *Assume the conditions of Theorem 3.1 are satisfied. Assume further that the assumptions discussed above hold. Let $\boldsymbol{\theta}^*$ be the true parameter of the data distribution. Then for every $M \in \mathbb{N}^+$,*

$$\hat{\boldsymbol{\theta}}_{N,M} \xrightarrow{P} \boldsymbol{\theta}^*, \quad N \rightarrow \infty.$$

Sketch of proof. We first prove that $J(\boldsymbol{\theta}; p_{\mathbf{v}}) = 0 \Leftrightarrow \boldsymbol{\theta} = \boldsymbol{\theta}^*$ (Lemma A.1 and Theorem 3.1). Then we prove the uniform convergence of $\hat{J}(\boldsymbol{\theta}; \mathbf{x}_1^N, \mathbf{v}_{11}^{NM})$ (Lemma A.3), which holds regardless of M . These two facts lead to consistency. For a complete proof, see Appendix A.1.3. \square

Remark 3.1. *In [HD05], the authors only showed that $J(\boldsymbol{\theta}) = 0 \Leftrightarrow \boldsymbol{\theta} = \boldsymbol{\theta}^*$, which leads to “local consistency” of score matching. This is a weaker notion of consistency compared to our settings.*

3.4.2 Asymptotic Normality

Asymptotic normality results can be very useful for approximate hypothesis testing and comparing different estimators. Below we show that $\hat{\boldsymbol{\theta}}_{N,M}$ is asymptotically normal when $N \rightarrow \infty$.

In addition to the assumptions in Section 3.4.1, we need an extra assumption to prove asymptotic normality. We require $p_m(\mathbf{x}; \boldsymbol{\theta})$ to have a stronger Lipschitz property (Assumption A.9).

For simplicity, we denote $\nabla_{\mathbf{x}} h(\mathbf{x})|_{\mathbf{x}=\mathbf{x}'}$ as $\nabla_{\mathbf{x}} h(\mathbf{x}')$, where $h(\cdot)$ is an arbitrary function. In the following, we will only show the asymptotic normality result for a specific $p_{\mathbf{v}}$. More

general results are in Appendix A.1.4.

Theorem 3.3 (Asymptotic normality, special case). *Assume the assumptions discussed above hold. If $p_{\mathbf{v}}$ is the multivariate Rademacher distribution, we have*

$$\sqrt{N}(\hat{\boldsymbol{\theta}}_{N,M} - \boldsymbol{\theta}^*) \xrightarrow{d} \mathcal{N}(0, \Sigma),$$

where

$$\Sigma := [\nabla_{\boldsymbol{\theta}}^2 J(\boldsymbol{\theta}^*)]^{-1} \left(\sum_{1 \leq i, j \leq D} V_{ij} + \frac{2}{M} \sum_{1 \leq i \neq j \leq D} W_{ij} \right) [\nabla_{\boldsymbol{\theta}}^2 J(\boldsymbol{\theta}^*)]^{-1}. \quad (3.9)$$

Here D is the dimension of data; V_{ij} and W_{ij} are p.s.d matrices depending on $p_m(\mathbf{x}; \boldsymbol{\theta}^*)$, and their definitions can be found in Appendix A.1.4.

Sketch of proof. Once we get the consistency (Theorem 3.2), the rest closely follows the proof of asymptotic normality of MLE [Vaa98]. A rigorous proof can be found in Appendix A.1.4. \square

Remark 3.2. As expected, larger M will lead to smaller asymptotic variance, as can be seen in Eq. (3.9).

Remark 3.3. As far as we know, there is no published proof of asymptotic normality for the standard (not sliced) score matching. However, by using the same techniques in our proofs, and under similar assumptions, we can conclude that the asymptotic variance of the score matching estimator is $[\nabla_{\boldsymbol{\theta}}^2 J(\boldsymbol{\theta}^*)]^{-1} \left(\sum_{ij} V_{ij} \right) [\nabla_{\boldsymbol{\theta}}^2 J(\boldsymbol{\theta}^*)]^{-1}$ (Corollary A.1), which will always be smaller than sliced score matching with multivariate Rademacher projections. However, the gap reduces when M increases.

3.4.3 Connection with Other Methods

Sliced score matching is widely connected to many other methods, and can be motivated from some different perspectives. Here we discuss a few of them.

Connection with NCE Noise contrastive estimation (NCE), proposed by [GH10], is another principle for training unnormalized statistical models. The method works by comparing $p_m(\mathbf{x}; \boldsymbol{\theta})$ with a noise distribution $p_n(\mathbf{x})$. We consider a special form of NCE

which minimizes the following objective

$$-\mathbb{E}_{p_d}[\log h(\mathbf{x}; \boldsymbol{\theta})] - \mathbb{E}_{p_n}[\log(1 - h(\mathbf{x}; \boldsymbol{\theta}))], \quad (3.10)$$

where $h(\mathbf{x}; \boldsymbol{\theta}) := \frac{p_m(\mathbf{x}; \boldsymbol{\theta})}{p_m(\mathbf{x}; \boldsymbol{\theta}) + p_m(\mathbf{x} - \mathbf{v}; \boldsymbol{\theta})}$, and we choose $p_n(\mathbf{x}) = p_d(\mathbf{x} + \mathbf{v})$. Assuming $\|\mathbf{v}\|_2$ to be small, Eq. (3.10) can be written as the following by Taylor expansion

$$\frac{1}{4} \mathbb{E}_{p_d} \left[\mathbf{v}^\top \nabla_{\mathbf{x}} \mathbf{s}_m(\mathbf{x}; \boldsymbol{\theta}) \mathbf{v} + \frac{1}{2} (\mathbf{s}_m(\mathbf{x}; \boldsymbol{\theta})^\top \mathbf{v})^2 \right] + 2 \log 2 + o(\|\mathbf{v}\|_2^2). \quad (3.11)$$

For derivation, see Proposition A.1 in Appendix A.1. A similar derivation can also be found in [GH11]. As a result of Eq. (3.11), if we choose some $p_{\mathbf{v}}$ and take the expectation of Eq. (3.10) w.r.t. $p_{\mathbf{v}}$, the objective will be equivalent to sliced score matching whenever $\|\mathbf{v}\|_2 \approx 0$.

Connection with Hutchinson's trick Hutchinson's trick [Hut89] is a stochastic algorithm to approximate $\text{tr}(\mathbf{A})$ for any square matrix \mathbf{A} . The idea is to choose a distribution of a random vector \mathbf{v} such that $\mathbb{E}_{p_{\mathbf{v}}}[\mathbf{v}\mathbf{v}^\top] = I$, and then we have $\text{tr}(\mathbf{A}) = \mathbb{E}_{p_{\mathbf{v}}}[\mathbf{v}^\top \mathbf{A} \mathbf{v}]$. Hence, using Hutchinson's trick, we can replace $\text{tr}(\nabla_{\mathbf{x}} \mathbf{s}_m(\mathbf{x}; \boldsymbol{\theta}))$ with $\mathbb{E}_{p_{\mathbf{v}}}[\mathbf{v}^\top \nabla_{\mathbf{x}} \mathbf{s}_m(\mathbf{x}; \boldsymbol{\theta}) \mathbf{v}]$ in the score matching objective $J(\boldsymbol{\theta})$. Then the finite sample objective of score matching becomes

$$\frac{1}{N} \sum_{i=1}^N \left(\frac{1}{M} \sum_{j=1}^M \mathbf{v}_{ij}^\top \nabla_{\mathbf{x}} \mathbf{s}_m(\mathbf{x}_i; \boldsymbol{\theta}) \mathbf{v}_{ij} \right) + \frac{1}{2} \|\mathbf{s}_m(\mathbf{x}_i; \boldsymbol{\theta})\|_2^2,$$

which is exactly the sliced score matching objective with variance reduction $\hat{J}_{\text{vr}}(\boldsymbol{\theta}; \mathbf{x}_1^N, \mathbf{v}_{11}^{NM})$.

3.5 Related Work

3.5.1 Scalable Score Matching

To the best of our knowledge, there are three existing methods that are able to scale up score matching to learning deep models on high dimensional data.

Denoising score matching Vincent [Vin11] proposes denoising score matching, a variant of score matching that completely circumvents the Hessian. Specifically, consider a noise distribution $q_\sigma(\tilde{\mathbf{x}} | \mathbf{x})$, and let $q_\sigma(\tilde{\mathbf{x}}) = \int q_\sigma(\tilde{\mathbf{x}} | \mathbf{x}) p_d(\mathbf{x}) d\mathbf{x}$. Denoising score matching applies

the original score matching to the noise-corrupted data distribution $q_\sigma(\tilde{\mathbf{x}})$, and the objective can be proven to be equivalent to the following up to a constant

$$\frac{1}{2} \mathbb{E}_{q_\sigma(\tilde{\mathbf{x}}|\mathbf{x})p_d(\mathbf{x})} [\|s_m(\tilde{\mathbf{x}}; \boldsymbol{\theta}) - \nabla_{\mathbf{x}} \log q_\sigma(\tilde{\mathbf{x}} | \mathbf{x})\|_2^2],$$

which can be estimated without computing any Hessian. Although denoising score matching is much faster than score matching, it has obvious drawbacks. First, it can only recover the noise corrupted data distribution. Furthermore, choosing the parameters of the noise distribution is highly non-trivial and in practice the performance can be very sensitive to σ , and heuristics have to be used in practice. For example, [Sar+18] propose a heuristic for choosing σ based on Parzen windows.

Approximate backpropagation Kingma and LeCun [KL10] propose a backpropagation method to approximately compute the trace of the Hessian. Because the backpropagation of the full Hessian scales quadratically w.r.t. the layer size, the authors limit backpropagation only to the diagonal so that it has the same cost as the usual backpropagation for computing loss gradients. However, there are no theoretical guarantees for the approximation errors. In fact, the authors only did experiments on networks with a single hidden layer, in which case the approximation is exact. Moreover, there is no direct support for the proposed approximate backpropagation method in modern automatic differentiation frameworks.

Curvature propagation Martens et al. [MSS12] estimate the trace term in score matching by applying curvature propagation to compute an unbiased, complex-valued estimator of the diagonal of the Hessian. The work claims that curvature propagation will have the smallest variance among a class of estimators, which includes the Hutchinson's estimator. However, their proof evaluates the pseudo-variance of the complex-valued estimator instead of the variance. In practice, curvature propagation can have large variance when the number of nodes in the network is large, because it introduces noise for each node in the network. Finally, implementing curvature propagation also requires manually modifying the backpropagation code, handling complex numbers in neural networks, and will be inefficient for networks of more general structures, such as recurrent neural networks.

3.5.2 Kernel Score Estimators

Two prior methods for score estimation are based on a generalized form of Stein’s identity [Ste81; GM17]:

$$\mathbb{E}_{q(\mathbf{x})}[\mathbf{h}(\mathbf{x})\nabla_{\mathbf{x}} \log q(\mathbf{x})^T + \nabla_{\mathbf{x}}\mathbf{h}(\mathbf{x})] = \mathbf{0}, \quad (3.12)$$

where $q(\mathbf{x})$ is a continuously differentiable density and $\mathbf{h}(\mathbf{x})$ is a function satisfying some regularity conditions. [LT18] propose to set $\mathbf{h}(\mathbf{x})$ as the feature map of some kernel in the *Stein class* [LLJ16] of q , and solve a finite-sample version of Eq. (3.12) to obtain estimates of $\nabla_{\mathbf{x}} \log q(\mathbf{x})$ at the sample points. We refer to this method as *Stein* in the experiments. Compared to *Stein*, their method is argued to have theoretical guarantees and principled out-of-sample estimation at an unseen test point. We refer to their method as *Spectral* in the experiments.

3.6 Experiments

Our experiments include two parts: (1) to test the effectiveness of sliced score matching (**SSM**) in learning deep models for density estimation, and (2) to test the ability of **SSM** in providing score estimates for applications such as VAEs with implicit encoders and WAEs. Unless specified explicitly, we choose $M = 1$ by default.

3.6.1 Density Estimation

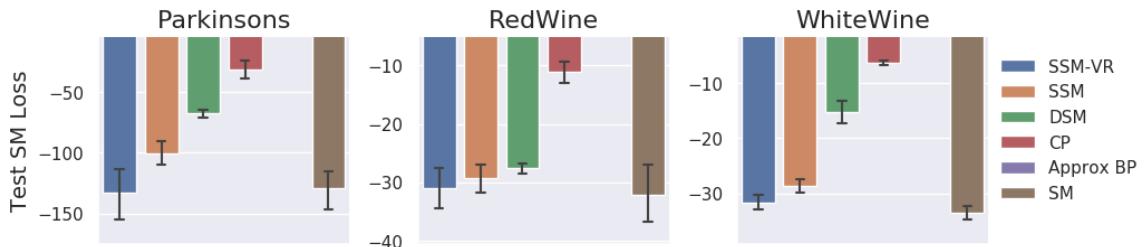


Figure 3.1: **SM** loss after training DKEF models on UCI datasets with different loss functions; lower is better. Results for approximate backpropagation are not shown because losses were larger than 10^9 .

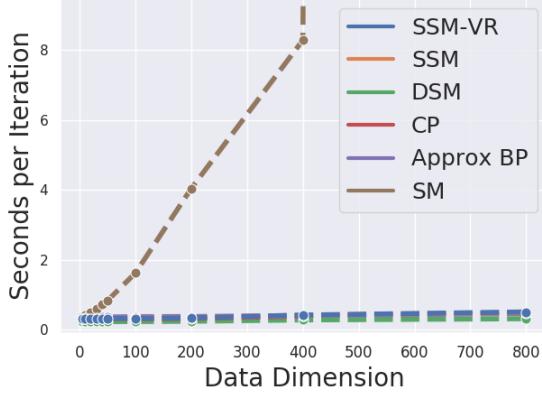


Figure 3.2: **SM** performance degrades linearly with the data dimension, while efficient approaches have relatively similar performance.

We evaluate **SSM** and its variance-reduced version (**SSM-VR**) for density estimation and compare against score matching (**SM**) and its three scalable baselines: denoising score matching (**DSM**), approximate backpropagation (**approx BP**), and curvature propagation (**CP**). All **SSM** methods in this section use the multivariate Rademacher distribution as $p_{\mathbf{v}}$. Our results demonstrate that: (1) **SSM** is comparable in performance to **SM**, (2) **SSM** outperforms other computationally efficient approximations to **SM**, and (3) unlike **SM**, **SSM** scales effectively to high dimensional data.

Deep Kernel Exponential Families

Models Deep kernel exponential families (DKEF) are unnormalized density models trained using **SM** [Wen+19]. DKEFs parameterize the unnormalized log density as $\log \tilde{p}_f(\mathbf{x}) = f(\mathbf{x}) + \log q_0(\mathbf{x})$, where f is a mixture of Gaussian kernels evaluated at different inducing points: $f(\mathbf{x}) = \sum_{l=1}^L \alpha_l k(\mathbf{x}, \mathbf{z}_l)$. The kernel is defined on the feature space of a neural network, and the network parameters are trained along with the inducing points \mathbf{z}_l . Further details of the model, which is identical to that in [Wen+19], are presented in Appendix B.1.2.

Setup Following the settings of [Wen+19], we trained models on three UCI datasets: Parkinsons, RedWine, and WhiteWine [DG17], and used their original code for **SM**. To compute the trace term exactly, Wenliang et al. [Wen+19]’s manual implementation of backpropagation takes over one thousand lines for a model that is four layers deep, while the implementation of **SSM** only takes several lines. For **DSM**, the value of σ is chosen by grid

search using the **SM** loss on a validation dataset. All models are trained with 15 different random seeds and training is stopped when validation loss does not improve for 200 steps.

Results Results in Fig. 3.1 demonstrate that **SSM-VR** performs comparably to **SM**, when evaluated using the **SM** loss on a held-out test set. We observe that variance reduction substantially improves the performance of **SSM**. In addition, **SSM** outperforms other computationally efficient approaches. **DSM** can perform comparably to **SSM** on RedWine. However, it is challenging to select σ for **DSM**. Models trained using σ from the heuristic in [Sar+18] are far from optimal (on both **SM** losses and likelihoods), and extensive grid search is needed to find the best σ . **CP** performs substantially worse, which is likely because it injects noise for each node in the computation graph, and the amount of noise introduced is too large for a neural-network-based kernel evaluated at 200 inducing points, which supports our hypothesis that **CP** does not work effectively for deep models. Results for **approx BP** are omitted because the losses exceeded 10^9 on all datasets. This is because **approx BP** provides a biased estimate of the Hessian without any error guarantees. All the results are similar when evaluating according to log-likelihood metric (Appendix B.1.2).

Scalability to high dimensional data We evaluate the computational efficiency of different losses on data of increasing dimensionality. We fit DKEF models to a multivariate standard normal in high dimensional spaces. The average running time per minibatch of 100 examples is reported in Fig. 3.2. **SM** performance degrades linearly with the dimension of the input data due to the computation of the Hessian, and creates out of memory errors for a 12GB GPU after the dimension increases beyond 400. **SSM** performs similarly to **DSM**, **approx BP** and **CP**, and they are all scalable to large dimensions.

Deep Flow Models

Setup As a sanity check, we also evaluate **SSM** by training a NICE flow model [DKB15], whose likelihood is tractable and can be compared to results obtained by **MLE**. The model we use has 20 hidden layers, and 1000 units per layer. Models are trained to fit MNIST handwritten digits, which are 784 dimensional images. Data are dequantized by adding uniform noise in the range $[-\frac{1}{512}, \frac{1}{512}]$, and transformed using a logit transformation, $\log(x) - \log(1 - x)$. We provide additional details in Appendix B.1.2.

	Test SM Loss	Test LL
MLE	-579	-791
SSM-VR	-8054	-3355
SSM	-2428	-2039
DSM ($\sigma = 0.10$)	-3035	-4363
DSM ($\sigma = 1.74$)	-97	-8082
CP	-1694	-1517
Approx BP	-48	-2288

Table 3.1: Score matching losses and log-likelihoods for NICE models on MNIST. $\sigma = 0.1$ is by grid search and $\sigma = 1.74$ is from the heuristic of [Sar+18].

Training with **SM** is extremely computationally expensive in this case. Our **SM** implementation based on auto-differentiation takes around 7 hours to finish one epoch of training, and 12 GB GPU memory cannot hold a batch size larger than 24, so we do not include these results. Since NICE has tractable likelihoods, we also evaluate **MLE** as a surrogate objective for minimizing the **SM** loss. Notably, likelihoods and **SM** losses might be uncorrelated when the model is mis-specified, which is likely to be the case for a complex dataset like MNIST.

Results **SM** losses and log-likelihoods on the test dataset are reported in Table 3.1, where models are evaluated using the best checkpoint in terms of the **SM** loss on a validation dataset over 100 epochs of training. **SSM-VR** greatly outperforms all the other methods on the **SM** loss. **DSM** performs worse than **SSM-VR**, and σ is still hard to tune. Specifically, following the heuristic in [Sar+18] leads to $\sigma = 1.74$, which performed the worst (on both log-likelihood and **SM** loss) of the eight choices of σ in our grid search. **Approx BP** has more success on NICE than for training DKEF models. This might be because the Hessians of hidden layers of NICE are closer to a diagonal matrix, which results in a smaller approximation error for **approx BP**. As in the DKEF experiments, **CP** performs worse. This is likely due to injecting noise to all hidden units, which will lead to large variance for a network as big as NICE. Unlike the DKEF experiments, we find that good log-likelihoods are less correlated with good **SM** loss, probably due to model mis-specification.

3.6.2 Score Estimation

We consider two typical tasks that require accurate score estimations: (1) training VAEs with an implicit encoder and (2) training Wasserstein Auto-Encoders. We show in both

Latent Dim	VAE		WAE	
	8	32	8	32
ELBO	96.87	89.06	N/A	N/A
SSM	95.50	89.25 (88.29 [†])	98.24	90.37
Stein	96.71	91.84	99.05	91.70
Spectral	96.60	94.67	98.81	92.55

Table 3.2: Negative log-likelihoods on MNIST, estimated by AIS. [†]The result of SSM with $M = 100$, in which case SSM matches the computational cost of kernel methods, which used 100 samples for each data point.

	Method \ Iteration	10k	40k	70k	100k
	Method	10k	40k	70k	100k
VAE	ELBO	96.20	73.70	69.42	66.32
	SSM	108.52	70.28	66.52	62.50
	Stein	126.60	118.87	120.51	126.76
	Spectral	131.90	125.04	128.36	133.93
WAE	SSM	84.11	61.09	56.23	54.33
	Stein	82.93	63.46	58.53	57.61
	Spectral	82.30	62.47	58.03	55.96

Table 3.3: FID scores of different methods versus number of training iterations on CelebA dataset.

tasks SSM outperforms previous score estimators. Samples generated by various algorithms can be found in Appendix B.1.1.

VAE with Implicit Encoders

Background Consider a latent variable model $p(\mathbf{x}, \mathbf{z})$, where \mathbf{x} and \mathbf{z} denote observed and latent variables respectively. A variational auto-encoder (VAE) is composed of two parts: 1) an encoder $p_{\theta}(\mathbf{x} | \mathbf{z})$ modeling the conditional distribution of \mathbf{x} given \mathbf{z} ; and a decoder $q_{\phi}(\mathbf{z} | \mathbf{x})$ that approximates the posterior distribution of the latent variable. A VAE is typically trained by maximizing the following evidence lower bound (ELBO)

$$\mathbb{E}_{p_d} [\mathbb{E}_{q_{\phi}(\mathbf{z} | \mathbf{x})} \log p_{\theta}(\mathbf{x} | \mathbf{z}) p(\mathbf{z}) - \mathbb{E}_{q_{\phi}(\mathbf{z} | \mathbf{x})} \log q_{\phi}(\mathbf{z} | \mathbf{x})],$$

where $p(\mathbf{z})$ denotes a pre-specified prior distribution. The expressive power of $q_\phi(\mathbf{z} \mid \mathbf{x})$ is critical to the success of variational learning. Typically, $q_\phi(\mathbf{z} \mid \mathbf{x})$ is chosen to be a simple distribution with tractable densities so that $H(q_\phi) := -\mathbb{E}_{q_\phi(\mathbf{z} \mid \mathbf{x})} \log q_\phi(\mathbf{z} \mid \mathbf{x})$ is tractable. We call this traditional approach “ELBO” in the experiments. With score estimation techniques, we can directly compute $\nabla_\phi H(q_\phi)$ for implicit distributions, which enables more flexible options for q_ϕ . We consider 3 score estimation techniques: **SSM**, **Stein** [LT18] and **Spectral** [SSZ18].

For a single data point \mathbf{x} , kernel score estimators need multiple samples from $q_\phi(\mathbf{z} \mid \mathbf{x})$ to estimate its score. On MNIST, we use 100 samples, as done in [SSZ18]. On CelebA, however, we can only take 20 samples due to GPU memory limitations. In contrast, **SSM** learns a score network $h(\mathbf{z} \mid \mathbf{x})$ along with $q_\phi(\mathbf{z} \mid \mathbf{x})$, which amortizes the cost of score estimation. Unless noted explicitly, we use one projection per data point ($M = 1$) for **SSM**, which is scalable to deep networks.

Setup We consider VAE training on MNIST and CelebA [Liu+15]. All images in CelebA are first cropped to a patch of 140×140 and then resized to 64×64 . We report negative likelihoods on MNIST, as estimated by AIS [Nea01] with 1000 intermediate distributions. We evaluate sample quality on CelebA with FID scores [Heu+17]. For fast AIS evaluation on MNIST, we use shallow fully-connected networks with 3 hidden layers. For CelebA experiments we use deep convolutional networks. The architectures of implicit encoders and score networks are straightforward modifications to the encoders of ELBO. More details are provided in Appendix B.1.2.

Results The negative likelihoods of different methods on MNIST are reported in the left part of Table 3.2. We note that **SSM** outperforms **Stein** and **Spectral** in all cases. When the latent dimension is 8, **SSM** outperforms ELBO, which indicates that the expressive power of implicit $q_\phi(\mathbf{z} \mid \mathbf{x})$ pays off. When the latent dimension is 32, the gaps between **SSM** and kernel methods are even larger, and the performance of **SSM** is still comparable to ELBO. Moreover, when $M = 100$ (matching the computation of kernel methods), **SSM** outperforms ELBO.

For CelebA, we provide FID scores of samples in the top part of Table 3.3. We observe that after 40k training iterations, **SSM** outperforms all other baselines. Kernel methods perform poorly in this case because only 20 samples per data point can be used due to

limited amount of GPU memory. Early during training, **SSM** does not perform as well. Since the score network is trained along with the encoder and decoder, a moderate number of training steps is needed to give an accurate score estimation (and better learning of the VAE).

WAEs

Background WAE is another method to learn latent variable models, which generally produces better samples than VAEs. Similar to a VAE, it contains an encoder $q_\phi(\mathbf{z} \mid \mathbf{x})$ and a decoder $p_\theta(\mathbf{x} \mid \mathbf{z})$ and both can be implicit distributions. Let $p(\mathbf{z})$ be a pre-defined prior distribution, and $q_\phi(\mathbf{z}) := \int q_\phi(\mathbf{z} \mid \mathbf{x}) p_d(\mathbf{x}) d\mathbf{x}$ denote the aggregated posterior distribution. Using a metric function $c(\cdot, \cdot)$ and KL divergence between $q_\phi(\mathbf{z})$ and $p(\mathbf{z})$, WAE minimizes the following objective

$$\mathbb{E}_{p_d} [\mathbb{E}_{q_\phi(\mathbf{z} \mid \mathbf{x})} [c(\mathbf{x}, p_\theta(\mathbf{x} \mid \mathbf{z})) - \lambda \log p(\mathbf{z})]] - \lambda H(q_\phi(\mathbf{z})).$$

Compared to $\nabla_\phi H(q_\phi(\mathbf{z} \mid \mathbf{x}))$, it is easier to estimate $\nabla_\phi H(q_\phi(\mathbf{z}))$ for kernel methods, because the samples of $q_\phi(\mathbf{z})$ can be collected by first sampling $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N \stackrel{\text{i.i.d.}}{\sim} p_d(\mathbf{x})$ and then sample one \mathbf{z} for each \mathbf{x}_i from $q_\phi(\mathbf{z} \mid \mathbf{x}_i)$. In contrast, multiple \mathbf{z} 's need to be sampled for each \mathbf{x}_i when estimating $\nabla_\phi H(q_\phi(\mathbf{z} \mid \mathbf{x}))$ with kernel approaches. For **SSM**, we use a score network $h(\mathbf{z})$ and train it alongside $q_\phi(\mathbf{z} \mid \mathbf{x})$.

Setup The setup for WAE experiments is largely the same as VAE. The architectures are very similar to those used in the VAE experiments, and the only difference is that we made decoders implicit, as suggested in [Tol+18]. More details can be found in Appendix B.1.2.

Results The negative likelihoods on MNIST are provided in the right part of Table 3.2. **SSM** outperforms both kernel methods, and achieves a larger performance gap when the latent dimension is higher. The likelihoods are lower than the VAE results as the WAE objective does not directly maximize likelihoods.

We show FID scores for CelebA experiments in the bottom part of Table 3.3. As expected, kernel methods perform much better than before, because it is faster to sample from $q_\phi(\mathbf{z})$. The FID scores are generally lower than those in VAE experiments, which supports previous results that WAEs generally obtain better samples. **SSM** outperforms both kernel methods

when the number of iterations is more than 40k. This shows the advantages of training a deep, expressive score network compared to using a fixed kernel in score estimation tasks.

3.7 Conclusion

We propose sliced score matching, a scalable method for learning unnormalized statistical models and estimating scores for implicit distributions. Compared to the original score matching and its variants, our estimator can scale to deep models on high dimensional data, while remaining easy to implement in modern automatic differentiation frameworks. Theoretically, our estimator is consistent and asymptotically normal under some regularity conditions. Experimental results demonstrate that our method outperforms competitors in learning deep energy-based models and provides more accurate estimates than kernel score estimators in training implicit VAEs and WAEs.

Chapter 4

Learning High Order Score Models with Denoising

In the previous chapter, we introduced sliced score training as an efficient way of training score models to estimate the first order derivative of the data density. In this chapter, we aim to estimate higher order derivatives, which provide additional local information about the data distribution and enable new applications. Although they can be estimated via automatic differentiation of a learned density model, this can amplify estimation errors and is expensive in high dimensional settings. To overcome these limitations, we propose a method to directly estimate high order derivatives (scores) of a data density from samples, inspired by the idea of denoising score matching [Vin11]. We first show that denoising score matching can be interpreted as a particular case of Tweedie’s formula. By leveraging Tweedie’s formula on higher order moments, we generalize denoising score matching to estimate higher order derivatives. We demonstrate empirically that models trained with the proposed method can approximate second order derivatives more efficiently and accurately than via automatic differentiation. Experimentally, we show that our models can be used to quantify uncertainty in denoising and to improve the mixing speed of Langevin dynamics via Ozaki discretization for sampling synthetic data and natural images.

This chapter was previously published as [Men+21]. Chenlin Meng, in particular, contributed significantly to the materials in this chapter. My contributions are conceiving the idea, designing the experiments, and helping with paper writing.

4.1 Introduction

The first order derivative of the log data density function, also known as *score*, has found many applications including image generation [SE19b; SE20; HJA20] (*cf.*, Chapters 5 and 6), image denoising [Sar+18; SH19] and audio synthesis [Kon+20]. Denoising score matching (DSM) [Vin11] provides an efficient way to estimate the *score* of the data density from samples and has been widely used for training score-based generative models and denoising [Sar+18; SH19]. High order derivatives of the data density, which we refer to as *high order scores*, provide a more accurate local approximation of the data density (*e.g.*, its curvature) and enable new applications. For instance, high order scores can improve the mixing speed for certain sampling methods [DK19; SZ+19; Mou+19], similar to how high order derivatives accelerate gradient descent in optimization [MG15]. In denoising problems, given a noisy data point, high order scores can be used to compute high order moments of the underlying noise-free data point, thus providing a way to quantify the uncertainty in denoising.

Existing methods for score estimation [HD05; Vin11; Son+19; ZSZ20], such as denoising score matching [Vin11], focus on estimating the *first order* score (*i.e.*, the Jacobian of the log density). In principle, high order scores can be estimated from a learned first order score model (or even a density model) via automatic differentiation. However, this approach is computationally expensive for high dimensional data and score models parameterized by deep neural networks. For example, given a D dimensional distribution, computing the $(n + 1)$ -th order score value from an existing n -th order score model by automatic differentiation is on the order of D times more expensive than evaluating the latter (*cf.*, Chapter 3). Moreover, computing higher-order scores by automatic differentiation might suffer from large estimation error, since a small training loss for the first order score does not always lead to a small estimation error for high order scores.

To overcome these limitations, we propose a new approach which directly models and estimates high order scores of a data density from samples. We draw inspiration from Tweedie’s formula [Efr11; Rob20], which connects the score function to a denoising problem, and show that denoising score matching (DSM) with Gaussian noise perturbation can be derived from Tweedie’s formula with the knowledge of least squares regression. We then provide a generalized version of Tweedie’s formula which allows us to further extend denoising score matching to estimate high order scores. In addition, we provide variance reduction techniques to improve the optimization of these newly introduced high order score

estimation objectives. With our approach, we can directly parameterize high order scores and learn them efficiently, sidestepping expensive automatic differentiation.

While our theory and estimation method is applicable to scores of any order, we focus on the *second order score* (*i.e.*, the Hessian of the log density) for empirical evaluation. Our experiments show that models learned with the proposed objective can approximate second order scores more accurately than applying automatic differentiation to lower order score models. Our approach is also more computationally efficient for high dimensional data, achieving up to $500\times$ speedups for second order score estimation on MNIST. In denoising problems, there could be multiple clean data points consistent with a noisy observation, and it is often desirable to measure the uncertainty of denoising results. As second order scores are closely related to the covariance matrix of the noise-free data conditioned on the noisy observation, we show that our estimated second order scores can provide extra insights into the solution of denoising problems by capturing and quantifying the uncertainty of denoising. We further show that our model can be used to improve the mixing speed of Langevin dynamics for sampling synthetic data and natural images. Our empirical results on second order scores, a special case of the general approach, demonstrate the potential and applications of our method for estimating high order scores.

4.2 Background

4.2.1 Scores of a Distribution

Definition 4.1. Given a probability density $p(\mathbf{x})$ over \mathbb{R}^D , we define the k -th order score $\mathbf{s}_k(\mathbf{x}) : \mathbb{R}^D \rightarrow \otimes^k \mathbb{R}^D$, where \otimes^k denotes k -fold tensor multiplications, to be a tensor with the (i_1, i_2, \dots, i_k) -th index given by $[\mathbf{s}_k(\mathbf{x})]_{i_1 i_2 \dots i_k} := \frac{\partial^k}{\partial x_{i_1} \partial x_{i_2} \dots \partial x_{i_k}} \log p(\mathbf{x})$, where $(i_1, i_2, \dots, i_k) \in \{1, \dots, D\}^k$.

As an example, when $k = 1$, the *first order score* is the gradient of $\log p(\mathbf{x})$ w.r.t. to \mathbf{x} , defined as $\mathbf{s}_1(\mathbf{x}) := \nabla_{\mathbf{x}} \log p(\mathbf{x})$. Intuitively, this is a vector field of the steepest ascent directions for the log-density. Note that the definition of *first order score* matches the definition of (Stein) *score* [HD05]. When $k = 2$, the *second order score* is the Hessian of $\log p(\mathbf{x})$ w.r.t. to \mathbf{x} . It gives the curvature of a density function, and with $\mathbf{s}_1(\mathbf{x})$ it can provide a better local approximation to $\log p(\mathbf{x})$.

4.2.2 Denoising Score Matching

Given a data distribution $p_{\text{data}}(\mathbf{x})$ and a model distribution $p(\mathbf{x}; \boldsymbol{\theta})$, the *score* functions of $p_{\text{data}}(\mathbf{x})$ and $p(\mathbf{x}; \boldsymbol{\theta})$ are defined as $\mathbf{s}_1(\mathbf{x}) := \nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x})$ and $\mathbf{s}_1(\mathbf{x}; \boldsymbol{\theta}) := \nabla_{\mathbf{x}} \log p(\mathbf{x}; \boldsymbol{\theta})$ respectively. Denoising score matching (DSM) [Vin11] perturbs a data sample $\mathbf{x} \sim p_{\text{data}}(\mathbf{x})$ with a pre-specified noise distribution $q_\sigma(\tilde{\mathbf{x}} | \mathbf{x})$ and then estimates the *score* of the perturbed data distribution $q_\sigma(\tilde{\mathbf{x}}) = \int q_\sigma(\tilde{\mathbf{x}} | \mathbf{x}) p_{\text{data}}(\mathbf{x}) d\mathbf{x}$ which we denote $\tilde{\mathbf{s}}_1(\tilde{\mathbf{x}}) := \nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}})$. DSM uses the following objective

$$\frac{1}{2} \mathbb{E}_{p_{\text{data}}(\mathbf{x})} \mathbb{E}_{q_\sigma(\tilde{\mathbf{x}} | \mathbf{x})} [\|\tilde{\mathbf{s}}_1(\tilde{\mathbf{x}}; \boldsymbol{\theta}) - \nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}} | \mathbf{x})\|_2^2]. \quad (4.1)$$

It is shown that under certain regularity conditions, minimizing Eq. (4.1) is equivalent to minimizing the score matching [HD05] loss between $\tilde{\mathbf{s}}_1(\tilde{\mathbf{x}}; \boldsymbol{\theta})$ and $\tilde{\mathbf{s}}_1(\tilde{\mathbf{x}})$ [Vin11] defined as

$$\frac{1}{2} \mathbb{E}_{p_{\text{data}}(\mathbf{x})} \mathbb{E}_{q_\sigma(\tilde{\mathbf{x}} | \mathbf{x})} [\|\tilde{\mathbf{s}}_1(\tilde{\mathbf{x}}; \boldsymbol{\theta}) - \tilde{\mathbf{s}}_1(\tilde{\mathbf{x}})\|_2^2]. \quad (4.2)$$

When $q_\sigma(\tilde{\mathbf{x}} | \mathbf{x}) = \mathcal{N}(\tilde{\mathbf{x}} | \mathbf{x}, \sigma^2 I)$, the objective becomes

$$\mathcal{L}_{\text{DSM}}(\boldsymbol{\theta}) = \frac{1}{2} \mathbb{E}_{p_{\text{data}}(\mathbf{x})} \mathbb{E}_{q_\sigma(\tilde{\mathbf{x}} | \mathbf{x})} \left[\left\| \tilde{\mathbf{s}}_1(\tilde{\mathbf{x}}; \boldsymbol{\theta}) + \frac{1}{\sigma^2} (\tilde{\mathbf{x}} - \mathbf{x}) \right\|_2^2 \right]. \quad (4.3)$$

Optimizing Eq. (4.3) can, intuitively, be understood as predicting $\frac{\tilde{\mathbf{x}} - \mathbf{x}}{\sigma^2}$, the added “noise” up to a constant, given the noisy input $\tilde{\mathbf{x}}$, and is thus related to denoising. Estimating the score of the noise perturbed distribution $q_\sigma(\tilde{\mathbf{x}})$ instead of the original (clean) data distribution $p_{\text{data}}(\mathbf{x})$ allows DSM to approximate scores more efficiently than other methods [HD05; Son+19]. When σ is close to zero, $q_\sigma(\tilde{\mathbf{x}}) \approx p_{\text{data}}(\mathbf{x})$ so the score of $q_\sigma(\tilde{\mathbf{x}})$ estimated by DSM will be close to that of $p_{\text{data}}(\mathbf{x})$. When σ is large, the estimated score for $q_\sigma(\tilde{\mathbf{x}})$ plays a crucial role in denoising [Sar+18] and learning score-based generative models [SE19b; SE20].

4.2.3 Tweedie’s Formula

Given a prior density $p_{\text{data}}(\mathbf{x})$, a noise distribution $q_\sigma(\tilde{\mathbf{x}} | \mathbf{x}) = \mathcal{N}(\tilde{\mathbf{x}} | \mathbf{x}, \sigma^2 I)$, and the noisy density $q_\sigma(\tilde{\mathbf{x}}) = \int p_{\text{data}}(\mathbf{x}) q_\sigma(\tilde{\mathbf{x}} | \mathbf{x}) d\mathbf{x}$, Tweedie’s formula [Rob20; Efr11] provides a close-form expression for the posterior expectation (the first moment) of \mathbf{x} conditioned on $\tilde{\mathbf{x}}$:

$$\mathbb{E}[\mathbf{x} | \tilde{\mathbf{x}}] = \tilde{\mathbf{x}} + \sigma^2 \tilde{\mathbf{s}}_1(\tilde{\mathbf{x}}), \quad (4.4)$$

where $\tilde{\mathbf{s}}_1(\tilde{\mathbf{x}}) := \nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}})$. Equation 4.4 implies that given a “noisy” observation $\tilde{\mathbf{x}} \sim q_\sigma(\tilde{\mathbf{x}})$, one can compute the expectation of the “clean” data point \mathbf{x} that may have produced $\tilde{\mathbf{x}}$. As a result, Equation 4.4 has become an important tool for denoising [SH19; Sar+18]. We provide the proof in Appendix A.2.

A less widely known fact is that Tweedies’ formula can be generalized to provide higher order moments of \mathbf{x} given $\tilde{\mathbf{x}}$, which we will leverage to derive the objective for learning higher order scores.

4.3 Estimating Higher Order Scores by Denoising

Below we demonstrate that DSM can be derived from Tweedie’s formula [Efr11; Rob20]. By leveraging the generalized Tweedie’s formula on high order moments of the posterior, we extend DSM to estimate higher order score functions.

4.3.1 DSM in the View of Tweedie’s Formula

The optimal solution to the least squares regression problem

$$\min_{\boldsymbol{\theta}} \mathbb{E}_{p_{\text{data}}(\mathbf{x})} \mathbb{E}_{q_\sigma(\tilde{\mathbf{x}}|\mathbf{x})} [\|\mathbf{h}(\tilde{\mathbf{x}}; \boldsymbol{\theta}) - \mathbf{x}\|_2^2] \quad (4.5)$$

is well-known to be the conditional expectation $\mathbf{h}(\tilde{\mathbf{x}}; \boldsymbol{\theta}^*) = \mathbb{E}[\mathbf{x} | \tilde{\mathbf{x}}]$. If we parameterize $\mathbf{h}(\tilde{\mathbf{x}}; \boldsymbol{\theta}) = \tilde{\mathbf{x}} + \sigma^2 \tilde{\mathbf{s}}_1(\tilde{\mathbf{x}}; \boldsymbol{\theta})$ where $\tilde{\mathbf{s}}_1(\tilde{\mathbf{x}}; \boldsymbol{\theta})$ is a first order score model with parameter $\boldsymbol{\theta}$, the least squares problem in Eq. (4.5) becomes equivalent to the DSM objective:

$$\min_{\boldsymbol{\theta}} \mathbb{E}_{p_{\text{data}}(\mathbf{x})} \mathbb{E}_{q_\sigma(\tilde{\mathbf{x}}|\mathbf{x})} [\|\sigma^2 \tilde{\mathbf{s}}_1(\tilde{\mathbf{x}}; \boldsymbol{\theta}) + \tilde{\mathbf{x}} - \mathbf{x}\|_2^2] = \min_{\boldsymbol{\theta}} 2\sigma^4 \cdot \mathcal{L}_{\text{DSM}}(\boldsymbol{\theta}). \quad (4.6)$$

From Tweedie’s formula, we know the optimal $\boldsymbol{\theta}^*$ satisfies $\mathbf{h}(\tilde{\mathbf{x}}; \boldsymbol{\theta}^*) = \tilde{\mathbf{x}} + \sigma^2 \tilde{\mathbf{s}}_1(\tilde{\mathbf{x}}; \boldsymbol{\theta}^*) = \mathbb{E}[\mathbf{x} | \tilde{\mathbf{x}}] = \tilde{\mathbf{x}} + \sigma^2 \tilde{\mathbf{s}}_1(\tilde{\mathbf{x}})$, from which we can conclude that $\tilde{\mathbf{s}}_1(\tilde{\mathbf{x}}; \boldsymbol{\theta}^*) = \tilde{\mathbf{s}}_1(\tilde{\mathbf{x}})$. This proves that minimizing the DSM objective in Eq. (4.6) recovers the first order score.

There are other ways to derive DSM. For example, Raphan and Simoncelli [RS11] provide a proof based on Bayesian least squares without relying on Tweedie’s formula. Stein’s Unbiased Risk Estimator (SURE) [Ste81] can also provide an alternative proof based on integration by parts. Compared to these methods, our derivation can be easily extended to learn high order scores, leveraging a more general version of Tweedie’s formula.

4.3.2 Second Order Denoising Score Matching

As a warm-up, we first consider the second order score, and later generalize to any desired order. Leveraging Tweedie's formula on $\mathbb{E}[\mathbf{x}\mathbf{x}^T | \tilde{\mathbf{x}}]$ and $\mathbb{E}[\mathbf{x} | \tilde{\mathbf{x}}]$, we obtain the following theorem.

Theorem 4.1. *Given a D -dimensional distribution $p(\mathbf{x})$ and $q_\sigma(\tilde{\mathbf{x}}) := \int p(\mathbf{x})q_\sigma(\tilde{\mathbf{x}}|\mathbf{x})d\mathbf{x}$, we have*

$$\mathbb{E}[\mathbf{x}\mathbf{x}^T | \tilde{\mathbf{x}}] = \mathbf{f}(\tilde{\mathbf{x}}, \tilde{\mathbf{s}}_1, \tilde{\mathbf{s}}_2) \quad (4.7)$$

$$\mathbb{E}[\mathbf{x}\mathbf{x}^T - \mathbf{x}\tilde{\mathbf{x}}^T - \tilde{\mathbf{x}}\mathbf{x}^T | \tilde{\mathbf{x}}] = \mathbf{h}(\tilde{\mathbf{x}}, \tilde{\mathbf{s}}_1, \tilde{\mathbf{s}}_2), \quad (4.8)$$

where $\mathbf{f}(\tilde{\mathbf{x}}, \tilde{\mathbf{s}}_1, \tilde{\mathbf{s}}_2)$ and $\mathbf{h}(\tilde{\mathbf{x}}, \tilde{\mathbf{s}}_1, \tilde{\mathbf{s}}_2)$ are polynomials of $\tilde{\mathbf{x}}, \tilde{\mathbf{s}}_1(\tilde{\mathbf{x}}), \tilde{\mathbf{s}}_2(\tilde{\mathbf{x}})$ defined as

$$\mathbf{f}(\tilde{\mathbf{x}}, \tilde{\mathbf{s}}_1, \tilde{\mathbf{s}}_2) = \tilde{\mathbf{x}}\tilde{\mathbf{x}}^T + \sigma^2\tilde{\mathbf{x}}\tilde{\mathbf{s}}_1(\tilde{\mathbf{x}})^T + \sigma^2\tilde{\mathbf{s}}_1(\tilde{\mathbf{x}})\tilde{\mathbf{x}}^T + \sigma^4\tilde{\mathbf{s}}_2(\tilde{\mathbf{x}}) + \sigma^4\tilde{\mathbf{s}}_1(\tilde{\mathbf{x}})\tilde{\mathbf{s}}_1(\tilde{\mathbf{x}})^T + \sigma^2I, \quad (4.9)$$

$$\mathbf{h}(\tilde{\mathbf{x}}, \tilde{\mathbf{s}}_1, \tilde{\mathbf{s}}_2) = -\tilde{\mathbf{x}}\tilde{\mathbf{x}}^T + \sigma^4\tilde{\mathbf{s}}_2(\tilde{\mathbf{x}}) + \sigma^4\tilde{\mathbf{s}}_1(\tilde{\mathbf{x}})\tilde{\mathbf{s}}_1(\tilde{\mathbf{x}})^T + \sigma^2I. \quad (4.10)$$

Here $\tilde{\mathbf{s}}_1(\tilde{\mathbf{x}})$ and $\tilde{\mathbf{s}}_2(\tilde{\mathbf{x}})$ denote the first and second order scores of $q_\sigma(\tilde{\mathbf{x}})$.

In Theorem 4.1, Eq. (4.9) is directly given by Tweedie's formula on $\mathbb{E}[\mathbf{x}\mathbf{x}^T | \tilde{\mathbf{x}}]$, and Eq. (4.10) is derived from Tweedie's formula on both $\mathbb{E}[\mathbf{x} | \tilde{\mathbf{x}}]$ and $\mathbb{E}[\mathbf{x}\mathbf{x}^T | \tilde{\mathbf{x}}]$. Given a noisy sample $\tilde{\mathbf{x}}$, Theorem 4.1 relates the second order moment of \mathbf{x} to the first order score $\tilde{\mathbf{s}}_1(\tilde{\mathbf{x}})$ and second order score $\tilde{\mathbf{s}}_2(\tilde{\mathbf{x}})$ of $q_\sigma(\tilde{\mathbf{x}})$. A detailed proof of Theorem 4.1 is given in Appendix A.2.

In the same way as how we derive DSM from Tweedie's formula in Section 4.3.1, we can obtain higher order score matching objectives with Eq. (4.9) and Eq. (4.10) as a least squares problem.

Theorem 4.2. *Suppose the first order score $\tilde{\mathbf{s}}_1(\tilde{\mathbf{x}})$ is given, we can learn a second order score model $\tilde{\mathbf{s}}_2(\tilde{\mathbf{x}}; \boldsymbol{\theta})$ by optimizing the following objectives*

$$\boldsymbol{\theta}^* = \arg \min_{\boldsymbol{\theta}} \mathbb{E}_{p_{data}(\mathbf{x})} \mathbb{E}_{q_\sigma(\tilde{\mathbf{x}}|\mathbf{x})} \left[\left\| \mathbf{x}\mathbf{x}^T - \mathbf{f}(\tilde{\mathbf{x}}, \tilde{\mathbf{s}}_1(\tilde{\mathbf{x}}), \tilde{\mathbf{s}}_2(\tilde{\mathbf{x}}; \boldsymbol{\theta})) \right\|_2^2 \right], \quad (4.11)$$

$$\boldsymbol{\theta}^* = \arg \min_{\boldsymbol{\theta}} \mathbb{E}_{p_{data}(\mathbf{x})} \mathbb{E}_{q_\sigma(\tilde{\mathbf{x}}|\mathbf{x})} \left[\left\| \mathbf{x}\mathbf{x}^T - \mathbf{x}\tilde{\mathbf{x}}^T - \tilde{\mathbf{x}}\mathbf{x}^T - \mathbf{h}(\tilde{\mathbf{x}}, \tilde{\mathbf{s}}_1(\tilde{\mathbf{x}}), \tilde{\mathbf{s}}_2(\tilde{\mathbf{x}}; \boldsymbol{\theta})) \right\|_2^2 \right] \quad (4.12)$$

where $\mathbf{f}(\cdot)$ and $\mathbf{h}(\cdot)$ are polynomials defined in Eq. (4.9) and Eq. (4.10). Assuming the model

has an infinite capacity, then the optimal parameter $\boldsymbol{\theta}^*$ satisfies $\tilde{\mathbf{s}}_2(\tilde{\mathbf{x}}; \boldsymbol{\theta}^*) = \tilde{\mathbf{s}}_2(\tilde{\mathbf{x}})$ for almost any $\tilde{\mathbf{x}}$.

Here Eq. (4.11) and Eq. (4.12) correspond to the least squares objective of Eq. (4.7) and Eq. (4.8) respectively, and have the same set of solutions assuming sufficient model capacity. In practice, we find that Eq. (4.12) has a much simpler form than Eq. (4.11), and will therefore use Eq. (4.12) in our experiments.

4.3.3 High Order Denoising Score Matching

Below we generalize our approach to even higher order scores by (i) leveraging Tweedie's formula to connect higher order moments of \mathbf{x} conditioned on $\tilde{\mathbf{x}}$ to higher order scores of $q_\sigma(\tilde{\mathbf{x}})$; and (ii) finding the corresponding least squares objective.

Theorem 4.3. $\mathbb{E}[\otimes^n \mathbf{x} | \tilde{\mathbf{x}}] = \mathbf{f}_n(\tilde{\mathbf{x}}, \tilde{\mathbf{s}}_1, \dots, \tilde{\mathbf{s}}_n)$, where $\otimes^n \mathbf{x} \in \mathbb{R}^{D^n}$ denotes n -fold tensor multiplications, $\mathbf{f}_n(\tilde{\mathbf{x}}, \tilde{\mathbf{s}}_1, \dots, \tilde{\mathbf{s}}_n)$ is a polynomial of $\{\tilde{\mathbf{x}}, \tilde{\mathbf{s}}_1(\tilde{\mathbf{x}}), \dots, \tilde{\mathbf{s}}_n(\tilde{\mathbf{x}})\}$ and $\tilde{\mathbf{s}}_k(\tilde{\mathbf{x}})$ represents the k -th order score of $q_\sigma(\tilde{\mathbf{x}}) = \int p_{\text{data}}(\mathbf{x}) q_\sigma(\tilde{\mathbf{x}} | \mathbf{x}) d\mathbf{x}$.

Theorem 4.3 shows that there exists an equality between (high order) moments of the posterior distribution of \mathbf{x} given $\tilde{\mathbf{x}}$ and (high order) scores with respect to $\tilde{\mathbf{x}}$. To get some intuition, for $n = 2$ the polynomial $\mathbf{f}_2(\tilde{\mathbf{x}}, \tilde{\mathbf{s}}_1, \tilde{\mathbf{s}}_2)$ is simply the function \mathbf{f} in Eq. (4.9). In Appendix A.2, we provide a recursive formula for obtaining the coefficients of \mathbf{f}_n in closed form.

Leveraging Theorem 4.3 and the least squares estimation of $\mathbb{E}[\otimes^k \mathbf{x} | \tilde{\mathbf{x}}]$, we can construct objectives for approximating the k -th order scores $\tilde{\mathbf{s}}_k(\tilde{\mathbf{x}})$ as in the following theorem.

Theorem 4.4. Given score functions $\tilde{\mathbf{s}}_1(\tilde{\mathbf{x}}), \dots, \tilde{\mathbf{s}}_{k-1}(\tilde{\mathbf{x}})$, a k -th order score model $\tilde{\mathbf{s}}_k(\tilde{\mathbf{x}}; \boldsymbol{\theta})$, and

$$\boldsymbol{\theta}^* = \arg \min_{\boldsymbol{\theta}} \mathbb{E}_{p_{\text{data}}(\mathbf{x})} \mathbb{E}_{q_\sigma(\tilde{\mathbf{x}} | \mathbf{x})} [\| \otimes^k \mathbf{x} - \mathbf{f}_k(\tilde{\mathbf{x}}, \tilde{\mathbf{s}}_1(\tilde{\mathbf{x}}), \dots, \tilde{\mathbf{s}}_{k-1}(\tilde{\mathbf{x}}), \tilde{\mathbf{s}}_k(\tilde{\mathbf{x}}; \boldsymbol{\theta})) \|^2].$$

We have $\tilde{\mathbf{s}}_k(\tilde{\mathbf{x}}; \boldsymbol{\theta}^*) = \tilde{\mathbf{s}}_k(\tilde{\mathbf{x}})$ for almost all $\tilde{\mathbf{x}}$.

As previously discussed, when σ approaches 0 such that $q_\sigma(\tilde{\mathbf{x}}) \approx p_{\text{data}}(\mathbf{x})$, $\tilde{\mathbf{s}}_k(\tilde{\mathbf{x}}; \boldsymbol{\theta}^*)$ well-approximates the k -th order score of $p_{\text{data}}(\mathbf{x})$.

4.4 Learning Second Order Score Models

Although our theory can be applied to scores of any order, we focus on second order scores for empirical analysis. In this section, we discuss the parameterization and empirical performance of the learned second order score models.

4.4.1 Instantiating Objectives for Second Order Score Models

In practice, we find that Eq. (4.12) has a much simpler expression than Eq. (4.11). Therefore, we propose to parameterize $\tilde{\mathbf{s}}_2(\tilde{\mathbf{x}})$ with a model $\tilde{\mathbf{s}}_2(\tilde{\mathbf{x}}; \boldsymbol{\theta})$, and optimize $\tilde{\mathbf{s}}_2(\tilde{\mathbf{x}}; \boldsymbol{\theta})$ with Eq. (4.12), which can be simplified to the following after combining Eq. (4.10) and Eq. (4.12):

$$\mathcal{L}_{D_2SM}(\boldsymbol{\theta}) := \mathbb{E}_{p_{\text{data}}(\mathbf{x})} \mathbb{E}_{q_\sigma(\tilde{\mathbf{x}}|\mathbf{x})} \left[\left\| \tilde{\mathbf{s}}_2(\tilde{\mathbf{x}}; \boldsymbol{\theta}) + \tilde{\mathbf{s}}_1(\tilde{\mathbf{x}}; \boldsymbol{\theta}) \tilde{\mathbf{s}}_1(\tilde{\mathbf{x}}; \boldsymbol{\theta})^\top + \frac{I - \mathbf{z}\mathbf{z}^\top}{\sigma^2} \right\|_2^2 \right], \quad (4.13)$$

where $\mathbf{z} := \frac{\tilde{\mathbf{x}} - \mathbf{x}}{\sigma}$. Note that Eq. (4.13) requires knowing the first order score $\tilde{\mathbf{s}}_1(\tilde{\mathbf{x}})$ in order to train the second order score model $\tilde{\mathbf{s}}_2(\tilde{\mathbf{x}}; \boldsymbol{\theta})$. We therefore use the following hybrid objective to simultaneously train both $\tilde{\mathbf{s}}_1(\tilde{\mathbf{x}}; \boldsymbol{\theta})$ and $\tilde{\mathbf{s}}_2(\tilde{\mathbf{x}}; \boldsymbol{\theta})$:

$$\mathcal{L}_{\text{joint}}(\boldsymbol{\theta}) = \mathcal{L}_{D_2SM}(\boldsymbol{\theta}) + \gamma \cdot \mathcal{L}_{DSM}(\boldsymbol{\theta}), \quad (4.14)$$

where $\mathcal{L}_{DSM}(\boldsymbol{\theta})$ is defined in Eq. (4.3) and $\gamma \in \mathbb{R}_{>0}$ is a tunable coefficient. The expectation for $\mathcal{L}_{D_2SM}(\boldsymbol{\theta})$ and $\mathcal{L}_{DSM}(\boldsymbol{\theta})$ in Eq. (4.14) can be estimated with samples, and we optimize the following unbiased estimator

$$\hat{\mathcal{L}}_{\text{joint}}(\boldsymbol{\theta}) = \frac{1}{N} \sum_{i=1}^N \left[\left\| \tilde{\mathbf{s}}_2(\tilde{\mathbf{x}}_i; \boldsymbol{\theta}) + \tilde{\mathbf{s}}_1(\tilde{\mathbf{x}}_i; \boldsymbol{\theta}) \tilde{\mathbf{s}}_1(\tilde{\mathbf{x}}_i; \boldsymbol{\theta})^\top + \frac{I - \mathbf{z}_i \mathbf{z}_i^\top}{\sigma^2} \right\|_2^2 + \frac{\gamma}{2} \left\| \tilde{\mathbf{s}}_1(\tilde{\mathbf{x}}_i; \boldsymbol{\theta}) + \frac{\mathbf{z}_i}{\sigma} \right\|_2^2 \right], \quad (4.15)$$

where we define $\mathbf{z}_i := \frac{\tilde{\mathbf{x}}_i - \mathbf{x}_i}{\sigma}$, and $\{\tilde{\mathbf{x}}_i\}_{i=1}^N$ are samples from $q_\sigma(\tilde{\mathbf{x}}) = \int p_{\text{data}}(\mathbf{x}) q_\sigma(\tilde{\mathbf{x}}|\mathbf{x}) d\mathbf{x}$ which can be obtained by adding noise to samples from $p_{\text{data}}(\mathbf{x})$. Similarly to DSM, when $\sigma \rightarrow 0$, the optimal model $\tilde{\mathbf{s}}_2(\tilde{\mathbf{x}}; \boldsymbol{\theta}^*)$ that minimizes Eq. (4.15) will be close to the second order score of $p_{\text{data}}(\mathbf{x})$ because $q_\sigma(\tilde{\mathbf{x}}) \approx p_{\text{data}}(\mathbf{x})$. When σ is large, the learned $\tilde{\mathbf{s}}_2(\tilde{\mathbf{x}}; \boldsymbol{\theta})$ can be applied to tasks such as uncertainty quantification for denoising, which will be discussed in Section 4.5.

For downstream tasks that require only the diagonal of $\tilde{\mathbf{s}}_2$, we can instead optimize a

simpler objective

$$\mathcal{L}_{\text{joint-diag}}(\boldsymbol{\theta}) := \mathcal{L}_{D_2\text{SM-diag}}(\boldsymbol{\theta}) + \gamma \cdot \mathcal{L}_{\text{DSM}}(\boldsymbol{\theta}), \quad \text{where} \quad (4.16)$$

$$\mathcal{L}_{D_2\text{SM-diag}}(\boldsymbol{\theta}) := \mathbb{E}_{p_{\text{data}}(\mathbf{x})} \mathbb{E}_{q_\sigma(\tilde{\mathbf{x}}|\mathbf{x})} \left[\left\| \text{diag}(\tilde{\mathbf{s}}_2(\tilde{\mathbf{x}}; \boldsymbol{\theta})) + \tilde{\mathbf{s}}_1(\tilde{\mathbf{x}}; \boldsymbol{\theta}) \odot \tilde{\mathbf{s}}_1(\tilde{\mathbf{x}}; \boldsymbol{\theta}) + \frac{\mathbf{1} - \mathbf{z} \odot \mathbf{z}^\top}{\sigma^2} \right\|_2^2 \right]. \quad (4.17)$$

Here $\text{diag}(\cdot)$ denotes the diagonal of a matrix and \odot denotes element-wise multiplication. Optimizing Eq. (4.16) only requires parameterizing $\text{diag}(\tilde{\mathbf{s}}_2(\tilde{\mathbf{x}}; \boldsymbol{\theta}))$, which can significantly reduce the memory and computational cost for training and running the second order score model. Similar to $\hat{\mathcal{L}}_{\text{joint}}(\boldsymbol{\theta})$, we estimate the expectation in Eq. (4.17) with empirical means.

4.4.2 Parameterizing Second Order Score Models

In practice, the performance of learning second order scores is affected by model parameterization. As many real world data distributions (*e.g.*, images) tend to lie on low dimensional manifolds [NM10; DF08; SR03], we propose to parametrize $\tilde{\mathbf{s}}_2(\tilde{\mathbf{x}}; \boldsymbol{\theta})$ with low rank matrices defined as below

$$\tilde{\mathbf{s}}_2(\tilde{\mathbf{x}}; \boldsymbol{\theta}) = \boldsymbol{\alpha}(\tilde{\mathbf{x}}; \boldsymbol{\theta}) + \boldsymbol{\beta}(\tilde{\mathbf{x}}; \boldsymbol{\theta}) \boldsymbol{\beta}(\tilde{\mathbf{x}}; \boldsymbol{\theta})^\top,$$

where $\boldsymbol{\alpha}(\cdot; \boldsymbol{\theta}) : \mathbb{R}^D \rightarrow \mathbb{R}^{D \times D}$ is a diagonal matrix, $\boldsymbol{\beta}(\cdot; \boldsymbol{\theta}) : \mathbb{R}^D \rightarrow \mathbb{R}^{D \times r}$ is a matrix with shape $D \times r$, and $r \leq D$ is a positive integer.

4.4.3 Antithetic Sampling for Variance Reduction

As the standard deviation of the perturbed noise σ approximates zero, training score models with denoising methods could suffer from a high variance. Inspired by a variance reduction method for DSM [Wan+20; SK21], we propose a variance reduction method for $D_2\text{SM}$

$$\mathcal{L}_{D_2\text{SM-VR}} = \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x})} \mathbb{E}_{\mathbf{z} \sim \mathcal{N}(0, I)} \left[\psi(\tilde{\mathbf{x}}_+)^2 + \psi(\tilde{\mathbf{x}}_-)^2 + 2 \frac{\mathbf{I} - \mathbf{z} \mathbf{z}^\top}{\sigma^2} \odot (\psi(\tilde{\mathbf{x}}_+) + \psi(\tilde{\mathbf{x}}_-) - 2\psi(\mathbf{x})) \right],$$

where $\tilde{\mathbf{x}}_+ = \mathbf{x} + \sigma \mathbf{z}$, $\tilde{\mathbf{x}}_- = \mathbf{x} - \sigma \mathbf{z}$ and $\psi = \tilde{\mathbf{s}}_2 + \tilde{\mathbf{s}}_1 \tilde{\mathbf{s}}_1^\top$. Instead of using independent noise samples, we apply antithetic sampling and use two correlated (opposite) noise vectors centered at \mathbf{x} . Similar to Eq. (4.14), we define $\mathcal{L}_{\text{joint-VR}} = \mathcal{L}_{D_2\text{SM-VR}} + \gamma \cdot \mathcal{L}_{\text{DSM-VR}}$, where $\mathcal{L}_{\text{DSM-VR}}$ is proposed in [Wan+20].

We empirically study the role of variance reduction (VR) in training models with DSM and $D_2\text{SM}$. We observe that VR is crucial for both DSM and $D_2\text{SM}$ when σ is approximately

zero, but is optional when σ is large enough. To see this, we consider a 2-d Gaussian distribution $\mathcal{N}(0, I)$ and train $\tilde{\mathbf{s}}_1(\tilde{\mathbf{x}}; \boldsymbol{\theta})$ and $\tilde{\mathbf{s}}_2(\tilde{\mathbf{x}}; \boldsymbol{\theta})$ with DSM and $D_2\text{SM}$ respectively. We plot the learning curves in Figs. 4.1a and 4.1b, and visualize the first dimension of the estimated scores for multiple noise scales σ in Figs. 4.1c and 4.1d. We observe that when $\sigma = 0.001$, both DSM and $D_2\text{SM}$ have trouble converging after a long period of training, while the VR counterparts converge quickly (see Fig. 4.1). When σ gets larger, DSM and $D_2\text{SM}$ without VR can both converge quickly and provide reasonable score estimations (Figs. 4.1c and 4.1d). We provide extra details in Appendix B.2.1.

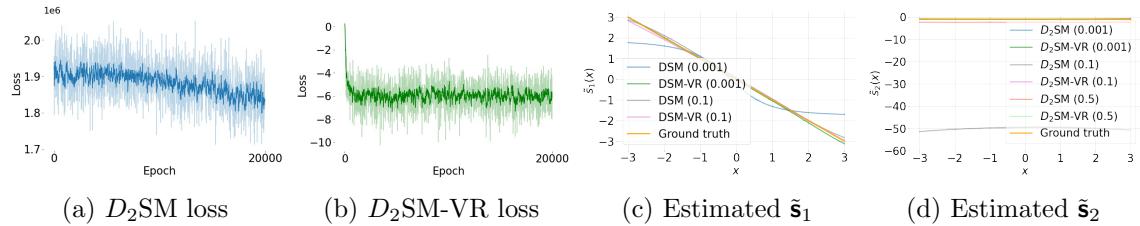


Figure 4.1: From left to right: (a) $D_2\text{SM}$ loss without variance reduction ($\sigma = 10^{-3}$). (b) $D_2\text{SM}$ loss with variance reduction ($\sigma = 10^{-3}$). (c) Estimated $\tilde{\mathbf{s}}_1$. (d) Estimated $\tilde{\mathbf{s}}_2$, where the estimation for $D_2\text{SM}$ (0.001) is too far from the ground truth to appear on the plot.

4.4.4 The Accuracy and Efficiency of Learning Second Order Scores

We show that the proposed method can estimate second order scores more efficiently and accurately than those obtained by automatic differentiation of a first order score model trained with DSM. We observe in our experiments that $\tilde{\mathbf{s}}_1(\tilde{\mathbf{x}}; \boldsymbol{\theta})$ jointly optimized via $\hat{\mathcal{L}}_{\text{joint}}$ or $\hat{\mathcal{L}}_{\text{joint-diag}}$ has a comparable empirical performance as trained directly by DSM, so we optimize $\tilde{\mathbf{s}}_1(\tilde{\mathbf{x}}; \boldsymbol{\theta})$ and $\tilde{\mathbf{s}}_2(\tilde{\mathbf{x}}; \boldsymbol{\theta})$ jointly in later experiments. We provide additional experimental details in Appendix B.2.1.

Learning accuracy We consider three synthetic datasets whose ground truth scores are available—a 100-dimensional correlated multivariate normal distribution and two high dimensional mixture of logistics distributions in Table 4.1. We study the performance of estimating $\tilde{\mathbf{s}}_2$ and the diagonal of $\tilde{\mathbf{s}}_2$. For the baseline, we estimate *second order scores* by taking automatic differentiation of $\tilde{\mathbf{s}}_1(\tilde{\mathbf{x}}; \boldsymbol{\theta})$ trained jointly with $\tilde{\mathbf{s}}_2(\tilde{\mathbf{x}}; \boldsymbol{\theta})$ using Eq. (4.15) or Eq. (4.17). As mentioned previously, $\tilde{\mathbf{s}}_1(\tilde{\mathbf{x}}; \boldsymbol{\theta})$ trained with the joint method has the same empirical performance as trained directly with DSM. For our method, we directly evaluate $\tilde{\mathbf{s}}_2(\tilde{\mathbf{x}}; \boldsymbol{\theta})$. We compute the mean squared error between estimated *second order scores* and

the ground truth score of the *clean* data since we use small σ and $q_\sigma(\tilde{\mathbf{x}}) \approx p_{\text{data}}(\mathbf{x})$ (see Table 4.1). We observe that $\tilde{\mathbf{s}}_2(\tilde{\mathbf{x}}; \boldsymbol{\theta})$ achieves better performance than the gradients of $\tilde{\mathbf{s}}_1(\tilde{\mathbf{x}}; \boldsymbol{\theta})$.

Table 4.1: Mean squared error between the estimated *second order scores* and the ground truth on 10^5 test samples. Each setup is trained with three random seeds and multiple noise scales σ .

Methods	$\sigma = 0.01$	$\sigma = 0.05$	$\sigma = 0.1$	Methods	$\sigma = 0.01$	$\sigma = 0.05$	$\sigma = 0.1$				
Multivariate normal (100-d)											
$\tilde{\mathbf{s}}_1$ grad (DSM)	43.80 ± 0.012	43.76 ± 0.001	43.75 ± 0.001	$\tilde{\mathbf{s}}_1$ grad (DSM-VR)	26.41 ± 0.55	26.13 ± 0.53	25.39 ± 0.50				
$\tilde{\mathbf{s}}_1$ grad (DSM-VR)	9.40 ± 0.049	9.39 ± 0.015	9.21 ± 0.020	$\tilde{\mathbf{s}}_2$ (Ours)	18.43 ± 0.11	18.50 ± 0.25	17.88 ± 0.15				
$\tilde{\mathbf{s}}_2$ (Ours, $r = 15$)	7.12 ± 0.319	6.91 ± 0.078	7.03 ± 0.039	Mixture of logistics diagonal estimation (80-d, 20 mixtures)							
$\tilde{\mathbf{s}}_2$ (Ours, $r = 20$)	5.24 ± 0.065	5.07 ± 0.047	5.13 ± 0.065	$\tilde{\mathbf{s}}_1$ grad (DSM-VR)	32.80 ± 0.34	32.44 ± 0.30	31.51 ± 0.43				
$\tilde{\mathbf{s}}_2$ (Ours, $r = 30$)	1.76 ± 0.038	2.05 ± 0.544	1.76 ± 0.045	$\tilde{\mathbf{s}}_2$ (Ours)	21.68 ± 0.18	22.23 ± 0.08	22.18 ± 0.08				

Computational efficiency Computing the gradients of $\tilde{\mathbf{s}}_1(\tilde{\mathbf{x}}; \boldsymbol{\theta})$ via automatic differentiation can be expensive for high dimensional data and deep neural networks. To see this, we consider two models—a 3-layer MLP and a U-Net [RPB15], which is used for image experiments in the subsequent sections. We consider a 100-d data distribution for the MLP model and a 784-d data distribution for the U-Net. We parameterize $\tilde{\mathbf{s}}_1$ and $\tilde{\mathbf{s}}_2$ with the same model architecture and use a batch size of 10 for both settings. We report the wall-clock time averaged in 7 runs used for estimating second order scores during test time on a TITAN Xp GPU in Table 4.2. We observe that $\tilde{\mathbf{s}}_2(\tilde{\mathbf{x}}; \boldsymbol{\theta})$ is $500\times$ faster than using automatic differentiation for $\tilde{\mathbf{s}}_1(\tilde{\mathbf{x}}; \boldsymbol{\theta})$ on the MNIST dataset.

4.5 Uncertainty Quantification with Second Order Score Models

Our second order score model $\tilde{\mathbf{s}}_2(\tilde{\mathbf{x}}; \boldsymbol{\theta})$ can capture and quantify the uncertainty of denoising on synthetic and real world image datasets, based on the following result by combining Eqs. (4.4) and (4.9)

$$\text{Cov}[\mathbf{x} \mid \tilde{\mathbf{x}}] := \mathbb{E}[\mathbf{x}\mathbf{x}^\top \mid \tilde{\mathbf{x}}] - \mathbb{E}[\mathbf{x} \mid \tilde{\mathbf{x}}]\mathbb{E}[\mathbf{x} \mid \tilde{\mathbf{x}}]^\top = \sigma^4 \tilde{\mathbf{s}}_2(\tilde{\mathbf{x}}) + \sigma^2 I \quad (4.18)$$

By estimating $\text{Cov}[\mathbf{x} \mid \tilde{\mathbf{x}}]$ via $\tilde{\mathbf{s}}_2(\tilde{\mathbf{x}})$, we gain insights into how pixels are correlated with each other under denoising settings, and which part of the pixels has large uncertainty. To

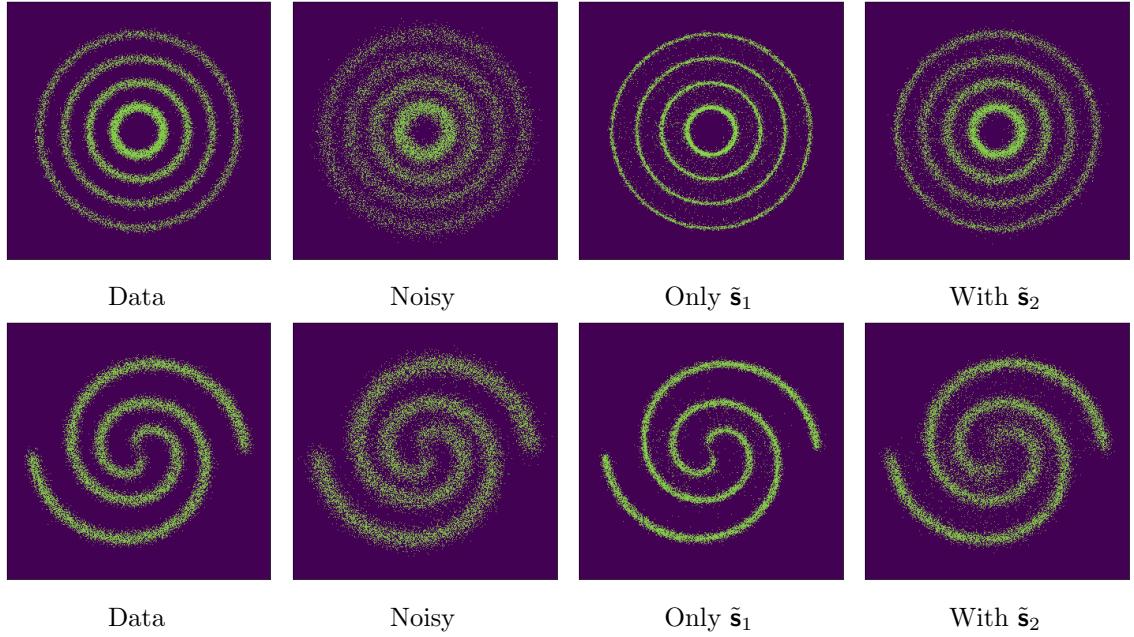


Figure 4.2: Denoising 2-d synthetic data. The incorporation of $\tilde{\mathbf{s}}_2$ improves uncertainty quantification.

examine the uncertainty given by our $\tilde{\mathbf{s}}_2(\tilde{\mathbf{x}}; \boldsymbol{\theta})$, we perform the following experiments (details in Appendix B.2.2).

Synthetic experiments We first consider 2-d synthetic datasets shown in Fig. 4.2, where we train $\tilde{\mathbf{s}}_1(\tilde{\mathbf{x}}; \boldsymbol{\theta})$ and $\tilde{\mathbf{s}}_2(\tilde{\mathbf{x}}; \boldsymbol{\theta})$ jointly with $\mathcal{L}_{\text{joint}}$. Given the trained score models, we estimate $\mathbb{E}[\mathbf{x} | \tilde{\mathbf{x}}]$ and $\text{Cov}[\mathbf{x} | \tilde{\mathbf{x}}]$ using Eq. (4.4) and Eq. (4.18). We approximate the posterior distribution $p(\mathbf{x} | \tilde{\mathbf{x}})$ with a conditional normal distribution $\mathcal{N}(\mathbf{x} | \mathbb{E}[\mathbf{x} | \tilde{\mathbf{x}}], \text{Cov}[\mathbf{x} | \tilde{\mathbf{x}}])$. We compare our result with that of Eq. (4.4), which only utilizes $\tilde{\mathbf{s}}_1$ (see Fig. 4.2). We observe that unlike Eq. (4.4), which is a point estimator, the incorporation of covariance matrices (estimated by $\tilde{\mathbf{s}}_2(\tilde{\mathbf{x}}; \boldsymbol{\theta})$) captures uncertainty in denoising.

Covariance diagonal visualizations We visualize the diagonal of the estimated $\text{Cov}[\mathbf{x} | \tilde{\mathbf{x}}]$ for MNIST and CIFAR-10 [KH+09] in Fig. 4.3. We find that the diagonal values are in general larger for pixels near the edges where there are multiple possibilities corresponding to the same noisy pixel. The diagonal values are smaller for the background pixels where there is less uncertainty. We also observe that covariance matrices corresponding to smaller noise scales tend to have smaller values on the diagonals, implying that the more noise an image has, the more uncertain the denoised results are.

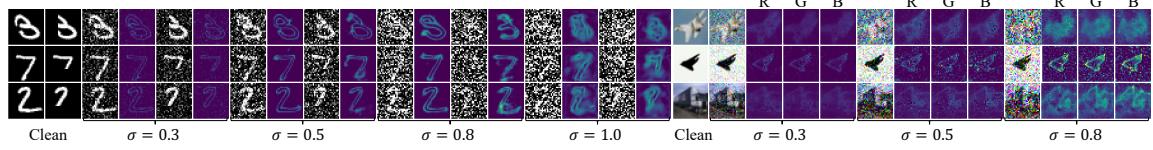


Figure 4.3: Visualizations of the estimated covariance matrix diagonals on MNIST and CIFAR-10. For CIFAR-10 images, we visualize the diagonal for R, G, B channels separately. Images corrupted with more noise tend to have larger covariance values, indicating larger uncertainty in denoising. Pixels in background have smaller values than pixels near edges, indicating more confident denoising.

Full covariance visualizations We visualize the eigenvectors (sorted by eigenvalues) of $\text{Cov}[\mathbf{x} | \tilde{\mathbf{x}}]$ estimated by $\tilde{\mathbf{s}}_2(\tilde{\mathbf{x}}; \boldsymbol{\theta})$ in Fig. 4.4. We observe that they can correspond to different digit identities, indicating uncertainty in the identity of the denoised image. This suggests $\text{Cov}[\mathbf{x} | \tilde{\mathbf{x}}]$ can capture additional information for uncertainty beyond its diagonal.

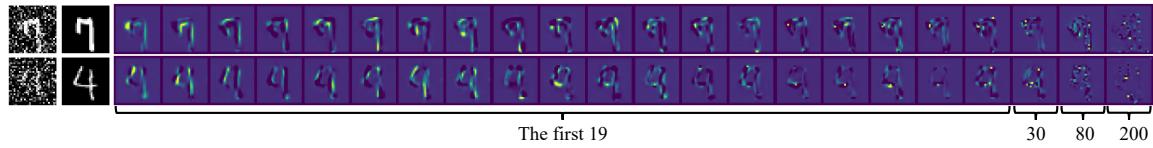


Figure 4.4: Eigenvectors of the estimated covariance matrix on MNIST. The first column shows the noisy images ($\sigma = 0.5$) and the second column shows clean images. The remaining columns show the first 19, plus the 30, 80 and 200-th eigenvectors of the matrix. We can see digit 7 and 9 in the eigenvectors corresponding to the noisy 7, and digit 4 and 9 in the second row, which implies that the estimated covariance matrix can capture different possibilities of the denoising results.

4.6 Sampling with Second Order Score Models

Here we show that our second order score model $\tilde{\mathbf{s}}_2(\tilde{\mathbf{x}}; \boldsymbol{\theta})$ can be used to improve the mixing speed of Langevin dynamics sampling.

Table 4.2: Speed analysis: direct modeling vs. autodiff.

Method \ Dimension	$D = 100$ (MLP)	$D = 784$ (U-Net)
Autodiff	$32100 \pm 156 \mu\text{s}$	$34600 \pm 194 \text{ ms}$
Ours (rank=20)	$380 \pm 7.9 \mu\text{s}$	$67.9 \pm 1.93 \text{ ms}$
Ours (rank=50)	$377 \pm 10.8 \mu\text{s}$	$72.5 \pm 1.93 \text{ ms}$
Ours (rank=200)	$546 \pm 1.91 \mu\text{s}$	$68.8 \pm 1.02 \text{ ms}$
Ours (rank=1000)	$1840 \pm 97.1 \mu\text{s}$	$69.4 \pm 2.63 \text{ ms}$

Table 4.3: ESS on synthetic datasets. Datasets are shown in Fig. 4.5. We use 32 chains, each with length 10000 and 1000 burn-in steps.

Method \ Metric	Dataset 1	Dataset 2
Langevin	ESS ↑	ESS ↑
Ozaki	21.81	26.33

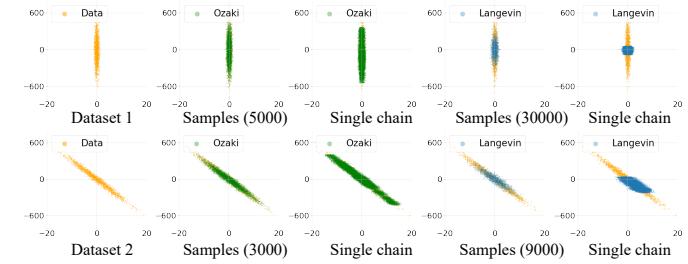


Figure 4.5: Sampling with Ozaki and Langevin dynamics. We tune the optimal step size separately for both methods. The number in the parenthesis (Column 2 and 4) stands for the iterations used for sampling. We observe that Ozaki obtains more reasonable samples than Langevin dynamics using 1/6 or 1/3 iterations. Column 3 and 5 show samples within a single chain with length 31000 and 1000 burn-in steps.

4.6.1 Background on the Sampling Methods

Langevin dynamics Langevin dynamics [BP07; WT11] sample from $p_{\text{data}}(\mathbf{x})$ using the first order score function $\mathbf{s}_1(\mathbf{x})$. Given a prior distribution $\pi(\mathbf{x})$, a fixed step size $\epsilon > 0$ and an initial value $\tilde{\mathbf{x}}_0 \sim \pi(\mathbf{x})$, Langevin dynamics update the samples iteratively as follows

$$\tilde{\mathbf{x}}_t = \tilde{\mathbf{x}}_{t-1} + \frac{\epsilon}{2} \mathbf{s}_1(\tilde{\mathbf{x}}_{t-1}) + \sqrt{\epsilon} \mathbf{z}_t, \quad (4.19)$$

where $\mathbf{z}_t \sim \mathcal{N}(0, I)$. As $\epsilon \rightarrow 0$ and $t \rightarrow \infty$, $\tilde{\mathbf{x}}_t$ is a sample from $p_{\text{data}}(\mathbf{x})$ under suitable conditions.

Ozaki sampling Langevin dynamics with Ozaki discretization [ST99] leverages second order information in $\mathbf{s}_2(\mathbf{x})$ to pre-condition Langevin dynamics:

$$\tilde{\mathbf{x}}_t = \tilde{\mathbf{x}}_{t-1} + M_{t-1} \mathbf{s}_1(\tilde{\mathbf{x}}_{t-1}) + \Sigma_{t-1}^{1/2} \mathbf{z}_t, \quad \mathbf{z}_t \sim \mathcal{N}(0, I) \quad (4.20)$$

where $M_{t-1} = (e^{\epsilon \mathbf{s}_2(\tilde{\mathbf{x}}_{t-1})} - I) \mathbf{s}_2(\tilde{\mathbf{x}}_{t-1})^{-1}$ and $\Sigma_{t-1} = (e^{2\epsilon \mathbf{s}_2(\tilde{\mathbf{x}}_{t-1})} - I) \mathbf{s}_2(\tilde{\mathbf{x}}_{t-1})^{-1}$. It is shown that under certain conditions, this variation can improve the convergence rate of Langevin dynamics [DK19]. In general, Eq. (4.20) is expensive to compute due to inversion, exponentiation and taking square root of matrices, so we simplify Eq. (4.20) by approximating $\mathbf{s}_2(\tilde{\mathbf{x}}_{t-1})$ with its diagonal in practice.

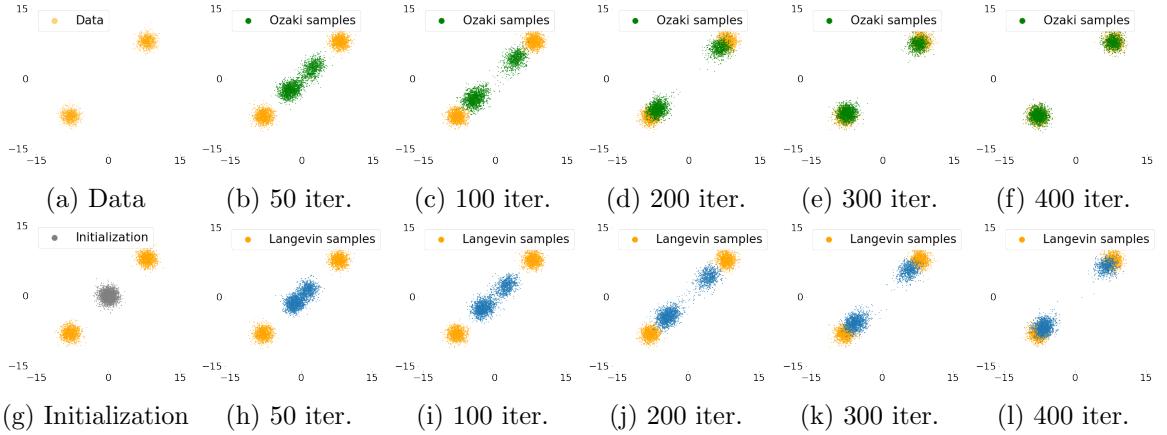


Figure 4.6: Sampling a two mode distribution. We use the same step size $\epsilon = 0.01$ for both methods. We observe that Ozaki sampling converges faster than Langevin sampling.

In our experiments, we only consider Ozaki sampling with \mathbf{s}_2 replaced by its diagonal in Eq. (4.20). As we use small σ , $\tilde{\mathbf{s}}_1 \approx \mathbf{s}_1$ and $\tilde{\mathbf{s}}_2 \approx \mathbf{s}_2$. We observe that $\text{diag}(\tilde{\mathbf{s}}_2(\tilde{\mathbf{x}}; \boldsymbol{\theta}))$ in Ozaki sampling can be computed in parallel with $\tilde{\mathbf{s}}_1(\tilde{\mathbf{x}}; \boldsymbol{\theta})$ on modern GPUs, making the wall-clock time per iteration of Ozaki sampling comparable to that of Langevin dynamics. Since we only use the diagonal of $\tilde{\mathbf{s}}_2(\tilde{\mathbf{x}}; \boldsymbol{\theta})$ in sampling, we can directly learn the diagonal of $\tilde{\mathbf{s}}_2(\tilde{\mathbf{x}})$ with Eq. (4.16).

4.6.2 Synthetic Datasets

We first consider 2-d synthetic datasets in Fig. 4.5 to compare the mixing speed of Ozaki sampling with Langevin dynamics. We search the optimal step size for each method and observe that Ozaki sampling can use a larger step size and converge faster than Langevin dynamics (see Fig. 4.5). We use the optimal step size for both methods and report the smallest effective sample size (ESS) of all the dimensions [SZE17; GC11] in Table 4.3. We observe that Ozaki sampling has better ESS values than Langevin dynamics, implying faster mixing speed. Even when using the same step size, Ozaki sampling still converges faster than Langevin dynamics on the two-model Gaussian dataset we consider (see Fig. 4.6). In all the experiments, we use $\sigma = 0.1$ and we provide more experimental details in Appendix B.2.3.

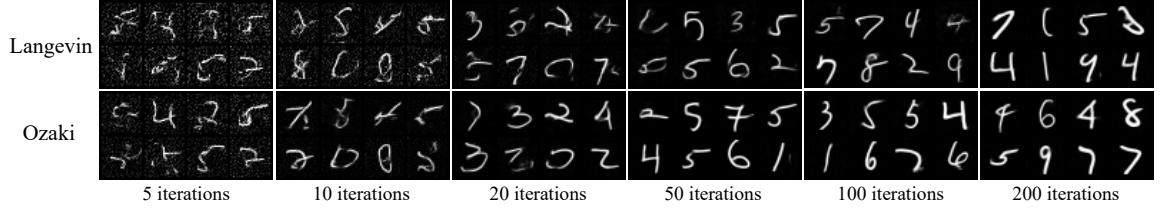


Figure 4.7: Sampling on MNIST. We observe that Ozaki sampling converges faster than Langevin dynamics. We use step size $\sigma = 0.02$ and initialize the chain with Gaussian noise for both methods.

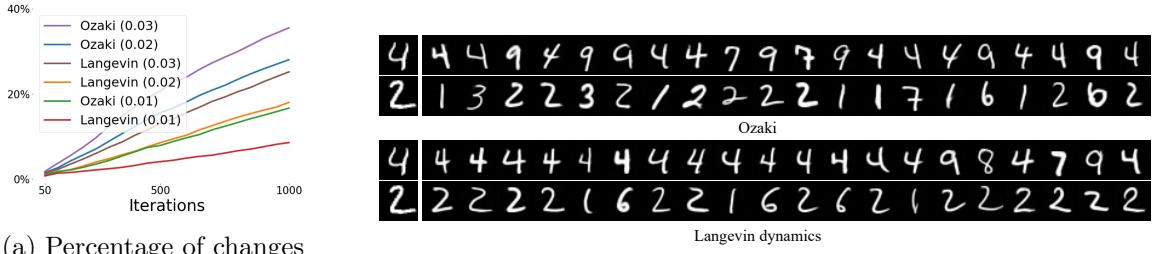


Figure 4.8: Sample diversity analysis. The number in the parenthesis in Fig. 4.8a denotes the step size. We initialize the chain with MNIST test images and report the percentage of images that have changed class labels from the initialization w.r.t. sampling iterations. We observe that Ozaki sampling has more diverse samples.

4.6.3 Image Datasets

Ozaki discretization with learned $\tilde{\mathbf{s}}_2(\tilde{\mathbf{x}}; \boldsymbol{\theta})$ produces more diverse samples and improve the mixing speed of Langevin dynamics on image datasets (see Fig. 4.7). To see this, we select ten different digits from MNIST test set and initialize 1000 different sampling chains for each image. We update the chains with Ozaki sampling and report the percentage of images that have class label changes after a fixed number of sampling iterations in Fig. 4.8a. We compare the results with Langevin dynamics with the same setting and observe that Ozaki sampling has more diverse samples within the same chain in a fixed amount of iterations. We provide more details in Appendix B.2.3.

4.7 Conclusion

We propose a method to directly estimate *high order scores* of a data density from samples. We first study the connection between Tweedie’s formula and denoising score matching (DSM) through the lens of least squares regression. We then leverage Tweedie’s formula on higher order moments, which allows us to generalize denoising score matching to estimate scores of any desired order. We demonstrate empirically that models trained with the proposed method can approximate second order scores more efficiently and accurately than applying automatic differentiation to a learned first order score model. In addition, we show that our models can be used to quantify uncertainty in denoising and to improve the mixing speed of Langevin dynamics via Ozaki discretization for sampling synthetic data and natural images. Besides the applications studied in this chapter, it would be interesting to study the application of high order scores for out of distribution detection. Due to limited computational resources, we only consider low resolution image datasets in this chapter. However, as a direct next step, we can apply our method to higher-resolution image datasets and explore its application to improve the sampling speed of score-based models [SE19b; SE20; HJA20] with Ozaki sampling. In general, when approximating the high-order scores with a diagonal or a low rank matrix, our training cost is comparable to standard denoising score matching, which is scalable to higher dimensional data. A larger rank typically requires more computation but could give better approximations to second-order scores. While we focused on images, this approach is likely applicable to other data modalities such as speech.

Part II

Generating Data Samples with Score Models

Chapter 5

Data Generation via Annealed Langevin Dynamics

Once a score model has been trained from data, how can we generate new samples from it? In this chapter, we introduce a new generative model whose samples are produced via Langevin dynamics using gradients of the data distribution estimated with score matching. Because gradients can be ill-defined and hard to estimate when the data resides on low-dimensional manifolds, we perturb the data with different levels of Gaussian noise, and jointly estimate the corresponding scores, *i.e.*, the vector fields of gradients of the perturbed data distribution for all noise levels. For sampling, we propose an annealed Langevin dynamics where we use gradients corresponding to gradually decreasing noise levels as the sampling process gets closer to the data manifold. Our framework allows flexible model architectures, requires no sampling during training or the use of adversarial methods, and provides a learning objective that can be used for principled model comparisons. Our models produce samples comparable to GANs on MNIST, CelebA and CIFAR-10 datasets, achieving a new state-of-the-art inception score of 8.87 on CIFAR-10. Additionally, we demonstrate that our models learn effective representations via image inpainting experiments.

This chapter was previously published as [SE19b].

5.1 Introduction

Generative models have many applications in machine learning. To list a few, they have been used to generate high-fidelity images [KLA19; BDS19], synthesize realistic speech and music

fragments [Oor+16a], improve the performance of semi-supervised learning [Kin+14; Dai+17], detect adversarial examples and other anomalous data [Son+18a], imitation learning [HE16], and explore promising states in reinforcement learning [Ost+17]. Recent progress is mainly driven by two approaches: likelihood-based methods [Gra13; KW14; DKB15; VKK16] and generative adversarial networks (GAN [Goo+14]). The former uses log-likelihood (or a suitable surrogate) as the training objective, while the latter uses adversarial training to minimize f -divergences [NCT16] or integral probability metrics [ACB17; Sri+09] between model and data distributions.

Although likelihood-based models and GANs have achieved great success, they have some intrinsic limitations. For example, likelihood-based models either have to use specialized architectures to build a normalized probability model (*e.g.*, autoregressive models, flow models), or use surrogate losses (*e.g.*, the evidence lower bound used in variational auto-encoders [KW14], contrastive divergence in energy-based models [Hin02]) for training. GANs avoid some of the limitations of likelihood-based models, but their training can be unstable due to the adversarial training procedure. In addition, the GAN objective is not suitable for evaluating and comparing different GAN models. While other objectives exist for generative modeling, such as noise contrastive estimation [GH10] and minimum probability flow [SBD11], these methods typically only work well for low-dimensional data.

In this chapter, we explore a new principle for generative modeling based on estimating and sampling from the *(Stein) score* [LLJ16] of the logarithmic data density, which is the gradient of the log-density function at the input data point. This is a vector field pointing in the direction where the log data density grows the most. We use a neural network trained with score matching [HD05] to learn this vector field from data. We then produce samples using Langevin dynamics, which approximately works by gradually moving a random initial sample to high density regions along the (estimated) vector field of scores. However, there are two main challenges with this approach. First, if the data distribution is supported on a low dimensional manifold—as it is often assumed for many real world datasets—the score will be undefined in the ambient space, and score matching will fail to provide a consistent score estimator. Second, the scarcity of training data in low data density regions, *e.g.*, far from the manifold, hinders the accuracy of score estimation and slows down the mixing of Langevin dynamics sampling. Since Langevin dynamics will often be initialized in low-density regions of the data distribution, inaccurate score estimation in these regions will negatively affect the sampling process. Moreover, mixing can be difficult because of the

need of traversing low density regions to transition between modes of the distribution.

To tackle these two challenges, we propose to *perturb the data with random Gaussian noise of various magnitudes*. Adding random noise ensures the resulting distribution does not collapse to a low dimensional manifold. Large noise levels will produce samples in low density regions of the original (unperturbed) data distribution, thus improving score estimation. Crucially, we train a single score network conditioned on the noise level and estimate the scores at all noise magnitudes. We then propose *an annealed version of Langevin dynamics*, where we initially use scores corresponding to the highest noise level, and gradually anneal down the noise level until it is small enough to be indistinguishable from the original data distribution. Our sampling strategy is inspired by simulated annealing [KGV83; Nea01] which heuristically improves optimization for multimodal landscapes.

Our approach has several desirable properties. First, our objective is tractable for almost all parameterizations of the score networks without the need of special constraints or architectures, and can be optimized without adversarial training, MCMC sampling, or other approximations during training. The objective can also be used to quantitatively compare different models on the same dataset. Experimentally, we demonstrate the efficacy of our approach on MNIST, CelebA [Liu+15], and CIFAR-10 [KH+09]. We show that the samples look comparable to those generated from modern likelihood-based models and GANs. On CIFAR-10, our model sets the new state-of-the-art inception score of 8.87 for unconditional generative models, and achieves a competitive FID score of 25.32. We show that the model learns meaningful representations of the data by image inpainting experiments.

5.2 Score-Based Generative Modeling

Suppose our dataset consists of i.i.d. samples $\{\mathbf{x}_i \in \mathbb{R}^D\}_{i=1}^N$ from an unknown data distribution $p_{\text{data}}(\mathbf{x})$. We define the *score* of a probability density $p(\mathbf{x})$ to be $\nabla_{\mathbf{x}} \log p(\mathbf{x})$. The *score network* $s_{\boldsymbol{\theta}} : \mathbb{R}^D \rightarrow \mathbb{R}^D$ is a neural network parameterized by $\boldsymbol{\theta}$, which will be trained to approximate the score of $p_{\text{data}}(\mathbf{x})$. The goal of generative modeling is to use the dataset to learn a model for generating new samples from $p_{\text{data}}(\mathbf{x})$. The framework of score-based generative modeling has two ingredients: score matching and Langevin dynamics.

5.2.1 Score Matching for Score Estimation

Score matching [HD05] is originally designed for learning non-normalized statistical models based on i.i.d. samples from an unknown data distribution. Following Chapter 3, we repurpose it for score estimation. Using score matching, we can directly train a score network $s_\theta(\mathbf{x})$ to estimate $\nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x})$ without training a model to estimate $p_{\text{data}}(\mathbf{x})$ first. Different from the typical usage of score matching, we opt not to use the gradient of an energy-based model as the score network to avoid extra computation due to higher-order gradients. The objective minimizes $\frac{1}{2} \mathbb{E}_{p_{\text{data}}} [\|s_\theta(\mathbf{x}) - \nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x})\|_2^2]$, which can be shown equivalent to the following up to a constant

$$\mathbb{E}_{p_{\text{data}}(\mathbf{x})} \left[\text{tr}(\nabla_{\mathbf{x}} s_\theta(\mathbf{x})) + \frac{1}{2} \|s_\theta(\mathbf{x})\|_2^2 \right], \quad (5.1)$$

where $\nabla_{\mathbf{x}} s_\theta(\mathbf{x})$ denotes the Jacobian of $s_\theta(\mathbf{x})$. As shown in Chapter 3, under some regularity conditions the minimizer of Eq. (5.3) (denoted as $s_{\theta^*}(\mathbf{x})$) satisfies $s_{\theta^*}(\mathbf{x}) = \nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x})$ almost surely. In practice, the expectation over $p_{\text{data}}(\mathbf{x})$ in Eq. (5.1) can be quickly estimated using data samples. However, score matching is not scalable to deep networks and high dimensional data (*cf.*, Chapter 3) due to the computation of $\text{tr}(\nabla_{\mathbf{x}} s_\theta(\mathbf{x}))$. Below we discuss two popular methods for large scale score matching.

Denoising score matching Denoising score matching [Vin11] is a variant of score matching that completely circumvents $\text{tr}(\nabla_{\mathbf{x}} s_\theta(\mathbf{x}))$. It first perturbs the data point \mathbf{x} with a pre-specified noise distribution $q_\sigma(\tilde{\mathbf{x}} | \mathbf{x})$ and then employs score matching to estimate the score of the perturbed data distribution $q_\sigma(\tilde{\mathbf{x}}) := \int q_\sigma(\tilde{\mathbf{x}} | \mathbf{x}) p_{\text{data}}(\mathbf{x}) d\mathbf{x}$. The objective was proved equivalent to the following:

$$\frac{1}{2} \mathbb{E}_{q_\sigma(\tilde{\mathbf{x}} | \mathbf{x}) p_{\text{data}}(\mathbf{x})} [\|s_\theta(\tilde{\mathbf{x}}) - \nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}} | \mathbf{x})\|_2^2]. \quad (5.2)$$

As shown in [Vin11], the optimal score network (denoted as $s_{\theta^*}(\mathbf{x})$) that minimizes Eq. (5.2) satisfies $s_{\theta^*}(\mathbf{x}) = \nabla_{\mathbf{x}} \log q_\sigma(\mathbf{x})$ almost surely. However, $s_{\theta^*}(\mathbf{x}) = \nabla_{\mathbf{x}} \log q_\sigma(\mathbf{x}) \approx \nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x})$ is true only when the noise is small enough such that $q_\sigma(\mathbf{x}) \approx p_{\text{data}}(\mathbf{x})$.

Sliced score matching Sliced score matching (*cf.*, Chapter 3) uses random projections to approximate $\text{tr}(\nabla_{\mathbf{x}} \mathbf{s}_{\theta}(\mathbf{x}))$ in score matching. The objective is

$$\mathbb{E}_{p_{\mathbf{v}}} \mathbb{E}_{p_{\text{data}}} \left[\mathbf{v}^{\top} \nabla_{\mathbf{x}} \mathbf{s}_{\theta}(\mathbf{x}) \mathbf{v} + \frac{1}{2} \|\mathbf{s}_{\theta}(\mathbf{x})\|_2^2 \right], \quad (5.3)$$

where $p_{\mathbf{v}}$ is a simple distribution of random vectors, *e.g.*, the multivariate standard normal. As shown in Chapter 3, the term $\mathbf{v}^{\top} \nabla_{\mathbf{x}} \mathbf{s}_{\theta}(\mathbf{x}) \mathbf{v}$ can be efficiently computed by forward mode auto-differentiation. Unlike denoising score matching which estimates the scores of *perturbed* data, sliced score matching provides score estimation for the original *unperturbed* data distribution, but requires around four times more computations due to the forward mode auto-differentiation.

5.2.2 Sampling with Langevin Dynamics

Langevin dynamics can produce samples from a probability density $p(\mathbf{x})$ using only the score function $\nabla_{\mathbf{x}} \log p(\mathbf{x})$. Given a fixed step size $\epsilon > 0$, and an initial value $\tilde{\mathbf{x}}_0 \sim \pi(\mathbf{x})$ with π being a prior distribution, the Langevin method recursively computes the following

$$\tilde{\mathbf{x}}_t = \tilde{\mathbf{x}}_{t-1} + \frac{\epsilon}{2} \nabla_{\mathbf{x}} \log p(\tilde{\mathbf{x}}_{t-1}) + \sqrt{\epsilon} \mathbf{z}_t, \quad (5.4)$$

where $\mathbf{z}_t \sim \mathcal{N}(0, \mathbf{I})$. The distribution of $\tilde{\mathbf{x}}_T$ equals $p(\mathbf{x})$ when $\epsilon \rightarrow 0$ and $T \rightarrow \infty$, in which case $\tilde{\mathbf{x}}_T$ becomes an exact sample from $p(\mathbf{x})$ under some regularity conditions [WT11]. When $\epsilon > 0$ and $T < \infty$, a Metropolis-Hastings update is needed to correct the error of Eq. (5.4), but it can often be ignored in practice [CFG14; DM19a; Nij+19]. In this chapter, we assume this error is negligible when ϵ is small and T is large.

Note that sampling from Eq. (5.4) only requires the score function $\nabla_{\mathbf{x}} \log p(\mathbf{x})$. Therefore, in order to obtain samples from $p_{\text{data}}(\mathbf{x})$, we can first train our score network such that $\mathbf{s}_{\theta}(\mathbf{x}) \approx \nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x})$ and then approximately obtain samples with Langevin dynamics using $\mathbf{s}_{\theta}(\mathbf{x})$. This is the key idea of our framework of *score-based generative modeling*.

5.3 Challenges of Score-Based Generative Modeling

In this section, we analyze more closely the idea of score-based generative modeling. We argue that there are two major obstacles that prevent a naïve application of this idea.

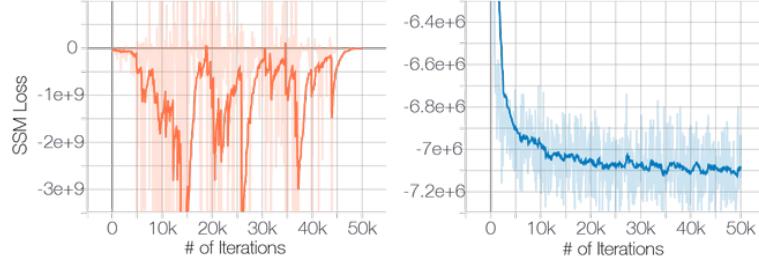


Figure 5.1: **Left:** Sliced score matching (SSM) loss w.r.t. iterations. No noise is added to data. **Right:** Same but data are perturbed with $\mathcal{N}(0, 0.0001)$.

5.3.1 The Manifold Hypothesis

The manifold hypothesis states that data in the real world tend to concentrate on low dimensional manifolds embedded in a high dimensional space (a.k.a., the ambient space). This hypothesis empirically holds for many datasets, and has become the foundation of manifold learning [BN03; RS00]. Under the manifold hypothesis, score-based generative models will face two key difficulties. First, since the score $\nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x})$ is a gradient taken in the *ambient space*, it is undefined when \mathbf{x} is confined to a low dimensional manifold. Second, the score matching objective Eq. (5.1) provides a consistent score estimator only when the support of the data distribution is the whole space (*cf.*, Theorem 2 in [HD05]), and will be inconsistent when the data reside on a low-dimensional manifold.

The negative effect of the manifold hypothesis on score estimation can be seen clearly from Fig. 5.1, where we train a ResNet (details in Appendix B.3.2) to estimate the data score on CIFAR-10. For fast training and faithful estimation of the data scores, we use the sliced score matching objective (Eq. (5.3)). As Fig. 5.1 (left) shows, when trained on the original CIFAR-10 images, the sliced score matching loss first decreases and then fluctuates irregularly. In contrast, if we perturb the data with a small Gaussian noise (such that the perturbed data distribution has full support over \mathbb{R}^D), the loss curve will converge (right panel). Note that the Gaussian noise $\mathcal{N}(0, 0.0001)$ we impose is very small for images with pixel values in the range $[0, 1]$, and is almost indistinguishable to human eyes.

5.3.2 Low Data Density Regions

The scarcity of data in low density regions can cause difficulties for both score estimation with score matching and MCMC sampling with Langevin dynamics.

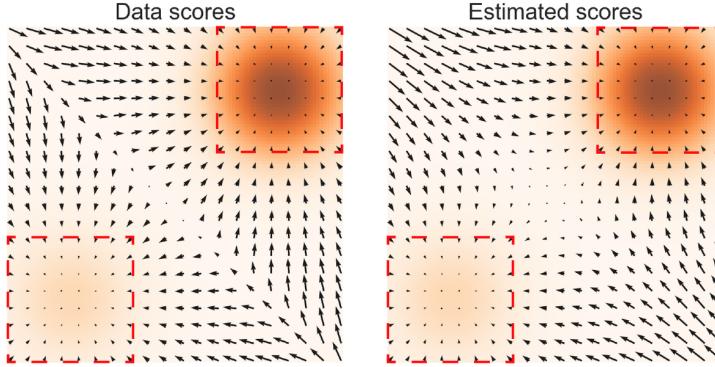


Figure 5.2: **Left:** $\nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x})$; **Right:** $s_{\theta}(\mathbf{x})$. The data density $p_{\text{data}}(\mathbf{x})$ is encoded using an orange colormap: darker color implies higher density. Red rectangles highlight regions where $\nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x}) \approx s_{\theta}(\mathbf{x})$.

Inaccurate Score Estimation with Score Matching

In regions of low data density, score matching may not have enough evidence to estimate score functions accurately, due to the lack of data samples. To see this, recall from Section 5.2.1 that score matching minimizes the expected squared error of the score estimates, *i.e.*, $\frac{1}{2}\mathbb{E}_{p_{\text{data}}}[\|\mathbf{s}_{\theta}(\mathbf{x}) - \nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x})\|_2^2]$. In practice, the expectation w.r.t. the data distribution is always estimated using i.i.d. samples $\{\mathbf{x}_i\}_{i=1}^N \stackrel{\text{i.i.d.}}{\sim} p_{\text{data}}(\mathbf{x})$. Consider any region $\mathcal{R} \subset \mathbb{R}^D$ such that $p_{\text{data}}(\mathcal{R}) \approx 0$. In most cases $\{\mathbf{x}_i\}_{i=1}^N \cap \mathcal{R} = \emptyset$, and score matching will not have sufficient data samples to estimate $\nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x})$ accurately for $\mathbf{x} \in \mathcal{R}$.

To demonstrate the negative effect of this, we provide the result of a toy experiment (details in Appendix B.3.2) in Fig. 5.2 where we use sliced score matching to estimate scores of a mixture of Gaussians $p_{\text{data}} = \frac{1}{5}\mathcal{N}((-5, -5), \mathbf{I}) + \frac{4}{5}\mathcal{N}((5, 5), \mathbf{I})$. As the figure demonstrates, score estimation is only reliable in the immediate vicinity of the modes of p_{data} , where the data density is high.

Slow Mixing of Langevin Dynamics

When two modes of the data distribution are separated by low density regions, Langevin dynamics will not be able to correctly recover the relative weights of these two modes in reasonable time, and therefore might not converge to the true distribution. Our analyses of this are largely inspired by [Wen+19], which analyzed the same phenomenon in the context of density estimation with score matching.

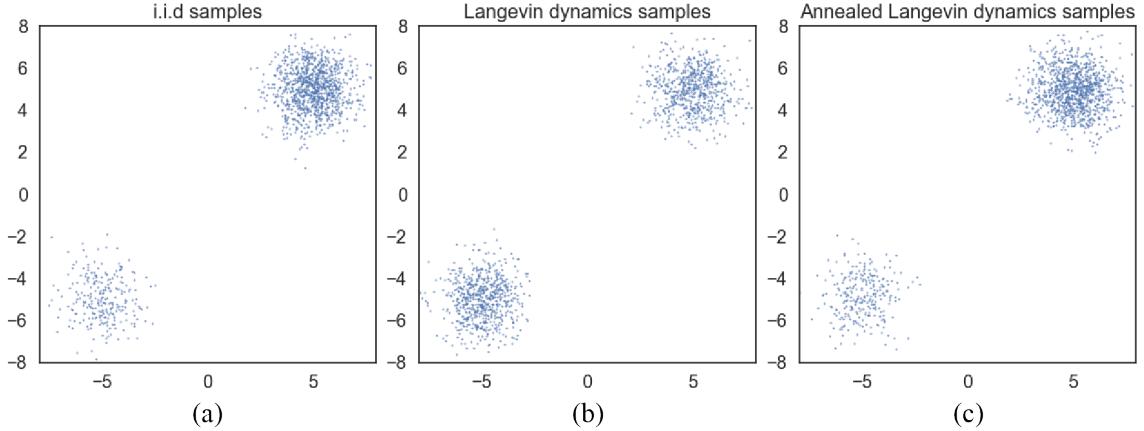


Figure 5.3: Samples from a mixture of Gaussian with different methods. (a) Exact sampling. (b) Sampling using Langevin dynamics with the exact scores. (c) Sampling using annealed Langevin dynamics with the exact scores. Clearly Langevin dynamics estimate the relative weights between the two modes incorrectly, while annealed Langevin dynamics recover the relative weights faithfully.

Consider a mixture distribution $p_{\text{data}}(\mathbf{x}) = \pi p_1(\mathbf{x}) + (1 - \pi)p_2(\mathbf{x})$, where $p_1(\mathbf{x})$ and $p_2(\mathbf{x})$ are normalized distributions with disjoint supports, and $\pi \in (0, 1)$. In the support of $p_1(\mathbf{x})$, $\nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x}) = \nabla_{\mathbf{x}}(\log \pi + \log p_1(\mathbf{x})) = \nabla_{\mathbf{x}} \log p_1(\mathbf{x})$, and in the support of $p_2(\mathbf{x})$, $\nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x}) = \nabla_{\mathbf{x}}(\log(1 - \pi) + \log p_2(\mathbf{x})) = \nabla_{\mathbf{x}} \log p_2(\mathbf{x})$. In either case, the score $\nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x})$ does not depend on π . Since Langevin dynamics use $\nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x})$ to sample from $p_{\text{data}}(\mathbf{x})$, the samples obtained will not depend on π . In practice, this analysis also holds when different modes have approximately disjoint supports—they may share the same support but be connected by regions of small data density. In this case, Langevin dynamics can produce correct samples in theory, but may require a very small step size and a very large number of steps to mix.

To verify this analysis, we test Langevin dynamics sampling for the same mixture of Gaussian used in Section 5.3.2 and provide the results in Fig. 5.3. We use the ground truth scores when sampling with Langevin dynamics. Comparing Fig. 5.3(b) with (a), it is obvious that the samples from Langevin dynamics have incorrect relative density between the two modes, as predicted by our analysis.

5.4 Noise Conditional Score Networks: Learning and Inference

We observe that perturbing data with random Gaussian noise makes the data distribution more amenable to score-based generative modeling. First, since the support of our Gaussian noise distribution is the whole space, the perturbed data will not be confined to a low dimensional manifold, which obviates difficulties from the manifold hypothesis and makes score estimation well-defined. Second, large Gaussian noise has the effect of filling low density regions in the original unperturbed data distribution; therefore score matching may get more training signal to improve score estimation. Furthermore, by using multiple noise levels we can obtain a sequence of noise-perturbed distributions that converge to the true data distribution. We can improve the mixing rate of Langevin dynamics on multimodal distributions by leveraging these intermediate distributions in the spirit of simulated annealing [KGV83] and annealed importance sampling [Nea01].

Built upon this intuition, we propose to improve score-based generative modeling by 1) *perturbing the data using various levels of noise*; and 2) *simultaneously estimating scores corresponding to all noise levels by training a single conditional score network*. After training, when using Langevin dynamics to generate samples, we initially use scores corresponding to large noise, and gradually anneal down the noise level. This helps smoothly transfer the benefits of large noise levels to low noise levels where the perturbed data are almost indistinguishable from the original ones. In what follows, we will elaborate more on the details of our method, including the architecture of our score networks, the training objective, and the annealing schedule for Langevin dynamics.

5.4.1 Noise Conditional Score Networks

Let $\{\sigma_i\}_{i=1}^L$ be a positive geometric sequence that satisfies $\frac{\sigma_1}{\sigma_2} = \dots = \frac{\sigma_{L-1}}{\sigma_L} > 1$. Let $q_\sigma(\mathbf{x}) := \int p_{\text{data}}(\mathbf{t}) \mathcal{N}(\mathbf{x} \mid \mathbf{t}, \sigma^2 \mathbf{I}) d\mathbf{t}$ denote the perturbed data distribution. We choose the noise levels $\{\sigma_i\}_{i=1}^L$ such that σ_1 is large enough to mitigate the difficulties discussed in Section 5.3, and σ_L is small enough to minimize the effect on data. We aim to train a conditional score network to jointly estimate the scores of all perturbed data distributions, *i.e.*, $\forall \sigma \in \{\sigma_i\}_{i=1}^L : \mathbf{s}_\theta(\mathbf{x}, \sigma) \approx \nabla_{\mathbf{x}} \log q_\sigma(\mathbf{x})$. Note that $\mathbf{s}_\theta(\mathbf{x}, \sigma) \in \mathbb{R}^D$ when $\mathbf{x} \in \mathbb{R}^D$. We call $\mathbf{s}_\theta(\mathbf{x}, \sigma)$ a *Noise Conditional Score Network (NCSN)*.

Similar to likelihood-based generative models and GANs, the design of model architectures

plays an important role in generating high quality samples. In this chapter, we mostly focus on architectures useful for image generation, and leave the architecture design for other domains as future work. Since the output of our noise conditional score network $s_{\theta}(\mathbf{x}, \sigma)$ has the same shape as the input image \mathbf{x} , we draw inspiration from successful model architectures for dense prediction of images (*e.g.*, semantic segmentation). In the experiments, our model $s_{\theta}(\mathbf{x}, \sigma)$ combines the architecture design of U-Net [RPB15] with dilated/atrous convolution [YK16; YKF17; Che+17]—both of which have been proved very successful in semantic segmentation. In addition, we adopt instance normalization in our score network, inspired by its superior performance in some image generation tasks [UVL16; DSK17; HB17], and we use a modified version of conditional instance normalization [DSK17] to provide conditioning on σ_i . More details on our architecture can be found in Appendix B.3.1.

5.4.2 Learning NCSNs via Score Matching

Both sliced and denoising score matching can train NCSNs. We adopt denoising score matching as it is slightly faster and naturally fits the task of estimating scores of noise-perturbed data distributions. However, we emphasize that empirically sliced score matching can train NCSNs as well as denoising score matching. We choose the noise distribution to be $q_{\sigma}(\tilde{\mathbf{x}} \mid \mathbf{x}) = \mathcal{N}(\tilde{\mathbf{x}} \mid \mathbf{x}, \sigma^2 \mathbf{I})$; therefore $\nabla_{\tilde{\mathbf{x}}} \log q_{\sigma}(\tilde{\mathbf{x}} \mid \mathbf{x}) = -(\tilde{\mathbf{x}} - \mathbf{x})/\sigma^2$. For a given σ , the denoising score matching objective (Eq. (5.2)) is

$$\ell(\boldsymbol{\theta}; \sigma) := \frac{1}{2} \mathbb{E}_{p_{\text{data}}(\mathbf{x})} \mathbb{E}_{\tilde{\mathbf{x}} \sim \mathcal{N}(\mathbf{x}, \sigma^2 \mathbf{I})} \left[\left\| s_{\boldsymbol{\theta}}(\tilde{\mathbf{x}}, \sigma) + \frac{\tilde{\mathbf{x}} - \mathbf{x}}{\sigma^2} \right\|_2^2 \right]. \quad (5.5)$$

Then, we combine Eq. (5.5) for all $\sigma \in \{\sigma_i\}_{i=1}^L$ to get one unified objective

$$\mathcal{L}(\boldsymbol{\theta}; \{\sigma_i\}_{i=1}^L) := \frac{1}{L} \sum_{i=1}^L \lambda(\sigma_i) \ell(\boldsymbol{\theta}; \sigma_i), \quad (5.6)$$

where $\lambda(\sigma_i) > 0$ is a coefficient function depending on σ_i . Assuming $s_{\boldsymbol{\theta}}(\mathbf{x}, \sigma)$ has enough capacity, $s_{\boldsymbol{\theta}*}(\mathbf{x}, \sigma)$ minimizes Eq. (5.6) if and only if $s_{\boldsymbol{\theta}*}(\mathbf{x}, \sigma_i) = \nabla_{\mathbf{x}} \log q_{\sigma_i}(\mathbf{x})$ for all $i \in \{1, 2, \dots, L\}$, because Eq. (5.6) is a conical combination of L denoising score matching objectives.

There can be many possible choices of $\lambda(\cdot)$. Ideally, we hope that the values of $\lambda(\sigma_i) \ell(\boldsymbol{\theta}; \sigma_i)$ for all $\{\sigma_i\}_{i=1}^L$ are roughly of the same order of magnitude. Empirically, we observe that

Algorithm 5.1 Annealed Langevin dynamics.

Require: $\{\sigma_i\}_{i=1}^L, \epsilon, T$.

- 1: Initialize $\tilde{\mathbf{x}}_0$
- 2: **for** $i \leftarrow 1$ to L **do**
- 3: $\alpha_i \leftarrow \epsilon \cdot \sigma_i^2 / \sigma_L^2$ $\triangleright \alpha_i$ is the step size.
- 4: **for** $t \leftarrow 1$ to T **do**
- 5: Draw $\mathbf{z}_t \sim \mathcal{N}(0, \mathbf{I})$
- 6: $\tilde{\mathbf{x}}_t \leftarrow \tilde{\mathbf{x}}_{t-1} + \frac{\alpha_i}{2} \mathbf{s}_\theta(\tilde{\mathbf{x}}_{t-1}, \sigma_i) + \sqrt{\alpha_i} \mathbf{z}_t$
- 7: **end for**
- 8: $\tilde{\mathbf{x}}_0 \leftarrow \tilde{\mathbf{x}}_T$
- 9: **end for**
- return** $\tilde{\mathbf{x}}_T$

when the score networks are trained to optimality, we approximately have $\|\mathbf{s}_\theta(\mathbf{x}, \sigma)\|_2 \propto 1/\sigma$. This inspires us to choose $\lambda(\sigma) = \sigma^2$. Because under this choice, we have $\lambda(\sigma)\ell(\boldsymbol{\theta}; \sigma) = \sigma^2\ell(\boldsymbol{\theta}; \sigma) = \frac{1}{2}\mathbb{E}[\|\sigma\mathbf{s}_\theta(\tilde{\mathbf{x}}, \sigma) + \frac{\tilde{\mathbf{x}} - \mathbf{x}}{\sigma}\|_2^2]$. Since $\frac{\tilde{\mathbf{x}} - \mathbf{x}}{\sigma} \sim \mathcal{N}(0, \mathbf{I})$ and $\|\sigma\mathbf{s}_\theta(\mathbf{x}, \sigma)\|_2 \propto 1$, we can easily conclude that the order of magnitude of $\lambda(\sigma)\ell(\boldsymbol{\theta}; \sigma)$ does not depend on σ .

We emphasize that our objective Eq. (5.6) requires no adversarial training, no surrogate losses, and no sampling from the score network during training (*e.g.*, unlike contrastive divergence). Also, it does not require $\mathbf{s}_\theta(\mathbf{x}, \sigma)$ to have special architectures in order to be tractable. In addition, when $\lambda(\cdot)$ and $\{\sigma_i\}_{i=1}^L$ are fixed, it can be used to quantitatively compare different NCSNs.

5.4.3 NCSN Inference via Annealed Langevin Dynamics

After the NCSN $\mathbf{s}_\theta(\mathbf{x}, \sigma)$ is trained, we propose a sampling approach—annealed Langevin dynamics (Algorithm 5.1)—to produce samples, inspired by simulated annealing [KGV83] and annealed importance sampling [Nea01]. As shown in Algorithm 5.1, we start annealed Langevin dynamics by initializing the samples from some fixed prior distribution, *e.g.*, uniform noise. Then, we run Langevin dynamics to sample from $q_{\sigma_1}(\mathbf{x})$ with step size α_1 . Next, we run Langevin dynamics to sample from $q_{\sigma_2}(\mathbf{x})$, starting from the final samples of the previous simulation and using a reduced step size α_2 . We continue in this fashion, using the final samples of Langevin dynamics for $q_{\sigma_{i-1}}(\mathbf{x})$ as the initial samples of Langevin dynamics for $q_{\sigma_i}(\mathbf{x})$, and tuning down the step size α_i gradually with $\alpha_i = \epsilon \cdot \sigma_i^2 / \sigma_L^2$. Finally, we run Langevin dynamics to sample from $q_{\sigma_L}(\mathbf{x})$, which is close to $p_{\text{data}}(\mathbf{x})$ when $\sigma_L \approx 0$.

Since the distributions $\{q_{\sigma_i}\}_{i=1}^L$ are all perturbed by Gaussian noise, their supports span the whole space and their scores are well-defined, avoiding difficulties from the manifold hypothesis. When σ_1 is sufficiently large, the low density regions of $q_{\sigma_1}(\mathbf{x})$ become small and the modes become less isolated. As discussed previously, this can make score estimation more accurate, and the mixing of Langevin dynamics faster. We can therefore assume that Langevin dynamics produce good samples for $q_{\sigma_1}(\mathbf{x})$. These samples are likely to come from high density regions of $q_{\sigma_1}(\mathbf{x})$, which means they are also likely to reside in the high density regions of $q_{\sigma_2}(\mathbf{x})$, given that $q_{\sigma_1}(\mathbf{x})$ and $q_{\sigma_2}(\mathbf{x})$ only slightly differ from each other. As score estimation and Langevin dynamics perform better in high density regions, samples from $q_{\sigma_1}(\mathbf{x})$ will serve as good initial samples for Langevin dynamics of $q_{\sigma_2}(\mathbf{x})$. Similarly, $q_{\sigma_{i-1}}(\mathbf{x})$ provides good initial samples for $q_{\sigma_i}(\mathbf{x})$, and finally we obtain samples of good quality from $q_{\sigma_L}(\mathbf{x})$.

There could be many possible ways of tuning α_i according to σ_i in Algorithm 5.1. Our choice is $\alpha_i \propto \sigma_i^2$. The motivation is to fix the magnitude of the “signal-to-noise” ratio $\frac{\alpha_i \mathbf{s}_\theta(\mathbf{x}, \sigma_i)}{2\sqrt{\alpha_i} \mathbf{z}}$ in Langevin dynamics. Note that $\mathbb{E}[\|\frac{\alpha_i \mathbf{s}_\theta(\mathbf{x}, \sigma_i)}{2\sqrt{\alpha_i} \mathbf{z}}\|_2^2] \approx \mathbb{E}[\frac{\alpha_i \|\mathbf{s}_\theta(\mathbf{x}, \sigma_i)\|_2^2}{4}] \propto \frac{1}{4} \mathbb{E}[\|\sigma_i \mathbf{s}_\theta(\mathbf{x}, \sigma_i)\|_2^2]$. Recall that empirically we found $\|\mathbf{s}_\theta(\mathbf{x}, \sigma)\|_2 \propto 1/\sigma$ when the score network is trained close to optimal, in which case $\mathbb{E}[\|\sigma_i \mathbf{s}_\theta(\mathbf{x}, \sigma_i)\|_2^2] \propto 1$. Therefore $\|\frac{\alpha_i \mathbf{s}_\theta(\mathbf{x}, \sigma_i)}{2\sqrt{\alpha_i} \mathbf{z}}\|_2 \propto \frac{1}{4} \mathbb{E}[\|\sigma_i \mathbf{s}_\theta(\mathbf{x}, \sigma_i)\|_2^2] \propto \frac{1}{4}$ does not depend on σ_i .

To demonstrate the efficacy of our annealed Langevin dynamics, we provide a toy example where the goal is to sample from a mixture of Gaussian with two well-separated modes using only scores. We apply Algorithm 5.1 to sample from the mixture of Gaussian used in Section 5.3.2. In the experiment, we choose $\{\sigma_i\}_{i=1}^L$ to be a geometric progression, with $L = 10$, $\sigma_1 = 10$ and $\sigma_{10} = 0.1$. The results are provided in Fig. 5.3. Comparing Fig. 5.3 (b) against (c), annealed Langevin dynamics correctly recover the relative weights between the two modes whereas standard Langevin dynamics fail.

5.5 Experiments

In this section, we demonstrate that our NCSNs are able to produce high quality image samples on several commonly used image datasets. In addition, we show that our models learn reasonable image representations by image inpainting experiments.

Table 5.1: Inception and FID scores for CIFAR-10

Model	Inception	FID
CIFAR-10 Unconditional		
PixelCNN [Oor+16b]	4.60	65.93
PixelIQN [ODM18]	5.29	49.46
EBM [DM19a]	6.02	40.58
WGAN-GP [Gul+17]	7.86 ± .07	36.4
MoLM [Rav+18]	7.90 ± .10	18.9
SNGAN [Miy+18]	8.22 ± .05	21.7
ProgressiveGAN [Kar+18]	8.80 ± .05	-
NCSN (Ours)	8.87 ± .12	25.32
CIFAR-10 Conditional		
EBM [DM19a]	8.30	37.9
SNGAN [Miy+18]	8.60 ± .08	25.5
BigGAN [BDS19]	9.22	14.73

Setup We use MNIST, CelebA [Liu+15], and CIFAR-10 [KH+09] datasets in our experiments. For CelebA, the images are first center-cropped to 140×140 and then resized to 32×32 . All images are rescaled so that pixel values are in $[0, 1]$. We choose $L = 10$ different standard deviations such that $\{\sigma_i\}_{i=1}^L$ is a geometric sequence with $\sigma_1 = 1$ and $\sigma_{10} = 0.01$. Note that Gaussian noise of $\sigma = 0.01$ is almost indistinguishable to human eyes for image data. When using annealed Langevin dynamics for image generation, we choose $T = 100$ and $\epsilon = 2 \times 10^{-5}$, and use uniform noise as our initial samples. We found the results are robust w.r.t. the choice of T , and ϵ between 5×10^{-6} and 5×10^{-5} generally works fine. We provide additional details on model architecture and settings in Appendices B.3.1 and B.3.2.

Image generation In Fig. 5.5, we show uncurated samples from annealed Langevin dynamics for MNIST, CelebA and CIFAR-10. As shown by the samples, our generated images have higher or comparable quality to those from modern likelihood-based models and GANs. To intuit the procedure of annealed Langevin dynamics, we provide intermediate samples in Fig. 5.4, where each row shows how samples evolve from pure random noise to high quality images. More samples from our approach can be found in Appendix B.3.3. We also show the nearest neighbors of generated images in the training dataset in Appendix B.3.3,

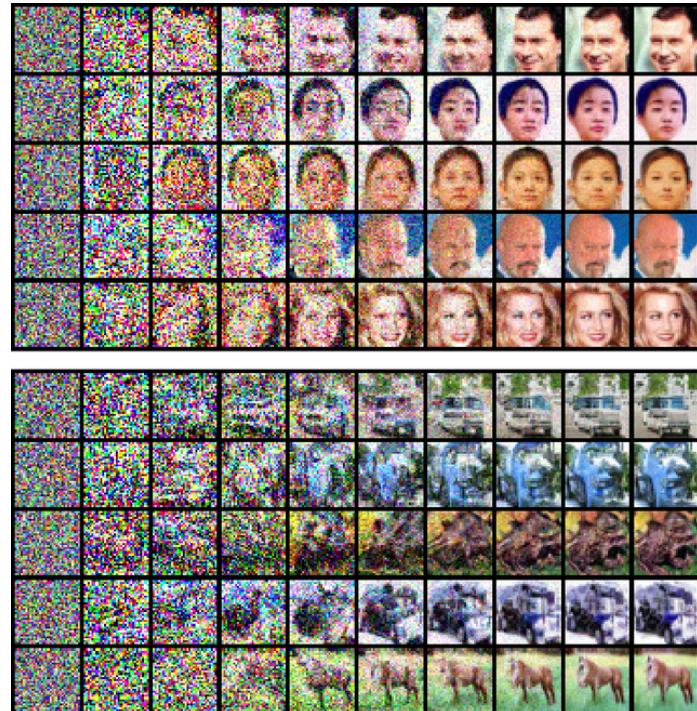


Figure 5.4: Intermediate samples of annealed Langevin dynamics.

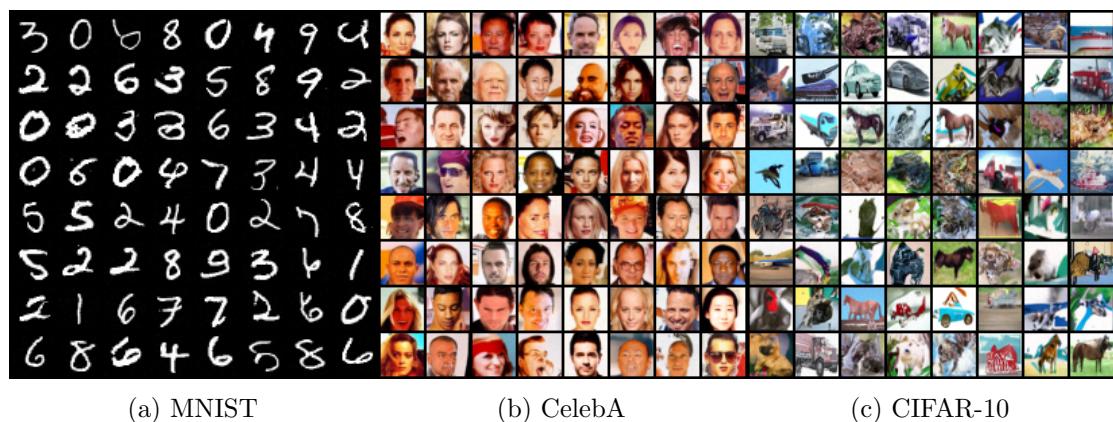


Figure 5.5: Uncurated samples on MNIST, CelebA, and CIFAR-10 datasets.



Figure 5.6: Image inpainting on CelebA (**left**) and CIFAR-10 (**right**). The leftmost column of each figure shows the occluded images, while the rightmost column shows the original images.

in order to demonstrate that our model is not simply memorizing training images. To show it is important to learn a conditional score network jointly for many noise levels and use annealed Langevin dynamics, we compare against a baseline approach where we only consider one noise level $\{\sigma_1 = 0.01\}$ and use the vanilla Langevin dynamics sampling method. Although this small added noise helps circumvent the difficulty of the manifold hypothesis (as shown by Fig. 5.1, things will completely fail if no noise is added), it is not large enough to provide information on scores in regions of low data density. As a result, this baseline fails to generate reasonable images, as shown by samples in Appendix B.3.3.

For quantitative evaluation, we report inception [Sal+16] and FID [Heu+17] scores on CIFAR-10 in Table 5.1. As an *unconditional* model, we achieve the state-of-the-art inception score of 8.87, which is even better than most reported values for *class-conditional* generative models. Our FID score 25.32 on CIFAR-10 is also comparable to top existing models, such as SNGAN [Miy+18]. We omit scores on MNIST and CelebA as the scores on these two datasets are not widely reported, and different preprocessing (such as the center crop size of CelebA) can lead to numbers not directly comparable.

Image inpainting In Fig. 5.6, we demonstrate that our score networks learn generalizable and semantically meaningful image representations that allow it to produce diverse image inpaintings. Note that some previous models such as PixelCNN can only impute images in the raster scan order. In contrast, our method can naturally handle images with occlusions of arbitrary shapes by a simple modification of the annealed Langevin dynamics procedure

(details in Appendix B.3.2). We provide more image inpainting results in Appendix B.3.3.

5.6 Related Work

Our approach has some similarities with methods that learn the transition operator of a Markov chain for sample generation [Ben+13; Soh+15; BHV17; Goy+17; SZE17]. For example, generative stochastic networks (GSN [Ben+13; Ala+16]) use denoising autoencoders to train a Markov chain whose equilibrium distribution matches the data distribution. Similarly, our method trains the score function used in Langevin dynamics to sample from the data distribution. However, GSN often starts the chain very close to a training data point, and therefore requires the chain to transition quickly between different modes. In contrast, our annealed Langevin dynamics are initialized from unstructured noise. Nonequilibrium Thermodynamics (NET [Soh+15]) use a hand-designed diffusion process to slowly transform data into random noise, and then learn to reverse this procedure by training an inverse diffusion. Compared to our approach, NET is restricted to maximum likelihood training with the evidence lower bound, whereas our models are trained by score matching.

Previous approaches such as Infusion Training (IT [BHV17]) and Variational Walkback (VW [Goy+17]) also employ different noise levels/temperatures to train transition operators of a Markov chain. Both IT and VW train their models by maximizing the evidence lower bound of a suitable marginal likelihood. In practice, they tend to produce blurry image samples, similar to variational autoencoders. In contrast, our objective is based on score matching instead of likelihood, and we can produce images comparable to GANs.

There are several structural differences that further distinguish our approach from previous methods discussed above. First, *we do not need to sample from a Markov chain during training*. In contrast, the walkback procedure of GSNs needs multiple runs of the chain to generate “negative samples”. Other methods including IT, and VW also need to simulate a Markov chain for every input to compute the training loss. This difference makes our approach more efficient and scalable for training deep models. Secondly, *our training and sampling methods are decoupled from each other*. For score estimation, both sliced and denoising score matching can be used. For sampling, any method based on scores is applicable, including Langevin dynamics and (potentially) Hamiltonian Monte Carlo [Nea+11]. Our framework allows arbitrary combinations of score estimators and (gradient-based) sampling approaches, whereas most previous methods, such as NET, tie the

model to a specific Markov chain. Finally, *our approach can be used to train energy-based models (EBM)* by using the gradient of an energy-based model as the score model. In contrast, it is unclear how previous methods that learn transition operators of Markov chains can be directly used for training EBMs.

Score matching was originally proposed for learning EBMs. However, many existing methods based on score matching are either not scalable [HD05] or fail to produce samples of comparable quality to VAEs or GANs [KL10; Sar+18]. To obtain better performance on training deep energy-based models, some recent works have resorted to contrastive divergence [Hin02], and propose to sample with Langevin dynamics for both training and testing [DM19a; Nij+19]. However, unlike our approach, contrastive divergence uses the computationally expensive procedure of Langevin dynamics as an inner loop during training. The idea of combining annealing with denoising score matching has also been investigated in previous work under different contexts. In [GS14; CS14; ZZ18], different annealing schedules on the noise for training denoising autoencoders are proposed. However, their work is on learning representations for improving the performance of classification, instead of generative modeling. The method of denoising score matching can also be derived from the perspective of Bayes least squares [RS07; RS11], using techniques of Stein’s Unbiased Risk Estimator [Miy61; Ste81].

5.7 Conclusion

We propose the framework of score-based generative modeling where we first estimate gradients of data densities via score matching, and then generate samples via Langevin dynamics. We analyze several challenges faced by a naïve application of this approach, and propose to tackle them by training Noise Conditional Score Networks (NCSN) and sampling with annealed Langevin dynamics. Our approach requires no adversarial training, no MCMC sampling during training, and no special model architectures. Experimentally, we show that our approach can generate high quality images that were previously only produced by the best likelihood-based models and GANs. We achieve the new state-of-the-art inception score on CIFAR-10, and an FID score comparable to SNGANs.

Chapter 6

Improved Techniques for Data Generation

In the previous chapter, we show that score-based generative models can produce high quality image samples comparable to GANs, without requiring adversarial optimization. However, existing training procedures are limited to images of low resolution (typically below 32×32), and can be unstable under some settings. In this chapter, we provide a new theoretical analysis of learning and sampling from score-based models in high dimensional spaces, explaining existing failure modes and motivating new solutions that generalize across datasets. To enhance stability, we also propose to maintain an exponential moving average of model weights. With these improvements, we can scale score-based generative models to various image datasets, with diverse resolutions ranging from 64×64 to 256×256 . Our score-based models can generate high-fidelity samples that rival best-in-class GANs on various image datasets, including CelebA, FFHQ, and several LSUN categories.

This chapter was previously published as [SE20].

6.1 Introduction

Score-based generative models represent probability distributions through score—a vector field pointing in the direction where the likelihood of data increases most rapidly. Remarkably, these score functions can be learned from data without requiring adversarial optimization, and can produce realistic image samples that rival GANs on simple datasets such as CIFAR-10 [KH+09].

Despite this success, existing score-based generative models only work on low resolution images (32×32) due to several limiting factors. First, the score function is learned via denoising score matching [HD05; Vin11; RS11]. Intuitively, this means a neural network (named the *score network*) is trained to denoise images blurred with Gaussian noise. A key insight from Chapter 5 is to perturb the data using *multiple* noise scales so that the score network captures both coarse and fine-grained image features. However, it is an open question how these noise scales should be chosen. The recommended settings in Chapter 5 work well for 32×32 images, but perform poorly when the resolution gets higher. Second, samples are generated by running Langevin dynamics [RT+96; WT11]. This method starts from white noise and progressively denoises it into an image using the score network. This procedure, however, might fail or take an extremely long time to converge when used in high-dimensions and with a necessarily imperfect (learned) score network.

We propose a set of techniques to scale score-based generative models to high resolution images. Based on a new theoretical analysis on a simplified mixture model, we provide a method to analytically compute an effective set of Gaussian noise scales from training data. Additionally, we propose an efficient architecture to amortize the score estimation task across a large (possibly infinite) number of noise scales with a single neural network. Based on a simplified analysis of the convergence properties of the underlying Langevin dynamics sampling procedure, we also derive a technique to approximately optimize its performance as a function of the noise scales. Combining these techniques with an exponential moving average (EMA) of model parameters, we are able to significantly improve the sample quality, and successfully scale to images of resolutions ranging from 64×64 to 256×256 , which was previously impossible for score-based generative models. As illustrated in Fig. 6.1, the samples are sharp and diverse.

6.2 Background

6.2.1 Langevin Dynamics

For any continuously differentiable probability density $p(\mathbf{x})$, we call $\nabla_{\mathbf{x}} \log p(\mathbf{x})$ its *score function*. In many situations the score function is easier to model and estimate than the original probability density function [HD05; Son+19]. For example, for an unnormalized density it does not depend on the partition function. Once the score function is known, we can employ Langevin dynamics to sample from the corresponding distribution. Given

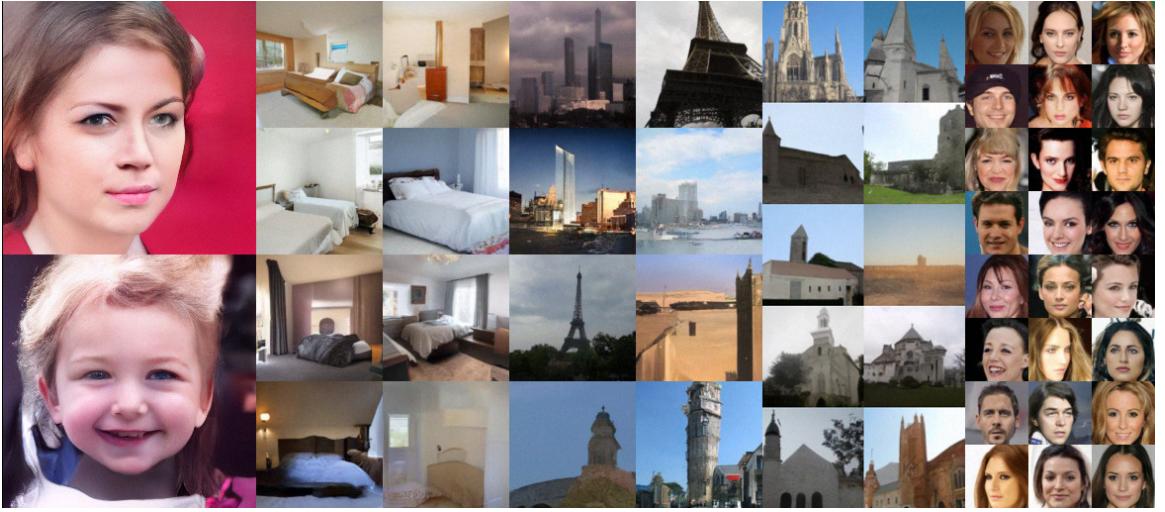


Figure 6.1: Generated samples on datasets of decreasing resolutions. From left to right: FFHQ 256×256 , LSUN bedroom 128×128 , LSUN tower 128×128 , LSUN church_outdoor 96×96 , and CelebA 64×64 .

a step size $\alpha > 0$, a total number of iterations T , and an initial sample \mathbf{x}_0 from any prior distribution $\pi(\mathbf{x})$, Langevin dynamics iteratively evaluate the following

$$\mathbf{x}_t \leftarrow \mathbf{x}_{t-1} + \alpha \nabla_{\mathbf{x}} \log p(\mathbf{x}_{t-1}) + \sqrt{2\alpha} \mathbf{z}_t, \quad 1 \leq t \leq T \quad (6.1)$$

where $\mathbf{z}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. When α is sufficiently small and T is sufficiently large, the distribution of \mathbf{x}_T will be close to $p(\mathbf{x})$ under some regularity conditions [RT+96; WT11]. Suppose we have a neural network $s_{\theta}(\mathbf{x})$ (called the *score network*) parameterized by θ , and it has been trained such that $s_{\theta}(\mathbf{x}) \approx \nabla_{\mathbf{x}} \log p(\mathbf{x})$. We can approximately generate samples from $p(\mathbf{x})$ using Langevin dynamics by replacing $\nabla_{\mathbf{x}} \log p(\mathbf{x}_{t-1})$ with $s_{\theta}(\mathbf{x}_{t-1})$ in Eq. (6.1). Note that Eq. (6.1) can be interpreted as noisy gradient ascent on the log-density $\log p(\mathbf{x})$.

6.2.2 Score-Based Generative Modeling

We can estimate the score function from data and generate new samples with Langevin dynamics. This idea was named *score-based generative modeling* in this dissertation. Because the estimated score function is inaccurate in regions without training data, Langevin dynamics may not converge correctly when a sampling trajectory encounters those regions (see more detailed analysis in Chapter 5). As a remedy, Chapter 5 proposes to perturb the data

with Gaussian noise of different intensities and jointly estimate the score functions of all noise-perturbed data distributions. During inference, they combine the information from all noise scales by sampling from each noise-perturbed distribution sequentially with Langevin dynamics.

More specifically, suppose we have an underlying data distribution $p_{\text{data}}(\mathbf{x})$ and consider a sequence of noise scales $\{\sigma_i\}_{i=1}^L$ that satisfies $\sigma_1 > \sigma_2 > \dots > \sigma_L$. Let $p_\sigma(\tilde{\mathbf{x}} | \mathbf{x}) = \mathcal{N}(\tilde{\mathbf{x}} | \mathbf{x}, \sigma^2 \mathbf{I})$, and denote the corresponding perturbed data distribution as $p_\sigma(\tilde{\mathbf{x}}) := \int p_\sigma(\tilde{\mathbf{x}} | \mathbf{x}) p_{\text{data}}(\mathbf{x}) d\mathbf{x}$. In Chapter 5, we proposed to estimate the score function of each $p_{\sigma_i}(\mathbf{x})$ by training a joint neural network $s_\theta(\mathbf{x}, \sigma)$ (called the *noise conditional score network*) with the following loss:

$$\frac{1}{2L} \sum_{i=1}^L \mathbb{E}_{p_{\text{data}}(\mathbf{x})} \mathbb{E}_{p_{\sigma_i}(\tilde{\mathbf{x}} | \mathbf{x})} \left[\left\| \sigma_i s_\theta(\tilde{\mathbf{x}}, \sigma_i) + \frac{\tilde{\mathbf{x}} - \mathbf{x}}{\sigma_i} \right\|_2^2 \right], \quad (6.2)$$

where all expectations can be efficiently estimated using empirical averages. When trained to the optimum (denoted as $s_{\theta^*}(\mathbf{x}, \sigma)$), the noise conditional score network (NCSN) satisfies $\forall i : s_{\theta^*}(\mathbf{x}, \sigma_i) = \nabla_{\mathbf{x}} \log p_{\sigma_i}(\mathbf{x})$ almost everywhere, assuming enough data and model capacity.

Algorithm 6.1 Annealed Langevin dynamics (Chapter 5)

Require: $\{\sigma_i\}_{i=1}^L, \epsilon, T$.

- 1: Initialize \mathbf{x}_0
- 2: **for** $i \leftarrow 1$ to L **do**
- 3: $\alpha_i \leftarrow \epsilon \cdot \sigma_i^2 / \sigma_L^2$ $\triangleright \alpha_i$ is the step size.
- 4: **for** $t \leftarrow 1$ to T **do**
- 5: Draw $\mathbf{z}_t \sim \mathcal{N}(0, I)$
- 6: $\mathbf{x}_t \leftarrow \mathbf{x}_{t-1} + \alpha_i s_\theta(\mathbf{x}_{t-1}, \sigma_i) + \sqrt{2\alpha_i} \mathbf{z}_t$
- 7: **end for**
- 8: $\mathbf{x}_0 \leftarrow \mathbf{x}_T$
- 9: **end for**
- 10: **if** denoise \mathbf{x}_T **then**
- 11: **return** $\mathbf{x}_T + \sigma_T^2 s_\theta(\mathbf{x}_T, \sigma_T)$
- 12: **else**
- 13: **return** \mathbf{x}_T
- 14: **end if**

After training an NCSN, one can generate samples by *annealed Langevin dynamics* (*cf.*, Chapter 5), a method that combines information from all noise scales. We provide its pseudo-code in Algorithm 6.1. The approach amounts to sampling from $p_{\sigma_1}(\mathbf{x}), p_{\sigma_2}(\mathbf{x}), \dots, p_{\sigma_L}(\mathbf{x})$

sequentially with Langevin dynamics with a special step size schedule $\alpha_i = \epsilon \sigma_i^2 / \sigma_L^2$ for the i -th noise scale. Samples from each noise scale are used to initialize Langevin dynamics for the next noise scale until reaching the smallest one, where it provides final samples for the NCSN.

Following the first public release of this work, Jolicoeur-Martineau et al. [Jol+21] noticed that adding an extra denoising step after the original annealed Langevin dynamics in Chapter 5, similar to [SH19; LCS19; KS20], often significantly improves FID scores [Heu+17] without affecting the visual appearance of samples. Instead of directly returning \mathbf{x}_T , this denoising step returns $\mathbf{x}_T + \sigma_T^2 \mathbf{s}_\theta(\mathbf{x}_T, \sigma_T)$ (see Algorithm 6.1), which essentially removes the unwanted noise $\mathcal{N}(\mathbf{0}, \sigma_T^2 \mathbf{I})$ from \mathbf{x}_T using Tweedie's formula [Efr11]. Therefore, we have updated results by incorporating this denoising trick, but kept some original results without this denoising step in Appendix B.4 for reference.

There are many design choices that are critical to the successful training and inference of NCSNs, including (i) the set of noise scales $\{\sigma_i\}_{i=1}^L$, (ii) the way that $\mathbf{s}_\theta(\mathbf{x}, \sigma)$ incorporates information of σ , (iii) the step size parameter ϵ and (iv) the number of sampling steps per noise scale T in Algorithm 6.1. Below we provide theoretically motivated ways to configure them without manual tuning, which significantly improve the performance of NCSNs on high resolution images.

6.3 Choosing Noise Scales

Noise scales are critical for the success of NCSNs. As shown in Chapter 5, score networks trained with a single noise can never produce convincing samples for large images. Intuitively, high noise facilitates the estimation of score functions, but also leads to corrupted samples; while lower noise gives clean samples but makes score functions harder to estimate. One should therefore leverage different noise scales together to get the best of both worlds.

When the range of pixel values is $[0, 1]$, Chapter 5 recommends choosing $\{\sigma_i\}_{i=1}^L$ as a geometric sequence where $L = 10$, $\sigma_1 = 1$, and $\sigma_L = 0.01$. It is reasonable that the smallest noise scale $\sigma_L = 0.01 \ll 1$, because we sample from perturbed distributions with descending noise scales and we want to add low noise at the end. However, some important questions remain unanswered, which turn out to be critical to the success of NCSNs on high resolution images: (i) Is $\sigma_1 = 1$ appropriate? If not, how should we adjust σ_1 for different datasets? (ii) Is geometric progression a good choice? (iii) Is $L = 10$ good across different datasets? If

not, how many noise scales are ideal?

Below we provide answers to the above questions, motivated by theoretical analyses on simple mathematical models. Our insights are effective for configuring score-based generative modeling in practice, as corroborated by experimental results in Section 6.6.

6.3.1 The Initial Noise Scale

The algorithm of annealed Langevin dynamics (Algorithm 6.1) is an iterative refining procedure that starts from generating coarse samples with rich variation under large noise, before converging to fine samples with less variation under small noise. The initial noise scale σ_1 largely controls the diversity of the final samples. In order to promote sample diversity, we might want to choose σ_1 to be as large as possible. However, an excessively large σ_1 will require more noise scales (to be discussed in Section 6.3.2) and make annealed Langevin dynamics more expensive. Below we present an analysis to guide the choice of σ_1 and provide a technique to strike the right balance.

Real-world data distributions are complex and hard to analyze, so we approximate them with empirical distributions. Suppose we have a dataset $\{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \dots, \mathbf{x}^{(N)}\}$ which is i.i.d. sampled from $p_{\text{data}}(\mathbf{x})$. Assuming N is sufficiently large, we have $p_{\text{data}}(\mathbf{x}) \approx \hat{p}_{\text{data}}(\mathbf{x}) := \frac{1}{N} \sum_{i=1}^N \delta(\mathbf{x} = \mathbf{x}^{(i)})$, where $\delta(\cdot)$ denotes a point mass distribution. When perturbed with $\mathcal{N}(\mathbf{0}, \sigma_1^2 \mathbf{I})$, the empirical distribution becomes $\hat{p}_{\sigma_1}(\mathbf{x}) := \frac{1}{N} \sum_{i=1}^N p^{(i)}(\mathbf{x})$, where $p^{(i)}(\mathbf{x}) := \mathcal{N}(\mathbf{x} | \mathbf{x}^{(i)}, \sigma_1^2 \mathbf{I})$. For generating diverse samples regardless of initialization, we naturally expect that Langevin dynamics can explore any component $p^{(i)}(\mathbf{x})$ when initialized from any other component $p^{(j)}(\mathbf{x})$, where $i \neq j$. The performance of Langevin dynamics is governed by the score function $\nabla_{\mathbf{x}} \log \hat{p}_{\sigma_1}(\mathbf{x})$ (see Eq. (6.1)).

Proposition 6.1. *Let $\hat{p}_{\sigma_1}(\mathbf{x}) := \frac{1}{N} \sum_{i=1}^N p^{(i)}(\mathbf{x})$, where $p^{(i)}(\mathbf{x}) := \mathcal{N}(\mathbf{x} | \mathbf{x}^{(i)}, \sigma_1^2 \mathbf{I})$. With $r^{(i)}(\mathbf{x}) := \frac{p^{(i)}(\mathbf{x})}{\sum_{k=1}^N p^{(k)}(\mathbf{x})}$, the score function is $\nabla_{\mathbf{x}} \log \hat{p}_{\sigma_1}(\mathbf{x}) = \sum_{i=1}^N r^{(i)}(\mathbf{x}) \nabla_{\mathbf{x}} \log p^{(i)}(\mathbf{x})$. Moreover,*

$$\mathbb{E}_{p^{(i)}(\mathbf{x})}[r^{(j)}(\mathbf{x})] \leq \frac{1}{2} \exp\left(-\frac{\|\mathbf{x}^{(i)} - \mathbf{x}^{(j)}\|_2^2}{8\sigma_1^2}\right). \quad (6.3)$$

In order for Langevin dynamics to transition from $p^{(i)}(\mathbf{x})$ to $p^{(j)}(\mathbf{x})$ easily for $i \neq j$, $\mathbb{E}_{p^{(i)}(\mathbf{x})}[r^{(j)}(\mathbf{x})]$ has to be relatively large, because otherwise $\nabla_{\mathbf{x}} \log \hat{p}_{\sigma_1}(\mathbf{x}) = \sum_{k=1}^N r^{(k)}(\mathbf{x}) \nabla_{\mathbf{x}} \log p^{(k)}(\mathbf{x})$ will ignore the component $p^{(j)}(\mathbf{x})$ (on average) when initialized with $\mathbf{x} \sim p^{(i)}(\mathbf{x})$.

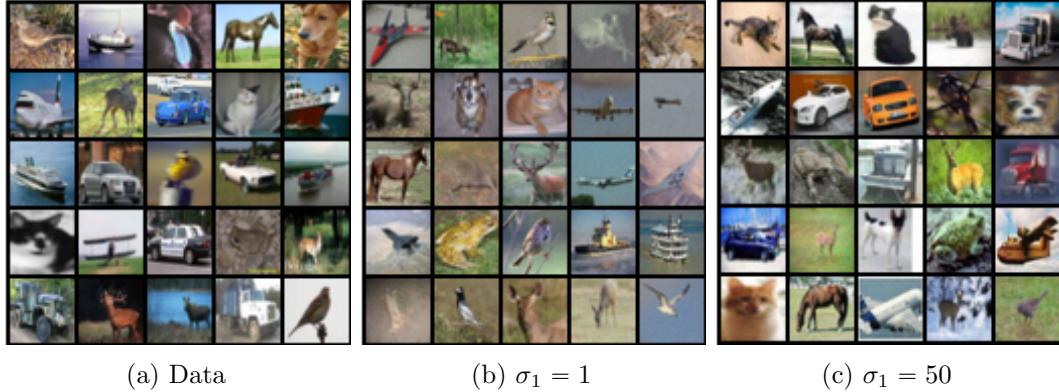


Figure 6.2: Running annealed Langevin dynamics to sample from a mixture of Gaussian centered at images in the CIFAR-10 test set.

and in such case Langevin dynamics will act as if $p^{(j)}(\mathbf{x})$ did not exist. The bound of Eq. (6.3) indicates that $\mathbb{E}_{p^{(i)}(\mathbf{x})}[r^{(j)}(\mathbf{x})]$ can decay exponentially fast if σ_1 is small compared to $\|\mathbf{x}^{(i)} - \mathbf{x}^{(j)}\|_2$. As a result, it is necessary for σ_1 to be numerically comparable to the maximum pairwise distances of data to facilitate transitioning of Langevin dynamics and hence improving sample diversity. In particular, we suggest:

Technique 6.1 (Initial noise scale). *Choose σ_1 to be as large as the maximum Euclidean distance between all pairs of training data points.*

Taking CIFAR-10 as an example, the median pairwise distance between all training images is around 18, so $\sigma_1 = 1$ as in Chapter 5 implies $\mathbb{E}[r(\mathbf{x})] < 10^{-17}$ and is unlikely to produce diverse samples as per our analysis. To test whether choosing σ_1 according to Technique 6.1 (*i.e.*, $\sigma_1 = 50$) gives significantly more diverse samples than using $\sigma_1 = 1$, we run annealed Langevin dynamics to sample from a mixture of Gaussian with 10000 components, where each component is centered at one CIFAR-10 test image. All initial samples are drawn from a uniform distribution over $[0, 1]^{32 \times 32 \times 3}$. This setting allows us to avoid confounders introduced by NCSN training because we use ground truth score functions. As shown in Fig. 6.2, samples in Fig. 6.2c (using Technique 6.1) exhibit comparable diversity to ground-truth images (Fig. 6.2a), and have better variety than Fig. 6.2b ($\sigma_1 = 1$). Quantitatively, the average pairwise distance of samples in Fig. 6.2c is 18.65, comparable to data (17.78) but much higher than that of Fig. 6.2b (10.12).

6.3.2 Other Noise Scales

After setting σ_L and σ_1 , we need to choose the number of noise scales L and specify the other elements of $\{\sigma_i\}_{i=1}^L$. As analyzed in Chapter 5, it is crucial for the success of score-based generative models to ensure that $p_{\sigma_i}(\mathbf{x})$ generates a sufficient number of training data in high density regions of $p_{\sigma_{i-1}}(\mathbf{x})$ for all $1 < i \leq L$. The intuition is we need reliable gradient signals for $p_{\sigma_i}(\mathbf{x})$ when initializing Langevin dynamics with samples from $p_{\sigma_{i-1}}(\mathbf{x})$.

However, an extensive grid search on $\{\sigma_i\}_{i=1}^L$ can be very expensive. To give some theoretical guidance on finding good noise scales, we consider a simple case where the dataset contains only one data point, or equivalently, $\forall 1 \leq i \leq L : p_{\sigma_i}(\mathbf{x}) = \mathcal{N}(\mathbf{x} \mid \mathbf{0}, \sigma_i^2 \mathbf{I})$. Our first step is to understand the distributions of $p_{\sigma_i}(\mathbf{x})$ better, especially when \mathbf{x} has high dimensionality. We can decompose $p_{\sigma_i}(\mathbf{x})$ in hyperspherical coordinates to $p(\phi)p_{\sigma_i}(r)$, where r and ϕ denote the radial and angular coordinates of \mathbf{x} respectively. Because $p_{\sigma_i}(\mathbf{x})$ is an isotropic Gaussian, the angular component $p(\phi)$ is uniform and shared across all noise scales. As for $p_{\sigma_i}(r)$, we have the following

Proposition 6.2. *Let $\mathbf{x} \in \mathbb{R}^D \sim \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I})$, and $r = \|\mathbf{x}\|_2$. We have*

$$p(r) = \frac{1}{2^{D/2-1}\Gamma(D/2)} \frac{r^{D-1}}{\sigma^D} \exp\left(-\frac{r^2}{2\sigma^2}\right) \quad \text{and} \quad r - \sqrt{D}\sigma \xrightarrow{d} \mathcal{N}(0, \sigma^2/2) \quad \text{when } D \rightarrow \infty.$$

In practice, dimensions of image data can range from several thousand to millions, and are typically large enough to warrant $p(r) \approx \mathcal{N}(r|\sqrt{D}\sigma, \sigma^2/2)$ with negligible error. We therefore take $p_{\sigma_i}(r) = \mathcal{N}(r|m_i, s_i^2)$ to simplify our analysis, where $m_i := \sqrt{D}\sigma$, and $s_i^2 := \sigma^2/2$.

Recall that our goal is to make sure samples from $p_{\sigma_i}(\mathbf{x})$ will cover high density regions of $p_{\sigma_{i-1}}(\mathbf{x})$. Because $p(\phi)$ is shared across all noise scales, $p_{\sigma_i}(\mathbf{x})$ already covers the angular component of $p_{\sigma_{i-1}}(\mathbf{x})$. Therefore, we need the radial components of $p_{\sigma_i}(\mathbf{x})$ and $p_{\sigma_{i-1}}(\mathbf{x})$ to have large overlap. Since $p_{\sigma_{i-1}}(r)$ has high density in $\mathcal{I}_{i-1} := [m_{i-1} - 3s_{i-1}, m_{i-1} + 3s_{i-1}]$ (employing the ‘‘three-sigma rule of thumb’’ [Gra06]), a natural choice is to fix $p_{\sigma_i}(r \in \mathcal{I}_{i-1}) = \Phi(\sqrt{2D}(\gamma_i - 1) + 3\gamma_i) - \Phi(\sqrt{2D}(\gamma_i - 1) - 3\gamma_i) = C$ with some moderately large constant $C > 0$ for all $1 < i \leq L$, where $\gamma_i := \sigma_{i-1}/\sigma_i$ and $\Phi(\cdot)$ is the CDF of standard Gaussian. This choice immediately implies that $\gamma_2 = \gamma_3 = \dots = \gamma_L$ and thus $\{\sigma_i\}_{i=1}^L$ is a geometric progression.

Ideally, we should choose as many noise scales as possible to make $C \approx 1$. However,

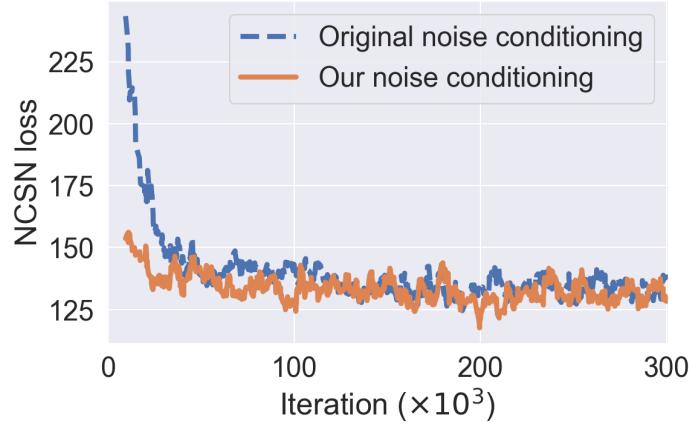


Figure 6.3: Training loss curves of two noise conditioning methods.

having too many noise scales will make sampling very costly, as we need to run Langevin dynamics for each noise scale in sequence. On the other hand, $L = 10$ (for 32×32 images) as in the original setting of Chapter 5 is arguably too small, for which $C = 0$ up to numerical precision. To strike a balance, we recommend $C \approx 0.5$ which performs well in our experiments. In summary,

Technique 6.2 (Other noise scales). *Choose $\{\sigma_i\}_{i=1}^L$ as a geometric progression with common ratio γ , such that $\Phi(\sqrt{2D}(\gamma - 1) + 3\gamma) - \Phi(\sqrt{2D}(\gamma - 1) - 3\gamma) \approx 0.5$.*

6.3.3 Incorporating the Noise Information

For high resolution images, we need a large σ_1 and a huge number of noise scales as per Techniques 6.1 and 6.2. Recall that the NCSN is a single amortized network that takes a noise scale and gives the corresponding score. In Chapter 5, we used a separate set of scale and bias parameters in normalization layers to incorporate the information from each noise scale. However, its memory consumption grows linearly w.r.t. L , and it is not applicable when the NCSN has no normalization layers.

We propose an efficient alternative that is easier to implement and more widely applicable. For $p_\sigma(\mathbf{x}) = \mathcal{N}(\mathbf{x} \mid \mathbf{0}, \sigma^2 \mathbf{I})$ analyzed in Section 6.3.2, we observe that $\mathbb{E}[\|\nabla_{\mathbf{x}} \log p_\sigma(\mathbf{x})\|_2] \approx \sqrt{D}/\sigma$. Moreover, as empirically noted in Chapter 5, $\|\mathbf{s}_\theta(\mathbf{x}, \sigma)\|_2 \propto 1/\sigma$ for a trained NCSN on real data. Because the norm of score functions scales inverse proportionally to σ , we can incorporate the noise information by rescaling the output of an unconditional score network $\mathbf{s}_\theta(\mathbf{x})$ with $1/\sigma$. This motivates our following recommendation

Technique 6.3 (Noise conditioning). *Parameterize the NCSN with $s_{\theta}(\mathbf{x}, \sigma) = s_{\theta}(\mathbf{x})/\sigma$, where $s_{\theta}(\mathbf{x})$ is an unconditional score network.*

It is typically hard for deep networks to automatically learn this rescaling, because σ_1 and σ_L can differ by several orders of magnitude. This simple choice is easier to implement, and can easily handle a large number of noise scales (even continuous ones). As shown in Fig. 6.3 (detailed settings in Appendix B.4.1), it achieves similar training losses compared to the original noise conditioning approach in Chapter 5, and generate samples of better quality (see Appendix B.4.2).

6.4 Configuring Annealed Langevin Dynamics

In order to sample from an NCSN with annealed Langevin dynamics, we need to specify the number of sampling steps per noise scale T and the step size parameter ϵ in Algorithm 6.1. In Chapter 5, we recommended $\epsilon = 2 \times 10^{-5}$ and $T = 100$. It remains unclear how we should change ϵ and T for different sets of noise scales.

To gain some theoretical insight, we revisit the setting in Section 6.3.2 where the dataset has one point (*i.e.*, $p_{\sigma_i}(\mathbf{x}) = \mathcal{N}(\mathbf{x} \mid \mathbf{0}, \sigma_i^2 \mathbf{I})$). Annealed Langevin dynamics connect two adjacent noise scales $\sigma_{i-1} > \sigma_i$ by initializing the Langevin dynamics for $p_{\sigma_i}(\mathbf{x})$ with samples obtained from $p_{\sigma_{i-1}}(\mathbf{x})$. When applying Langevin dynamics to $p_{\sigma_i}(\mathbf{x})$, we have $\mathbf{x}_{t+1} \leftarrow \mathbf{x}_t + \alpha \nabla_{\mathbf{x}} \log p_{\sigma_i}(\mathbf{x}_t) + \sqrt{2\alpha} \mathbf{z}_t$, where $\mathbf{x}_0 \sim p_{\sigma_{i-1}}(\mathbf{x})$ and $\mathbf{z}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. The distribution of \mathbf{x}_T can be computed in closed form:

Proposition 6.3. *Let $\gamma = \frac{\sigma_{i-1}}{\sigma_i}$. For $\alpha = \epsilon \cdot \frac{\sigma_i^2}{\sigma_L^2}$ (as in Algorithm 6.1), we have $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, s_T^2 \mathbf{I})$, where*

$$\frac{s_T^2}{\sigma_i^2} = \left(1 - \frac{\epsilon}{\sigma_L^2}\right)^{2T} \left(\gamma^2 - \frac{2\epsilon}{\sigma_L^2 - \sigma_L^2 \left(1 - \frac{\epsilon}{\sigma_L^2}\right)^2} \right) + \frac{2\epsilon}{\sigma_L^2 - \sigma_L^2 \left(1 - \frac{\epsilon}{\sigma_L^2}\right)^2}. \quad (6.4)$$

When $\{\sigma_i\}_{i=1}^L$ is a geometric progression as advocated by Technique 6.2, we immediately see that s_T^2/σ_i^2 is identical across all $1 < i \leq T$ because of the shared γ . Furthermore, the value of s_T^2/σ_i^2 has no explicit dependency on the dimensionality D .

For better mixing of annealed Langevin dynamics, we hope s_T^2/σ_i^2 approaches 1 across all noise scales, which can be achieved by finding ϵ and T that minimize the difference between Eq. (6.4) and 1. Unfortunately, this often results in an unnecessarily large T that makes

sampling very expensive for large L . As an alternative, we propose to first choose T based on a reasonable computing budget (typically $T \times L$ is several thousand), and subsequently find ϵ by making Eq. (6.4) as close to 1 as possible. In summary:

Technique 6.4 (selecting T and ϵ). *Choose T as large as allowed by a computing budget and then select an ϵ that makes Eq. (6.4) maximally close to 1.*

We follow this guidance to generate all samples in this chapter, except for those from the original NCSN where we adopt the same settings as in Chapter 5. When finding ϵ with Technique 6.4 and Eq. (6.4), we recommend performing grid search over ϵ , rather than using gradient-based optimization methods.

6.5 Improving Stability with Moving Average

Unlike GANs, score-based generative models have one unified objective (Eq. (6.2)) and require no adversarial training. However, even though the loss function of NCSNs typically decreases steadily over the course of training, we observe that the generated image samples sometimes exhibit unstable visual quality, especially for images of larger resolutions. We empirically demonstrate this fact by training NCSNs on CIFAR-10 32×32 and CelebA [Liu+15] 64×64 following the settings of Chapter 5, which exemplifies typical behavior on other image datasets. We report FID scores [Heu+17] computed on 1000 samples every 5000 iterations. Results in Fig. 6.4 are computed with the denoising step, but results without the denoising step are similar (see Fig. B.16 in Appendix B.4.2). As shown in Figs. 6.4 and B.16, the FID scores for the vanilla NCSN often fluctuate significantly during training. Additionally, samples from the vanilla NCSN sometimes exhibit characteristic artifacts: image samples from the same checkpoint have strong tendency to have a common color shift. Moreover, samples are shifted towards different colors throughout training. We provide more samples in Appendix B.4.2 to manifest this artifact.

This issue can be easily fixed by exponential moving average (EMA). Specifically, let θ_i denote the parameters of an NCSN after the i -th training iteration, and θ' be an independent copy of the parameters. We update θ' with $\theta' \leftarrow m\theta' + (1 - m)\theta_i$ after each optimization step, where m is the momentum parameter and typically $m = 0.999$. When producing samples, we use $s_{\theta'}(\mathbf{x}, \sigma)$ instead of $s_{\theta_i}(\mathbf{x}, \sigma)$. As shown in Fig. 6.4, EMA can effectively stabilize FIDs, remove artifacts (more samples in Appendix B.4.2) and give better FID

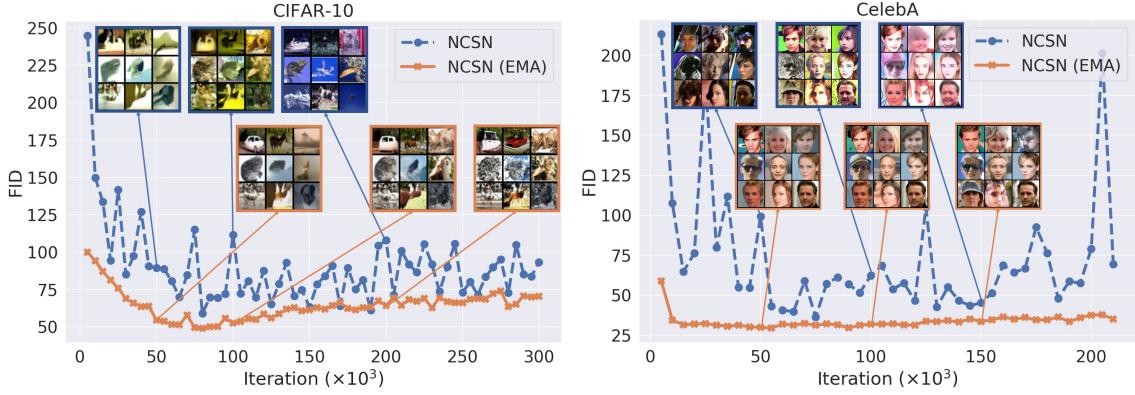


Figure 6.4: FIDs and color artifacts over the course of training (best viewed in color). The FIDs of NCSN have much higher volatility compared to NCSN with EMA. Samples from the vanilla NCSN often have obvious color shifts. All FIDs are computed with the denoising step.

scores in most cases. Empirically, we observe the effectiveness of EMA is universal across a large number of different image datasets. As a result, we recommend the following rule of thumb:

Technique 6.5 (EMA). *Apply exponential moving average to parameters when sampling.*

6.6 Combining All Techniques Together

Employing Techniques 6.1 to 6.5, we build NCSNs that can readily work across a large number of different datasets, including high resolution images that were previously out of reach with score-based generative modeling. Our modified model is named NCSNv2. For a complete description on experimental details and more results, please refer to Appendices B.4.1 and B.4.2.

Quantitative results: We consider CIFAR-10 32×32 and CelebA 64×64 where NCSN and NCSNv2 both produce reasonable samples. We report FIDs (lower is better) every 5000 iterations of training on 1000 samples and give results in Fig. 6.5 (with denoising) and Fig. B.17 (without denoising, deferred to Appendix B.4.2). As shown in Figs. 6.5 and B.17, we observe that the FID scores of NCSNv2 (with all techniques applied) are on average better than those of NCSN, and have much smaller variance over the course of training. As in Chapter 5, we select checkpoints with the smallest FIDs (on 1000 samples) encountered

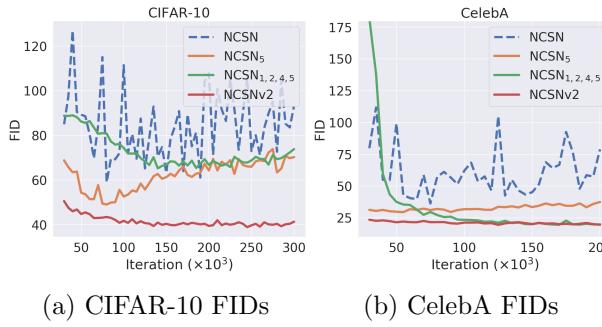


Figure 6.5: FIDs for different groups of techniques. Subscripts of “NCSN” are IDs of techniques in effect. “NCSNV2” uses all techniques. Results are computed with the denoising step.

Table 6.1: Inception and FID scores.

Model	Inception \uparrow	FID \downarrow
CIFAR-10 Unconditional		
PixelCNN [Oor+16b]	4.60	65.93
IGEBM [DM19a]	6.02	40.58
WGAN-GP [Gul+17]	$7.86 \pm .07$	36.4
SNGAN [Miy+18]	$8.22 \pm .05$	21.7
NCSN [SE19b]	$8.87 \pm .12$	25.32
NCSN (w/ denoising)	$7.32 \pm .12$	29.8
NCSNV2 (w/o denoising)	$8.73 \pm .13$	31.75
NCSNV2 (w/ denoising)	$8.40 \pm .07$	10.87
CelebA 64 \times 64		
NCSN (w/o denoising)	-	26.89
NCSN (w/ denoising)	-	25.30
NCSNV2 (w/o denoising)	-	28.86
NCSNV2 (w/ denoising)	-	10.23

during training, and compute full FID and Inception scores on more samples from them. As shown by results in Table 6.1, NCSNV2 (w/ denoising) is able to significantly improve the FID scores of NCSN on both CIFAR-10 and CelebA, while bearing a slight loss of Inception scores on CIFAR-10. However, we note that Inception and FID scores have known issues [BS18; Saj+18] and they should be interpreted with caution as they may not correlate with visual quality in the expected way. In particular, they can be sensitive to slight noise perturbations [ROV19], as shown by the difference of scores with and without denoising in Table 6.1. To verify that NCSNV2 indeed generates better images than NCSN, we provide additional uncurated samples in Appendix B.4.2 for visual comparison.

Ablation studies: We conduct ablation studies to isolate the contributions of different techniques. We partition all techniques into three groups: (i) Technique 6.5, (ii) Techniques 6.1, 6.2 and 6.4, and (iii) Technique 6.3, where different groups can be applied simultaneously. Techniques 6.1, 6.2 and 6.4 are grouped together because Techniques 6.1 and 6.2 collectively determine the set of noise scales, and to sample from NCSNs trained with these noise scales we need Technique 6.4 to configure annealed Langevin dynamics properly. We test the performance of successively removing groups (iii), (ii), (i) from NCSNV2, and report results in Fig. 6.5 for sampling with denoising and in Fig. B.17 (Appendix B.4.2) for sampling without denoising. All groups of techniques improve over the vanilla NCSN. Although the FID scores are not strictly increasing when removing (iii), (ii), and (i) progressively, we note that FIDs may not always correlate with sample quality well. In fact, we do observe decreasing sample quality by visual inspection (see Appendix B.4.2), and combining

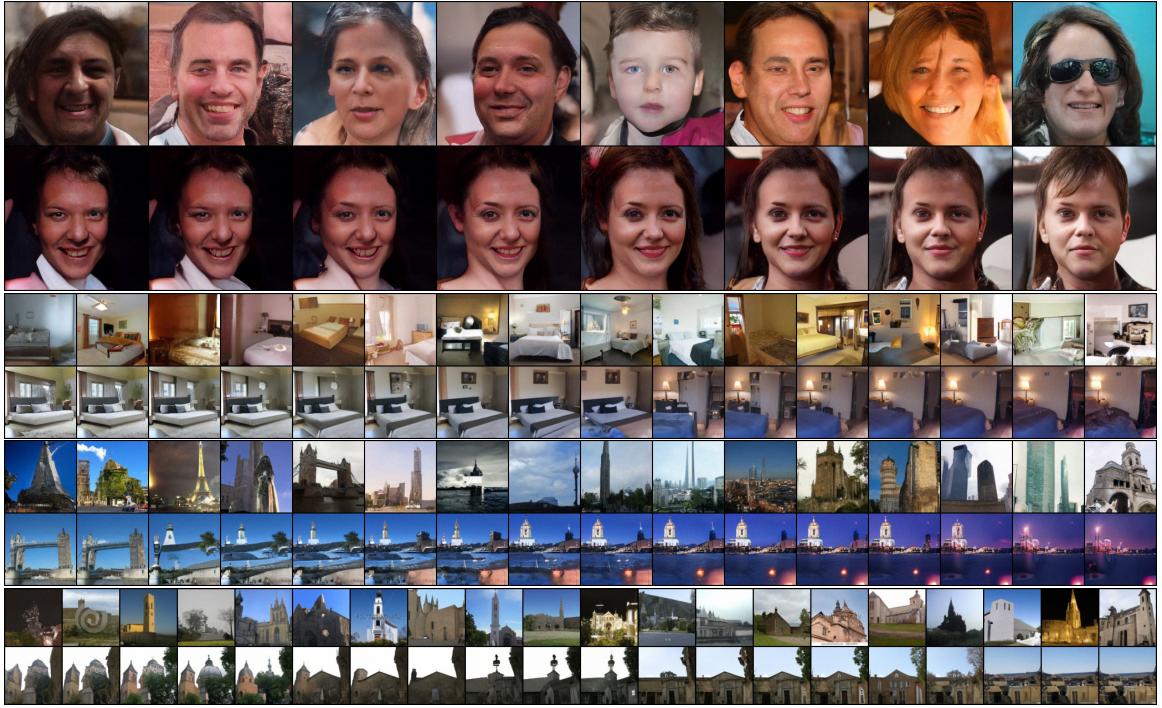


Figure 6.6: From top to bottom: FFHQ 256^2 , LSUN bedroom 128^2 , LSUN tower 128^2 , and LSUN church_outdoor 96^2 . Within each group of images: the first row shows uncurated samples from NCSNv2, and the second shows the interpolation results between the leftmost and rightmost samples with NCSNv2. You may zoom in to view more details.

all techniques gives the best samples.

Towards higher resolution: The original NCSN only succeeds at generating images of low resolution. In fact, Chapter 5 only tested it on MNIST 28×28 and CelebA/CIFAR-10 32×32 . For slightly larger images such as CelebA 64×64 , NCSN can generate images of consistent global structure, yet with strong color artifacts that are easily noticeable (see Fig. 6.4 and compare Fig. B.18a with Fig. B.18b). For images with resolutions beyond 96×96 , NCSN will completely fail to produce samples with correct structure or color (see



Figure 6.7: NCSN vs. NCSNv2 samples on LSUN church_outdoor (a)(b) and LSUN bedroom (c)(d).

Fig. 6.7). All samples shown here are generated without the denoising step, but since σ_L is very small, they are visually indistinguishable from ones with the denoising step.

By combining Techniques 6.1 to 6.5, NCSNv2 can work on images of much higher resolution. Note that we directly calculated the noise scales for training NCSNs, and computed the step size for annealed Langevin dynamics sampling without manual hyper-parameter tuning. The network architectures are the same across datasets, except that for ones with higher resolution we use more layers and more filters to ensure the receptive field and model capacity are large enough (see details in Appendix B.4.1). In Fig. 6.6 and 6.1, we show NCSNv2 is capable of generating high-fidelity image samples with resolutions ranging from 96×96 to 256×256 . To show that this high sample quality is not a result of dataset memorization, we provide the loss curves for training/test, as well as nearest neighbors for samples in Appendix B.4.2. In addition, NCSNv2 can produce smooth interpolations between two given samples as in Fig. 6.6 (details in Appendix B.4.1), indicating the ability to learn generalizable image representations.

6.7 Conclusion

Motivated by both theoretical analyses and empirical observations, we propose a set of techniques to improve score-based generative models. Our techniques significantly improve the training and sampling processes, lead to better sample quality, and enable high-fidelity image generation at high resolutions. Although our techniques work well without manual tuning, we believe that the performance can be improved even more by fine-tuning various hyper-parameters. Future directions include theoretical understandings on the sample quality of score-based generative models, as well as alternative noise distributions to Gaussian perturbations.

Chapter 7

Data Generation with Differential Equations

Creating noise from data is easy; creating data from noise is generative modeling. In previous chapters, we notice that the sample quality of NCSNs improves on several image datasets as we increase the number of noise levels. Given this observation, one natural question to ask is whether it is possible to employ an infinite number of noise levels to maximize the performance of score-based generative models.

In this chapter, we investigate this possibility using the tools of stochastic differential equations (SDEs). Specifically, we present an SDE that smoothly transforms a complex data distribution to a known prior distribution by slowly injecting noise, and a corresponding reverse-time SDE that transforms the prior distribution back into the data distribution by slowly removing the noise. Crucially, the reverse-time SDE depends only on the time-dependent gradient field (a.k.a., score) of the perturbed data distribution. We can accurately estimate these scores with neural networks, and use numerical SDE solvers to generate samples. We show that this framework encapsulates previous approaches in score-based generative modeling and diffusion probabilistic modeling, allowing for new sampling procedures and new modeling capabilities. In particular, we introduce a predictor-corrector framework to correct errors in the evolution of the discretized reverse-time SDE. We also derive an equivalent neural ODE that samples from the same distribution as the SDE with a deterministic sampling procedure and improved sampling efficiency. In addition, we provide a new way to solve inverse problems with score-based models, as demonstrated with experiments on class-conditional generation, image inpainting, and colorization. Combined with multiple

architectural improvements, we achieve record-breaking performance for unconditional image generation on CIFAR-10 with an Inception score of 9.89 and FID of 2.20, and demonstrate high fidelity generation of 1024×1024 images for the first time from a score-based generative model.

This chapter is primarily based on our previous publication [Son+21b].

7.1 Introduction

Two successful classes of probabilistic generative models involve sequentially corrupting training data with slowly increasing noise, and then learning to reverse this corruption in order to form a generative model of the data. *Score matching with Langevin dynamics* (SMILD) [SE19b] (see also Chapter 5) estimates the *score* (*i.e.*, the gradient of the log probability density with respect to data) at each noise scale, and then uses Langevin dynamics to sample from a sequence of decreasing noise scales during generation. *Denoising diffusion probabilistic modeling* (DDPM) [Soh+15; HJA20] trains a sequence of probabilistic models to reverse each step of the noise corruption, using knowledge of the functional form of the reverse distributions to make training tractable. For continuous state spaces, the DDPM training objective implicitly computes scores at each noise scale. We can therefore refer to these two model classes together as *score-based generative models*.

Score-based generative models, and related techniques [BHV17; Goy+17; DM19a], have proven effective at generation of images [SE19b; SE20; HJA20], audio [Che+21b; Kon+20], graphs [Niu+20], and shapes [Cai+20a]. To enable new sampling methods and further extend the capabilities of score-based generative models, we propose a unified framework that generalizes previous approaches through the lens of stochastic differential equations (SDEs).

Specifically, instead of perturbing data with a finite number of noise distributions, we consider a continuum of distributions that evolve over time according to a diffusion process. This process progressively diffuses a data point into random noise, and is given by a prescribed SDE that does not depend on the data and has no trainable parameters. By reversing this process, we can smoothly mold random noise into data for sample generation. Crucially, this reverse process satisfies a reverse-time SDE [And82], which can be derived from the forward SDE given the score of the marginal probability densities as a function of time. We can therefore approximate the reverse-time SDE by training a time-dependent neural network to

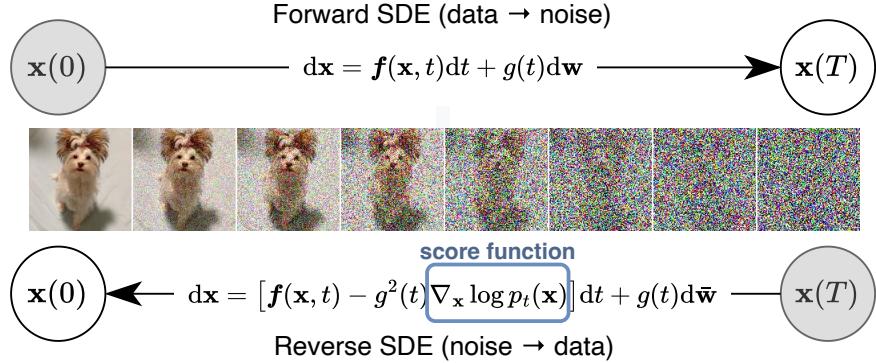


Figure 7.1: **Solving a reverse-time SDE yields a score-based generative model.** Transforming data to a simple noise distribution can be accomplished with a continuous-time SDE. This SDE can be reversed if we know the score of the distribution at each intermediate time step, $\nabla_{\mathbf{x}} \log p_t(\mathbf{x})$.

estimate the scores, and then produce samples using numerical SDE solvers. Our key idea is summarized in Fig. 7.1.

Our proposed framework has several theoretical and practical contributions:

Flexible sampling: We can employ any general-purpose SDE solver to integrate the reverse-time SDE for sampling. In addition, we propose two special methods not viable for general SDEs: (i) Predictor-Corrector (PC) samplers that combine numerical SDE solvers with score-based MCMC approaches, such as Langevin MCMC [Par81] and HMC [Nea+11]; and (ii) deterministic samplers based on the probability flow ordinary differential equation (ODE). The former *unifies and improves* over existing sampling methods for score-based models. The latter allows for *fast adaptive sampling* via black-box ODE solvers, *flexible data manipulation* via latent codes, and a *uniquely identifiable encoding*.

Controllable generation: We can modulate the generation process by conditioning on information not available during training, because the conditional reverse-time SDE can be efficiently estimated from *unconditional* scores. This enables applications such as class-conditional generation, image inpainting, colorization and other inverse problems, all achievable using a single unconditional score-based model without re-training.

Unified framework: Our framework provides a unified way to explore and tune various SDEs for improving score-based generative models. The methods of SMLD and DDPM can be amalgamated into our framework as discretizations of two separate SDEs. Although DDPM [HJA20] was recently reported to achieve higher sample quality than SMLD [SE19b; SE20] (Chapters 5 and 6), we show that with better architectures and new sampling algorithms allowed by our framework, the latter can catch up—it achieves new state-of-the-art Inception score (9.89) and FID score (2.20) on CIFAR-10, as well as high-fidelity generation of 1024×1024 images for the first time from a score-based model.

7.2 Background

7.2.1 Denoising Score Matching with Langevin Dynamics (SMLD)

Let $p_\sigma(\tilde{\mathbf{x}} | \mathbf{x}) := \mathcal{N}(\tilde{\mathbf{x}}; \mathbf{x}, \sigma^2 \mathbf{I})$ be a perturbation kernel, and $p_\sigma(\tilde{\mathbf{x}}) := \int p_{\text{data}}(\mathbf{x}) p_\sigma(\tilde{\mathbf{x}} | \mathbf{x}) d\mathbf{x}$, where $p_{\text{data}}(\mathbf{x})$ denotes the data distribution. Consider a sequence of positive noise scales $\sigma_{\min} = \sigma_1 < \sigma_2 < \dots < \sigma_N = \sigma_{\max}$. Typically, σ_{\min} is small enough such that $p_{\sigma_{\min}}(\mathbf{x}) \approx p_{\text{data}}(\mathbf{x})$, and σ_{\max} is large enough such that $p_{\sigma_{\max}}(\mathbf{x}) \approx \mathcal{N}(\mathbf{x}; \mathbf{0}, \sigma_{\max}^2 \mathbf{I})$. Chapter 5 proposes to train a Noise Conditional Score Network (NCSN), denoted by $s_\theta(\mathbf{x}, \sigma)$, with a weighted sum of denoising score matching [Vin11] objectives:

$$\boldsymbol{\theta}^* = \arg \min_{\boldsymbol{\theta}} \sum_{i=1}^N \sigma_i^2 \mathbb{E}_{p_{\text{data}}(\mathbf{x})} \mathbb{E}_{p_{\sigma_i}(\tilde{\mathbf{x}} | \mathbf{x})} [\|s_\theta(\tilde{\mathbf{x}}, \sigma_i) - \nabla_{\tilde{\mathbf{x}}} \log p_{\sigma_i}(\tilde{\mathbf{x}} | \mathbf{x})\|_2^2]. \quad (7.1)$$

Given sufficient data and model capacity, the optimal score-based model $s_{\theta^*}(\mathbf{x}, \sigma)$ matches $\nabla_{\mathbf{x}} \log p_\sigma(\mathbf{x})$ almost everywhere for $\sigma \in \{\sigma_i\}_{i=1}^N$. For sampling, we run M steps of Langevin MCMC in Chapter 5 to get a sample for each $p_{\sigma_i}(\mathbf{x})$ sequentially:

$$\mathbf{x}_i^m = \mathbf{x}_i^{m-1} + \epsilon_i s_{\theta^*}(\mathbf{x}_i^{m-1}, \sigma_i) + \sqrt{2\epsilon_i} \mathbf{z}_i^m, \quad m = 1, 2, \dots, M, \quad (7.2)$$

where $\epsilon_i > 0$ is the step size, and \mathbf{z}_i^m is standard normal. The above is repeated for $i = N, N-1, \dots, 1$ in turn with $\mathbf{x}_N^0 \sim \mathcal{N}(\mathbf{x} | \mathbf{0}, \sigma_{\max}^2 \mathbf{I})$ and $\mathbf{x}_i^0 = \mathbf{x}_{i+1}^M$ when $i < N$. As $M \rightarrow \infty$ and $\epsilon_i \rightarrow 0$ for all i , \mathbf{x}_1^M becomes an exact sample from $p_{\sigma_{\min}}(\mathbf{x}) \approx p_{\text{data}}(\mathbf{x})$ under some regularity conditions.

7.2.2 Denoising Diffusion Probabilistic Models (DDPM)

Sohl-Dickstein et al. [Soh+15] and Ho et al. [HJA20] consider a sequence of positive noise scales $0 < \beta_1, \beta_2, \dots, \beta_N < 1$. For each training data point $\mathbf{x}_0 \sim p_{\text{data}}(\mathbf{x})$, a discrete Markov chain $\{\mathbf{x}_0, \mathbf{x}_1, \dots, \mathbf{x}_N\}$ is constructed such that $p(\mathbf{x}_i | \mathbf{x}_{i-1}) = \mathcal{N}(\mathbf{x}_i; \sqrt{1 - \beta_i} \mathbf{x}_{i-1}, \beta_i \mathbf{I})$, and therefore $p_{\alpha_i}(\mathbf{x}_i | \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_i; \sqrt{\alpha_i} \mathbf{x}_0, (1 - \alpha_i) \mathbf{I})$, where $\alpha_i := \prod_{j=1}^i (1 - \beta_j)$. Similar to SMLD, we can denote the perturbed data distribution as $p_{\alpha_i}(\tilde{\mathbf{x}}) := \int p_{\text{data}}(\mathbf{x}) p_{\alpha_i}(\tilde{\mathbf{x}} | \mathbf{x}) d\mathbf{x}$. The noise scales are prescribed such that \mathbf{x}_N is approximately distributed according to $\mathcal{N}(\mathbf{0}, \mathbf{I})$. A variational Markov chain in the reverse direction is parameterized with $p_{\theta}(\mathbf{x}_{i-1} | \mathbf{x}_i) = \mathcal{N}(\mathbf{x}_{i-1}; \frac{1}{\sqrt{1 - \beta_i}} (\mathbf{x}_i + \beta_i \mathbf{s}_{\theta}(\mathbf{x}_i, i)), \beta_i \mathbf{I})$, and trained with a re-weighted variant of the evidence lower bound (ELBO):

$$\boldsymbol{\theta}^* = \arg \min_{\boldsymbol{\theta}} \sum_{i=1}^N (1 - \alpha_i) \mathbb{E}_{p_{\text{data}}(\mathbf{x})} \mathbb{E}_{p_{\alpha_i}(\tilde{\mathbf{x}} | \mathbf{x})} [\|\mathbf{s}_{\theta}(\tilde{\mathbf{x}}, i) - \nabla_{\tilde{\mathbf{x}}} \log p_{\alpha_i}(\tilde{\mathbf{x}} | \mathbf{x})\|_2^2]. \quad (7.3)$$

After solving Eq. (7.3) to get the optimal model $\mathbf{s}_{\theta^*}(\mathbf{x}, i)$, samples can be generated by starting from $\mathbf{x}_N \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and following the estimated reverse Markov chain as below

$$\mathbf{x}_{i-1} = \frac{1}{\sqrt{1 - \beta_i}} (\mathbf{x}_i + \beta_i \mathbf{s}_{\theta^*}(\mathbf{x}_i, i)) + \sqrt{\beta_i} \mathbf{z}_i, \quad i = N, N-1, \dots, 1. \quad (7.4)$$

We call this method *ancestral sampling*, since it amounts to performing ancestral sampling from the graphical model $\prod_{i=1}^N p_{\theta}(\mathbf{x}_{i-1} | \mathbf{x}_i)$. The objective Eq. (7.3) described here is L_{simple} in [HJA20], written in a form to expose more similarity to Eq. (7.1). Like Eq. (7.1), Eq. (7.3) is also a weighted sum of denoising score matching objectives, which implies that the optimal model, $\mathbf{s}_{\theta^*}(\tilde{\mathbf{x}}, i)$, matches the score of the perturbed data distribution, $\nabla_{\tilde{\mathbf{x}}} \log p_{\alpha_i}(\tilde{\mathbf{x}})$. Notably, the weights of the i -th summand in Eq. (7.1) and Eq. (7.3), namely σ_i^2 and $(1 - \alpha_i)$, are related to corresponding perturbation kernels in the same functional form: $\sigma_i^2 \propto 1/\mathbb{E}[\|\nabla_{\tilde{\mathbf{x}}} \log p_{\alpha_i}(\tilde{\mathbf{x}} | \mathbf{x})\|_2^2]$ and $(1 - \alpha_i) \propto 1/\mathbb{E}[\|\nabla_{\tilde{\mathbf{x}}} \log p_{\alpha_i}(\tilde{\mathbf{x}} | \mathbf{x})\|_2^2]$.

7.3 Score-Based Generative Modeling with SDEs

Perturbing data with multiple noise scales is key to the success of previous methods. We propose to generalize this idea further to an infinite number of noise scales, such that perturbed data distributions evolve according to an SDE as the noise intensifies. An overview of our framework is given in Fig. 7.2.

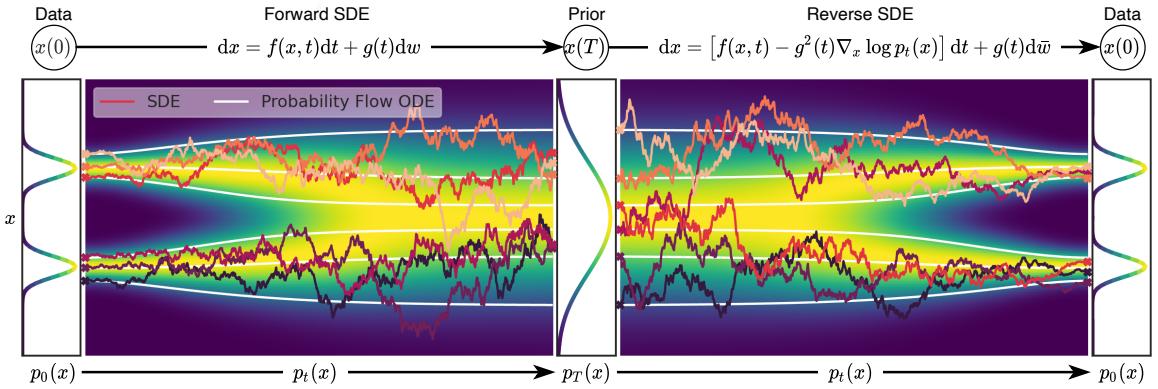


Figure 7.2: Overview of score-based generative modeling through SDEs. We can map data to a noise distribution (the prior) with an SDE (Section 7.3.1), and reverse this SDE for generative modeling (Section 7.3.2). We can also reverse the associated probability flow ODE (Section 7.4.3), which yields a deterministic process that samples from the same distribution as the SDE. Both the reverse-time SDE and probability flow ODE can be obtained by estimating the score $\nabla_{\mathbf{x}} \log p_t(\mathbf{x})$ (Section 7.3.3).

7.3.1 Perturbing Data with SDEs

Our goal is to construct a diffusion process $\{\mathbf{x}(t)\}_{t=0}^T$ indexed by a continuous time variable $t \in [0, T]$, such that $\mathbf{x}(0) \sim p_0$, for which we have a dataset of i.i.d. samples, and $\mathbf{x}(T) \sim p_T$, for which we have a tractable form to generate samples efficiently. In other words, p_0 is the data distribution and p_T is the prior distribution. This diffusion process can be modeled as the solution to an Itô SDE:

$$d\mathbf{x} = \mathbf{f}(\mathbf{x}, t) dt + g(t) d\mathbf{w}, \quad (7.5)$$

where \mathbf{w} is the standard Wiener process (a.k.a., Brownian motion), $\mathbf{f}(\cdot, t) : \mathbb{R}^d \rightarrow \mathbb{R}^d$ is a vector-valued function called the *drift* coefficient of $\mathbf{x}(t)$, and $g(\cdot) : \mathbb{R} \rightarrow \mathbb{R}$ is a scalar function known as the *diffusion* coefficient of $\mathbf{x}(t)$. For ease of presentation we assume the diffusion coefficient is a scalar (instead of a $d \times d$ matrix) and does not depend on \mathbf{x} , but our theory can be generalized to hold in those cases (see Appendix B.5.1). The SDE has a unique strong solution as long as the coefficients are globally Lipschitz in both state and time [Oks13]. We hereafter denote by $p_t(\mathbf{x})$ the probability density of $\mathbf{x}(t)$, and use $p_{st}(\mathbf{x}(t) | \mathbf{x}(s))$ to denote the transition kernel from $\mathbf{x}(s)$ to $\mathbf{x}(t)$, where $0 \leq s < t \leq T$.

Typically, p_T is an unstructured prior distribution that contains no information of p_0 ,

such as a Gaussian distribution with fixed mean and variance. There are various ways of designing the SDE in Eq. (7.5) such that it diffuses the data distribution into a fixed prior distribution. We provide several examples later in Section 7.3.4 that are derived from continuous generalizations of SMLD and DDPM.

7.3.2 Generating Samples by Reversing the SDE

By starting from samples of $\mathbf{x}(T) \sim p_T$ and reversing the process, we can obtain samples $\mathbf{x}(0) \sim p_0$. A remarkable result from [And82] states that the reverse of a diffusion process is also a diffusion process, running backwards in time and given by the reverse-time SDE:

$$d\mathbf{x} = [\mathbf{f}(\mathbf{x}, t) - g(t)^2 \nabla_{\mathbf{x}} \log p_t(\mathbf{x})] dt + g(t) d\bar{\mathbf{w}}, \quad (7.6)$$

where $\bar{\mathbf{w}}$ is a standard Wiener process when time flows backwards from T to 0, and dt is an infinitesimal negative timestep. Once the score of each marginal distribution, $\nabla_{\mathbf{x}} \log p_t(\mathbf{x})$, is known for all t , we can derive the reverse diffusion process from Eq. (7.6) and simulate it to sample from p_0 .

7.3.3 Estimating Scores for the SDE

The score of a distribution can be estimated by training a score-based model on samples with score matching [HD05; Son+19]. To estimate $\nabla_{\mathbf{x}} \log p_t(\mathbf{x})$, we can train a time-dependent score-based model $\mathbf{s}_{\theta}(\mathbf{x}, t)$ via a continuous generalization to Eqs. (7.1) and (7.3):

$$\boldsymbol{\theta}^* = \arg \min_{\boldsymbol{\theta}} \mathbb{E}_t \left\{ \lambda(t) \mathbb{E}_{\mathbf{x}(0)} \mathbb{E}_{\mathbf{x}(t)|\mathbf{x}(0)} \left[\left\| \mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}(t), t) - \nabla_{\mathbf{x}(t)} \log p_{0t}(\mathbf{x}(t) | \mathbf{x}(0)) \right\|_2^2 \right] \right\}. \quad (7.7)$$

Here $\lambda : [0, T] \rightarrow \mathbb{R}_{>0}$ is a positive weighting function, t is uniformly sampled over $[0, T]$, $\mathbf{x}(0) \sim p_0(\mathbf{x})$ and $\mathbf{x}(t) \sim p_{0t}(\mathbf{x}(t) | \mathbf{x}(0))$. With sufficient data and model capacity, score matching ensures that the optimal solution to Eq. (7.7), denoted by $\mathbf{s}_{\boldsymbol{\theta}^*}(\mathbf{x}, t)$, equals $\nabla_{\mathbf{x}} \log p_t(\mathbf{x})$ for almost all \mathbf{x} and t . As in SMLD and DDPM, we can typically choose $\lambda \propto 1/\mathbb{E}[\|\nabla_{\mathbf{x}(t)} \log p_{0t}(\mathbf{x}(t) | \mathbf{x}(0))\|_2^2]$. Note that Eq. (7.7) uses denoising score matching, but other score matching objectives, such as sliced score matching [Son+19] (Chapter 3) and finite-difference score matching [Pan+20] are also applicable here.

We typically need to know the transition kernel $p_{0t}(\mathbf{x}(t) | \mathbf{x}(0))$ to efficiently solve Eq. (7.7). When $\mathbf{f}(\cdot, t)$ is affine, the transition kernel is always a Gaussian distribution,

where the mean and variance are often known in closed-forms and can be obtained with standard techniques (see Section 5.5 in [SS19]). For more general SDEs, we may solve Kolmogorov's forward equation [Oks13] to obtain $p_{0t}(\mathbf{x}(t) | \mathbf{x}(0))$. Alternatively, we can simulate the SDE to sample from $p_{0t}(\mathbf{x}(t) | \mathbf{x}(0))$ and replace denoising score matching in Eq. (7.7) with sliced score matching for model training, which bypasses the computation of $\nabla_{\mathbf{x}(t)} \log p_{0t}(\mathbf{x}(t) | \mathbf{x}(0))$ (see Appendix B.5.1).

7.3.4 Examples: VE, VP SDEs and Beyond

The noise perturbations used in SMLD and DDPM can be regarded as discretizations of two different SDEs. Below we provide a brief discussion and relegate more details to Appendix B.5.2.

When using a total of N noise scales, each perturbation kernel $p_{\sigma_i}(\mathbf{x} | \mathbf{x}_0)$ of SMLD corresponds to the distribution of \mathbf{x}_i in the following Markov chain:

$$\mathbf{x}_i = \mathbf{x}_{i-1} + \sqrt{\sigma_i^2 - \sigma_{i-1}^2} \mathbf{z}_{i-1}, \quad i = 1, \dots, N, \quad (7.8)$$

where $\mathbf{z}_{i-1} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, and we have introduced $\sigma_0 = 0$ to simplify the notation. In the limit of $N \rightarrow \infty$, $\{\sigma_i\}_{i=1}^N$ becomes a function $\sigma(t)$, \mathbf{z}_i becomes $\mathbf{z}(t)$, and the Markov chain $\{\mathbf{x}_i\}_{i=1}^N$ becomes a continuous stochastic process $\{\mathbf{x}(t)\}_{t=0}^1$, where we have used a continuous time variable $t \in [0, 1]$ for indexing, rather than an integer i . The process $\{\mathbf{x}(t)\}_{t=0}^1$ is given by the following SDE

$$d\mathbf{x} = \sqrt{\frac{d[\sigma^2(t)]}{dt}} d\mathbf{w}. \quad (7.9)$$

Likewise for the perturbation kernels $\{p_{\alpha_i}(\mathbf{x} | \mathbf{x}_0)\}_{i=1}^N$ of DDPM, the discrete Markov chain is

$$\mathbf{x}_i = \sqrt{1 - \beta_i} \mathbf{x}_{i-1} + \sqrt{\beta_i} \mathbf{z}_{i-1}, \quad i = 1, \dots, N. \quad (7.10)$$

As $N \rightarrow \infty$, Eq. (7.10) converges to the following SDE,

$$d\mathbf{x} = -\frac{1}{2} \beta(t) \mathbf{x} dt + \sqrt{\beta(t)} d\mathbf{w}. \quad (7.11)$$

Therefore, the noise perturbations used in SMLD and DDPM correspond to discretizations

of SDEs Eqs. (7.9) and (7.11). Interestingly, the SDE of Eq. (7.9) always gives a process with exploding variance when $t \rightarrow \infty$, whilst the SDE of Eq. (7.11) yields a process with a fixed variance of one when the initial distribution has unit variance (proof in Appendix B.5.2). Due to this difference, we hereafter refer to Eq. (7.9) as the Variance Exploding (VE) SDE, and Eq. (7.11) the Variance Preserving (VP) SDE.

Inspired by the VP SDE, we propose a new type of SDEs, given by

$$d\mathbf{x} = -\frac{1}{2}\beta(t)\mathbf{x} dt + \sqrt{\beta(t)(1 - e^{-2\int_0^t \beta(s) ds})} d\mathbf{w}. \quad (7.12)$$

When using the same $\beta(t)$ and starting from the same initial distribution, the variance of the stochastic process induced by Eq. (7.12) is always bounded by the VP SDE at every intermediate time step (proof in Appendix B.5.2). For this reason, we name Eq. (7.12) the sub-VP SDE.

Since VE, VP and sub-VP SDEs all have affine drift coefficients, their perturbation kernels $p_{0t}(\mathbf{x}(t) | \mathbf{x}(0))$ are all Gaussian and can be computed in closed-forms, as discussed in Section 7.3.3. This makes training with Eq. (7.7) particularly efficient.

7.4 Solving the Reverse SDE

After training a time-dependent score-based model s_θ , we can use it to construct the reverse-time SDE and then simulate it with numerical approaches to generate samples from p_0 .

7.4.1 General-Purpose Numerical SDE Solvers

Numerical solvers provide approximate trajectories from SDEs. Many general-purpose numerical methods exist for solving SDEs, such as Euler-Maruyama and stochastic Runge-Kutta methods [KP13], which correspond to different discretizations of the stochastic dynamics. We can apply any of them to the reverse-time SDE for sample generation.

Ancestral sampling, the sampling method of DDPM (Eq. (7.4)), actually corresponds to one special discretization of the reverse-time VP SDE (Eq. (7.11)) (see Appendix B.5.5). Deriving the ancestral sampling rules for new SDEs, however, can be non-trivial. To remedy this, we propose *reverse diffusion samplers* (details in Appendix B.5.5), which discretize the reverse-time SDE in the same way as the forward one, and thus can be readily derived

Table 7.1: Comparing different reverse-time SDE solvers on CIFAR-10. Shaded regions are obtained with the same computation (number of score function evaluations). Mean and standard deviation are reported over five sampling runs. “P1000” or “P2000”: predictor-only samplers using 1000 or 2000 steps. “C2000”: corrector-only samplers using 2000 steps. “PC1000”: Predictor-Corrector (PC) samplers using 1000 predictor and 1000 corrector steps.

FID↓\ Sampler ↓\ Predictor	Variance Exploding SDE (SMLD)				Variance Preserving SDE (DDPM)			
	P1000	P2000	C2000	PC1000	P1000	P2000	C2000	PC1000
	ancestral sampling	$4.98 \pm .06$	$4.88 \pm .06$		$3.62 \pm .03$	$3.24 \pm .02$	$3.24 \pm .02$	
reverse diffusion	$4.79 \pm .07$	$4.74 \pm .08$	$20.43 \pm .07$		$3.60 \pm .02$	$3.21 \pm .02$	$3.19 \pm .02$	$19.06 \pm .06$
probability flow	$15.41 \pm .15$	$10.54 \pm .08$			$3.51 \pm .04$	$3.59 \pm .04$	$3.23 \pm .03$	

given the forward discretization. As shown in Table 7.1, reverse diffusion samplers perform slightly better than ancestral sampling for both SMLD and DDPM models on CIFAR-10 (DDPM-type ancestral sampling is also applicable to SMLD models, see Appendix B.5.6.)

7.4.2 Predictor-Corrector Samplers

Unlike generic SDEs, we have additional information that can be used to improve solutions. Since we have a score-based model $s_{\theta^*}(\mathbf{x}, t) \approx \nabla_{\mathbf{x}} \log p_t(\mathbf{x})$, we can employ score-based MCMC approaches, such as Langevin MCMC [Par81; GM94] or HMC [Nea+11] to sample from p_t directly, and correct the solution of a numerical SDE solver.

Specifically, at each time step, the numerical SDE solver first gives an estimate of the sample at the next time step, playing the role of a “predictor”. Then, the score-based MCMC approach corrects the marginal distribution of the estimated sample, playing the role of a “corrector”. The idea is analogous to Predictor-Corrector methods, a family of numerical continuation techniques for solving systems of equations [AG12], and we similarly name our hybrid sampling algorithms *Predictor-Corrector* (PC) samplers. Please find pseudo-code and a complete description in Appendix B.5.7. PC samplers generalize the original sampling methods of SMLD and DDPM: the former uses an identity function as the predictor and annealed Langevin dynamics as the corrector, while the latter uses ancestral sampling as the predictor and identity as the corrector.

We test PC samplers on SMLD and DDPM models (see Algorithms B.4 and B.5 in Appendix B.5.7) trained with original discrete objectives given by Eqs. (7.1) and (7.3). This exhibits the compatibility of PC samplers to score-based models trained with a fixed

number of noise scales. We summarize the performance of different samplers in Table 7.1, where probability flow is a predictor to be discussed in Section 7.4.3. Detailed experimental settings and additional results are given in Appendix B.5.7. We observe that our reverse diffusion sampler always outperform ancestral sampling, and corrector-only methods (C2000) perform worse than other competitors (P2000, PC1000) with the same computation (In fact, we need way more corrector steps per noise scale, and thus more computation, to match the performance of other samplers.) For all predictors, adding one corrector step for each predictor step (PC1000) doubles computation but always improves sample quality (against P1000). Moreover, it is typically better than doubling the number of predictor steps without adding a corrector (P2000), where we have to interpolate between noise scales in an ad hoc manner (detailed in Appendix B.5.7) for SMLD/DDPM models. In Fig. B.43 (Appendix B.5.7), we additionally provide qualitative comparison for models trained with the continuous objective Eq. (7.7) on 256×256 LSUN images and the VE SDE, where PC samplers clearly surpass predictor-only samplers under comparable computation, when using a proper number of corrector steps.

7.4.3 Probability Flow and Connection to Neural ODEs

Score-based models enable another numerical method for solving the reverse-time SDE. For all diffusion processes, there exists a corresponding *deterministic process* whose trajectories share the same marginal probability densities $\{p_t(\mathbf{x})\}_{t=0}^T$ as the SDE. This deterministic process satisfies an ODE (more details in Appendix B.5.4):

$$d\mathbf{x} = \left[\mathbf{f}(\mathbf{x}, t) - \frac{1}{2}g(t)^2 \nabla_{\mathbf{x}} \log p_t(\mathbf{x}) \right] dt, \quad (7.13)$$

which can be determined from the SDE once scores are known. We name the ODE in Eq. (7.13) the *probability flow ODE*. When the score function is approximated by the time-dependent score-based model, which is typically a neural network, this is an example of a neural ODE [Che+18a].

Manipulating latent representations By integrating Eq. (7.13), we can encode any data point $\mathbf{x}(0)$ into a latent space $\mathbf{x}(T)$. Decoding can be achieved by integrating a corresponding ODE for the reverse-time SDE. As is done with other invertible models such as neural ODEs and normalizing flows [DSB17; KD18], we can manipulate this latent representation for image editing, such as interpolation, and temperature scaling (see Fig. 7.3

Table 7.2: CIFAR-10 sample quality.

Model	FID↓	IS↑
Conditional		
BigGAN [BDS19]	14.73	9.22
StyleGAN2-ADA [Kar+20a]	2.42	10.14
Unconditional		
StyleGAN2-ADA [Kar+20a]	2.92	9.83
NCSN [SE19b]	25.32	$8.87 \pm .12$
NCSNv2 [SE20]	10.87	$8.40 \pm .07$
DDPM [HJA20]	3.17	$9.46 \pm .11$
DDPM++	2.78	9.64
DDPM++ cont. (VP)	2.55	9.58
DDPM++ cont. (sub-VP)	2.61	9.56
DDPM++ cont. (deep, VP)	2.41	9.68
DDPM++ cont. (deep, sub-VP)	2.41	9.57
NCSN++	2.45	9.73
NCSN++ cont. (VE)	2.38	9.83
NCSN++ cont. (deep, VE)	2.20	9.89

and Appendix B.5.4).

Uniquely identifiable encoding Unlike most current invertible models, our encoding is *uniquely identifiable*, meaning that with sufficient training data, model capacity, and optimization accuracy, the encoding for an input is uniquely determined by the data distribution [RMK20]. This is because our forward SDE, Eq. (7.5), has no trainable parameters, and its associated probability flow ODE, Eq. (7.13), provides the same trajectories given perfectly estimated scores. We provide additional empirical verification on this property in Appendix B.5.4.

Efficient sampling As with neural ODEs, we can sample $\mathbf{x}(0) \sim p_0$ by solving Eq. (7.13) from different final conditions $\mathbf{x}(T) \sim p_T$. Using a fixed discretization strategy we can generate competitive samples, especially when used in conjunction with correctors (Table 7.1, “probability flow sampler”, details in Appendix B.5.4). Using a black-box ODE solver [DP80] not only produces high quality samples (details in Appendix B.5.4), but also allows us to explicitly trade-off accuracy for efficiency. With a larger error tolerance, the number of function evaluations can be reduced by over 90% without affecting the visual quality of samples (Fig. 7.3).

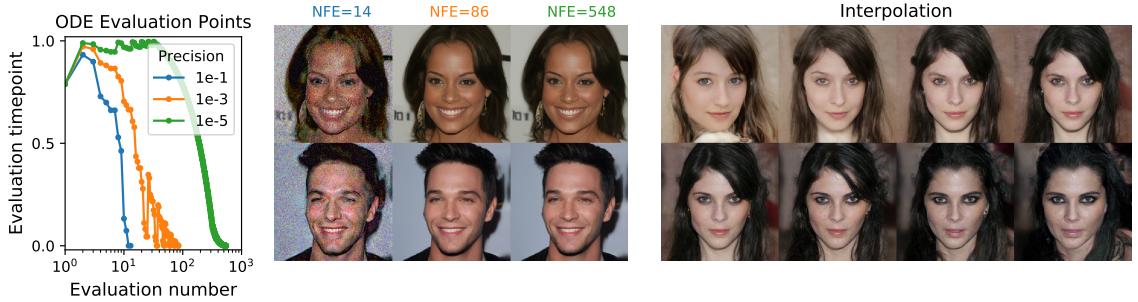


Figure 7.3: **Probability flow ODE** enables fast sampling with adaptive stepsizes as the numerical precision is varied (*left*), and reduces the number of score function evaluations (NFE) without harming quality (*middle*). The invertible mapping from latents to images allows for interpolations (*right*).

7.4.4 Architecture Improvements

We explore several new architecture designs for score-based models using both VE and VP SDEs (details in Appendix B.5.8), where we train models with the same discrete objectives as in SMLD/DDPM. We directly transfer the architectures for VP SDEs to sub-VP SDEs due to their similarity. Our optimal architecture for the VE SDE, named NCSN++, achieves an FID of 2.45 on CIFAR-10 with PC samplers, while our optimal architecture for the VP SDE, called DDPM++, achieves 2.78.

By switching to the continuous training objective in Eq. (7.7), and increasing the network depth, we can further improve sample quality for all models. The resulting architectures are denoted as NCSN++ cont. and DDPM++ cont. in Table 7.2 for VE and VP/sub-VP SDEs respectively. Results reported in Table 7.2 are for the checkpoint with the smallest FID over the course of training, where samples are generated with PC samplers. As shown in Table 7.2, VE SDEs can provide better sample quality than VP/sub-VP SDEs.

Our best model for sample quality, NCSN++ cont. (deep, VE), doubles the network depth and sets new records for both inception score and FID on unconditional generation for CIFAR-10. Surprisingly, we can achieve better FID than the previous best conditional generative model without requiring labeled data. With all improvements together, we also obtain the first set of high-fidelity samples on CelebA-HQ 1024×1024 from score-based models (see Appendix B.5.8).

7.5 Controllable Generation

The continuous structure of our framework allows us to not only produce data samples from p_0 , but also from $p_0(\mathbf{x}(0) \mid \mathbf{y})$ if $p_t(\mathbf{y} \mid \mathbf{x}(t))$ is known. Given a forward SDE as in Eq. (7.5), we can sample from $p_t(\mathbf{x}(t) \mid \mathbf{y})$ by starting from $p_T(\mathbf{x}(T) \mid \mathbf{y})$ and solving a conditional reverse-time SDE:

$$d\mathbf{x} = \{\mathbf{f}(\mathbf{x}, t) - g(t)^2 [\nabla_{\mathbf{x}} \log p_t(\mathbf{x}) + \nabla_{\mathbf{x}} \log p_t(\mathbf{y} \mid \mathbf{x})]\} dt + g(t) d\bar{\mathbf{w}}. \quad (7.14)$$

In general, we can use Eq. (7.14) to solve a large family of *inverse problems* with score-based generative models, once given an estimate of the gradient of the forward process, $\nabla_{\mathbf{x}} \log p_t(\mathbf{y} \mid \mathbf{x}(t))$. In some cases, it is possible to train a separate model to learn the forward process $\log p_t(\mathbf{y} \mid \mathbf{x}(t))$ and compute its gradient. Otherwise, we may estimate the gradient with heuristics and domain knowledge. In Appendix B.5.9, we provide a broadly applicable method for obtaining such an estimate without the need of training auxiliary models.

We consider three applications of controllable generation with this approach: class-conditional generation, image imputation and colorization. When \mathbf{y} represents class labels, we can train a time-dependent classifier $p_t(\mathbf{y} \mid \mathbf{x}(t))$ for class-conditional sampling. Since the forward SDE is tractable, we can easily create training data $(\mathbf{x}(t), \mathbf{y})$ for the time-dependent classifier by first sampling $(\mathbf{x}(0), \mathbf{y})$ from a dataset, and then sampling $\mathbf{x}(t) \sim p_{0t}(\mathbf{x}(t) \mid \mathbf{x}(0))$. Afterwards, we may employ a mixture of cross-entropy losses over different time steps, like Eq. (7.7), to train the time-dependent classifier $p_t(\mathbf{y} \mid \mathbf{x}(t))$. We provide class-conditional CIFAR-10 samples in Fig. 7.4 (left), and relegate more details and results to Appendix B.5.9.

Imputation is a special case of conditional sampling. Suppose we have an incomplete data point \mathbf{y} where only some subset, $\Omega(\mathbf{y})$ is known. Imputation amounts to sampling from $p(\mathbf{x}(0) \mid \Omega(\mathbf{y}))$, which we can accomplish using an unconditional model (see Appendix B.5.9). Colorization is a special case of imputation, except that the known data dimensions are coupled. We can decouple these data dimensions with an orthogonal linear transformation, and perform imputation in the transformed space (details in Appendix B.5.9). Fig. 7.4 (right) shows results for inpainting and colorization achieved with unconditional time-dependent score-based models.



Figure 7.4: *Left:* Class-conditional samples on 32×32 CIFAR-10. Top four rows are automobiles and bottom four rows are horses. *Right:* Inpainting (top two rows) and colorization (bottom two rows) results on 256×256 LSUN. First column is the original image, second column is the masked/gray-scale image, remaining columns are sampled image completions or colorizations.

7.6 Conclusion

We presented a framework for score-based generative modeling based on SDEs. Our work enables a better understanding of existing approaches, new sampling algorithms, uniquely identifiable encoding, latent code manipulation, and brings new conditional generation abilities to the family of score-based generative models.

While our proposed sampling approaches improve results and enable more efficient sampling, they remain slower at sampling than GANs [Goo+14] on the same datasets. Identifying ways of combining the stable learning of score-based generative models with the fast sampling of implicit models like GANs remains an important research direction. Additionally, the breadth of samplers one can use when given access to score functions introduces a number of hyper-parameters. Future work would benefit from improved methods to automatically select and tune these hyper-parameters, as well as more extensive investigation on the merits and limitations of various samplers.

Chapter 8

Medical Image Reconstruction with Score Models

In the previous chapter, we show that score-based generative models provide a general tool for solving inverse problems, as demonstrated in image inpainting, image colorization, and class-conditional image generation, all with unconditional score models. In this chapter, we employ score-based generative models to solve important inverse problems in medical image reconstruction, showcasing their enormous potential for broadly impacting other fields of science and engineering.

Reconstructing medical images from partial measurements is an important inverse problem in Computed Tomography (CT) and Magnetic Resonance Imaging (MRI). Existing solutions based on machine learning typically train a model to directly map measurements to medical images, leveraging a training dataset of paired images and measurements. These measurements are typically synthesized from images using a fixed physical model of the measurement process, which hinders the generalization capability of models to unknown measurement processes. To address this issue, we propose a fully unsupervised technique for inverse problem solving, leveraging the recently introduced score-based generative models. Specifically, we first train a score-based generative model on medical images to capture their prior distribution. Given measurements and a physical model of the measurement process at test time, we introduce a sampling method to reconstruct an image consistent with both the prior and the observed measurements. Our method does not assume a fixed measurement process during training, and can thus be flexibly adapted to different measurement processes at test time. Empirically, we observe comparable or better performance to supervised

learning techniques in several medical imaging tasks in CT and MRI, while demonstrating significantly better generalization to unknown measurement processes.

This chapter was previously published as [Son+22].

8.1 Introduction

Computed Tomography (CT) and Magnetic Resonance Imaging (MRI) are commonly used imaging tools for medical diagnosis. Reconstructing CT and MRI images from raw measurements (sinograms for CT and k-spaces for MRI) are well-known inverse problems. Specifically, measurements in CT are given by X-ray projections of an object from various directions, and measurements in MRI are obtained by inspecting the Fourier spectrum of an object with magnetic fields. However, since obtaining the full sinogram for CT causes excessive ionizing radiation for patients, and measuring the full k-space of MRI is very time-consuming, it has become important to reduce the number of measurements in CT and MRI. In many cases, only partial measurements, such as sparse-view sinograms and downsampled k-spaces, are available. Due to this loss of information, the inverse problems in CT and MRI are often ill-posed, making image reconstruction especially challenging.

With the rise of machine learning, many methods [Zhu+18; Mar+18; SZX19; Wür+18; GK18; Wei+20] have been proposed for medical image reconstruction using a small number of measurements. Most of these methods are supervised learning techniques. They learn to directly map partial measurements to medical images, by training on a large dataset comprising pairs of CT/MRI images and measurements. These measurements need to be synthesized from medical images with a fixed physical model of the measurement process. However, when the measurement process changes, such as using a different number of CT projections or different downsampling ratio of MRI k-spaces, we have to re-collect the paired dataset with the new measurement process and re-train the model. This prevents models from generalizing effectively to new measurement processes, leading to counter-intuitive instabilities such as more measurements causing worse performance [Ant+20].

In this chapter, we sidestep this difficulty completely by proposing unsupervised methods that do not require a paired dataset for training, and therefore are not restricted to a fixed measurement process. Our main idea is to learn the prior distribution of medical images with a generative model in order to infer the lost information due to partial measurements. Specifically, we propose to train a score-based generative model on medical images as the

data prior, due to its strong performance in image generation [HJA20; DN21]. Given a trained score-based generative model, we provide a family of sampling algorithms to create image samples that are consistent with the observed measurements and the estimated data prior, leveraging the physical measurement process. Once our model is trained, it can be used to solve any inverse problem within the same image domain, as long as the mapping from images to measurements is linear, which holds for a large number of medical imaging applications.

We evaluate the performance of our method on several tasks in CT and MRI. Empirically, we observe comparable or better performance compared to supervised learning counterparts, even when evaluated with the same measurement process in their training. In addition, we are able to uniformly surpass all baselines when changing the number of measurements, *e.g.*, using a different number of projections in sparse-view CT or changing the k-space downsampling ratio in undersampled MRI. Moreover, we show that by plugging in a different measurement process, we can use a single model to perform both sparse-view CT reconstruction and metal artifact removal for CT imaging with metallic implants. To the best of our knowledge, this is the first time that generative models are reported successful on clinical CT data. Collectively, these empirical results indicate that our method is a competitive alternative to supervised techniques in medical image reconstruction and artifact removal, and has the potential to be a universal tool for solving many inverse problems within the same image domain.

8.2 Background

8.2.1 Linear Inverse Problems

An inverse problem seeks to recover an unknown signal from a set of observed measurements. Specifically, suppose $\mathbf{x} \in \mathbb{R}^n$ is an unknown signal, and $\mathbf{y} \in \mathbb{R}^m = \mathbf{A}\mathbf{x} + \boldsymbol{\epsilon}$ is a noisy observation given by m linear measurements, where the measurement acquisition process is represented by a linear operator $\mathbf{A} \in \mathbb{R}^{m \times n}$, and $\boldsymbol{\epsilon} \in \mathbb{R}^m$ represents a noise vector. Solving a linear inverse problem amounts to recovering the signal \mathbf{x} from its measurement \mathbf{y} . Without further assumptions, the problem is ill-defined when $m < n$, so we additionally assume that \mathbf{x} is sampled from a prior distribution $p(\mathbf{x})$. In this probabilistic formulation, the measurement and signal are connected through a measurement distribution $p(\mathbf{y} | \mathbf{x}) = q_{\boldsymbol{\epsilon}}(\mathbf{y} - \mathbf{A}\mathbf{x})$, where $q_{\boldsymbol{\epsilon}}$ denotes the noise distribution of $\boldsymbol{\epsilon}$. Given $p(\mathbf{y} | \mathbf{x})$ and $p(\mathbf{x})$, we can solve the inverse

problem by sampling from the posterior distribution $p(\mathbf{x} | \mathbf{y})$.

Examples of linear inverse problems in medical imaging include image reconstruction for CT and MRI. In both cases, the signal \mathbf{x} is a medical image. The measurement \mathbf{y} in CT is a sinogram formed by X-ray projections of the image from various angular directions [Buz11], while the measurement \mathbf{y} in MRI consists of spatial frequencies in the Fourier space of the image (a.k.a. the k-space in the MRI community) [VB13].

8.2.2 Score-Based Generative Models

When solving inverse problems in medical imaging, we are given an observation \mathbf{y} , the measurement distribution $p(\mathbf{y} | \mathbf{x})$ and aim to sample from the posterior distribution $p(\mathbf{x} | \mathbf{y})$. The prior distribution $p(\mathbf{x})$ is typically unknown, but we can train generative models on a dataset $\{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \dots, \mathbf{x}^{(N)}\} \sim p(\mathbf{x})$ to estimate this prior distribution. Given an estimate of $p(\mathbf{x})$ and the measurement distribution $p(\mathbf{y} | \mathbf{x})$, the posterior distribution $p(\mathbf{x} | \mathbf{y})$ can be determined through Bayes' rule.

We propose to estimate the prior distribution of medical images using score-based generative models [SE19b; HJA20; Son+21b], whose iterative sampling procedure makes it especially easy for controllable generation conditioned on an observation \mathbf{y} . Specifically, we adopt the formulation of score-based generative models in Chapter 7, where we leverage a Markovian diffusion process to progressively perturb data to noise, and then smoothly convert noise to samples of the data distribution by estimating and simulating its time reversal. We provide an illustration of this generative modeling framework in Fig. 8.1.

Perturbation process Suppose the dataset is sampled from an unknown data distribution $p(\mathbf{x})$. We perturb data points with a stochastic process over a time horizon $[0, 1]$, governed by a linear stochastic differential equation (SDE) of the following form

$$d\mathbf{x}_t = f(t)\mathbf{x}_t dt + g(t) d\mathbf{w}_t, \quad t \in [0, 1], \quad (8.1)$$

where $f : [0, 1] \rightarrow \mathbb{R}$, $g : [0, 1] \rightarrow \mathbb{R}$, $\{\mathbf{w}_t \in \mathbb{R}^n\}_{t \in [0, 1]}$ denotes a standard Wiener process (a.k.a., Brownian motion), and $\{\mathbf{x}_t \in \mathbb{R}^n\}_{t \in [0, 1]}$ symbolizes the trajectory of random variables in the stochastic process. We further denote the marginal probability distribution of \mathbf{x}_t as $p_t(\mathbf{x})$, and the transition distribution from \mathbf{x}_0 to \mathbf{x}_t as $p_{0t}(\mathbf{x}_t | \mathbf{x}_0)$. By definition, we clearly have $p_0(\mathbf{x}) \equiv p(\mathbf{x})$. Moreover, the functions $f(t)$ and $g(t)$ are specifically chosen

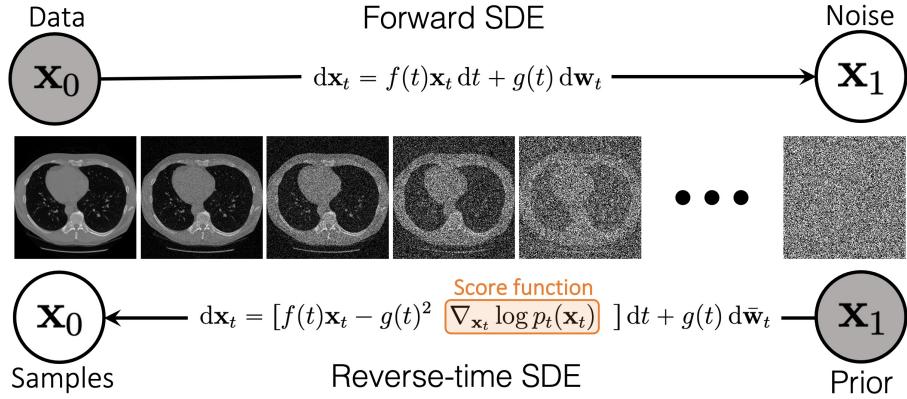


Figure 8.1: We can smoothly perturb images to noise by following the trajectory of an SDE. By estimating the score function $\nabla_{\mathbf{x}} \log p_t(\mathbf{x})$ with neural networks (called score models), it is possible to approximate the reverse SDE and then solve it to generate image samples from noise.

such that for any initial distribution $p_0(\mathbf{x})$, the distribution at the end of the perturbation process, $p_1(\mathbf{x})$, is close to a pre-defined noise distribution $\pi(\mathbf{x})$. In addition, the transition density $p_{0t}(\mathbf{x}_t | \mathbf{x}_0)$ is always a conditional linear Gaussian distribution, taking the form $p_{0t}(\mathbf{x}_t | \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t | \alpha(t)\mathbf{x}_0, \beta^2(t)\mathbf{I})$ where $\alpha : [0, 1] \rightarrow \mathbb{R}$ and $\beta : [0, 1] \rightarrow \mathbb{R}$ can be derived analytically from $f(t)$ and $g(t)$ [SS19]. Examples of such SDEs include Variance Exploding (VE), Variance Preserving (VP), and subVP SDEs proposed in Chapter 7. We found VE SDEs performed the best in our experiments.

Reverse process By reversing the perturbation process in Eq. (8.1), we can start from a noise sample $\mathbf{x}_1 \sim p_1(\mathbf{x})$ and gradually remove the noise therein to obtain a data sample $\mathbf{x}_0 \sim p_0(\mathbf{x}) \equiv p(\mathbf{x})$. Crucially, the time reversal of Eq. (8.1) is given by the following reverse-time SDE from Chapter 7

$$d\mathbf{x}_t = [f(t)\mathbf{x}_t - g(t)^2 \nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t)] dt + g(t) d\bar{\mathbf{w}}_t, \quad t \in [0, 1], \quad (8.2)$$

where $\{\bar{\mathbf{w}}_t\}_{t \in [0,1]}$ denotes a standard Wiener process in the reverse-time direction, and dt represents an infinitesimal negative time step, since the above SDE must be solved backwards from $t = 1$ to $t = 0$. The quantity $\nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t)$ is known as the *score function* of $p_t(\mathbf{x}_t)$. By the definition of time reversal, the trajectory of the reverse stochastic process given by Eq. (8.2) is $\{\mathbf{x}_t\}_{t \in [0,1]}$, same as the one from the forward SDE in Eq. (8.1).

Algorithm 8.1 Unconditional sampling

Require: N

- 1: $\hat{\mathbf{x}}_1 \sim \pi(\mathbf{x})$, $\Delta t \leftarrow \frac{1}{N}$
- 2: **for** $i = N - 1$ **to** 0 **do**
- 3: $t \leftarrow \frac{i+1}{N}$
- 4: $\hat{\mathbf{x}}_{t-\Delta t} \leftarrow \hat{\mathbf{x}}_t - f(t)\hat{\mathbf{x}}_t\Delta t$
- 5: $\hat{\mathbf{x}}_{t-\Delta t} \leftarrow \hat{\mathbf{x}}_{t-\Delta t} + g(t)^2 s_{\theta^*}(\hat{\mathbf{x}}_t, t)\Delta t$
- 6: $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
- 7: $\hat{\mathbf{x}}_{t-\Delta t} \leftarrow \hat{\mathbf{x}}_{t-\Delta t} + g(t)\sqrt{\Delta t} \mathbf{z}$
- 8: **end for**
- 9: **return** $\hat{\mathbf{x}}_0$

Algorithm 8.2 Inverse problem solving

Require: N, \mathbf{y}, λ

- 1: $\hat{\mathbf{x}}_1 \sim \pi(\mathbf{x})$, $\Delta t \leftarrow \frac{1}{N}$
- 2: **for** $i = N - 1$ **to** 0 **do**
- 3: $t \leftarrow \frac{i+1}{N}$
- 4: $\hat{\mathbf{y}}_t \sim p_{0t}(\mathbf{y}_t | \mathbf{y})$
- 5: $\hat{\mathbf{x}}_t \leftarrow \mathbf{T}^{-1}[\lambda \Lambda \mathcal{P}^{-1}(\Lambda) \hat{\mathbf{y}}_t + (1 - \lambda) \Lambda \mathbf{T} \hat{\mathbf{x}}_t + (\mathbf{I} - \Lambda) \mathbf{T} \hat{\mathbf{x}}_t]$
- 6: $\hat{\mathbf{x}}_{t-\Delta t} \leftarrow \hat{\mathbf{x}}_t - f(t)\hat{\mathbf{x}}_t\Delta t$
- 7: $\hat{\mathbf{x}}_{t-\Delta t} \leftarrow \hat{\mathbf{x}}_{t-\Delta t} + g(t)^2 s_{\theta^*}(\hat{\mathbf{x}}_t, t)\Delta t$
- 8: $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
- 9: $\hat{\mathbf{x}}_{t-\Delta t} \leftarrow \hat{\mathbf{x}}_{t-\Delta t} + g(t)\sqrt{\Delta t} \mathbf{z}$
- 10: **end for**
- 11: **return** $\hat{\mathbf{x}}_0$

Sampling Given an initial sample from $p_1(\mathbf{x})$, as well as scores at each intermediate time step, $\nabla_{\mathbf{x}} \log p_t(\mathbf{x})$, we can simulate the reverse-time SDE in Eq. (8.2) to obtain samples from the data distribution $p_0(\mathbf{x}) \equiv p(\mathbf{x})$. In practice, the initial sample is approximately drawn from $\pi(\mathbf{x})$ since $\pi(\mathbf{x}) \approx p_1(\mathbf{x})$, and the scores are estimated by training a neural network $s_{\theta}(\mathbf{x}, t)$ (named the *score model*) on a dataset $\{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \dots, \mathbf{x}^{(N)}\} \sim p(\mathbf{x})$ with denoising score matching [Vin11; Son+21b], *i.e.*, solving the following objective

$$\theta^* = \arg \min_{\theta} \frac{1}{N} \sum_{i=1}^N \mathbb{E}_{t \sim \mathcal{U}[0,1]} \mathbb{E}_{\mathbf{x}_t^{(i)} \sim p_{0t}(\mathbf{x}_t^{(i)} | \mathbf{x}^{(i)})} \left[\left\| s_{\theta}(\mathbf{x}_t^{(i)}, t) - \nabla_{\mathbf{x}_t^{(i)}} \log p_{0t}(\mathbf{x}_t^{(i)} | \mathbf{x}^{(i)}) \right\|_2^2 \right],$$

where $\mathcal{U}[0, 1]$ denotes a uniform distribution over $[0, 1]$. The theory of denoising score matching ensures that $s_{\theta^*}(\mathbf{x}, t) \approx \nabla_{\mathbf{x}} \log p_t(\mathbf{x})$. After training this score model, we plug it into Eq. (8.2) and solve the resulting reverse-time SDE

$$d\mathbf{x}_t = [f(t)\mathbf{x}_t - g(t)^2 s_{\theta^*}(\mathbf{x}_t, t)] dt + g(t) d\bar{\mathbf{w}}_t, \quad t \in [0, 1], \quad (8.3)$$

for sample generation. One sampling method is to use the Euler-Maruyama discretization for solving Eq. (8.3), as given in Algorithm 8.1. Other sampling methods include annealed Langevin dynamics [SE19b] (Chapter 5), probability flow ODE solvers, and Predictor-Corrector samplers (*cf.*, Chapter 7).

8.3 Solving Inverse Problems with Score-Based Generative Models

With score-based generative modeling, we can train a score model $s_{\theta^*}(\mathbf{x}, t)$ to generate unconditional samples from the prior distribution of medical images $p(\mathbf{x})$. To solve inverse problems however, we will need to sample from the posterior $p(\mathbf{x} | \mathbf{y})$. This can be accomplished by conditioning the original stochastic process $\{\mathbf{x}_t\}_{t \in [0,1]}$ on an observation \mathbf{y} , yielding a *conditional* stochastic process $\{\mathbf{x}_t | \mathbf{y}\}_{t \in [0,1]}$. We denote the marginal distribution at t as $p_t(\mathbf{x}_t | \mathbf{y})$, and our goal is to sample from $p_0(\mathbf{x}_0 | \mathbf{y})$, the same distribution as $p(\mathbf{x} | \mathbf{y})$ by definition. Much like generating unconditional samples by solving the reverse-time SDE in Eq. (8.2), we can reverse the conditional stochastic process $\{\mathbf{x}_t | \mathbf{y}\}_{t \in [0,1]}$ to sample from the posterior distribution $p_0(\mathbf{x}_0 | \mathbf{y})$ by solving the following *conditional* reverse-time SDE from Chapter 7:

$$d\mathbf{x}_t = [f(t)\mathbf{x}_t - g(t)^2 \nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t | \mathbf{y})] dt + g(t) d\bar{\mathbf{w}}_t, \quad t \in [0, 1]. \quad (8.4)$$

The conditional score function $\nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t | \mathbf{y})$ is a critical part of Eq. (8.4), yet it is non-trivial to compute. One solution is to estimate the score function by training a new score model $s_{\theta^*}(\mathbf{x}_t, \mathbf{y}, t)$ that explicitly depends on \mathbf{y} [Son+21b; DN21], such that $s_{\theta^*}(\mathbf{x}_t, \mathbf{y}, t) \approx \nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t | \mathbf{y})$. However, this requires paired data $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^N$ for training and has the same drawbacks as supervised learning techniques. We do not consider this approach in this chapter.

An unsupervised alternative is to approximate the conditional score function with an unconditionally-trained score model $s_{\theta^*}(\mathbf{x}_t, t) \approx \nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t)$ and the measurement distribution $p(\mathbf{y} | \mathbf{x})$. Many existing works [Son+21b; KVE21; KS20; Jal+21] have implemented this idea in different ways. However, the methods in [KVE21] and [KS20] both require computing the singular value decomposition (SVD) of $\mathbf{A} \in \mathbb{R}^{m \times n}$, which can be difficult for many measurement processes in medical imaging. The method proposed in [Jal+21] is only designed for a specific sampling method called annealed Langevin dynamics (ALD in Chapter 5), which proves to be inferior to more advanced sampling algorithms such as Predictor-Corrector methods in Chapter 7.

In what follows, we propose a new conditional sampling approach for inverse problem solving with score-based generative models. Our method is computationally efficient for

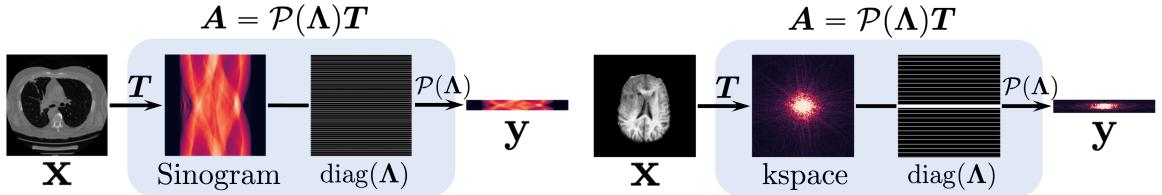


Figure 8.2: Linear measurement processes for sparse-view CT (*left*) and undersampled MRI (*right*).

medical image reconstruction, and is applicable to a large family of iterative sampling methods for score-based generative models. At a high level, we first train an unconditional score model $s_{\theta^*}(\mathbf{x}, t)$ on medical images without assuming any measurement process. Given an observation \mathbf{y} at test time, we form a stochastic process $\{\mathbf{y}_t\}_{t \in [0,1]}$ by adding appropriate noise to \mathbf{y} . We then discretize the reverse-time SDE in Eq. (8.3) with existing unconditional samplers for $s_{\theta^*}(\mathbf{x}, t)$, while incorporating the conditional information from \mathbf{y} with a proximal optimization step to generate intermediate samples that are consistent with $\{\mathbf{y}_t\}_{t \in [0,1]}$.

8.3.1 A Convenient Form of the Linear Measurement Process

Many different measurement processes in medical imaging share same components of computation. For example, sparse-view CT reconstruction and metal artifact removal for CT both involve computing the same Radon transform. Similarly, MRI measurement processes require computing the same spatial Fourier transform regardless of different downsampling ratios. To rigorously characterize this structure of measurement processes, we propose a special formulation of \mathbf{A} that is efficient to obtain in medical imaging applications. Without loss of generality, we assume that the linear operator \mathbf{A} has full rank, *i.e.*, $\text{rank}(\mathbf{A}) = \min(n, m) = m$. The result below gives the alternative formulation of \mathbf{A} :

Proposition 8.1. *If $\text{rank}(\mathbf{A}) = m$, then there exist an invertible matrix $\mathbf{T} \in \mathbb{R}^{n \times n}$, and a diagonal matrix $\Lambda \in \{0, 1\}^{n \times n}$ with $\text{tr}(\Lambda) = m$, such that $\mathbf{A} = \mathcal{P}(\Lambda)\mathbf{T}$. Here $\mathcal{P}(\Lambda) \in \{0, 1\}^{m \times n}$ is an operator that, when multiplied with any vector $\mathbf{a} \in \mathbb{R}^n$, reduces its dimensionality to m by removing each i -th element of \mathbf{a} for $i = 1, 2, \dots, n$ if $\Lambda_{ii} = 0$.*

We illustrate this decomposition for CT/MRI in Fig. 8.2. Many measurement processes in medical imaging share the same \mathbf{T} , even if they correspond to different \mathbf{A} . For example, \mathbf{T} corresponds to the Radon transform and Fourier transform in sparse-view CT

and undersampled MRI respectively, regardless of the number of measurements, *i.e.*, CT projections and k-space downsampling ratios. For both sparse-view CT reconstruction and metal artifact removal for CT images, the operator \mathbf{T} is the Radon transform (see Fig. B.54). Intuitively, $\text{diag}(\boldsymbol{\Lambda})$ can be viewed as a subsampling mask on the sinogram/k-space, and $\mathcal{P}(\boldsymbol{\Lambda})$ subsamples the sinogram/k-space into an observation \mathbf{y} with a smaller size according to this subsampling mask. In addition, we note that \mathbf{T}^{-1} can be efficiently implemented with the inverse Radon transform or the inverse Fourier transform in CT/MRI applications.

8.3.2 Incorporating Observations into the Sampling Process

In what follows, we show that the decomposition in Proposition 8.1 provides an efficient way to generate approximate samples from the conditional stochastic process $\{\mathbf{x}_t \mid \mathbf{y}\}_{t \in [0,1]}$ with an *unconditional* score model $s_{\theta^*}(\mathbf{x}, t)$. The basic idea is to “hijack” the unconditional sampling process of score-based generative models to incorporate an observed measurement \mathbf{y} .

As we have already discussed, it is difficult to directly solve $\{\mathbf{x}_t \mid \mathbf{y}\}_{t \in [0,1]}$ for sample generation. To bypass this difficulty, we first consider a related stochastic process that is much easier to sample from. Recall that $p_{0t}(\mathbf{x}_t \mid \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t \mid \alpha(t)\mathbf{x}_0, \beta^2(t)\mathbf{I})$ where $\alpha(t)$ and $\beta(t)$ can be derived from $f(t)$ and $g(t)$ [Son+21b] (Chapter 7). Given the unconditional stochastic process $\{\mathbf{x}_t\}_{t \in [0,1]}$, we define $\{\mathbf{y}_t\}_{t \in [0,1]}$, where $\mathbf{y}_t = \mathbf{A}\mathbf{x}_t + \alpha(t)\boldsymbol{\epsilon}$. Unlike $\{\mathbf{x}_t \mid \mathbf{y}\}_{t \in [0,1]}$, the conditional stochastic process $\{\mathbf{y}_t \mid \mathbf{y}\}_{t \in [0,1]}$ is fully tractable. First, we have $\mathbf{y}_0 = \mathbf{A}\mathbf{x}_0 + \alpha(0)\boldsymbol{\epsilon} = \mathbf{A}\mathbf{x}_0 + \boldsymbol{\epsilon} = \mathbf{y}$. Since $p_{0t}(\mathbf{x}_t \mid \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t \mid \alpha(t)\mathbf{x}_0, \beta^2(t)\mathbf{I})$, we have $\mathbf{x}_t = \alpha(t)\mathbf{x}_0 + \beta(t)\mathbf{z}$, where $\mathbf{z} \in \mathbb{R}^n \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. By definition, $\mathbf{y}_t = \mathbf{A}\mathbf{x}_t + \alpha(t)\boldsymbol{\epsilon}$, so we have $\mathbf{y}_t = \mathbf{A}(\alpha(t)\mathbf{x}_0 + \beta(t)\mathbf{z}) + \alpha(t)\boldsymbol{\epsilon} = \alpha(t)(\mathbf{y} - \boldsymbol{\epsilon}) + \beta(t)\mathbf{A}\mathbf{z} + \alpha(t)\boldsymbol{\epsilon} = \alpha(t)\mathbf{y} + \beta(t)\mathbf{A}\mathbf{z}$. Therefore, we can easily generate a sample $\hat{\mathbf{y}}_t \sim p_t(\mathbf{y}_t \mid \mathbf{y})$ by first drawing $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and then computing $\hat{\mathbf{y}}_t = \alpha(t)\mathbf{y} + \beta(t)\mathbf{A}\mathbf{z}$.

The key of our approach is to modify any existing iterative sampling algorithm designed for the unconditional stochastic process $\{\mathbf{x}_t\}_{t \in [0,1]}$ so that the samples are consistent with $\{\mathbf{y}_t \mid \mathbf{y}\}_{t \in [0,1]}$. In general, an iterative sampling process of score-based generative models selects a sequence of time steps $\{0 = t_0 < t_1 < \dots < t_N = 1\}$ and iterates according to

$$\hat{\mathbf{x}}_{t_{i-1}} = \mathbf{h}(\hat{\mathbf{x}}_{t_i}, \mathbf{z}_i, s_{\theta^*}(\hat{\mathbf{x}}_{t_i}, t_i)), \quad i = N, N-1, \dots, 1, \quad (8.5)$$

where $\hat{\mathbf{x}}_{t_N} \sim \pi(\mathbf{x})$, $\mathbf{z}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, and θ^* denotes the parameters in an unconditional score

model $s_{\theta^*}(\mathbf{x}, t)$. Here the iteration function \mathbf{h} takes a noisy sample $\hat{\mathbf{x}}_{t_i}$ and reduces the noise therein to generate $\hat{\mathbf{x}}_{t_{i-1}}$, using the unconditional score model $s_{\theta^*}(\mathbf{x}, t)$. For example, for the Euler-Maruyama sampler detailed in Algorithm 8.1, this iteration function is given by

$$\mathbf{h}(\hat{\mathbf{x}}_{t_i}, \mathbf{z}_i, s_{\theta^*}(\hat{\mathbf{x}}_{t_i}, t_i)) = \hat{\mathbf{x}}_{t_i} - f(t_i)\hat{\mathbf{x}}_{t_i}/N + g(t_i)^2 s_{\theta^*}(\hat{\mathbf{x}}_{t_i}, t_i)/N + g(t_i)\mathbf{z}_i/\sqrt{N}.$$

Samples obtained by this procedure $\{\hat{\mathbf{x}}_{t_i}\}_{i=0}^N$ constitute an approximation of $\{\mathbf{x}_t\}_{t \in [0,1]}$, where the last sample $\hat{\mathbf{x}}_{t_0}$ can be viewed as an approximate sample from $p_0(\mathbf{x})$. Most existing sampling methods for score-based generative models are instances of this iterative sampling paradigm, including Algorithm 8.1, ALD from Chapter 5, probability flow ODEs and Predictor-Corrector samplers from Chapter 7.

To enforce the constraint implied by $\{\mathbf{y}_t \mid \mathbf{y}\}_{t \in [0,1]}$, we prepend an additional step to the iteration rule in Eq. (8.5), leading to

$$\hat{\mathbf{x}}'_{t_i} = \mathbf{k}(\hat{\mathbf{x}}_{t_i}, \hat{\mathbf{y}}_{t_i}, \lambda) \quad (8.6)$$

$$\hat{\mathbf{x}}_{t_{i-1}} = \mathbf{h}(\hat{\mathbf{x}}'_{t_i}, \mathbf{z}_i, s_{\theta^*}(\hat{\mathbf{x}}_{t_i}, t_i)), \quad i = N, N-1, \dots, 1, \quad (8.7)$$

where $\hat{\mathbf{x}}_{t_N} \sim \pi(\mathbf{x})$, $\hat{\mathbf{y}}_{t_i} \sim p_{t_i}(\mathbf{y}_{t_i} \mid \mathbf{y})$, and $0 \leq \lambda \leq 1$ is a hyper-parameter. We provide an illustration of this process in Fig. 8.3. The iteration function $\mathbf{k}(\cdot, \hat{\mathbf{y}}_{t_i}, \lambda) : \mathbb{R}^n \rightarrow \mathbb{R}^n$ promotes data consistency by solving a proximal optimization step [Nes03; BBV04; Ham+] that simultaneously minimizes the distance between $\hat{\mathbf{x}}'_{t_i}$ and $\hat{\mathbf{x}}_{t_i}$, and the distance between $\hat{\mathbf{x}}'_{t_i}$ and the hyperplane $\{\mathbf{x} \in \mathbb{R}^n \mid \mathbf{A}\mathbf{x} = \hat{\mathbf{y}}_{t_i}\}$, with a hyperparameter $0 \leq \lambda \leq 1$ balancing between the two:

$$\hat{\mathbf{x}}'_{t_i} = \arg \min_{\mathbf{z} \in \mathbb{R}^n} \{(1 - \lambda) \|\mathbf{z} - \hat{\mathbf{x}}_{t_i}\|_{\mathbf{T}}^2 + \min_{\mathbf{u} \in \mathbb{R}^n} \lambda \|\mathbf{z} - \mathbf{u}\|_{\mathbf{T}}^2\} \quad \text{s.t.} \quad \mathbf{A}\mathbf{u} = \hat{\mathbf{y}}_{t_i}. \quad (8.8)$$

Recall that $\mathbf{A} = \mathcal{P}(\boldsymbol{\Lambda})\mathbf{T}$ according to Proposition 8.1. In the equation above we choose the norm $\|\mathbf{a}\|_{\mathbf{T}}^2 := \|\mathbf{T}\mathbf{a}\|_2^2$ to simplify our theoretical analysis. The decomposition in Proposition 8.1 allows us to derive a closed-form solution to the optimization problem in Eq. (8.8), as given below:

Theorem 8.1. *The solution of Eq. (8.8) can be given by*

$$\hat{\mathbf{x}}'_{t_i} = \mathbf{T}^{-1}[\lambda \boldsymbol{\Lambda} \mathcal{P}^{-1}(\boldsymbol{\Lambda}) \hat{\mathbf{y}}_{t_i} + (1 - \lambda) \boldsymbol{\Lambda} \mathbf{T} \hat{\mathbf{x}}_{t_i} + (\mathbf{I} - \boldsymbol{\Lambda}) \mathbf{T} \hat{\mathbf{x}}_{t_i}], \quad (8.9)$$

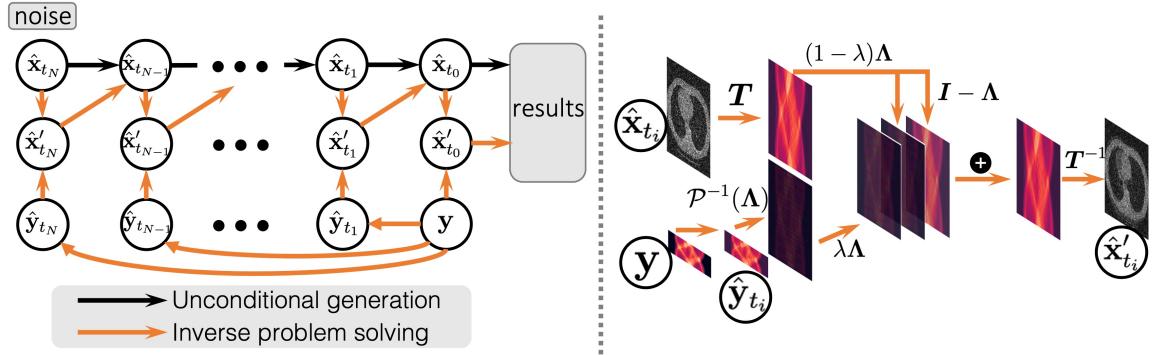


Figure 8.3: (Left) An overview of our method for solving inverse problems with score-based generative models. (Right) An illustration about how to combine $\hat{\mathbf{x}}_{t_i}$ and \mathbf{y} to form $\hat{\mathbf{x}}'_{t_i}$.

where $\mathcal{P}^{-1}(\Lambda) : \mathbb{R}^m \rightarrow \mathbb{R}^n$ denotes any right inverse of $\mathcal{P}(\Lambda)$.

See Fig. 8.3 for an illustration of the function $\hat{\mathbf{x}}'_{t_i} = \mathbf{k}(\hat{\mathbf{x}}_{t_i}, \hat{\mathbf{y}}_{t_i}, \lambda)$. The right inverse $\mathcal{P}^{-1}(\Lambda)$ increases the dimensionality of a vector $\mathbf{a} \in \mathbb{R}^m$ to n by putting its entries on every index i of an n -dimensional vector where $\Lambda_{ii} = 1$. Recall that in sparse-view CT or undersampled MRI, $\text{diag}(\Lambda)$ represents a subsampling mask, and $\mathcal{P}(\Lambda)$ subsamples the full sinogram/k-space to generate the observation \mathbf{y} . In this case, $\mathcal{P}^{-1}(\Lambda)$ pads the observation \mathbf{y} so that it has the same size as the full sinogram/k-space.

When $\lambda = 0$, $\hat{\mathbf{x}}'_{t_i} = \mathbf{k}(\hat{\mathbf{x}}_{t_i}, \hat{\mathbf{y}}_{t_i}, 0) = \hat{\mathbf{x}}_{t_i}$ completely ignores the constraint $\mathbf{A}\hat{\mathbf{x}}'_{t_i} = \hat{\mathbf{y}}_{t_i}$, in which case our sampling method in Eq. (8.7) performs unconditional generation. On the other hand, when $\lambda = 1$, $\hat{\mathbf{x}}'_{t_i} = \mathbf{k}(\hat{\mathbf{x}}_{t_i}, \hat{\mathbf{y}}_{t_i}, 1)$ satisfies $\mathbf{A}\hat{\mathbf{x}}'_{t_i} = \hat{\mathbf{y}}_{t_i}$ exactly. When the measurement is noisy, we choose $0 < \lambda < 1$ to allow slackness in the constraint $\mathbf{A}\hat{\mathbf{x}}'_{t_i} = \hat{\mathbf{y}}_{t_i}$. The value of λ is important for balancing between $\hat{\mathbf{x}}'_{t_i} \approx \hat{\mathbf{x}}_{t_i}$ and $\mathbf{A}\hat{\mathbf{x}}'_{t_i} \approx \hat{\mathbf{y}}_{t_i}$. In practice, we use Bayesian optimization to tune this λ automatically on a validation dataset. When the measurement process contains no noise, we replace $\hat{\mathbf{x}}_{t_0}$ with $\mathbf{k}(\hat{\mathbf{x}}_{t_0}, \mathbf{y}, 1)$ at the last sampling step to guarantee $\mathbf{A}\hat{\mathbf{x}}_{t_0} = \mathbf{y}$.

In summary, our method given in Eq. (8.7) introduces minimal modifications to an existing iterative sampling method of score-based generative models. For example, we can convert the sampler in Algorithm 8.1 to an inverse problem solver in Algorithm 8.2 by adding/modifying just three lines of pseudo-code. Unlike the concurrent work [Jal+21], our method is not limited to annealed Langevin dynamics (ALD). As demonstrated in our experiments, we outperform [Jal+21] even with the same ALD sampler, and can widen the performance gap further by using more advanced approaches like the Predictor-Corrector sampler in

Chapter 7. Unlike [KS20; KVE21], we rely on the efficient alternative representation of \mathbf{A} given in Section 8.3.1, and do not require expensive SVD computation.

8.4 Experiments

We aim to answer the following questions in this section: (1) Can we directly compete with best-in-class supervised learning techniques for the same measurement process used in their training, even though our approach is fully unsupervised? (2) Can our method generalize better to new measurement processes? (3) How do we fare against other unsupervised approaches? To study these questions, we experiment on several tasks in medical imaging, including sparse-view CT reconstruction, metal artifact removal (MAR) for CT, and undersampled MRI reconstruction. More experimental details are provided in Appendix B.6.

Datasets We consider two datasets for CT experiments. The first is the Lung Image Database Consortium (LIDC) image collection dataset [Arm+11; Cla+13] where we slice the original 3D CT volumes to obtain 130304 2D images of resolution 320×320 for training. The second is the Low Dose CT (LDCT) Image and Projection dataset [Moe+21] that contains CT scans of multiple anatomic sites, including head, chest, and abdomen, from which we generate 47006 2D image slices of resolution 512×512 for training. We simulate CT measurements (sinograms) with a parallel-beam geometry using projection angles equally distributed across 180 degrees. For MAR experiments, we follow [Yu+20] to synthesize metal artifacts. For undersampled MRI experiments, we use the Brain Tumor Segmentation (BraTS) 2021 dataset [Men+14; Bak+17], where we slice 3D MRI volumes to get 297270 images of resolution 240×240 as the training dataset. We simulate MRI measurements with Fast Fourier Transform using a single-coil setup, and follow [Zbo+18; Kno+20] to undersample the k-space with an equispaced Cartesian mask. The performance is measured on 1000 test images with peak signal-to-noise ratio (PSNR) and structural similarity (SSIM).

Standard techniques in medical imaging We include two standard learning-free techniques as baselines for sparse-view CT reconstruction. The first is filtered back projection on sparse-view sinograms, which is denoted by “FBP”. The second is an iterative reconstruction method with total variation regularization called FISTA-TV [BT09]. For MAR experiments, we include another learning-free baseline called linear interpolation [LI, KHE87].

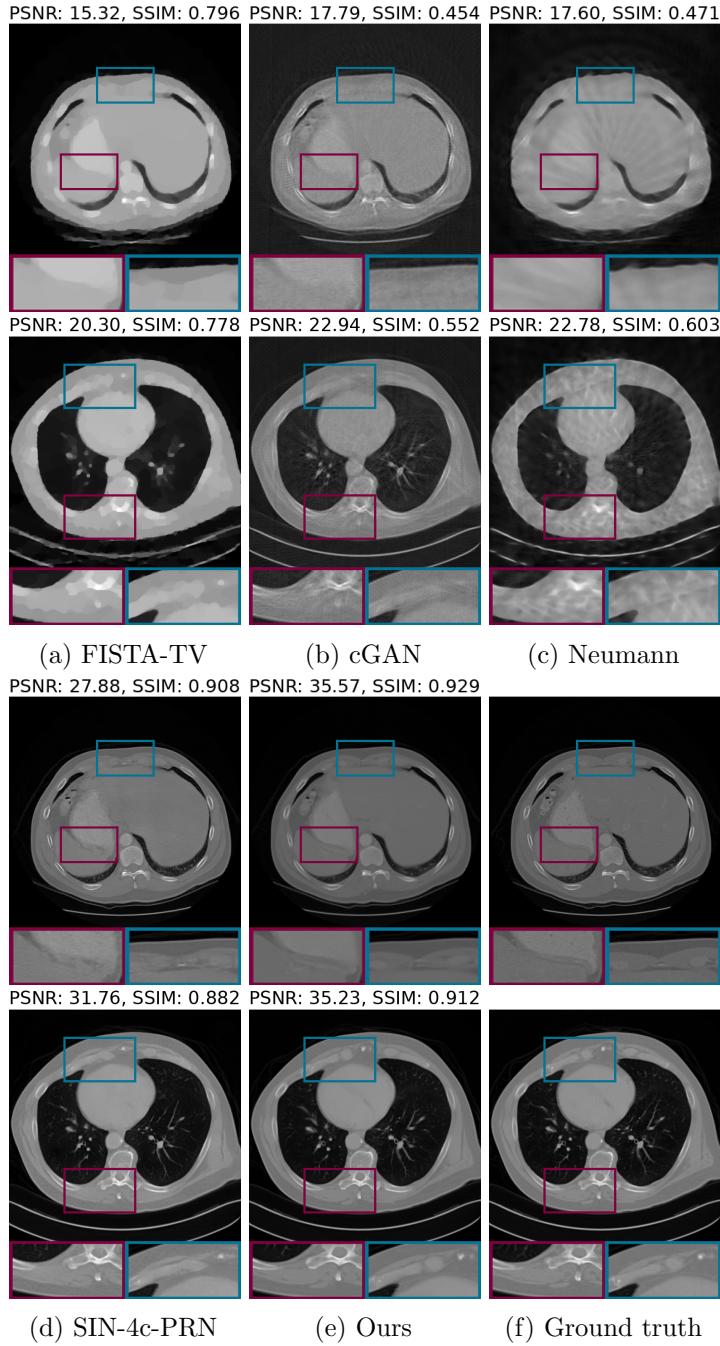


Figure 8.4: Examples of sparse-view CT reconstruction results on LIDC 320×320 (*Top row*) and LDCT 512×512 (*Bottom row*), all with 23 projections. You may zoom in to view more details.

Table 8.1: Results for undersampled MRI reconstruction on BraTS. First two methods are supervised learning techniques trained with $8\times$ acceleration. The others are unsupervised techniques.

Method	24× Acceleration		8× Acceleration		4× Acceleration	
	PSNR↑	SSIM↑	PSNR↑	SSIM↑	PSNR↑	SSIM↑
Cascade DenseNet	23.39 \pm 2.17	0.765 \pm 0.042	28.35 \pm 2.30	0.845 \pm 0.038	30.97 \pm 2.33	0.902 \pm 0.028
DuDoRNet	18.46 \pm 3.05	0.662 \pm 0.093	37.88\pm3.03	0.985\pm0.007	30.53 \pm 4.13	0.891 \pm 0.071
Score SDE	27.83 \pm 2.73	0.849 \pm 0.038	35.04 \pm 2.11	0.943 \pm 0.016	37.55 \pm 2.08	0.960 \pm 0.013
Langevin	28.80 \pm 3.21	0.873 \pm 0.039	36.44 \pm 2.28	0.952 \pm 0.016	38.76 \pm 2.32	0.966 \pm 0.012
Ours	29.42\pm3.03	0.880\pm0.035	37.63 \pm 2.70	0.958 \pm 0.015	39.91\pm2.67	0.965\pm0.013

Table 8.2: Results for sparse-view CT reconstruction on LIDC and LDCT. FISTA-TV is a standard iterative reconstruction method that does not need training. cGAN, Neumann, and SIN-4c-PRN are supervised learning techniques trained with 23 projection angles.

Method	Projections	LIDC 320 × 320		LDCT 512 × 512	
		PSNR↑	SSIM↑	PSNR↑	SSIM↑
FBP	23	10.18 \pm 1.38	0.230 \pm 0.072	10.11 \pm 1.19	0.302 \pm 0.078
FISTA-TV	23	20.08 \pm 4.89	0.799 \pm 0.061	21.88 \pm 4.42	0.850 \pm 0.067
cGAN	23	19.83 \pm 3.07	0.479 \pm 0.103	19.90 \pm 2.52	0.545 \pm 0.065
Neumann	23	17.18 \pm 3.79	0.454 \pm 0.128	18.83 \pm 3.29	0.525 \pm 0.073
SIN-4c-PRN	23	30.48 \pm 3.99	0.895 \pm 0.047	34.82 \pm 3.55	0.877 \pm 0.116
Ours	10	29.52 \pm 2.63	0.823 \pm 0.061	28.96 \pm 4.41	0.849 \pm 0.086
	20	34.40 \pm 2.66	0.895 \pm 0.048	36.80 \pm 4.50	0.936 \pm 0.058
	23	35.24\pm2.71	0.905\pm0.046	37.41\pm4.62	0.941\pm0.057

Table 8.3: MAR results on LIDC.

Method	PSNR↑	SSIM↑
LI	26.30 ± 2.62	0.910 ± 0.028
cGANMAR	27.27 ± 1.96	0.927 ± 0.060
SNMAR	27.28 ± 1.43	0.937 ± 0.048
Ours	32.16 ± 2.32	0.939 ± 0.022

Supervised learning baselines For sparse-view CT on both LIDC and LDCT, we include cGAN [GK18], Neumann [GOW19], and SIN-4c-PRN [Wei+20] as supervised learning baselines. We follow the settings in [Wei+20] and train all methods with 23 projection angles. For MAR, we use cGANMAR [Wan+18] and SNMAR [Yu+20] as the baselines. For undersampled MRI on BraTS, we compare against Cascade DenseNet [ZFZ19] and DuDoRNet [ZZ20], which are both trained with a $8\times$ acceleration factor by measuring only 1/8 of the full k-space.

Unsupervised learning baselines For unsupervised techniques, so far only score-based generative models have witnessed success on clinic data. We compare with several existing methods that apply score-based generative models to inverse problem solving. Specifically, we consider the ‘‘Langevin’’ approach proposed in [Jal+21], and the ‘‘Score SDE’’ method in Chapter 7, where the former is limited to annealed Langevin dynamics (ALD) sampling, and the latter was based on a crude approximation to the conditional score function $\nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t | \mathbf{y})$ in Eq. (8.4), and was proposed as a theoretical possibility in Appendix B.5.9 without experiments. We only focus on undersampled MRI for these baselines, since it is the only medical imaging problem ever tackled with score-based generative models before our work. All methods share the same score models and only differ in terms of inference. We make sure all sampling algorithms have comparable number of iteration steps (N in Eqs. (8.5) and (8.7)).

Competing with supervised learning approaches Thanks to the outstanding sample quality of score-based generative models, we can achieve comparable or better performance than best-in-class supervised learning methods even for the same measurement process used in their training. As shown in Table 8.2, we outperform the top supervised learning technique SIN-4c-PRN on sparse-view CT reconstruction by a significant margin, on both the LIDC

and LDCT datasets. Our results with 20 measurements are even better than supervised learning counterparts with 23 measurements. In Fig. 8.4, we provide a visual comparison of the reconstruction quality for various methods, where it is clear to see that our method can recover more details faithfully. From results in Table 8.3, we also outperform the top supervised learning method SNMAR on metal artifact removal. As shown in Fig. B.53, our method generates images with less artifacts and preserves the structure better. For undersampled MRI reconstruction results given in Tables 8.1 and 8.3, our method is ranked the 2nd for the case of $8\times$ acceleration, with comparable performance to the top supervised method DuDoRNet.

Generalizing to different number of measurements Since our approach is fully unsupervised, we can naturally apply the same score model to different measurement processes. We first consider changing the number of measurements at the test time, *e.g.*, using different number of projection angles (*resp.* different acceleration factors) for sparse-view CT (*resp.* undersampled MRI) reconstruction. As shown in Table 8.1 and Fig. 8.5 (*Left*), we achieve the best performance on undersampled MRI for both $24\times$ and $4\times$ acceleration factors, whereas DuDoRNet fails to generalize when the acceleration factor changes. The other supervised learning approach Cascade DenseNet demonstrates limited adaptability by building a model architecture inspired by the physical measurement process of MRI, but fails to yield top-level performance. For sparse-view CT reconstruction, all supervised learning methods struggle to generalize to different projection angles, as shown in Fig. 8.5 (*Center*).

Generalizing to different measurement processes in CT We can perform both sparse-view CT reconstruction and metal artifact removal (MAR) with a single score model trained on CT images. These two tasks are inverse problems in CT imaging with different measurement processes \mathbf{A} , but they share the same \mathbf{T} in the decomposition of Proposition 8.1. We provide a visualization of the measurement process corresponding to MAR in Fig. B.54. As shown in Table 8.3, we can outperform supervised learning techniques specifically designed and trained for MAR, while using the same score model used in sparse-view CT reconstruction on LIDC.

Comparing against existing score-based methods We compare our method against Langevin [Jal+21] and Score SDE [Son+21b] (Chapter 7) for undersampled MRI reconstruction on BraTS. Two variants of our approach are considered, which respectively use

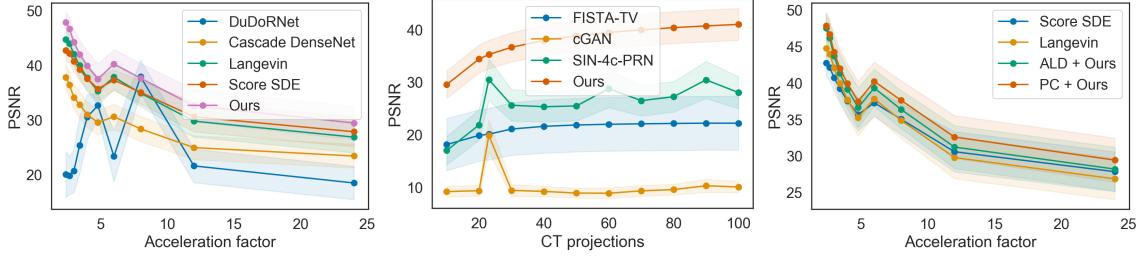


Figure 8.5: Performance vs. numbers of measurements. Shaded areas represent standard deviation. (*Left*) MRI on BraTS. (*Center*) CT on LIDC. (*Right*) Comparing score-based generative models for undersampled MRI reconstruction on BraTS.

annealed Langevin dynamics (ALD) and the Predictor-Corrector (PC) sampler for score-based generative models as the backend. We denote the former by “ALD + Ours”, and the latter by “PC + Ours” (our default method for all other experiments). Recall that Langevin uses ALD as the sampler, same as “ALD + Ours”. All results are provided in Fig. 8.5 (*Right*). We observe that “ALD + Ours” uniformly outperform Langevin and Score SDE across all numbers of measurements in the experiment. Moreover, “PC + Ours” can further improve “ALD + Ours”, demonstrating the power of switching to more advanced sampling methods of score-based generative models in our proposed approach.

8.5 Conclusion

We propose a new method to solve linear inverse problems with score-based generative models. Our method is fully unsupervised, requires no paired data for training, can flexibly adapt to different measurement processes at test time, and only requires minimal modifications to a large number of existing sampling methods of score-based generative models. Empirical results demonstrate that our method can match or outperform existing supervised learning counterparts on image reconstruction for sparse-view CT and undersampled MRI, and has better generalization to new measurement processes, such as using a different number of projections or downsampling ratios in CT/MRI, and tackling both sparse-view CT reconstruction and metal artifact removal with a single model.

Part III

Evaluating Probability Values with Score Models

Chapter 9

Computing, Bounding and Optimizing Likelihoods

Probability values, or likelihoods, are very important quantities in generative modeling. They enable broad applications ranging from outlier detection, data compression, model comparison, to generative classification. In Chapter 7, we derived an ordinary differential equation, called the probability flow ODE, to sample from score models in a deterministic way. In this chapter, we delve deeper into this ODE formulation to reveal its significance in likelihood evaluation and maximum likelihood training of score models.

Specifically, we show that the log-likelihood of continuous-time score-based generative models (which we call score-based generative models in this chapter) can be accurately computed by solving the probability flow ODE with numerical methods. Yet, this procedure requires hundreds of iterations, making maximum likelihood training of score models prohibitively slow. To address this difficulty, we show that the weighted combination of score matching losses, *i.e.*, the objective function that enables efficient training of score-based generative models, actually upper bounds the negative log-likelihood with a special weighting function. By providing an efficient bound on the log-likelihood, this weighted score matching objective enables practical approximate maximum likelihood training of score-based generative models. Empirically, we observe that maximum likelihood training consistently improves the likelihood of score-based generative models across multiple datasets, stochastic processes, and model architectures. Our best models achieve negative log-likelihoods of 2.83 and 3.76 bits/dim on CIFAR-10 and ImageNet 32×32 without any data augmentation, on par with state-of-the-art autoregressive models on these tasks.

This chapter is primarily based on our previous publications [Son+21b] and [Son+21a].

9.1 Introduction

Score-based generative models [SE19b; SE20; Son+21b] (Chapters 5 to 7) and diffusion probabilistic models [Soh+15; HJA20] have recently achieved state-of-the-art sample quality in a number of tasks, including image generation [Son+21b; DN21], audio synthesis [Che+21b; Kon+20; Pop+21], and shape generation [Cai+20b]. Both families of models perturb data with a sequence of noise distributions, and generate samples by learning to reverse this path from noise to data. Through stochastic calculus, these approaches can be unified into a single framework [Son+21b] (Chapter 7) based on continuous-time diffusion, which we refer to as score-based diffusion models in this chapter.

The framework of score-based diffusion models involves gradually diffusing the data distribution towards a given noise distribution using a stochastic differential equation (SDE), and learning the time reversal of this SDE for sample generation. Crucially, the reverse-time SDE has a closed-form expression which depends solely on the time-dependent gradient field (a.k.a., score) of the perturbed data distribution. This gradient field can be efficiently estimated by training a neural network (called a score-based model as in Chapters 5 and 6) with a weighted combination of score matching losses [HD05; Vin11; Son+19] as the objective. A special property of score-based diffusion models is the probability flow ODE, a deterministic dynamical system that produces samples from the same distribution as obtained by the reverse-time SDE.

In this chapter, we make an important observation that the probability flow ODE is essentially a continuous normalizing flow (CNF, [Che+18a; Gra+18]), thus allowing tractable likelihood computation with numerical ODE solvers. Compared to vanilla CNFs, score-based diffusion models are much more efficient to train. This is because the maximum likelihood objective for training CNFs requires running an expensive ODE solver for every optimization step, while the weighted combination of score matching losses for training score-based models does not. However, unlike maximum likelihood training, minimizing a combination of score matching losses does not necessarily lead to better likelihood values. Since better likelihoods are useful for applications including compression [Hoo+19; HLA19; TBB19], semi-supervised learning [Dai+17], adversarial purification [Son+18a], and comparing against likelihood-based generative models, we seek a training objective for score-based diffusion models that

is as efficient as score matching but also promotes higher likelihoods.

We show that such an objective can be readily obtained through slight modification of the weighted combination of score matching losses. Our theory reveals that with a specific choice of weighting, which we term the *likelihood weighting*, the combination of score matching losses actually upper bounds the negative log-likelihood. We further prove that this upper bound becomes tight when our score-based model corresponds to the true time-dependent gradient field of a certain reverse-time SDE. Using likelihood weighting increases the variance of our objective, which we counteract by introducing a variance reduction technique based on importance sampling. Our bound is analogous to the classic evidence lower bound used for training latent-variable models in the variational autoencoding framework [KW14; RMW14b], and can be viewed as a continuous-time generalization of [Soh+15].

With our likelihood weighting, we can minimize the weighted combination of score matching losses for approximate maximum likelihood training of score-based diffusion models. Compared to weightings in Chapter 7, we consistently improve likelihood values across multiple datasets, model architectures, and SDEs, with only slight degradation of Fréchet Inception distances [Heu+17]. Moreover, our upper bound on negative log-likelihood allows training with variational dequantization [Ho+19], with which we reach negative log-likelihood of **2.83** bits/dim on CIFAR-10 [KNH14] and **3.76** bits/dim on ImageNet 32×32 [VKK16] with no data augmentation. Our models present the first instances of normalizing flows which achieve comparable likelihood to cutting-edge autoregressive models.

9.2 Score-Based Diffusion Models

Score-based diffusion models are deep generative models that smoothly transform data to noise using a diffusion process, and synthesize samples by learning and simulating the time-reversal of this diffusion. The overall approach is illustrated in Fig. 9.1.

9.2.1 Diffusing Data to Noise with an SDE

Let $p(\mathbf{x})$ denote the unknown distribution of a dataset consisting of D -dimensional i.i.d. samples. Score-based diffusion models [Son+21b] (*cf.*, Chapter 7) employ a stochastic differential equation (SDE) to diffuse $p(\mathbf{x})$ towards a noise distribution. The SDEs are of

the form

$$d\mathbf{x} = \mathbf{f}(\mathbf{x}, t) dt + g(t) d\mathbf{w}, \quad (9.1)$$

where $\mathbf{f}(\cdot, t) : \mathbb{R}^D \rightarrow \mathbb{R}^D$ is the drift coefficient, $g(t) \in \mathbb{R}$ is the diffusion coefficient, and $\mathbf{w} \in \mathbb{R}^D$ denotes a standard Wiener process (a.k.a., Brownian motion). Intuitively, we can interpret $d\mathbf{w}$ as infinitesimal Gaussian noise. The solution of an SDE is a diffusion process $\{\mathbf{x}(t)\}_{t \in [0, T]}$, where $[0, T]$ is a fixed time horizon. We let $p_t(\mathbf{x})$ denote the marginal distribution of $\mathbf{x}(t)$, and $p_{0t}(\mathbf{x}' | \mathbf{x})$ denote the transition distribution from $\mathbf{x}(0)$ to $\mathbf{x}(t)$. Note that by definition we always have $p_0 = p$ when using an SDE to perturb the data distribution.

The role of the SDE is to smooth the data distribution by adding noise, gradually removing structure until little of the original signal remains. In the framework of score-based diffusion models, we choose $\mathbf{f}(\mathbf{x}, t)$, $g(t)$, and T such that the diffusion process $\{\mathbf{x}(t)\}_{t \in [0, T]}$ approaches some analytically tractable prior distribution $\pi(\mathbf{x})$ at $t = T$, meaning $p_T(\mathbf{x}) \approx \pi(\mathbf{x})$. Three families of SDEs suitable for this task are outlined in Chapter 7, namely Variance Exploding (VE) SDEs, Variance Preserving (VP) SDEs, and subVP SDEs.

9.2.2 Generating Samples with the Reverse SDE

Sample generation in score-based diffusion models relies on time-reversal of the diffusion process. For well-behaved drift and diffusion coefficients, the forward diffusion described in Eq. (9.1) has an associated reverse-time diffusion process [And82; HP86] given by the following SDE

$$d\mathbf{x} = [\mathbf{f}(\mathbf{x}, t) - g(t)^2 \nabla_{\mathbf{x}} \log p_t(\mathbf{x})] dt + g(t) d\bar{\mathbf{w}}, \quad (9.2)$$

where $\bar{\mathbf{w}}$ is now a standard Wiener process in the reverse-time direction. Here dt represents an infinitesimal negative time step, meaning that the above SDE must be solved from $t = T$ to $t = 0$. This reverse-time SDE results in exactly the same diffusion process $\{\mathbf{x}(t)\}_{t \in [0, T]}$ as Eq. (9.1), assuming it is initialized with $\mathbf{x}(T) \sim p_T(\mathbf{x})$. This result allows for the construction of diffusion-based generative models, and its functional form reveals the key target for learning: the time-dependent score function $\nabla_{\mathbf{x}} \log p_t(\mathbf{x})$. Again, see Fig. 9.1 for a helpful visualization of this two-part formulation.

In order to estimate $\nabla_{\mathbf{x}} \log p_t(\mathbf{x})$ from a given dataset, we fit the parameters of a neural

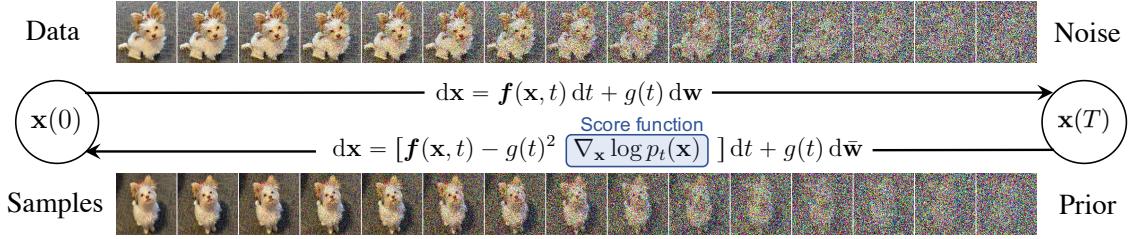


Figure 9.1: We can use an SDE to diffuse data to a simple noise distribution. This SDE can be reversed once we know the score of the marginal distribution at each intermediate time step, $\nabla_{\mathbf{x}} \log p_t(\mathbf{x})$.

network $s_{\theta}(\mathbf{x}, t)$, termed a *score-based model*, such that $s_{\theta}(\mathbf{x}, t) \approx \nabla_{\mathbf{x}} \log p_t(\mathbf{x})$ for almost all $\mathbf{x} \in \mathbb{R}^D$ and $t \in [0, T]$. Unlike many likelihood-based generative models, a score-based model does not need to satisfy the integral constraints of a density function, and is therefore much easier to parameterize. Good score-based models should keep the following least squares loss small

$$\mathcal{J}_{\text{SM}}(\boldsymbol{\theta}; \lambda(\cdot)) := \frac{1}{2} \int_0^T \mathbb{E}_{p_t(\mathbf{x})} [\lambda(t) \|\nabla_{\mathbf{x}} \log p_t(\mathbf{x}) - s_{\theta}(\mathbf{x}, t)\|_2^2] dt, \quad (9.3)$$

where $\lambda: [0, T] \rightarrow \mathbb{R}_{>0}$ is a positive weighting function. The integrand features the well-known score matching [HD05] objective $\mathbb{E}_{p_t(\mathbf{x})} [\|\nabla_{\mathbf{x}} \log p_t(\mathbf{x}) - s_{\theta}(\mathbf{x}, t)\|_2^2]$. We therefore refer to Eq. (9.3) as a weighted combination of score matching losses.

With score matching techniques [Vin11; Son+19], we can compute Eq. (9.3) up to an additive constant and minimize it for training score-based models. For example, we can use denoising score matching [Vin11] to transform $\mathcal{J}_{\text{SM}}(\boldsymbol{\theta}; \lambda(\cdot))$ into the following, which is equivalent up to a constant independent of $\boldsymbol{\theta}$:

$$\mathcal{J}_{\text{DSM}}(\boldsymbol{\theta}; \lambda(\cdot)) := \frac{1}{2} \int_0^T \mathbb{E}_{p(\mathbf{x}) p_{0t}(\mathbf{x}' | \mathbf{x})} [\lambda(t) \|\nabla_{\mathbf{x}'} \log p_{0t}(\mathbf{x}' | \mathbf{x}) - s_{\theta}(\mathbf{x}', t)\|_2^2] dt. \quad (9.4)$$

Whenever the drift coefficient $f_{\theta}(\mathbf{x}, t)$ is linear in \mathbf{x} (which is true for all SDEs in Chapter 7), the transition density $p_{0t}(\mathbf{x}' | \mathbf{x})$ is a tractable Gaussian distribution. We can form a Monte Carlo estimate of both the time integral and expectation in $\mathcal{J}_{\text{DSM}}(\boldsymbol{\theta}; \lambda(\cdot))$ with a sample $(t, \mathbf{x}, \mathbf{x}')$, where t is uniformly drawn from $[0, T]$, $\mathbf{x} \sim p(\mathbf{x})$ is a sample from the dataset, and $\mathbf{x}' \sim p_{0t}(\mathbf{x}' | \mathbf{x})$. The gradient $\nabla_{\mathbf{x}'} \log p_{0t}(\mathbf{x}' | \mathbf{x})$ can also be computed in closed form since $p_{0t}(\mathbf{x}' | \mathbf{x})$ is Gaussian.

After training a score-based model $s_{\theta}(\mathbf{x}, t)$ with $\mathcal{J}_{\text{DSM}}(\theta; \lambda(\cdot))$, we can plug it into the reverse-time SDE in Eq. (9.2). Samples are then generated by solving this reverse-time SDE with numerical SDE solvers, given an initial sample from $\pi(\mathbf{x})$ at $t = T$. Since the forward SDE Eq. (9.1) is designed such that $p_T(\mathbf{x}) \approx \pi(\mathbf{x})$, the reverse-time SDE will closely trace the diffusion process given by Eq. (9.1) in the reverse time direction, and yield an approximate data sample at $t = 0$ (as visualized in Fig. 9.1).

9.3 Likelihood of Score-Based Diffusion Models

The forward and backward diffusion processes in score-based diffusion models induce two probabilistic models for which we can define a likelihood. The first probabilistic model, denoted as $p_{\theta}^{\text{SDE}}(\mathbf{x})$, is given by the approximate reverse-time SDE constructed from our score-based model $s_{\theta}(\mathbf{x}, t)$. In particular, suppose $\{\hat{\mathbf{x}}_{\theta}(t)\}_{t \in [0, T]}$ is a stochastic process given by

$$d\hat{\mathbf{x}} = [\mathbf{f}(\hat{\mathbf{x}}, t) - g(t)^2 s_{\theta}(\hat{\mathbf{x}}, t)] dt + g(t) d\bar{\mathbf{w}}, \quad \hat{\mathbf{x}}_{\theta}(T) \sim \pi. \quad (9.5)$$

We define p_{θ}^{SDE} as the marginal distribution of $\hat{\mathbf{x}}_{\theta}(0)$. The probabilistic model p_{θ}^{SDE} is jointly defined by the score-based model $s_{\theta}(\mathbf{x}, t)$, the prior π , plus the drift and diffusion coefficients of the forward SDE in Eq. (9.1). We can obtain a sample $\hat{\mathbf{x}}_{\theta}(0) \sim p_{\theta}^{\text{SDE}}$ by numerically solving the reverse-time SDE in Eq. (9.5) with an initial noise vector $\hat{\mathbf{x}}_{\theta}(T) \sim \pi$.

The other probabilistic model, denoted $p_{\theta}^{\text{ODE}}(\mathbf{x})$, is derived from the SDE's associated *probability flow ODE* (cf., Chapter 7). Every SDE has a corresponding probability flow ODE whose marginal distribution at each time t matches that of the SDE, so that they share the same $p_t(\mathbf{x})$ for all time. In particular, the ODE corresponding to the SDE in Eq. (9.1) is given by

$$\frac{d\mathbf{x}}{dt} = \mathbf{f}(\mathbf{x}, t) - \frac{1}{2}g(t)^2 \nabla_{\mathbf{x}} \log p_t(\mathbf{x}). \quad (9.6)$$

Unlike the SDEs in Eq. (9.1) and Eq. (9.2), this ODE describes fully deterministic dynamics for the process. Notably, it still features the same time-dependent score function $\nabla_{\mathbf{x}} \log p_t(\mathbf{x})$. By approximating this score function with our model $s_{\theta}(\mathbf{x}, t)$, the probability flow ODE

becomes

$$\frac{d\tilde{\mathbf{x}}}{dt} = \mathbf{f}(\tilde{\mathbf{x}}, t) - \frac{1}{2}g(t)^2\mathbf{s}_{\theta}(\tilde{\mathbf{x}}, t). \quad (9.7)$$

In fact, this ODE is an instance of a continuous normalizing flow (CNF) [Gra+18], and we can quantify how the ODE dynamics transform volumes across time in exactly the same way as these traditional flow-based models [Che+18a]. Given a prior distribution $\pi(\mathbf{x})$, and a trajectory function $\tilde{\mathbf{x}}_{\theta}: [0, T] \rightarrow \mathbb{R}^D$ satisfying the ODE in Eq. (9.7), we define p_{θ}^{ODE} as the marginal distribution of $\tilde{\mathbf{x}}_{\theta}(0)$ when $\tilde{\mathbf{x}}_{\theta}(T) \sim \pi$. Similarly to p_{θ}^{SDE} , the model p_{θ}^{ODE} is jointly defined by the score-based model $\mathbf{s}_{\theta}(\mathbf{x}, t)$, the prior π , and the forward SDE in Eq. (9.1). Leveraging the instantaneous change-of-variables formula [Che+18a], we can evaluate $\log p_{\theta}^{\text{ODE}}(\mathbf{x})$ exactly with numerical ODE solvers. Since p_{θ}^{ODE} is a CNF, we can generate a sample $\tilde{\mathbf{x}}_{\theta}(0) \sim p_{\theta}^{\text{ODE}}$ by numerically solving the ODE in Eq. (9.7) with an initial value $\tilde{\mathbf{x}}_{\theta}(T) \sim \pi$.

Although computing $\log p_{\theta}^{\text{ODE}}(\mathbf{x})$ is tractable, training p_{θ}^{ODE} with maximum likelihood will require calling an ODE solver for every optimization step [Che+18a; Gra+18], which can be prohibitively expensive for large-scale score-based models. Unlike p_{θ}^{ODE} , we cannot evaluate $\log p_{\theta}^{\text{SDE}}(\mathbf{x})$ exactly for an arbitrary data point \mathbf{x} . However, we have a lower bound on $\log p_{\theta}^{\text{SDE}}(\mathbf{x})$ which allows both efficient evaluation and optimization, as will be shown in Section 9.4.2.

9.4 Bounding the Likelihood of Score-Based Diffusion Models

Many applications benefit from models which achieve high likelihood. One example is lossless compression, where log-likelihood directly corresponds to the minimum expected number of bits needed to encode a message. Indeed, popular likelihood-based models such as variational autoencoders and normalizing flows have already found success in image compression [TBB19; HLA19; Hoo+19]. Despite some known drawbacks [TOB16b], likelihood is still one of the most popular metrics for evaluating and comparing generative models.

Maximizing the likelihood of score-based diffusion models can be accomplished by either maximizing the likelihood of p_{θ}^{SDE} or p_{θ}^{ODE} . Although p_{θ}^{ODE} is a continuous normalizing flow (CNF) and its log-likelihood is tractable, training with maximum likelihood is expensive. As mentioned already, it requires solving an ODE at every optimization step in order to evaluate the log-likelihood on a batch of training data. In contrast, training with the

weighted combination of score matching losses is much more efficient, yet in general it does not directly promote high likelihood of either p_{θ}^{SDE} or p_{θ}^{ODE} .

In what follows, we show that with a specific choice of the weighting function $\lambda(t)$, the combination of score matching losses $\mathcal{J}_{\text{SM}}(\boldsymbol{\theta}; \lambda(\cdot))$ actually becomes an upper bound on $D_{\text{KL}}(p \parallel p_{\theta}^{\text{SDE}})$, and can therefore serve as an efficient proxy for maximum likelihood training. In addition, we provide a related lower bound on $\log p_{\theta}^{\text{SDE}}(\mathbf{x})$ that can be evaluated efficiently on any individual data point \mathbf{x} .

9.4.1 Bounding the KL Divergence with Likelihood Weighting

It is well-known that maximizing the log-likelihood of a probabilistic model is equivalent to minimizing the KL divergence from the data distribution to the model distribution. We show in the following theorem that for the model p_{θ}^{SDE} , this KL divergence can be upper bounded by $\mathcal{J}_{\text{SM}}(\boldsymbol{\theta}; \lambda(\cdot))$ when using the weighting function $\lambda(t) = g(t)^2$, where $g(t)$ is the diffusion coefficient of SDE in Eq. (9.1).

Theorem 9.1. *Let $p(\mathbf{x})$ be the data distribution, $\pi(\mathbf{x})$ be a known prior distribution, and p_{θ}^{SDE} be defined as in Section 9.3. Suppose $\{\mathbf{x}(t)\}_{t \in [0, T]}$ is a stochastic process defined by the SDE in Eq. (9.1) with $\mathbf{x}(0) \sim p$, where the marginal distribution of $\mathbf{x}(t)$ is denoted as p_t . Under some regularity conditions detailed in Appendix A.5, we have*

$$D_{\text{KL}}(p \parallel p_{\theta}^{\text{SDE}}) \leq \mathcal{J}_{\text{SM}}(\boldsymbol{\theta}; g(\cdot)^2) + D_{\text{KL}}(p_T \parallel \pi). \quad (9.8)$$

Sketch of proof. Let $\boldsymbol{\mu}$ and $\boldsymbol{\nu}$ denote the path measures of SDEs in Eq. (9.1) and Eq. (9.5) respectively. Intuitively, $\boldsymbol{\mu}$ is the joint distribution of the diffusion process $\{\mathbf{x}(t)\}_{t \in [0, T]}$ given in Section 9.2.1, and $\boldsymbol{\nu}$ represents the joint distribution of the process $\{\hat{\mathbf{x}}_{\theta}(t)\}_{t \in [0, T]}$ defined in Section 9.3. Since we can marginalize $\boldsymbol{\mu}$ and $\boldsymbol{\nu}$ to obtain distributions p and p_{θ}^{SDE} , the data processing inequality gives $D_{\text{KL}}(p \parallel p_{\theta}^{\text{SDE}}) \leq D_{\text{KL}}(\boldsymbol{\mu} \parallel \boldsymbol{\nu})$. From the chain rule for the KL divergence, we also have $D_{\text{KL}}(\boldsymbol{\mu} \parallel \boldsymbol{\nu}) = D_{\text{KL}}(p_T \parallel \pi) + \mathbb{E}_{p_T(\mathbf{z})}[D_{\text{KL}}(\boldsymbol{\mu}(\cdot \mid \mathbf{x}(T) = \mathbf{z}) \parallel \boldsymbol{\nu}(\cdot \mid \hat{\mathbf{x}}_{\theta}(T) = \mathbf{z}))]$, where the KL divergence in the final term can be computed by applying the Girsanov theorem [Oks13] to Eq. (9.5) and the reverse-time SDE of Eq. (9.1). \square

When the prior distribution π is fixed, Theorem 9.1 guarantees that optimizing the weighted combination of score matching losses $\mathcal{J}_{\text{SM}}(\boldsymbol{\theta}; g(\cdot)^2)$ is equivalent to minimizing an upper bound on the KL divergence from the data distribution p to the model distribution

Table 9.1: SDEs and their corresponding weightings for score matching losses.

SDE	Formula	$\lambda(t)$ in Chapter 7	likelihood weighting
VE	$d\mathbf{x} = \sigma(t) d\mathbf{w}$	$\sigma^2(t)$	$\sigma^2(t)$
VP	$d\mathbf{x} = -\frac{1}{2}\beta(t)\mathbf{x} dt + \sqrt{\beta(t)} d\mathbf{w}$	$1 - e^{-\int_0^t \beta(s) ds}$	$\beta(t)$
subVP	$d\mathbf{x} = -\frac{1}{2}\beta(t)\mathbf{x} dt + \sqrt{\beta(t)(1 - e^{-2\int_0^t \beta(s) ds})} d\mathbf{w}$	$(1 - e^{-\int_0^t \beta(s) ds})^2$	$\beta(t)(1 - e^{-2\int_0^t \beta(s) ds})$

p_{θ}^{SDE} . Due to well-known equivalence between minimizing KL divergence and maximizing likelihood, we have the following corollary.

Corollary 9.1. *Consider the same conditions and notations in Theorem 9.1. When π is a fixed prior distribution that does not depend on θ , we have*

$$-\mathbb{E}_{p(\mathbf{x})}[\log p_{\theta}^{\text{SDE}}(\mathbf{x})] \leq \mathcal{J}_{\text{SM}}(\theta; g(\cdot)^2) + C_1 = \mathcal{J}_{\text{DSM}}(\theta; g(\cdot)^2) + C_2,$$

where C_1 and C_2 are constants independent of θ .

In light of the result in Corollary 9.1, we henceforth term $\lambda(t) = g(t)^2$ the *likelihood weighting*. The original weighting functions in Chapter 7 are inspired from earlier work such as [SE19b; SE20] and [HJA20], which are motivated by balancing different score matching losses in the combination, and justified by empirical performance. In contrast, likelihood weighting is motivated from maximizing the likelihood of a probabilistic model induced by the diffusion process, and derived by theoretical analysis. There are three types of SDEs considered in Chapter 7: the Variance Exploding (VE) SDE, the Variance Preserving (VP) SDE, and the subVP SDE. In Table 9.1, we summarize all these SDEs and contrast their original weighting functions with our likelihood weighting. For VE SDE, our likelihood weighting incidentally coincides with the original weighting used in Chapter 7, whereas for VP and subVP SDEs they differ from one another.

Theorem 9.1 leaves two questions unanswered. First, what are the conditions for the bound to be tight (become an equality)? Second, is there any connection between p_{θ}^{SDE} and p_{θ}^{ODE} under some conditions? We provide both answers in the following theorem.

Theorem 9.2. *Suppose $p(\mathbf{x})$ and $q(\mathbf{x})$ have continuous second-order derivatives and finite second moments. Let $\{\mathbf{x}(t)\}_{t \in [0, T]}$ be the diffusion process defined by the SDE in Eq. (9.1). We use p_t and q_t to denote the distributions of $\mathbf{x}(t)$ when $\mathbf{x}(0) \sim p$ and $\mathbf{x}(0) \sim q$, and assume*

they satisfy the same assumptions in Appendix A.5. Under the conditions $q_T = \pi$ and $\mathbf{s}_\theta(\mathbf{x}, t) \equiv \nabla_{\mathbf{x}} \log q_t(\mathbf{x})$ for all $t \in [0, T]$, we have the following equivalence in distributions

$$p_\theta^{\text{SDE}} = p_\theta^{\text{ODE}} = q. \quad (9.9)$$

Moreover, we have

$$D_{\text{KL}}(p \parallel p_\theta^{\text{SDE}}) = \mathcal{J}_{\text{SM}}(\boldsymbol{\theta}; g(\cdot)^2) + D_{\text{KL}}(p_T \parallel \pi). \quad (9.10)$$

Sketch of proof. When $\mathbf{s}_\theta(\mathbf{x}, t)$ matches $\nabla_{\mathbf{x}} \log q_t(\mathbf{x})$, they both represent the time-dependent score of the same stochastic process so we immediately have $p_\theta^{\text{SDE}} = q$. According to the theory of probability flow ODEs, we also have $p_\theta^{\text{ODE}} = q = p_\theta^{\text{SDE}}$. To prove Eq. (9.10), we note that $D_{\text{KL}}(p \parallel p_\theta^{\text{SDE}}) = D_{\text{KL}}(p \parallel q) = D_{\text{KL}}(p_T \parallel q_T) - \int_0^T \frac{d}{dt} D_{\text{KL}}(p_t \parallel q_t) dt = D_{\text{KL}}(p_T \parallel \pi) - \int_0^T \frac{d}{dt} D_{\text{KL}}(p_t \parallel q_t) dt$. We can now complete the proof by simplifying the integrand using the Fokker–Planck equation of p_t and q_t followed by integration by parts. \square

In practice, the conditions of Theorem 9.2 are hard to satisfy since our score-based model $\mathbf{s}_\theta(\mathbf{x}, t)$ will not exactly match the score function $\nabla_{\mathbf{x}} \log q_t(\mathbf{x})$ of some reverse-time diffusion process with the initial distribution $q_T = \pi$. In other words, our score model may not be a valid time-dependent score function of a stochastic process with an appropriate initial distribution. Therefore, although score matching with likelihood weighting performs approximate maximum likelihood training for p_θ^{SDE} , we emphasize that it is not theoretically guaranteed to make the likelihood of p_θ^{ODE} better. That said, p_θ^{ODE} will closely match p_θ^{SDE} if our score-based model well-approximates the true score such that $\mathbf{s}_\theta(\mathbf{x}, t) \approx \nabla_{\mathbf{x}} \log p_t(\mathbf{x})$ for all \mathbf{x} and $t \in [0, T]$. Moreover, we empirically observe in our experiments (see Table 9.2) that training with the likelihood weighting is actually able to consistently improve the likelihood of p_θ^{ODE} across multiple datasets, SDEs, and model architectures.

9.4.2 Bounding the Log-Likelihood on Individual data points

The bound in Theorem 9.1 is for the entire distributions of p and p_θ^{SDE} , but we often seek to bound the log-likelihood for an individual data point \mathbf{x} . In addition, $\mathcal{J}_{\text{SM}}(\boldsymbol{\theta}; \lambda(\cdot))$ in the bound is not directly computable due to the unknown quantity $\nabla_{\mathbf{x}} \log p_t(\mathbf{x})$, and can only be evaluated up to an additive constant through $\mathcal{J}_{\text{DSM}}(\boldsymbol{\theta}; \lambda(\cdot))$ (as we already discussed in Section 9.2.2). Therefore, the bound in Theorem 9.1 is only suitable for training purposes.

To address these issues, we provide the following bounds for individual data points.

Theorem 9.3. *Let $p_{0t}(\mathbf{x}' | \mathbf{x})$ denote the transition distribution from $p_0(\mathbf{x})$ to $p_t(\mathbf{x})$ for the SDE in Eq. (9.1). With the same notations and conditions in Theorem 9.1, we have*

$$-\log p_{\boldsymbol{\theta}}^{\text{SDE}}(\mathbf{x}) \leq \mathcal{L}_{\boldsymbol{\theta}}^{\text{SM}}(\mathbf{x}) = \mathcal{L}_{\boldsymbol{\theta}}^{\text{DSM}}(\mathbf{x}), \quad (9.11)$$

where $\mathcal{L}_{\boldsymbol{\theta}}^{\text{SM}}(\mathbf{x})$ is defined as

$$\begin{aligned} -\mathbb{E}_{p_{0T}(\mathbf{x}'|\mathbf{x})}[\log \pi(\mathbf{x}')] + \frac{1}{2} \int_0^T \mathbb{E}_{p_{0t}(\mathbf{x}'|\mathbf{x})} & \left[2g(t)^2 \nabla_{\mathbf{x}'} \cdot \mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}', t) \right. \\ & \left. + g(t)^2 \|\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}', t)\|_2^2 - 2\nabla_{\mathbf{x}'} \cdot \mathbf{f}(\mathbf{x}', t) \right] dt, \end{aligned}$$

and $\mathcal{L}_{\boldsymbol{\theta}}^{\text{DSM}}(\mathbf{x})$ is given by

$$\begin{aligned} -\mathbb{E}_{p_{0T}(\mathbf{x}'|\mathbf{x})}[\log \pi(\mathbf{x}')] + \frac{1}{2} \int_0^T \mathbb{E}_{p_{0t}(\mathbf{x}'|\mathbf{x})} & \left[g(t)^2 \|\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}', t) - \nabla_{\mathbf{x}'} \log p_{0t}(\mathbf{x}' | \mathbf{x})\|_2^2 \right] dt \\ & - \frac{1}{2} \int_0^T \mathbb{E}_{p_{0t}(\mathbf{x}'|\mathbf{x})} \left[g(t)^2 \|\nabla_{\mathbf{x}'} \log p_{0t}(\mathbf{x}' | \mathbf{x})\|_2^2 + 2\nabla_{\mathbf{x}'} \cdot \mathbf{f}(\mathbf{x}', t) \right] dt. \end{aligned}$$

Sketch of proof. For any continuous data distribution p , we have $-\mathbb{E}_{p(\mathbf{x})}[\log p_{\boldsymbol{\theta}}^{\text{SDE}}(\mathbf{x})] = D_{\text{KL}}(p \parallel p_{\boldsymbol{\theta}}^{\text{SDE}}) + \mathcal{H}(p)$, where $\mathcal{H}(p)$ denotes the differential entropy of p . The KL term can be bounded according to Theorem 9.1, while the differential entropy has an identity similar to Theorem 9.2 (see Theorem A.1 in Appendix A.5). Combining the bound of $D_{\text{KL}}(p \parallel p_{\boldsymbol{\theta}}^{\text{SDE}})$ and the identity of $\mathcal{H}(p)$, we obtain a bound on $-\mathbb{E}_{p(\mathbf{x})}[\log p_{\boldsymbol{\theta}}^{\text{SDE}}(\mathbf{x})]$ that holds for all continuous distribution p . Removing the expectation over p on both sides then gives us a bound on $-\log p_{\boldsymbol{\theta}}^{\text{SDE}}(\mathbf{x})$ for an individual data point \mathbf{x} . We can simplify this bound to $\mathcal{L}_{\boldsymbol{\theta}}^{\text{SM}}(\mathbf{x})$ and $\mathcal{L}_{\boldsymbol{\theta}}^{\text{DSM}}(\mathbf{x})$ with similar techniques to [HD05] and [Vin11]. \square

We provide two equivalent bounds $\mathcal{L}_{\boldsymbol{\theta}}^{\text{SM}}(\mathbf{x})$ and $\mathcal{L}_{\boldsymbol{\theta}}^{\text{DSM}}(\mathbf{x})$. The former bears resemblance to score matching while the second resembles denoising score matching. Both admit efficient unbiased estimators when $\mathbf{f}(\cdot, t)$ is linear, as the time integrals and expectations in $\mathcal{L}_{\boldsymbol{\theta}}^{\text{SM}}(\mathbf{x})$ and $\mathcal{L}_{\boldsymbol{\theta}}^{\text{DSM}}(\mathbf{x})$ can be estimated by samples of the form (t, \mathbf{x}') , where t is uniformly sampled over $[0, T]$, and $\mathbf{x}' \sim p_{0t}(\mathbf{x}' | \mathbf{x})$. Since the transition distribution $p_{0t}(\mathbf{x}' | \mathbf{x})$ is a tractable Gaussian when $\mathbf{f}(\cdot, t)$ is linear, we can easily sample from it as well as evaluating $\nabla_{\mathbf{x}'} \log p_{0t}(\mathbf{x}' | \mathbf{x})$ for computing $\mathcal{L}_{\boldsymbol{\theta}}^{\text{DSM}}(\mathbf{x})$. Moreover, the divergences $\nabla_{\mathbf{x}} \cdot \mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}, t)$ and

$\nabla_{\mathbf{x}} \cdot \mathbf{f}(\mathbf{x}, t)$ in $\mathcal{L}_{\theta}^{\text{SM}}(\mathbf{x})$ and $\mathcal{L}_{\theta}^{\text{DSM}}(\mathbf{x})$ have efficient unbiased estimators via the Skilling–Hutchinson trick [Ski89; Hut89].

We can view $\mathcal{L}_{\theta}^{\text{DSM}}(\mathbf{x})$ as a continuous-time generalization of the evidence lower bound (ELBO) in diffusion probabilistic models [Soh+15; HJA20]. Our bounds in Theorem 9.3 are not only useful for optimizing and estimating $\log p_{\theta}^{\text{SDE}}(\mathbf{x})$, but also for training the drift and diffusion coefficients $\mathbf{f}(\mathbf{x}, t)$ and $g(t)$ jointly with the score-based model $s_{\theta}(\mathbf{x}, t)$; we leave this avenue of research for future work. In addition, we can plug the bounds in Theorem 9.3 into any objective that involves minimizing $-\log p_{\theta}^{\text{SDE}}(\mathbf{x})$ to obtain an efficient surrogate. Section 9.5.2 provides an example, where we perform variational dequantization to further improve the likelihood of score-based diffusion models.

Similar to the observation in Section 9.4.1, $\mathcal{L}_{\theta}^{\text{SM}}(\mathbf{x})$ and $\mathcal{L}_{\theta}^{\text{DSM}}(\mathbf{x})$ are not guaranteed to upper bound $-\log p_{\theta}^{\text{ODE}}(\mathbf{x})$. However, they should become approximate upper bounds when $s_{\theta}(\mathbf{x}, t)$ is trained sufficiently close to the ground truth. In fact, we empirically observe that $-\log p_{\theta}^{\text{ODE}}(\mathbf{x}) \leq \mathcal{L}_{\theta}^{\text{SM}}(\mathbf{x}) = \mathcal{L}_{\theta}^{\text{DSM}}(\mathbf{x})$ holds true for \mathbf{x} sampled from the dataset in all experiments.

9.4.3 Numerical Stability

So far we have assumed that the SDEs are defined in the time horizon $[0, T]$ in all theoretical analysis. In practice, however, we often face numerical instabilities when $t \rightarrow 0$. To avoid them, we choose a small non-zero starting time $\epsilon > 0$, and train/evaluate score-based diffusion models in the time horizon $[\epsilon, T]$ instead of $[0, T]$. Since ϵ is small, training score-based diffusion models with likelihood weighting still approximately maximizes their model likelihood. Yet at test time, the likelihood bound as computed in Theorem 9.3 is slightly biased, rendering the values not directly comparable to results reported in other works. We use Jensen’s inequality to correct for this bias in our experiments, for which we provide a detailed explanation in Appendix B.7.2.

9.4.4 Related Work

Our result in Theorem 9.2 can be viewed as a generalization of De Bruijn’s identity ([Sta59], Eq. 2.12) from its original differential form to an integral form. De Bruijn’s identity relates the rate of change of the Shannon entropy under an additive Gaussian noise channel to the Fisher information, a result which can be interpreted geometrically as relating the rate

of change of the volume of a distribution’s typical set to its surface area. Barron [Bar86] (Lemma 1) builds on this result and presents an integral and relative form of de Bruijn’s identity which relates the KL divergence to the integral of the relative Fisher information for a distribution of interest and a reference standard normal. More generally, various identities and inequalities involving the (relative) Shannon entropy and (relative) Fisher information have found use in proofs of the central limit theorem [JB04]. Lyu [Lyu09] (Theorem 1) covers similar ground to the relative form of de Bruijn’s identity, but is perhaps the first to consider its implications for learning in probabilistic models by framing the discussion in terms of the score matching objective ([HD05], Eq. 2).

9.5 Improving the Likelihood of Score-Based Diffusion Models

Our theoretical analysis implies that training with the likelihood weighting should improve the likelihood of score-based diffusion models. To verify this empirically, we test likelihood weighting with different model architectures, SDEs, and datasets. We observe that switching to likelihood weighting increases the variance of the training objective and propose to counteract it with importance sampling. We additionally combine our bound with variational dequantization [Ho+19] which narrows the gap between the likelihood of continuous and discrete probability models. All combined, we observe consistent improvement of likelihoods for both p_{θ}^{SDE} and p_{θ}^{ODE} across all settings. We term the model p_{θ}^{ODE} trained in this way *ScoreFlow*, and show that it achieves excellent likelihoods on CIFAR-10 [KNH14] and ImageNet 32×32 [VKK16], on a par with cutting-edge autoregressive models.

9.5.1 Variance Reduction via Importance Sampling

As mentioned in Section 9.2.2, we typically use Monte Carlo sampling to approximate the time integral in $\mathcal{J}_{\text{DSM}}(\theta; \lambda(\cdot))$ during training. In particular, we first uniformly sample a time step $t \sim \mathcal{U}[0, T]$, and then use the denoising score matching loss at t as an estimate for the whole time integral. This Monte Carlo approximation is much faster than computing the time integral accurately, but introduces additional variance to the training loss.

We empirically observe that this Monte Carlo approximation suffers from a larger variance when using our likelihood weighting instead of the original weightings in Chapter 7.

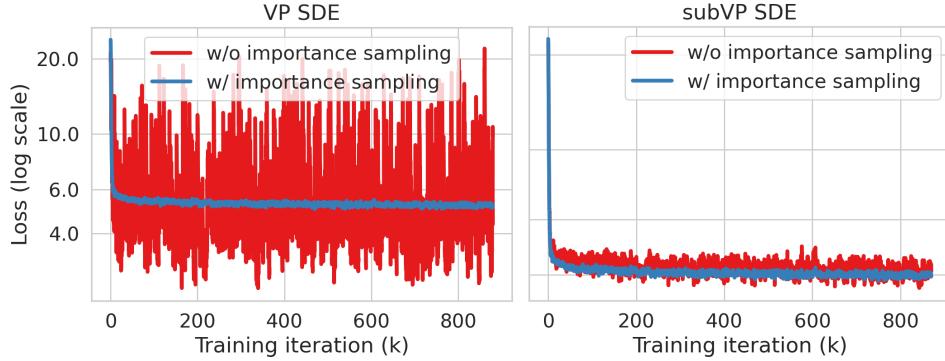


Figure 9.2: Learning curves with the likelihood weighting on the CIFAR-10 dataset (smoothed with exponential moving average). Importance sampling significantly reduces the loss variance.

Leveraging importance sampling, we propose a new Monte Carlo approximation that significantly reduces the variance of learning curves under likelihood weighting, as demonstrated in Fig. 9.2. In fact, with importance sampling, the loss variance (after convergence) decreases from 98.48 to 0.068 on CIFAR-10, and decreases from 0.51 to 0.043 on ImageNet.

Let $\lambda(t) = \alpha(t)^2$ denote the weightings in Chapter 7 (reproduced in Table 9.1), and recall that our likelihood weighting is $\lambda(t) = g(t)^2$. Since $\alpha(t)^2$ empirically leads to lower variance, we can use a proposal distribution $p(t) := g(t)^2/\alpha(t)^2 Z$ to change the weighting in $\mathcal{J}_{\text{DSM}}(\theta; g(\cdot)^2)$ from $g(t)^2$ to $\alpha(t)^2$ with importance sampling, where Z is a normalizing constant that ensures $\int p(t) dt = 1$. Specifically, for any function $h(t)$, we estimate the time integral $\int_0^T g(t)^2 h(t) dt$ with

$$\int_0^T g(t)^2 h(t) dt = Z \int_0^T p(t) \alpha(t)^2 h(t) dt \approx TZ\alpha(\tilde{t})^2 h(\tilde{t}), \quad (9.12)$$

where \tilde{t} is a sample from $p(t)$. When training score-based models with likelihood weighting, $h(t)$ corresponds to the denoising score matching loss at time t .

Nichol and Dhariwal [ND21] also observed that optimizing the ELBO for diffusion probabilistic models has large variance, and proposed to reduce it with importance sampling. They built their proposal distribution based on historical loss values stored at thousands of discrete time steps. Despite this similarity, our method is easier to implement without needing to maintain history, can be used for evaluation, and is particularly suited to the continuous-time setting.

9.5.2 Variational Dequantization

Digital images are discrete data, and must be dequantized when training continuous density models like normalizing flows [DKB15; DSB17] and score-based diffusion models. One popular approach to this is uniform dequantization [UML13; TOB16b], where we add small uniform noise over $[0, 1)$ to images taking values in $\{0, 1, \dots, 255\}$. As shown in [TOB16b], training a continuous model $p_{\theta}(\mathbf{x})$ on uniformly dequantized data implicitly maximizes a lower bound on the log-likelihood of a certain discrete model $P_{\theta}(\mathbf{x})$. Due to the gap between $p_{\theta}(\mathbf{x})$ and $P_{\theta}(\mathbf{x})$, comparing the likelihood of continuous density models to models which fit discrete data directly, such as autoregressive models [VKK16] or variational autoencoders, naturally puts the former at a disadvantage.

To minimize the gap between $p_{\theta}(\mathbf{x})$ and $P_{\theta}(\mathbf{x})$, Ho et al. [Ho+19] proposes variational dequantization, where a separate normalizing flow model $q_{\phi}(\mathbf{u} \mid \mathbf{x})$ is trained to produce the dequantization noise by optimizing the following objective

$$\max_{\phi} \mathbb{E}_{\mathbf{x} \sim p(\mathbf{x})} \mathbb{E}_{\mathbf{u} \sim q_{\phi}(\cdot \mid \mathbf{x})} [\log p_{\theta}(\mathbf{x} + \mathbf{u}) - \log q_{\phi}(\mathbf{u} \mid \mathbf{x})]. \quad (9.13)$$

Plugging in the lower bound on $\log p_{\theta}(\mathbf{x})$ from Theorem 9.3, we can optimize Eq. (9.13) to improve the likelihood of score-based diffusion models.

9.5.3 Experiments

We empirically test the performance of likelihood weighting, importance sampling and variational dequantization across multiple architectures of score-based models, SDEs, and datasets. In particular, we consider DDPM++ (“Baseline” in Table 9.2) and DDPM++(deep) (“Deep” in Table 9.2) models with VP and subVP SDEs (Chapter 7) on CIFAR-10 [KNH14] and ImageNet 32×32 [VKK16] datasets. We omit experiments on the VE SDE since (i) under this SDE our likelihood weighting is the same as the original weighting in Chapter 7; (ii) we empirically observe that the best VE SDE model achieves around 3.4 bits/dim on CIFAR-10 in our experiments, which is significantly worse than other SDEs. For each experiment, we report $-\mathbb{E}[\log p_{\theta}^{\text{ODE}}(\mathbf{x})]$ (“Negative log-likelihood” in Table 9.2), and the upper bound $\mathbb{E}[\mathcal{L}_{\theta}^{\text{DSM}}(\mathbf{x})]$ on $-\mathbb{E}[\log p_{\theta}^{\text{SDE}}(\mathbf{x})]$ (“Bound” in Table 9.2). In addition, we report FID scores [Heu+17] for samples from p_{θ}^{ODE} , produced by solving the corresponding ODE with the Dormand–Prince RK45 [DP80] solver. Unless otherwise noted, we apply horizontal flipping as data augmentation for training models on CIFAR-10, so as to match

Table 9.2: Negative log-likelihood (bits/dim) and sample quality (FID scores) on CIFAR-10 and ImageNet 32×32 . Abbreviations: “NLL” for “negative log-likelihood”; “Uni. deq.” for “Uniform dequantization”; “Var. deq.” for “Variational dequantization”; “LW” for “likelihood weighting”; and “IS” for “importance sampling”. Bold indicates best result in the corresponding column. Shaded rows represent models trained with both likelihood weighting and importance sampling.

Model	SDE	CIFAR-10					ImageNet 32×32				
		Uni. deq.		Var. deq.		FID↓	Uni. deq.		Var. deq.		FID↓
		NLL↓	Bound↓	NLL↓	Bound↓		NLL↓	Bound↓	NLL↓	Bound↓	
Baseline	VP	3.16	3.28	3.04	3.14	3.98	3.90	3.96	3.84	3.91	8.34
Baseline + LW	VP	3.06	3.18	2.94	3.03	5.18	3.91	3.96	3.86	3.92	17.75
Baseline + LW + IS	VP	2.95	3.08	2.83	2.94	6.03	3.86	3.92	3.80	3.88	11.15
Deep	VP	3.13	3.25	3.01	3.10	3.09	3.89	3.95	3.84	3.90	8.40
Deep + LW	VP	3.06	3.17	2.93	3.02	7.88	3.91	3.96	3.86	3.92	17.73
Deep + LW + IS	VP	2.93	3.06	2.80	2.92	5.34	3.85	3.92	3.79	3.88	11.20
Baseline	subVP	2.99	3.09	2.88	2.98	3.20	3.87	3.92	3.82	3.88	8.71
Baseline + LW	subVP	2.97	3.07	2.86	2.96	7.33	3.87	3.92	3.82	3.88	12.99
Baseline + LW + IS	subVP	2.94	3.05	2.84	2.94	5.58	3.84	3.91	3.79	3.87	10.57
Deep	subVP	2.96	3.06	2.85	2.95	2.86	3.86	3.91	3.81	3.87	8.87
Deep + LW	subVP	2.95	3.05	2.85	2.94	6.57	3.88	3.93	3.83	3.88	16.55
Deep + LW + IS	subVP	2.90	3.02	2.81	2.90	5.40	3.82	3.90	3.76	3.86	10.18

the settings in [Son+21b; HJA20]. Detailed description of all our experiments can be found in Appendices B.7 and B.7.2.

We summarize all results in Table 9.2. Our key observations are as follows:

1. Although Theorem 9.3 only guarantees $\mathbb{E}[\mathcal{L}_{\theta}^{\text{DSM}}(\mathbf{x})] \geq -\mathbb{E}[\log p_{\theta}^{\text{SDE}}(\mathbf{x})]$, and in general we have $p_{\theta}^{\text{SDE}} \neq p_{\theta}^{\text{ODE}}$, we still find that $\mathbb{E}[\mathcal{L}_{\theta}^{\text{DSM}}(\mathbf{x})]$ (“Bound” in Table 9.2) $\geq -\mathbb{E}[\log p_{\theta}^{\text{ODE}}(\mathbf{x})]$ (“NLL” in Table 9.2) in all our settings.
2. When all conditions are fixed except for the weighting in the training objective, having a lower value of the bound for p_{θ}^{SDE} always leads to a lower negative log-likelihood for p_{θ}^{ODE} .
3. With only likelihood weighting, we can uniformly improve the likelihood of p_{θ}^{ODE} and the bound of p_{θ}^{SDE} on CIFAR-10 across model architectures and SDEs, but it is not sufficient to guarantee likelihood improvement on ImageNet 32×32 .
4. By combining importance sampling and likelihood weighting, we are able to achieve uniformly better likelihood for p_{θ}^{ODE} and bounds for p_{θ}^{SDE} across all model architectures,

Table 9.3: NLLs on CIFAR-10 and ImageNet 32x32.

Model	CIFAR-10	ImageNet
FFJORD [Gra+18]	3.40	-
Flow++ [Ho+19]	3.08	3.86
Gated PixelCNN [Oor+16b]	3.03	3.83
VFlow [Che+20]	2.98	3.83
PixelCNN++ [Sal+17]	2.92	-
NVAE [VK20]	2.91	3.92
Image Transformer [Par+18]	2.90	3.77
Very Deep VAE [Chi21]	2.87	3.80
PixelSNAIL [Che+18b]	2.85	3.80
δ -VAE [Raz+19]	2.83	3.77
Sparse Transformer [Chi+19]	2.80	-
ScoreFlow (Ours)	2.83	3.76

SDEs, and datasets, with only slight degradation of sample quality as measured by FID [Heu+17].

5. Variational dequantization uniformly improves both the bound for p_{θ}^{SDE} and the negative log-likelihood (NLL) of p_{θ}^{ODE} in all settings, regardless of likelihood weighting.

Our experiments confirm that with importance sampling, likelihood weighting is not only effective for maximizing the lower bound for the log-likelihood of p_{θ}^{SDE} , but also improving the log-likelihood of p_{θ}^{ODE} . In agreement with [HJA20; ND21], we observe that models achieving better likelihood tend to have worse FIDs. However, we emphasize that this degradation of FID is small, and samples actually have no obvious difference in visual quality (see Figs. B.55 and B.56). To trade likelihood for FID, we can use weighting functions that interpolate between likelihood weighting and the original weighting functions in Chapter 7. Our FID scores are still much better than most other likelihood-based models.

We term p_{θ}^{ODE} a *ScoreFlow* when its corresponding score-based model $s_{\theta}(\mathbf{x}, t)$ is trained with likelihood weighting, importance sampling, and variational dequantization combined. It can be viewed as a continuous normalizing flow, but is parameterized by a score-based model and trained in a more efficient way. With variational dequantization, we show ScoreFlows obtain competitive negative log-likelihoods (NLLs) of 2.83 bits/dim on CIFAR-10 and 3.76 bits/dim on ImageNet 32×32. Here the ScoreFlow on CIFAR-10 is trained without horizontal flipping (different from the setting in Table 9.2). As shown in Table 9.3, our results are

on a par with the state-of-the-art autoregressive models on these tasks, and outperform all existing normalizing flow models. The likelihood for CIFAR-10 can be significantly improved by incorporating advanced data augmentation, as demonstrated in [Jun+20; SD21]. While we do not compare against these approaches, we believe that incorporating the same data augmentation techniques could also improve the likelihood of ScoreFlows.

9.6 Conclusion

We connect score-based diffusion models with continuous normalizing flows through the probability flow ODE, thereby showing that their log-likelihoods can be computed accurately with numerical ODE solvers. In addition, we propose an efficient training objective for approximate maximum likelihood training of score-based diffusion models that is much faster than calling an expensive ODE solver at each training iteration. Our theoretical analysis reveals that the weighted combination of score matching losses upper bounds the negative log-likelihood when using a particular weighting function which we term the likelihood weighting. By minimizing this upper bound, we consistently improve the likelihood of score-based diffusion models across multiple model architectures, SDEs, and datasets. When combined with variational dequantization, we achieve competitive likelihoods on CIFAR-10 and ImageNet 32×32 , matching the performance of best-in-class autoregressive models.

Our upper bound is analogous to the evidence lower bound commonly used for training variational autoencoders. Aside from promoting higher likelihood, the bound can be combined with other objectives that depend on the negative log-likelihood, and also enables joint training of the forward and backward SDEs, which we leave as a future research direction. Our results suggest that score-based diffusion models are competitive alternatives to continuous normalizing flows which enjoy the same tractable likelihood computation but with more efficient maximum likelihood training.

Chapter 10

Conclusion

We have presented score-based generative modeling, a new family of generative techniques that model, estimate and sample from the score function of the underlying data distribution. Contrary to existing likelihood-based methods (*e.g.*, variational autoencoders [KW14; RMW14a], normalizing flows [DKB15; DSB17], autoregressive models [BB99; UML13; VKK16], and energy-based models [Hin02; SK21] as discussed in Sections 2.2.1 and 2.2.2), score-based generative models bypass the challenge of normalization in probabilistic modeling (*cf.*, Section 2.1.3) without limiting model flexibility, while allowing accurate computation of probability values. Contrary to generative adversarial networks (GANs, [Goo+14]), score-based generative models avoid the instability of adversarial optimization while producing new data samples with comparable or even better quality. Furthermore, score-based generative models excel at solving challenging ill-posed inverse problems that are ubiquitous in science and engineering, as demonstrated by their success in computer vision applications (Chapter 7) and medical image reconstruction (Chapter 8).

Despite clear advantages of score-based generative models showcased in this dissertation, several research questions still remain unanswered. First of all, since score functions are gradients of log-densities, they are ill-defined for discrete data distributions. On the other hand, generative models for discrete data, such as large language models [Bro+20], have become predominant in artificial intelligence. How can we generalize score functions to discrete data domains while making sure the success of score-based generative modeling carries over? Second, the iterative sampling process of score-based generative models involves repeated model evaluation, which can be prohibitively slow for applications that require real-time generation. How can we make score-based models generate faster while

keeping all their existing benefits? Third but not least, while many generative models (*e.g.*, variational autoencoders [KW14; RMW14a], restricted Boltzmann machines [Hin02]) can extract useful representations from unlabeled data to benefit downstream applications, score-based generative models do not provide a direct way to do so due to lack of a low-dimensional latent space. How can we endow score-based generative models with such latent structures? Collectively, these questions shed lights on future research directions for score-based generative modeling.

The past few years have witnessed unprecedented progress in generative modeling including but not limited to score-based approaches. These latest advancements have enabled new capabilities of artificial intelligence (AI) but also posed new challenges, some of which we highlight below.

Societal impacts of generative models The power to generate realistic content at large scale could have unexpected societal impacts. Misuse of generative models has already produced photo and video “deepfakes” for harassment and spreading disinformation. When deployed in production, generative models can amplify various social biases (*e.g.*, gender, ethnicity, nationality) that may exist in the original training dataset, causing harm to marginalized communities and raising ethical/legal concerns. To minimize the potential negative consequences of generative models on society, we need to develop methods for detecting fake content created by generative models and mitigating biases of generative models, among other directions in algorithmic fairness and AI ethics. On the other hand, generative techniques may be useful for addressing negative societal impacts of other machine learning algorithms. For example, we have already shown in [Son+18a] and [Son+18b] that generative models can be used to verify and improve the robustness of classifiers. Further progress in generative modeling is thus likely to be conducive to building more reliable machine learning systems.

Accessible generative modeling Despite the empirical success on many tasks, generative models still require considerable expertise for training and tuning when applied to new settings. For example, the training of GANs [Goo+14] is notoriously unstable, and practitioners have to rely on heuristics to select model architectures and optimizers for successful training. Similarly, autoregressive generative models [VKK16] require a prescribed ordering of data dimensions, whereas many data domains do not admit a natural ordering.

Score-based generative models have largely sidestepped these issues, but they still have many hyperparameters that require manual tuning. To make generative techniques broadly accessible to everyone, and to foster interdisciplinary collaborations, it is important to build models that can work off-the-shelf for new settings without considerable domain expertise. It is also very valuable to develop open source software packages that ease the effort of implementing generative models for non-experts.

Generative modeling for science and engineering The ability of generative models to capture the underlying structure in complex data is broadly useful in many disciplines of science and engineering. As already mentioned in this dissertation, generative models can play a crucial role for solving ill-posed inverse problems, whose applications span medical imaging, remote sensing, radar, astronomy, geophysics, acoustics, oceanology, *etc.* Given the success of score-based generative modeling in medical image reconstruction (Chapter 8), it is exciting to explore its potential in other inverse problems. Two possible examples are seismic tomography for detecting mineral concentrations [See+99], and cryogenic electron tomography for analyzing virus structures [Sub+07]. Improving the solution of these inverse problems can reduce the cost of searching for valuable minerals, and speed up the research on biohazards.

Looking forward, we believe progress in generative modeling not only benefits immediate applications of data generation, like synthesizing large-scale datasets, generating high quality artwork, creating molecules for drug discovery and material design, but also deepens our understandings of the nature of data, and as a result, advances data-driven techniques for artificial intelligence as a whole.

Appendix A

Additional Proofs

A.1 Proofs for Chapter 3

A.1.1 Notations

Below we provide a summary of the most commonly used notations used in the proofs. First, we denote the data distribution as $p_d(\mathbf{x})$ and assume that the training/test data $\{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\}$ are i.i.d. samples of $p_d(\mathbf{x})$. The model is denoted as $p_m(\mathbf{x}; \boldsymbol{\theta})$, where $\boldsymbol{\theta}$ is restricted to a parameter space Θ . Note that $p_m(\mathbf{x}; \mathbf{v})$ can be *an unnormalized energy-based model*. We use \mathbf{v} to represent a random vector with the same dimension of input \mathbf{x} . This vector \mathbf{v} is often called the *projection vector*, and we use $p_{\mathbf{v}}$ to denote its distribution.

Next, we introduce several shorthand notations for quantities related to $p_m(\mathbf{x}; \boldsymbol{\theta})$ and $p_d(\mathbf{x})$. The log-likelihood $\log p_m(\mathbf{x}; \boldsymbol{\theta})$ and $\log p_d(\mathbf{x})$ are respectively denoted as $l_m(\mathbf{x}; \boldsymbol{\theta})$ and $l_d(\mathbf{x})$. The (Stein) score function $\nabla_{\mathbf{x}} \log p_m(\mathbf{x}; \boldsymbol{\theta})$ and $\nabla_{\mathbf{x}} \log p_d(\mathbf{x})$ are written as $\mathbf{s}_m(\mathbf{x}; \boldsymbol{\theta})$ and $\mathbf{s}_d(\mathbf{x})$, and finally the Hessian of $\log p_m(\mathbf{x}; \boldsymbol{\theta})$ w.r.t. \mathbf{x} is denoted as $\nabla_{\mathbf{x}} \mathbf{s}_m(\mathbf{x}; \boldsymbol{\theta})$.

We also adopt some convenient notations for collections. In particular, we use \mathbf{x}_1^N to denote a collection of N vectors $\{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\}$ and use \mathbf{v}_{11}^{NM} to denote $N \times M$ vectors $\{\mathbf{v}_{11}, \mathbf{v}_{12}, \dots, \mathbf{v}_{1M}, \mathbf{v}_{21}, \mathbf{v}_{22}, \dots, \mathbf{v}_{2M}, \dots, \mathbf{v}_{N1}, \mathbf{v}_{N2}, \dots, \mathbf{v}_{NM}\}$.

A.1.2 Basic Properties

The following regularity conditions are needed for integration by parts and identifiability.

Assumption A.1 (Regularity of score functions). *The model score function $\mathbf{s}_m(\mathbf{x})$ and data score function $\mathbf{s}_d(\mathbf{x})$ are both differentiable. They additionally satisfy $\mathbb{E}_{p_d}[\|\mathbf{s}_m(\mathbf{x})\|_2^2] < \infty$*

and $\mathbb{E}_{p_d}[\|s_d(\mathbf{x})\|_2^2] < \infty$.

Assumption A.2 (Regularity of projection vectors). *The projection vectors satisfy $\mathbb{E}_{p_{\mathbf{v}}}[\|\mathbf{v}\|_2^2] < \infty$, and $\mathbb{E}_{p_{\mathbf{v}}}[\mathbf{v}\mathbf{v}^\top] > 0$.*

Assumption A.3 (Boundary conditions). $\forall \boldsymbol{\theta} \in \Theta, \lim_{\|\mathbf{x}\| \rightarrow \infty} s_m(\mathbf{x}; \boldsymbol{\theta}) p_d(\mathbf{x}) = 0$.

Assumption A.4 (Identifiability). *The model family $\{p_m(\mathbf{x}; \boldsymbol{\theta}) \mid \boldsymbol{\theta} \in \Theta\}$ is well-specified, i.e., $p_d(\mathbf{x}) = p_m(\mathbf{x}; \boldsymbol{\theta}^*)$. Furthermore, $p_m(\mathbf{x}; \boldsymbol{\theta}) \neq p_m(\mathbf{x}; \boldsymbol{\theta}^*)$ whenever $\boldsymbol{\theta} \neq \boldsymbol{\theta}^*$.*

Assumption A.5 (Positiveness). $p_m(\mathbf{x}; \boldsymbol{\theta}) > 0, \forall \boldsymbol{\theta} \in \Theta$ and $\forall \mathbf{x}$.

Theorem 3.1. *Assume $s_m(\mathbf{x}; \boldsymbol{\theta})$, $s_d(\mathbf{x})$ and $p_{\mathbf{v}}$ satisfy some regularity conditions (Assumption A.1, Assumption A.2). Under proper boundary conditions (Assumption A.3), we have*

$$\begin{aligned} L(\boldsymbol{\theta}; p_{\mathbf{v}}) &:= \frac{1}{2} \mathbb{E}_{p_{\mathbf{v}}} \mathbb{E}_{p_d} \left[(\mathbf{v}^\top s_m(\mathbf{x}; \boldsymbol{\theta}) - \mathbf{v}^\top s_d(\mathbf{x}))^2 \right] \\ &= \mathbb{E}_{p_{\mathbf{v}}} \mathbb{E}_{p_d} \left[\mathbf{v}^\top \nabla_{\mathbf{x}} s_m(\mathbf{x}; \boldsymbol{\theta}) \mathbf{v} + \frac{1}{2} \left(\mathbf{v}^\top s_m(\mathbf{x}; \boldsymbol{\theta}) \right)^2 \right] + C, \end{aligned} \quad (\text{A.1})$$

where C is a constant w.r.t. $\boldsymbol{\theta}$.

Proof. The basic idea of this proof is similar to that of Theorem 1 in [HD05]. First, note that $L(\boldsymbol{\theta}, p_{\mathbf{v}})$ can be expanded to

$$\begin{aligned} L(\boldsymbol{\theta}, p_{\mathbf{v}}) &= \frac{1}{2} \mathbb{E}_{p_{\mathbf{v}}} \mathbb{E}_{p_d} \left[(\mathbf{v}^\top s_m(\mathbf{x}; \boldsymbol{\theta}) - \mathbf{v}^\top s_d(\mathbf{x}))^2 \right] \\ &\stackrel{(i)}{=} \frac{1}{2} \mathbb{E}_{p_{\mathbf{v}}} \mathbb{E}_{p_d} \left[(\mathbf{v}^\top s_m(\mathbf{x}; \boldsymbol{\theta}))^2 + (\mathbf{v}^\top s_d(\mathbf{x}))^2 - 2(\mathbf{v}^\top s_m(\mathbf{x}; \boldsymbol{\theta}))(\mathbf{v}^\top s_d(\mathbf{x}; \boldsymbol{\theta})) \right] \end{aligned} \quad (\text{A.2})$$

$$= \mathbb{E}_{p_{\mathbf{v}}} \mathbb{E}_{p_d} \left[-(\mathbf{v}^\top s_m(\mathbf{x}; \boldsymbol{\theta}))(\mathbf{v}^\top s_d(\mathbf{x}; \boldsymbol{\theta})) + \frac{1}{2} \left(\mathbf{v}^\top s_m(\mathbf{x}; \boldsymbol{\theta}) \right)^2 \right] + C, \quad (\text{A.3})$$

where (i) is due to the assumptions of bounded expectations. We have absorbed the second term in the bracket of Eq. (A.2) into C since it does not depend on $\boldsymbol{\theta}$. Now what we need to prove is

$$-\mathbb{E}_{p_{\mathbf{v}}} \mathbb{E}_{p_d}[(\mathbf{v}^\top s_m(\mathbf{x}; \boldsymbol{\theta}))(\mathbf{v}^\top s_d(\mathbf{x}; \boldsymbol{\theta}))] = \mathbb{E}_{p_{\mathbf{v}}} \mathbb{E}_{p_d}[\mathbf{v}^\top \nabla_{\mathbf{x}} s_m(\mathbf{x}; \boldsymbol{\theta}) \mathbf{v}]. \quad (\text{A.4})$$

This can be shown by first observing that

$$-\mathbb{E}_{p_{\mathbf{v}}} \mathbb{E}_{p_d}[(\mathbf{v}^\top s_m(\mathbf{x}; \boldsymbol{\theta}))(\mathbf{v}^\top s_d(\mathbf{x}; \boldsymbol{\theta}))]$$

$$\begin{aligned}
&= -\mathbb{E}_{p_{\mathbf{v}}} \int p_d(\mathbf{x})(\mathbf{v}^T \mathbf{s}_m(\mathbf{x}; \boldsymbol{\theta}))(\mathbf{v}^T \mathbf{s}_d(\mathbf{x}; \boldsymbol{\theta})) d\mathbf{x} \\
&= -\mathbb{E}_{p_{\mathbf{v}}} \int p_d(\mathbf{x})(\mathbf{v}^T \nabla_{\mathbf{x}} \log p_m(\mathbf{x}; \boldsymbol{\theta}))(\mathbf{v}^T \nabla_{\mathbf{x}} \log p_d(\mathbf{x})) d\mathbf{x} \\
&= -\mathbb{E}_{p_{\mathbf{v}}} \int (\mathbf{v}^T \nabla_{\mathbf{x}} \log p_m(\mathbf{x}; \boldsymbol{\theta}))(\mathbf{v}^T \nabla_{\mathbf{x}} p_d(\mathbf{x})) d\mathbf{x} \\
&= -\mathbb{E}_{p_{\mathbf{v}}} \sum_{i=1}^D \int (\mathbf{v}^T \nabla_{\mathbf{x}} \log p_m(\mathbf{x}; \boldsymbol{\theta})) v_i \frac{\partial p_d(\mathbf{x})}{\partial x_i} d\mathbf{x}, \tag{A.5}
\end{aligned}$$

where we assume $\mathbf{x} \in \mathbb{R}^D$. Then, applying multivariate integration by parts (*cf.*, Lemma 4 in [HD05]), we obtain

$$\begin{aligned}
&\left| \mathbb{E}_{p_{\mathbf{v}}} \sum_{i=1}^D \int (\mathbf{v}^T \mathbf{s}_m(\mathbf{x}; \boldsymbol{\theta})) v_i \frac{\partial p_d(\mathbf{x})}{\partial x_i} d\mathbf{x} + \mathbb{E}_{p_{\mathbf{v}}} \sum_{i=1}^D \int v_i p_d(\mathbf{x}) \mathbf{v}^T \frac{\partial \mathbf{s}_m(\mathbf{x}; \boldsymbol{\theta})}{\partial x_i} d\mathbf{x} \right| \\
&= \left| \mathbb{E}_{p_{\mathbf{v}}} \left[\sum_{i=1}^D \lim_{x_i \rightarrow \infty} (\mathbf{v}^T \mathbf{s}_m(\mathbf{x}; \boldsymbol{\theta})) v_i p_d(\mathbf{x}) - \sum_{i=1}^D \lim_{x_i \rightarrow -\infty} (\mathbf{v}^T \mathbf{s}_m(\mathbf{x}; \boldsymbol{\theta})) v_i p_d(\mathbf{x}) \right] \right| \\
&\leq \sum_{i=1}^D \lim_{x_i \rightarrow \infty} \sum_{j=1}^D \mathbb{E}_{p_{\mathbf{v}}} |v_i v_j| |\mathbf{s}_{m,j}(\mathbf{x}; \boldsymbol{\theta}) p_d(\mathbf{x})| + \sum_{i=1}^D \lim_{x_i \rightarrow -\infty} \sum_{j=1}^D \mathbb{E}_{p_{\mathbf{v}}} |v_i v_j| |\mathbf{s}_{m,j}(\mathbf{x}; \boldsymbol{\theta}) p_d(\mathbf{x})| \\
&\stackrel{(i)}{\leq} \sum_{i=1}^D \lim_{x_i \rightarrow \infty} \sum_{j=1}^D \sqrt{\mathbb{E}_{p_{\mathbf{v}}} v_i^2 \mathbb{E}_{p_{\mathbf{v}}} v_j^2} |\mathbf{s}_{m,j}(\mathbf{x}; \boldsymbol{\theta}) p_d(\mathbf{x})| \\
&\quad + \sum_{i=1}^D \lim_{x_i \rightarrow -\infty} \sum_{j=1}^D \sqrt{\mathbb{E}_{p_{\mathbf{v}}} v_i^2 \mathbb{E}_{p_{\mathbf{v}}} v_j^2} |\mathbf{s}_{m,j}(\mathbf{x}; \boldsymbol{\theta}) p_d(\mathbf{x})| \\
&\stackrel{(ii)}{=} 0,
\end{aligned}$$

where $\mathbf{s}_{m,j}(\mathbf{x}; \boldsymbol{\theta})$ denotes the j -th component of $\mathbf{s}_m(\mathbf{x}; \boldsymbol{\theta})$. In the above derivation, (i) is due to Cauchy-Schwarz inequality and (ii) is from the assumption that $\mathbb{E}_{p_{\mathbf{v}}} [\|\mathbf{v}\|^2] < \infty$ and $s(\mathbf{x}; \boldsymbol{\theta}) p_d(\mathbf{x})$ vanishes at infinity.

Now returning to Eq. (A.5), we have

$$\begin{aligned}
-\mathbb{E}_{p_{\mathbf{v}}} \sum_{i=1}^D \int (\mathbf{v}^T \nabla_{\mathbf{x}} \log p_m(\mathbf{x}; \boldsymbol{\theta})) v_i \frac{\partial p_d(\mathbf{x})}{\partial x_i} d\mathbf{x} &= \mathbb{E}_{p_{\mathbf{v}}} \sum_{i=1}^D \int v_i p_d(\mathbf{x}) \mathbf{v}^T \frac{\partial \mathbf{s}_m(\mathbf{x}; \boldsymbol{\theta})}{\partial x_i} d\mathbf{x} \\
&= \mathbb{E}_{p_{\mathbf{v}}} \int p_d(\mathbf{x}) \mathbf{v}^T \nabla_{\mathbf{x}} \mathbf{s}_m(\mathbf{x}; \boldsymbol{\theta}) \mathbf{v} d\mathbf{x},
\end{aligned}$$

which proves Eq. (A.4) and the proof is completed. \square

Lemma A.1. *Assume our model family is well-specified and identifiable (Assumption A.4). Assume further that the densities are all positive (Assumption A.5). When $p_{\mathbf{v}}$ satisfies some regularity conditions (Assumption A.2), we have*

$$L(\boldsymbol{\theta}; p_{\mathbf{v}}) = 0 \Leftrightarrow \boldsymbol{\theta} = \boldsymbol{\theta}^*.$$

Proof. First, since $p_d(\mathbf{x}) = p_m(\mathbf{x}; \boldsymbol{\theta}^*) > 0$, $L(\boldsymbol{\theta}; p_{\mathbf{v}}) = 0$ implies

$$\begin{aligned} & \frac{1}{2} \mathbb{E}_{p_{\mathbf{v}}} (\mathbf{v}^T (\mathbf{s}_m(\mathbf{x}; \boldsymbol{\theta}) - \mathbf{s}_d(\mathbf{x})))^2 = 0 \\ \Leftrightarrow & \mathbb{E}_{p_{\mathbf{v}}} \mathbf{v}^T (\mathbf{s}_m(\mathbf{x}; \boldsymbol{\theta}) - \mathbf{s}_d(\mathbf{x})) (\mathbf{s}_m(\mathbf{x}; \boldsymbol{\theta}) - \mathbf{s}_d(\mathbf{x}))^T \mathbf{v} = 0 \\ \Leftrightarrow & (\mathbf{s}_m(\mathbf{x}; \boldsymbol{\theta}) - \mathbf{s}_d(\mathbf{x}))^T \mathbb{E}_{p_{\mathbf{v}}} [\mathbf{v} \mathbf{v}^T] (\mathbf{s}_m(\mathbf{x}; \boldsymbol{\theta}) - \mathbf{s}_d(\mathbf{x})) = 0 \\ \stackrel{(i)}{\Leftrightarrow} & \mathbf{s}_m(\mathbf{x}; \boldsymbol{\theta}) - \mathbf{s}_d(\mathbf{x}) = 0 \\ \Leftrightarrow & \log p_m(\mathbf{x}; \boldsymbol{\theta}) = \log p_d(\mathbf{x}) + C \end{aligned}$$

where (i) holds because $\mathbb{E}_{p_{\mathbf{v}}} [\mathbf{v} \mathbf{v}^T]$ is positive definite. Because $p_m(\mathbf{x}; \boldsymbol{\theta})$ and $p_d(\mathbf{x})$ are normalized probability density functions, we have $p_m(\mathbf{x}; \boldsymbol{\theta}) = p_d(\mathbf{x})$. The identifiability assumption gives $\boldsymbol{\theta} = \boldsymbol{\theta}^*$. This concludes the left to right direction of the implication and the converse direction is trivial. \square

A.1.3 Consistency

In addition to the assumptions in Theorem 3.1 and Lemma A.1, we need the following regularity conditions to prove the consistency of $\hat{\boldsymbol{\theta}}_{N,M} := \arg \min_{\boldsymbol{\theta} \in \Theta} \hat{J}(\boldsymbol{\theta}; \mathbf{x}_1^N, \mathbf{v}_{11}^{NM})$.

Assumption A.6 (Compactness). *The parameter space Θ is compact.*

Assumption A.7 (Lipschitz continuity). *Both $\nabla_{\mathbf{x}} \mathbf{s}_m(\mathbf{x}; \boldsymbol{\theta})$ and $\mathbf{s}_m(\mathbf{x}; \boldsymbol{\theta}) \mathbf{s}_m(\mathbf{x}; \boldsymbol{\theta})^T$ are Lipschitz continuous in terms of Frobenious norm, i.e.,*

$$\forall \boldsymbol{\theta}_1 \in \Theta, \boldsymbol{\theta}_2 \in \Theta, \|\nabla_{\mathbf{x}} \mathbf{s}_m(\mathbf{x}; \boldsymbol{\theta}_1) - \nabla_{\mathbf{x}} \mathbf{s}_m(\mathbf{x}; \boldsymbol{\theta}_2)\|_F \leq L_1(\mathbf{x}) \|\boldsymbol{\theta}_1 - \boldsymbol{\theta}_2\|_2$$

and

$$\left\| \mathbf{s}_m(\mathbf{x}; \boldsymbol{\theta}_1) \mathbf{s}_m(\mathbf{x}; \boldsymbol{\theta}_1)^T - \mathbf{s}_m(\mathbf{x}; \boldsymbol{\theta}_2) \mathbf{s}_m(\mathbf{x}; \boldsymbol{\theta}_2)^T \right\|_F \leq L_2(\mathbf{x}) \|\boldsymbol{\theta}_1 - \boldsymbol{\theta}_2\|_2.$$

In addition, we require $\mathbb{E}_{p_d}[L_1^2(\mathbf{x})] < \infty$ and $\mathbb{E}_{p_d}[L_2^2(\mathbf{x})] < \infty$.

Assumption A.8 (Bounded moments of projection vectors). $\mathbb{E}_{p_{\mathbf{v}}}[\|\mathbf{v}\mathbf{v}^T\|_F^2] < \infty$.

Lemma A.2. Suppose $s_m(\mathbf{x}; \boldsymbol{\theta})$ is sufficiently smooth (Assumption A.7) and $p_{\mathbf{v}}$ has bounded higher-order moments (Assumption A.8). Let $f(\boldsymbol{\theta}; \mathbf{x}, \mathbf{v}) := \mathbf{v}^T \nabla_{\mathbf{x}} s_m(\mathbf{x}; \boldsymbol{\theta}) \mathbf{v} + \frac{1}{2} (\mathbf{v}^T s_m(\mathbf{x}; \boldsymbol{\theta}))^2$. Then $f(\boldsymbol{\theta}; \mathbf{x}, \mathbf{v})$ is Lipschitz continuous with constant $L(\mathbf{x}, \mathbf{v})$ and $\mathbb{E}_{p_d, p_{\mathbf{v}}} [L^2(\mathbf{x}, \mathbf{v})] < \infty$.

Proof. Let $A(\boldsymbol{\theta}) := \nabla_{\mathbf{x}} s_m(\mathbf{x}; \boldsymbol{\theta})$ and $B(\boldsymbol{\theta}) := s_m(\mathbf{x}; \boldsymbol{\theta}) s_m(\mathbf{x}; \boldsymbol{\theta})^T$. Consider $\boldsymbol{\theta}_1 \in \Theta, \boldsymbol{\theta}_2 \in \Theta$ and let D be the dimension of \mathbf{v} , we have

$$\begin{aligned} & |f(\boldsymbol{\theta}_1; \mathbf{x}, \mathbf{v}) - f(\boldsymbol{\theta}_2; \mathbf{x}, \mathbf{v})| \\ &= \sum_{i=1}^D \sum_{j=1}^D \left[v_i v_j (A(\boldsymbol{\theta}_1)_{ij} - A(\boldsymbol{\theta}_2)_{ij}) + \frac{1}{2} v_i v_j (B(\boldsymbol{\theta}_1)_{ij} - B(\boldsymbol{\theta}_2)_{ij}) \right] \\ &\stackrel{(i)}{\leqslant} \sqrt{\sum_{i=1}^D \sum_{j=1}^D v_i^2 v_j^2} \sqrt{\sum_{i=1}^D \sum_{j=1}^D \left[(A(\boldsymbol{\theta}_1)_{ij} - A(\boldsymbol{\theta}_2)_{ij}) + \frac{1}{2} (B(\boldsymbol{\theta}_1)_{ij} - B(\boldsymbol{\theta}_2)_{ij}) \right]^2} \\ &\stackrel{(ii)}{\leqslant} \sqrt{\sum_{i=1}^D \sum_{j=1}^D v_i^2 v_j^2} \sqrt{\sum_{i=1}^D \sum_{j=1}^D 2(A(\boldsymbol{\theta}_1)_{ij} - A(\boldsymbol{\theta}_2)_{ij})^2 + \frac{1}{2} (B(\boldsymbol{\theta}_1)_{ij} - B(\boldsymbol{\theta}_2)_{ij})^2} \\ &\stackrel{(iii)}{\leqslant} \sqrt{\sum_{i=1}^D \sum_{j=1}^D v_i^2 v_j^2} \sqrt{2L_1^2(\mathbf{x}) \|\boldsymbol{\theta}_1 - \boldsymbol{\theta}_2\|_2^2 + \frac{1}{2} L_2^2(\mathbf{x}) \|\boldsymbol{\theta}_1 - \boldsymbol{\theta}_2\|_2^2} \\ &= \sqrt{\sum_{i=1}^D \sum_{j=1}^D v_i^2 v_j^2} \sqrt{2L_1^2(\mathbf{x}) + \frac{1}{2} L_2^2(\mathbf{x})} \|\boldsymbol{\theta}_1 - \boldsymbol{\theta}_2\|_2, \end{aligned}$$

where (i) is Cauchy-Schwarz inequality, (ii) is Jensen's inequality, and (iii) is due to Assumption A.7. Now let

$$L(\mathbf{x}, \mathbf{v}) := \sqrt{\sum_{i=1}^D \sum_{j=1}^D v_i^2 v_j^2} \sqrt{2L_1^2(\mathbf{x}) + \frac{1}{2} L_2^2(\mathbf{x})}.$$

Then

$$\begin{aligned} \mathbb{E}_{p_d, p_{\mathbf{v}}} [L^2(\mathbf{x}, \mathbf{v})] &\stackrel{(i)}{=} \mathbb{E}_{p_{\mathbf{v}}} \left[\sum_{i=1}^D \sum_{j=1}^D v_i^2 v_j^2 \right] \mathbb{E}_{p_d} \left[2L_1^2(\mathbf{x}) + \frac{1}{2} L_2^2(\mathbf{x}) \right] \\ &\stackrel{(ii)}{<} \infty, \end{aligned}$$

where (i) results from the independence of \mathbf{v}, \mathbf{x} , and (ii) is due to Assumptions A.7 and A.8.

□

Lemma A.3 (Uniform convergence of the expected error). *Under Assumptions A.6 to A.8, we have*

$$\mathbb{E}_{p_{\mathbf{v}}, p_d} \left[\sup_{\boldsymbol{\theta} \in \Theta} |\hat{J}(\boldsymbol{\theta}; \mathbf{x}_1^N, \mathbf{v}_{11}^{NM}) - J(\boldsymbol{\theta}; p_{\mathbf{v}})| \right] \leq O \left(\text{diam}(\Theta) \sqrt{\frac{D}{N}} \right) \quad (\text{A.6})$$

where $\text{diam}(\cdot)$ denotes the diameter and D is the dimension of Θ .

Proof. The proof consists of 3 steps. First, we use the symmetrization trick to get rid of the inner expectation term $J(\boldsymbol{\theta}; p_{\mathbf{v}}) = \mathbb{E}_{p_{\mathbf{v}}, p_d} [\hat{J}(\boldsymbol{\theta}; \mathbf{x}_1^N, \mathbf{v}_{11}^{NM})]$. Second, we use chaining to get an upper bound that involves integration of the metric entropy. Finally, we upper bound the metric entropy to obtain the uniform convergence bound.

Step 1: From Jensen's inequality, we obtain

$$\begin{aligned} & \mathbb{E}_{p_{\mathbf{v}}, p_d} \left[\sup_{\boldsymbol{\theta} \in \Theta} |\hat{J}(\boldsymbol{\theta}; \mathbf{x}_1^N, \mathbf{v}_{11}^{NM}) - J(\boldsymbol{\theta}; p_{\mathbf{v}})| \right] \\ &= \mathbb{E}_{p_{\mathbf{v}}, p_d} \left[\sup_{\boldsymbol{\theta} \in \Theta} |\hat{J}(\boldsymbol{\theta}; \mathbf{x}_1^N, \mathbf{v}_{11}^{NM}) - \mathbb{E}_{p_{\mathbf{v}}, p_d} [\hat{J}(\boldsymbol{\theta}; \mathbf{x}_1^N, \mathbf{v}_{11}^{NM})]| \right] \\ &= \mathbb{E}_{p_{\mathbf{v}}, p_d} \left[\sup_{\boldsymbol{\theta} \in \Theta} |\hat{J}(\boldsymbol{\theta}; \mathbf{x}_1^N, \mathbf{v}_{11}^{NM}) - \mathbb{E}_{p_{\mathbf{v}}, p_d} [\hat{J}(\boldsymbol{\theta}; \mathbf{x}'_1^N, \mathbf{v}'_{11}^{NM})]| \right] \\ &\leq \mathbb{E}_{p_{\mathbf{v}}, p_d} \left[\sup_{\boldsymbol{\theta} \in \Theta} |\hat{J}(\boldsymbol{\theta}; \mathbf{x}_1^N, \mathbf{v}_{11}^{NM}) - \hat{J}(\boldsymbol{\theta}; \mathbf{x}'_1^N, \mathbf{v}'_{11}^{NM})| \right], \end{aligned}$$

where $\mathbf{x}'_1^N, \mathbf{v}'_{11}^{NM}$ are independent copies of \mathbf{x}_1^N and \mathbf{v}_{11}^{NM} . Let $\{\epsilon_i\}_{i=1}^N$ be a set of independent Rademacher random variables, we have

$$\begin{aligned} & \mathbb{E}_{p_{\mathbf{v}}, p_d} \left[\sup_{\boldsymbol{\theta} \in \Theta} |\hat{J}(\boldsymbol{\theta}; \mathbf{x}_1^N, \mathbf{v}_{11}^{NM}) - \hat{J}(\boldsymbol{\theta}; \mathbf{x}'_1^N, \mathbf{v}'_{11}^{NM})| \right] \\ &= \mathbb{E}_{p_{\mathbf{v}}, p_d} \left[\sup_{\boldsymbol{\theta} \in \Theta} \frac{1}{N} \frac{1}{M} \left| \sum_{i=1}^N \sum_{j=1}^M \underbrace{\mathbf{v}_{ij}^\top \nabla_{\mathbf{x}} s_m(\mathbf{x}_i; \boldsymbol{\theta}) \mathbf{v}_{ij} + \frac{1}{2} \left(\mathbf{v}_{ij}^\top s_m(\mathbf{x}_i; \boldsymbol{\theta}) \right)^2}_{:= f(\boldsymbol{\theta}; \mathbf{x}_i, \mathbf{v}_{ij})} \right| \right] \\ &\quad - \left[\underbrace{\mathbf{v}'_{ij}^\top \nabla_{\mathbf{x}} s_m(\mathbf{x}'_i; \boldsymbol{\theta}) \mathbf{v}'_{ij} + \frac{1}{2} \left(\mathbf{v}'_{ij}^\top s_m(\mathbf{x}'_i; \boldsymbol{\theta}) \right)^2}_{:= f(\boldsymbol{\theta}; \mathbf{x}'_i, \mathbf{v}'_{ij})} \right] \Big| \\ &= \mathbb{E}_{p_{\mathbf{v}}, p_d} \left[\sup_{\boldsymbol{\theta} \in \Theta} \frac{1}{N} \frac{1}{M} \left| \sum_{i=1}^N \sum_{j=1}^M f(\boldsymbol{\theta}; \mathbf{x}_i, \mathbf{v}_{ij}) - f(\boldsymbol{\theta}; \mathbf{x}'_i, \mathbf{v}'_{ij}) \right| \right] \end{aligned} \quad (\text{A.7})$$

$$\begin{aligned}
& \stackrel{(i)}{=} \mathbb{E} \left[\sup_{\boldsymbol{\theta} \in \Theta} \left| \frac{1}{N} \frac{1}{M} \sum_{i=1}^N \sum_{j=1}^M \epsilon_i (f(\boldsymbol{\theta}; \mathbf{x}_i, \mathbf{v}_{ij}) - f(\boldsymbol{\theta}; \mathbf{x}'_i, \mathbf{v}'_{ij})) \right| \right] \\
& \stackrel{(ii)}{\leq} \mathbb{E} \left[\sup_{\boldsymbol{\theta} \in \Theta} \left| \frac{1}{N} \frac{1}{M} \sum_{i=1}^N \sum_{j=1}^M \epsilon_i f(\boldsymbol{\theta}; \mathbf{x}_i, \mathbf{v}_{ij}) \right| \right] + \mathbb{E} \left[\sup_{\boldsymbol{\theta} \in \Theta} \left| \frac{1}{N} \frac{1}{M} \sum_{i=1}^N \sum_{j=1}^M \epsilon_i f(\boldsymbol{\theta}; \mathbf{x}'_i, \mathbf{v}'_{ij}) \right| \right] \\
& = 2 \mathbb{E} \left[\sup_{\boldsymbol{\theta} \in \Theta} \left| \frac{1}{N} \frac{1}{M} \sum_{i=1}^N \sum_{j=1}^M \epsilon_i f(\boldsymbol{\theta}; \mathbf{x}_i, \mathbf{v}_{ij}) \right| \right], \tag{A.8}
\end{aligned}$$

where (i) is because the quantity is symmetric about 0 in distribution, and (ii) is due to Jensen's inequality.

Step 2: First note that given $\mathbf{x}_i, \mathbf{v}_{ij}$, $\frac{\epsilon_i}{M} \sum_{j=1}^M f(\boldsymbol{\theta}; \mathbf{x}_i, \mathbf{v}_{ij})$ is a zero-mean sub-Gaussian process w.r.t. $\boldsymbol{\theta}$. This can be observed from

$$\begin{aligned}
& \mathbb{E}_{\epsilon_i} \left[e^{\lambda \frac{\epsilon_i}{M} \sum_{j=1}^M [f(\boldsymbol{\theta}_1; \mathbf{x}_i, \mathbf{v}_{ij}) - f(\boldsymbol{\theta}_2; \mathbf{x}_i, \mathbf{v}_{ij})]} \right] \\
& \stackrel{(i)}{\leq} \exp \left\{ \frac{\lambda^2}{2M^2} \left(\sum_{j=1}^M [f(\boldsymbol{\theta}_1; \mathbf{x}_i, \mathbf{v}_{ij}) - f(\boldsymbol{\theta}_2; \mathbf{x}_i, \mathbf{v}_{ij})] \right)^2 \right\} \\
& \stackrel{(ii)}{\leq} \exp \left\{ \frac{\lambda^2}{2M^2} M \sum_{j=1}^M [f(\boldsymbol{\theta}_1; \mathbf{x}_i, \mathbf{v}_{ij}) - f(\boldsymbol{\theta}_2; \mathbf{x}_i, \mathbf{v}_{ij})]^2 \right\} \\
& \stackrel{(iii)}{\leq} \exp \left\{ \frac{\lambda^2}{2M} \sum_{j=1}^M L^2(\mathbf{x}_i, \mathbf{v}_{ij}) \|\boldsymbol{\theta}_1 - \boldsymbol{\theta}_2\|^2 \right\},
\end{aligned}$$

where (i) holds because ϵ_i is a 1-sub-Gaussian random variable, (ii) is from Cauchy-Schwarz inequality and (iii) is due to Lemma A.2. As a result, $\frac{1}{N} \frac{1}{M} \sum_{i=1}^N \sum_{j=1}^M \epsilon_i f(\boldsymbol{\theta}; \mathbf{x}_i, \mathbf{v}_{ij})$ is a zero-mean sub-Gaussian random process with metric

$$d(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2) = \frac{1}{\sqrt{NM}} \sqrt{\frac{1}{NM} \sum_{i=1}^N \sum_{j=1}^M L^2(\mathbf{x}_i, \mathbf{v}_{ij}) \|\boldsymbol{\theta}_1 - \boldsymbol{\theta}_2\|^2}.$$

Since Θ is compact, the diameter of Θ with respect to the Euclidean norm $\|\cdot\|_2$ is finite and we denote it as $\text{diam}(\Theta) < \infty$. Then, Dudley's entropy integral [Dud67] gives

$$\mathbb{E} \left[\sup_{\boldsymbol{\theta} \in \Theta} \left| \frac{1}{N} \frac{1}{M} \sum_{i=1}^N \sum_{j=1}^M \epsilon_i f(\boldsymbol{\theta}; \mathbf{x}_i, \mathbf{v}_{ij}) \right| \right]$$

$$\leq O(1) \mathbb{E} \left[\int_0^{\frac{1}{\sqrt{N}} \sqrt{\frac{1}{NM} \sum_{i=1}^N \sum_{j=1}^M L^2(\mathbf{x}_i, \mathbf{v}_{ij})} \operatorname{diam}(\Theta)} \sqrt{\log N(\Theta, d, \epsilon)} d\epsilon \right]. \quad (\text{A.9})$$

Here $\log N(\Theta, d, \epsilon)$ is the metric entropy of Θ with metric

$$d(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2) = \frac{1}{\sqrt{N}} \sqrt{\frac{1}{NM} \sum_{i=1}^N \sum_{j=1}^M L^2(\mathbf{x}_i, \mathbf{v}_{ij}) \|\boldsymbol{\theta}_1 - \boldsymbol{\theta}_2\|_2}$$

and size ϵ .

Step 3: When the dimension of $\boldsymbol{\theta} \in \Theta$ is D , it is known that the ϵ -covering number of Θ with Euclidean distance is

$$N(\Theta, \|\cdot\|, \epsilon) \leq \left(1 + \frac{\operatorname{diam}(\Theta)}{\epsilon}\right)^D.$$

Therefore, $N(\Theta, d, \epsilon)$ can be bounded by

$$N(\Theta, d, \epsilon) \leq \left(1 + \sqrt{\frac{1}{NM} \sum_{i=1}^N \sum_{j=1}^M L^2(\mathbf{x}_i, \mathbf{v}_{ij})} \frac{\operatorname{diam}(\Theta)}{\sqrt{N}\epsilon}\right)^D.$$

Hence, the metric integral can be bounded

$$\begin{aligned} & \int_0^{\frac{1}{\sqrt{N}} \sqrt{\frac{1}{NM} \sum_{i=1}^N \sum_{j=1}^M L^2(\mathbf{x}_i, \mathbf{v}_{ij})} \operatorname{diam}(\Theta)} \sqrt{\log N(\Theta, d, \epsilon)} d\epsilon \\ & \leq \int_0^{\frac{1}{\sqrt{N}} \sqrt{\frac{1}{NM} \sum_{i=1}^N \sum_{j=1}^M L^2(\mathbf{x}_i, \mathbf{v}_{ij})} \operatorname{diam}(\Theta)} \sqrt{D \log \left(1 + \sqrt{\frac{1}{NM} \sum_{i=1}^N \sum_{j=1}^M L^2(\mathbf{x}_i, \mathbf{v}_{ij})} \frac{\operatorname{diam}(\Theta)}{\sqrt{N}\epsilon}\right)} d\epsilon \\ & \leq \int_0^{\frac{1}{\sqrt{N}} \sqrt{\frac{1}{NM} \sum_{i=1}^N \sum_{j=1}^M L^2(\mathbf{x}_i, \mathbf{v}_{ij})} \operatorname{diam}(\Theta)} \sqrt{\sqrt{\frac{1}{NM} \sum_{i=1}^N \sum_{j=1}^M L^2(\mathbf{x}_i, \mathbf{v}_{ij})} \frac{D \operatorname{diam}(\Theta)}{\sqrt{N}\epsilon}} d\epsilon \\ & = 2 \sqrt{\frac{1}{NM} \sum_{i=1}^N \sum_{j=1}^M L^2(\mathbf{x}_i, \mathbf{v}_{ij})} \sqrt{\frac{D}{N}} \operatorname{diam}(\Theta) \end{aligned} \quad (\text{A.10})$$

Finally, combining Eq. (A.8), Eq. (A.9) and Eq. (A.10) gives us

$$\mathbb{E}_{p_{\mathbf{v}}, p_d} \left[\sup_{\boldsymbol{\theta} \in \Theta} |\hat{J}(\boldsymbol{\theta}; \mathbf{x}_1^N, \mathbf{v}_{11}^{NM}) - J(\boldsymbol{\theta}; p_{\mathbf{v}})| \right]$$

$$\begin{aligned}
&\leq 4O(1)\mathbb{E}_{p_{\mathbf{v}}, p_d} \left[\sqrt{\frac{1}{NM} \sum_{i=1}^N \sum_{j=1}^M L^2(\mathbf{x}_i, \mathbf{v}_{ij})} \sqrt{\frac{D}{N}} \text{diam}(\Theta) \right] \\
&\stackrel{(i)}{\leq} O(1) \sqrt{\frac{D}{N}} \text{diam}(\Theta) \sqrt{\mathbb{E}_{p_{\mathbf{v}}, p_d} \left[\frac{1}{NM} \sum_{i=1}^N \sum_{j=1}^M L^2(\mathbf{x}_i, \mathbf{v}_{ij}) \right]} \\
&\stackrel{(ii)}{\leq} O(1) \text{diam}(\Theta) \sqrt{\frac{D}{N}},
\end{aligned}$$

where (i) is due to Jensen's inequality and (ii) results from $\mathbb{E}[L^2(\mathbf{x}, \mathbf{v})] < \infty$, and the compactness of Θ guarantees that the bound is finite. \square

Theorem 3.2. Suppose all the previous assumptions hold (Assumptions A.1 to A.8). Assume further the conditions of Theorem 3.1 and Lemma A.1 are satisfied. Let $\boldsymbol{\theta}^*$ be the true parameter of the data distribution, and $\hat{\boldsymbol{\theta}}_{N,M}$ be the empirical estimator defined by

$$\hat{\boldsymbol{\theta}}_{N,M} := \arg \min_{\boldsymbol{\theta} \in \Theta} \hat{J}(\boldsymbol{\theta}; \mathbf{x}_1^N, \mathbf{v}_{11}^{NM})$$

Then, $\hat{\boldsymbol{\theta}}_{N,M}$ is consistent, meaning that

$$\hat{\boldsymbol{\theta}}_{N,M} \xrightarrow{p} \boldsymbol{\theta}^*$$

as $N \rightarrow \infty$.

Proof. Note that Theorem 3.1 and Lemma A.1 together imply that $\boldsymbol{\theta}^* = \arg \min_{\boldsymbol{\theta} \in \Theta} J(\boldsymbol{\theta}; p_{\mathbf{v}})$. Then, we will show $J(\hat{\boldsymbol{\theta}}_{N,M}; p_{\mathbf{v}}) \xrightarrow{p} J(\boldsymbol{\theta}^*; p_{\mathbf{v}})$ when $N \rightarrow \infty$. This can be done by noticing

$$J(\hat{\boldsymbol{\theta}}_{N,M}; p_{\mathbf{v}}) - J(\boldsymbol{\theta}^*; p_{\mathbf{v}}) = J(\hat{\boldsymbol{\theta}}_{N,M}; p_{\mathbf{v}}) - \hat{J}(\hat{\boldsymbol{\theta}}_{N,M}; \mathbf{x}_1^N, \mathbf{v}_{11}^{NM}) \quad (\text{A.11})$$

$$+ \hat{J}(\hat{\boldsymbol{\theta}}_{N,M}; \mathbf{x}_1^N, \mathbf{v}_{11}^{NM}) - \hat{J}(\boldsymbol{\theta}^*; \mathbf{x}_1^N, \mathbf{v}_{11}^{NM})$$

$$+ \hat{J}(\boldsymbol{\theta}^*; \mathbf{x}_1^N, \mathbf{v}_{11}^{NM}) - J(\boldsymbol{\theta}^*; p_{\mathbf{v}})$$

$$\leq \sup_{\boldsymbol{\theta} \in \Theta} |\hat{J}(\boldsymbol{\theta}; \mathbf{x}_1^N, \mathbf{v}_{11}^{NM}) - J(\boldsymbol{\theta}; p_{\mathbf{v}})| + |\hat{J}(\boldsymbol{\theta}^*; \mathbf{x}_1^N, \mathbf{v}_{11}^{NM}) - J(\boldsymbol{\theta}^*; p_{\mathbf{v}})|$$

$$\leq 2 \sup_{\boldsymbol{\theta} \in \Theta} |\hat{J}(\boldsymbol{\theta}; \mathbf{x}_1^N, \mathbf{v}_{11}^{NM}) - J(\boldsymbol{\theta}; p_{\mathbf{v}})| \quad (\text{A.12})$$

We can easily conclude that Eq. (A.12) is $o_p(1)$ with the help of Lemma A.3, because

$$P \left(\sup_{\boldsymbol{\theta} \in \Theta} |\hat{J}(\boldsymbol{\theta}; \mathbf{x}_1^N, \mathbf{v}_{11}^{NM}) - J(\boldsymbol{\theta}; p_{\mathbf{v}})| > t \right)$$

$$\leq \mathbb{E} \left[\sup_{\boldsymbol{\theta} \in \Theta} |\hat{J}(\boldsymbol{\theta}; \mathbf{x}_1^N, \mathbf{v}_{11}^{NM}) - J(\boldsymbol{\theta}; p_{\mathbf{v}})| \right] / t \leq O(1) \sqrt{\frac{1}{Nt^2}} \rightarrow 0,$$

as $N \rightarrow \infty$. From Lemma A.1 we also have $L(\hat{\boldsymbol{\theta}}_{N,M}; p_{\mathbf{v}}) - L(\boldsymbol{\theta}^*; p_{\mathbf{v}}) > 0$ if $\hat{\boldsymbol{\theta}}_{N,M} \neq \boldsymbol{\theta}$. As shown by Theorem 3.1, this is the same as $J(\hat{\boldsymbol{\theta}}_{N,M}; p_{\mathbf{v}}) - J(\boldsymbol{\theta}^*; p_{\mathbf{v}}) > 0$ if $\hat{\boldsymbol{\theta}}_{N,M} \neq \boldsymbol{\theta}$. Therefore Eq. (A.12) = $o_p(1)$ gives $J(\hat{\boldsymbol{\theta}}_{N,M}; p_{\mathbf{v}}) \xrightarrow{p} J(\boldsymbol{\theta}^*; p_{\mathbf{v}})$.

Next, we show $\hat{\boldsymbol{\theta}}_{N,M} \xrightarrow{p} \boldsymbol{\theta}^*$. This can be inferred from $J(\hat{\boldsymbol{\theta}}_{N,M}; p_{\mathbf{v}}) \xrightarrow{p} J(\boldsymbol{\theta}^*; p_{\mathbf{v}})$ because $J(\boldsymbol{\theta}; p_{\mathbf{v}})$ is continuous (Assumption A.7) and Θ is compact (Assumption A.6). The proof is reductio ad absurdum. Specifically, assume that $\hat{\boldsymbol{\theta}}_{N,M} \not\xrightarrow{p} \boldsymbol{\theta}^*$. We know $\exists \epsilon > 0, \delta > 0, \forall K > 0, \exists N > K, M > 0$ such that $P(\|\hat{\boldsymbol{\theta}}_{N,M} - \boldsymbol{\theta}^*\| \geq \epsilon) \geq \delta$. Note that $J(\boldsymbol{\theta}; p_{\mathbf{v}}) = \mathbb{E}[f(\boldsymbol{\theta}; \mathbf{x}, \mathbf{v})]$, $f(\boldsymbol{\theta}; \mathbf{x}, \mathbf{v})$ is Lipschitz continuous and $\mathbb{E}[L(\boldsymbol{\theta}; \mathbf{x}, \mathbf{v})] \leq \sqrt{\mathbb{E}[L^2(\boldsymbol{\theta}; \mathbf{x}, \mathbf{v})]} < \infty$. This implies that $J(\boldsymbol{\theta}; p_{\mathbf{v}})$ is continuous w.r.t. $\boldsymbol{\theta}$. Since Θ is compact and $J(\boldsymbol{\theta}; p_{\mathbf{v}})$ is continuous, we can define a compact set $\mathcal{S}_\epsilon := \{\boldsymbol{\theta} \in \Theta \mid \|\boldsymbol{\theta} - \boldsymbol{\theta}^*\| \geq \epsilon\}$ and let $\boldsymbol{\theta}_{\mathcal{S}_\epsilon} := \arg \min_{\boldsymbol{\theta} \in \mathcal{S}_\epsilon} J(\boldsymbol{\theta}; p_{\mathbf{v}})$. Observe that

$$\begin{aligned} p(J(\hat{\boldsymbol{\theta}}_{N,M}; p_{\mathbf{v}}) \geq J(\boldsymbol{\theta}_{\mathcal{S}_\epsilon}; p_{\mathbf{v}})) &= p(|J(\hat{\boldsymbol{\theta}}_{N,M}; p_{\mathbf{v}}) - J(\boldsymbol{\theta}^*; p_{\mathbf{v}})| \geq J(\boldsymbol{\theta}_{\mathcal{S}_\epsilon}; p_{\mathbf{v}}) - J(\boldsymbol{\theta}^*; p_{\mathbf{v}})) \\ &\geq p(\|\hat{\boldsymbol{\theta}}_{N,M} - \boldsymbol{\theta}^*\| \geq \epsilon) \geq \delta. \end{aligned}$$

However, the fact that $p(|J(\hat{\boldsymbol{\theta}}_{N,M}; p_{\mathbf{v}}) - J(\boldsymbol{\theta}^*; p_{\mathbf{v}})| \geq J(\boldsymbol{\theta}_{\mathcal{S}_\epsilon}; p_{\mathbf{v}}) - J(\boldsymbol{\theta}^*; p_{\mathbf{v}})) \geq \delta$ holds for arbitrarily large N contradicts $J(\hat{\boldsymbol{\theta}}_{N,M}; p_{\mathbf{v}}) \xrightarrow{p} J(\boldsymbol{\theta}^*; p_{\mathbf{v}})$. \square

A.1.4 Asymptotic Normality

Notations To simplify notations we use $\partial_i \partial_j h(\cdot) := (\nabla_{\mathbf{x}}^2 h(\cdot))_{ij}$, $\partial_i h(\cdot) := (\nabla_{\mathbf{x}} h(\cdot))_i$, and denote $\nabla_{\mathbf{x}} h(\mathbf{x})|_{\mathbf{x}=\mathbf{x}'}$ as $\nabla_{\mathbf{x}} h(\mathbf{x}')$. Here $h(\cdot)$ denotes some arbitrary function. Let $l_m := \log p_m(\mathbf{x}; \boldsymbol{\theta})$, $l_m(\mathbf{x}; \boldsymbol{\theta}) := \log p_m(\mathbf{x}; \boldsymbol{\theta})$ and further adopt the following notations

$$\begin{aligned} J(\boldsymbol{\theta}) &:= \mathbb{E}_{p_d} \left[\text{tr}(\nabla_{\mathbf{x}} s_m(\mathbf{x}; \boldsymbol{\theta})) + \frac{1}{2} \|s_m(\mathbf{x}; \boldsymbol{\theta})\|_2^2 \right] \\ f(\boldsymbol{\theta}; \mathbf{x}, \mathbf{v}) &:= \mathbf{v}^T \nabla_{\mathbf{x}} s_m(\mathbf{x}; \boldsymbol{\theta}) \mathbf{v} + \frac{1}{2} (\mathbf{v}^T s_m(\mathbf{x}; \boldsymbol{\theta}))^2 \\ f(\boldsymbol{\theta}; \mathbf{x}, \mathbf{v}_1^M) &:= \frac{1}{M} \sum_{j=1}^M f(\boldsymbol{\theta}; \mathbf{x}, \mathbf{v}_j) = \frac{1}{M} \sum_{j=1}^M \mathbf{v}_j^T \nabla_{\mathbf{x}} s_m(\mathbf{x}; \boldsymbol{\theta}) \mathbf{v}_j + \frac{1}{2} (\mathbf{v}_j^T s_m(\mathbf{x}; \boldsymbol{\theta}))^2 \\ f(\boldsymbol{\theta}; \mathbf{x}) &:= \text{tr}(\nabla_{\mathbf{x}} s_m(\mathbf{x}; \boldsymbol{\theta})) + \frac{1}{2} \|s_m(\mathbf{x}; \boldsymbol{\theta})\|_2^2 \\ \Sigma_{ij} &:= (\mathbb{E}_{p_{\mathbf{v}}} [\mathbf{v} \mathbf{v}^T])_{ij} \\ \mathfrak{S}_{ijpq} &:= \mathbb{E}_{p_{\mathbf{v}}} [v_i v_j v_p v_q] \end{aligned}$$

$$\begin{aligned}\mathfrak{V}_{ijpq} &:= \mathbb{E}_{p_d} \left[\left(\nabla_{\boldsymbol{\theta}} \partial_i \partial_j l_m + \frac{1}{2} \nabla_{\boldsymbol{\theta}} (\partial_i l_m \partial_j l_m) \right) \left(\nabla_{\boldsymbol{\theta}} \partial_p \partial_q l_m + \frac{1}{2} \nabla_{\boldsymbol{\theta}} (\partial_p l_m \partial_q l_m) \right)^T \right] \Big|_{\boldsymbol{\theta}=\boldsymbol{\theta}^*} \\ V_{ij} &:= \mathfrak{V}_{iijj} \\ W_{ij} &:= \mathfrak{V}_{ijij}\end{aligned}$$

For the proof of asymptotic normality we need the following extra assumptions.

Assumption A.9 (Lipschitz smoothness on second derivatives). *For $\boldsymbol{\theta}_1, \boldsymbol{\theta}_2$ near $\boldsymbol{\theta}^*$, and $\forall i, j$,*

$$\begin{aligned}\|\nabla_{\boldsymbol{\theta}}^2 \partial_i \partial_j l_m(\mathbf{x}; \boldsymbol{\theta}_1) - \nabla_{\boldsymbol{\theta}}^2 \partial_i \partial_j l_m(\mathbf{x}; \boldsymbol{\theta}_2)\|_F &\leq M_{ij}(\mathbf{x}) \|\boldsymbol{\theta}_1 - \boldsymbol{\theta}_2\|_2 \\ \|\nabla_{\boldsymbol{\theta}}^2 \partial_i l_m(\mathbf{x}; \boldsymbol{\theta}_1) \partial_j l_m(\mathbf{x}; \boldsymbol{\theta}_1) - \nabla_{\boldsymbol{\theta}}^2 \partial_i l_m(\mathbf{x}; \boldsymbol{\theta}_2) \partial_j l_m(\mathbf{x}; \boldsymbol{\theta}_2)\|_F &\leq N_{ij}(\mathbf{x}) \|\boldsymbol{\theta}_1 - \boldsymbol{\theta}_2\|_2\end{aligned}$$

and

$$\mathbb{E}_{p_d}[M_{ij}^2(\mathbf{x})] < \infty, \quad \mathbb{E}_{p_d}[N_{ij}^2(\mathbf{x})] < \infty, \quad \forall i, j.$$

Lemma A.4. *Suppose $l_m(\mathbf{x}; \boldsymbol{\theta})$ is sufficiently smooth (Assumption A.9) and $p_{\mathbf{v}}$ has bounded moments (Assumption A.2 and Assumption A.8). Let*

$$\nabla_{\boldsymbol{\theta}}^2 f(\boldsymbol{\theta}; \mathbf{x}, \mathbf{v}_1^M) := \frac{1}{M} \sum_{i=1}^M \nabla_{\boldsymbol{\theta}}^2 \mathbf{v}_i^T \nabla_{\mathbf{x}} s_m(\mathbf{x}; \boldsymbol{\theta}) \mathbf{v}_i + \frac{1}{2} \nabla_{\boldsymbol{\theta}}^2 (\mathbf{v}_i^T s_m(\mathbf{x}; \boldsymbol{\theta}))^2.$$

Then $\nabla_{\boldsymbol{\theta}}^2 f(\boldsymbol{\theta}; \mathbf{x}, \mathbf{v})$ is Lipschitz continuous, i.e., for $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$ close to $\boldsymbol{\theta}^$, there exists a Lipschitz constant $L(\mathbf{x}, \mathbf{v}_1^M)$ such that*

$$\|\nabla_{\boldsymbol{\theta}}^2 f(\boldsymbol{\theta}_1; \mathbf{x}, \mathbf{v}_1^M) - \nabla_{\boldsymbol{\theta}}^2 f(\boldsymbol{\theta}_2; \mathbf{x}, \mathbf{v}_1^M)\|_F \leq L(\mathbf{x}, \mathbf{v}_1^M) \|\boldsymbol{\theta}_1 - \boldsymbol{\theta}_2\|_2,$$

and $\mathbb{E}_{p_d, p_{\mathbf{v}}}[L^2(\mathbf{x}, \mathbf{v}_1^M)] < \infty$.

Proof. First, we write out $\nabla_{\boldsymbol{\theta}}^2 f(\boldsymbol{\theta}_1; \mathbf{x}, \mathbf{v}_1^M) - \nabla_{\boldsymbol{\theta}}^2 f(\boldsymbol{\theta}_2; \mathbf{x}, \mathbf{v}_1^M)$ according to the definitions. Let $A_{ij}(\boldsymbol{\theta}) := \nabla_{\boldsymbol{\theta}}^2 \partial_i \partial_j l_m(\mathbf{x}; \boldsymbol{\theta})$ and $B_{ij}(\boldsymbol{\theta}) := \nabla_{\boldsymbol{\theta}}^2 \partial_i l_m(\mathbf{x}; \boldsymbol{\theta}) \partial_j l_m(\mathbf{x}; \boldsymbol{\theta})$. Then,

$$\begin{aligned}\nabla_{\boldsymbol{\theta}}^2 f(\boldsymbol{\theta}_1; \mathbf{x}, \mathbf{v}_1^M) - \nabla_{\boldsymbol{\theta}}^2 f(\boldsymbol{\theta}_2; \mathbf{x}, \mathbf{v}_1^M) &= \frac{1}{M} \sum_{i,j,k} v_{k,i} v_{k,j} \left[A_{ij}(\boldsymbol{\theta}_1) - A_{ij}(\boldsymbol{\theta}_2) + \frac{1}{2} (B_{ij}(\boldsymbol{\theta}_1) - B_{ij}(\boldsymbol{\theta}_2)) \right].\end{aligned}$$

Then, Cauchy-Schwarz and Jensen's inequality give

$$\begin{aligned}
& \|\nabla_{\boldsymbol{\theta}}^2 f(\boldsymbol{\theta}_1; \mathbf{x}, \mathbf{v}_1^M) - \nabla_{\boldsymbol{\theta}}^2 f(\boldsymbol{\theta}_2; \mathbf{x}, \mathbf{v}_1^M)\|_F^2 \\
&= \sum_{l,m} \left(\frac{1}{M} \sum_{i,j,k} v_{k,i} v_{k,j} \left[A_{ij}(\boldsymbol{\theta}_1)_{lm} - A_{ij}(\boldsymbol{\theta}_2)_{lm} + \frac{1}{2} (B_{ij}(\boldsymbol{\theta}_1)_{lm} - B_{ij}(\boldsymbol{\theta}_2)_{lm}) \right] \right)^2 \\
&\leq \sum_{l,m} \left(\frac{1}{M} \sqrt{\sum_{i,j} \left(\sum_k v_{k,i} v_{k,j} \right)^2} \cdot \sqrt{\sum_{i,j} \left[A_{ij}(\boldsymbol{\theta}_1)_{lm} - A_{ij}(\boldsymbol{\theta}_2)_{lm} + \frac{1}{2} (B_{ij}(\boldsymbol{\theta}_1)_{lm} - B_{ij}(\boldsymbol{\theta}_2)_{lm}) \right]^2} \right)^2 \\
&\leq \sum_{l,m} \frac{1}{M^2} \left(\sum_{i,j} \left(\sum_k v_{k,i} v_{k,j} \right)^2 \right) \\
&\quad \cdot \left(\sum_{i,j} \left(2 \left[A_{ij}(\boldsymbol{\theta}_1)_{lm} - A_{ij}(\boldsymbol{\theta}_2)_{lm} \right]^2 + \frac{1}{2} \left[(B_{ij}(\boldsymbol{\theta}_1)_{lm} - B_{ij}(\boldsymbol{\theta}_2)_{lm}) \right]^2 \right) \right) \\
&= \frac{1}{M^2} \left(\sum_{i,j} \left(\sum_k v_{k,i} v_{k,j} \right)^2 \right) \left(\sum_{i,j} \left(\sum_{l,m} 2 \left[A_{ij}(\boldsymbol{\theta}_1)_{lm} - A_{ij}(\boldsymbol{\theta}_2)_{lm} \right]^2 \right. \right. \\
&\quad \left. \left. + \sum_{l,m} \frac{1}{2} \left[(B_{ij}(\boldsymbol{\theta}_1)_{lm} - B_{ij}(\boldsymbol{\theta}_2)_{lm}) \right]^2 \right) \right) \\
&= \frac{1}{M^2} \left(\sum_{i,j,p,q} v_{p,i} v_{p,j} v_{q,i} v_{q,j} \right) \left(\sum_{i,j} \left(2 \|A_{ij}(\boldsymbol{\theta}_1) - A_{ij}(\boldsymbol{\theta}_2)\|_F^2 + \frac{1}{2} \|V_{ij}(\boldsymbol{\theta}_1) - V_{ij}(\boldsymbol{\theta}_2)\|_F^2 \right) \right) \\
&\leq \underbrace{\frac{1}{M^2} \left(\sum_{i,j,p,q} v_{p,i} v_{p,j} v_{q,i} v_{q,j} \right) \left(\sum_{i,j} \left(2M_{ij}^2 + \frac{1}{2} N_{ij}^2 \right) \right)}_{:= L^2(\mathbf{x}, \mathbf{v}_1^M)} \|\boldsymbol{\theta}_1 - \boldsymbol{\theta}_2\|_2^2
\end{aligned}$$

Next, we bound the expectation

$$\begin{aligned}
\mathbb{E}_{p_d, p_{\mathbf{v}}} [L^2(\mathbf{x}, \mathbf{v}_1^M)] &= \frac{1}{M^2} \mathbb{E}_{p_{\mathbf{v}}} \left[\sum_{i,j,p,q} v_{p,i} v_{p,j} v_{q,i} v_{q,j} \right] \mathbb{E}_{p_d} \left[\sum_{ij} 2M_{ij}^2(\mathbf{x}) + \frac{1}{2} N_{ij}^2(\mathbf{x}) \right] \\
&\stackrel{(i)}{\leq} O(1) \mathbb{E}_{p_{\mathbf{v}}} \left[\sum_{i,j,p,q} v_{p,i} v_{p,j} v_{q,i} v_{q,j} \right] \\
&= O(1) \left(\sum_{ij} M(M-1) \mathbb{E}_{p_{\mathbf{v}}} [v_i v_j]^2 + M \mathbb{E}_{p_{\mathbf{v}}} [(v_i v_j)^2] \right) \\
&= O(1) \left(M(M-1) \left\| \mathbb{E}_{p_{\mathbf{v}}} [\mathbf{v} \mathbf{v}^\top] \right\|_F^2 + M \mathbb{E}_{p_{\mathbf{v}}} [\left\| \mathbf{v} \mathbf{v}^\top \right\|_F^2] \right) \\
&\stackrel{(ii)}{\leq} O(1) \left(M(M-1) \mathbb{E}_{p_{\mathbf{v}}} [\left\| \mathbf{v} \mathbf{v}^\top \right\|_F^2] + M \mathbb{E}_{p_{\mathbf{v}}} [\left\| \mathbf{v} \mathbf{v}^\top \right\|_F^2] \right) \stackrel{(iii)}{<} \infty,
\end{aligned}$$

where (i) is due to Assumption A.9, (ii) is Jensen's, and (iii) is because of Assumptions A.2 and A.8. \square

Lemma A.5. Assume that conditions in Theorem 3.1 and Lemma A.1 hold, and $p_{\mathbf{v}}$ has bounded higher-order moments (Assumption A.8). Then

$$\text{Var}_{p_d, p_{\mathbf{v}}} [\nabla_{\boldsymbol{\theta}} f(\boldsymbol{\theta}^*; \mathbf{x}, \mathbf{v}_1^M)] = \sum_{i,j,p,q} \left[\left(1 - \frac{1}{M}\right) \Sigma_{ij} \Sigma_{pq} + \frac{1}{M} \mathfrak{S}_{ijpq} \right] \mathfrak{V}_{ijpq}, \quad (\text{A.13})$$

where $\nabla_{\boldsymbol{\theta}} f(\boldsymbol{\theta}^*; \mathbf{x}, \mathbf{v}_1^M) = \nabla_{\boldsymbol{\theta}} f(\boldsymbol{\theta}; \mathbf{x}, \mathbf{v}_1^M)|_{\boldsymbol{\theta}=\boldsymbol{\theta}^*}$

In particular, if $p_{\mathbf{v}} \sim \mathcal{N}(0, I)$, we have

$$\text{Var}_{p_d, p_{\mathbf{v}}} [\nabla_{\boldsymbol{\theta}} f(\boldsymbol{\theta}^*; \mathbf{x}, \mathbf{v}_1^M)] = \sum_{ij} V_{ij} + \frac{2}{M} \sum_i V_{ii} + \frac{2}{M} \sum_{i \neq j} W_{ij}.$$

If $p_{\mathbf{v}}$ is the distribution of multivariate Rademacher random variables, we have

$$\text{Var}_{p_d, p_{\mathbf{v}}} [\nabla_{\boldsymbol{\theta}} f(\boldsymbol{\theta}^*; \mathbf{x}, \mathbf{v}_1^M)] = \sum_{ij} V_{ij} + \frac{2}{M} \sum_{i \neq j} W_{ij}.$$

Proof. Since $\boldsymbol{\theta}^*$ is the true parameter of the data distribution, we have

$$\mathbb{E}_{p_d, p_{\mathbf{v}}} [\nabla_{\boldsymbol{\theta}} f(\boldsymbol{\theta}^*; \mathbf{x}, \mathbf{v}_1^M)] = \nabla_{\boldsymbol{\theta}} \mathbb{E}_{p_d, p_{\mathbf{v}}} [f(\boldsymbol{\theta}^*; \mathbf{x}, \mathbf{v}_1^M)] = \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}^*; p_{\mathbf{v}}) = 0.$$

Therefore, Eq. (A.13) can be expanded as

$$\begin{aligned} & \text{Var}_{p_d, p_{\mathbf{v}}} [\nabla_{\boldsymbol{\theta}} f(\boldsymbol{\theta}^*; \mathbf{x}, \mathbf{v}_1^M)] \\ &= \mathbb{E}_{p_d, p_{\mathbf{v}}} [\nabla_{\boldsymbol{\theta}} f(\boldsymbol{\theta}^*; \mathbf{x}, \mathbf{v}_1^M) \nabla_{\boldsymbol{\theta}} f(\boldsymbol{\theta}^*; \mathbf{x}, \mathbf{v}_1^M)^T] \\ &= \mathbb{E} \left[\sum_{i,j,p,q} \left(\frac{1}{M^2} \sum_{k,l} v_{k,i} v_{k,j} v_{l,p} v_{l,q} \right) \right. \\ & \quad \left. \left(\nabla_{\boldsymbol{\theta}} \partial_i \partial_j l_m + \frac{1}{2} \nabla_{\boldsymbol{\theta}} (\partial_i l_m \partial_j l_m) \right) \left(\nabla_{\boldsymbol{\theta}} \partial_p \partial_q l_m + \frac{1}{2} \nabla_{\boldsymbol{\theta}} (\partial_p l_m \partial_q l_m) \right)^T \right] \\ &= \sum_{i,j,p,q} \underbrace{\mathbb{E} \left[\frac{1}{M^2} \sum_{k,l} v_{k,i} v_{k,j} v_{l,p} v_{l,q} \right]}_{:= E_1} \\ & \quad \underbrace{\mathbb{E} \left[\left(\nabla_{\boldsymbol{\theta}} \partial_i \partial_j l_m + \frac{1}{2} \nabla_{\boldsymbol{\theta}} (\partial_i l_m \partial_j l_m) \right) \left(\nabla_{\boldsymbol{\theta}} \partial_p \partial_q l_m + \frac{1}{2} \nabla_{\boldsymbol{\theta}} (\partial_p l_m \partial_q l_m) \right)^T \right]}_{\mathfrak{V}_{ijpq}}. \end{aligned}$$

Continuing on E_1 , we have that

$$\begin{aligned} E_1 &= \frac{1}{M^2} \sum_{k \neq l} \mathbb{E}[v_{k,i} v_{k,j}] \mathbb{E}[v_{l,p} v_{l,q}] + \frac{1}{M^2} \sum_k \mathbb{E}[v_{k,i} v_{k,j} v_{k,p} v_{k,q}] \\ &= \left(1 - \frac{1}{M}\right) \Sigma_{ij} \Sigma_{pq} + \frac{1}{M} \mathfrak{S}_{ijpq}, \end{aligned}$$

which leads to Eq. (A.13). Note that Assumption A.2 guarantees that $|\Sigma_{ij}| < \infty$ and Assumption A.8 ensures $|\mathfrak{S}_{ijpq}| < \infty$.

If $p_{\mathbf{v}} \sim \mathcal{N}(0, I)$, \mathfrak{S} and Σ have the following simpler forms

$$\begin{aligned} \Sigma_{ij} &= \delta_{ij} \\ \mathfrak{S}_{ijpq} &= \begin{cases} 3, & i = j = p = q \\ 1, & i = j \neq p = q \text{ or } i = p \neq j = q \text{ or } i = q \neq j = p \\ 0, & \text{otherwise} \end{cases}. \end{aligned}$$

Then, if we assume that the second derivatives of l_m are continuous, we have $\partial_i \partial_j l_m = \partial_j \partial_i l_m$, and the variance Eq. (A.13) can also be simplified to

$$\begin{aligned} \text{Var}_{p_d, p_{\mathbf{v}}} [\nabla_{\boldsymbol{\theta}} f(\boldsymbol{\theta}^*; \mathbf{x}, \mathbf{v}_1^M)] &= \sum_{i \neq j} V_{ij} + \frac{M+2}{M} \sum_i V_{ii} + \frac{2}{M} \sum_{i \neq j} W_{ij} = \sum_{ij} V_{ij} + \frac{2}{M} \sum_i V_{ii} + \frac{2}{M} \sum_{i \neq j} W_{ij}. \end{aligned}$$

Similarly, if $p_{\mathbf{v}} \sim \mathcal{U}(\{\pm 1\}^D)$, Eq. (A.13) has the simplified form

$$\text{Var}_{p_d, p_{\mathbf{v}}} [\nabla_{\boldsymbol{\theta}} f(\boldsymbol{\theta}^*; \mathbf{x}, \mathbf{v}_1^M)] = \sum_{ij} V_{ij} + \frac{2}{M} \sum_{i \neq j} W_{ij}.$$

□

Theorem 3.3. *With the notations and assumptions in Lemma A.4, Lemma A.5 and Theorem 3.2, we have*

$$\begin{aligned} &\sqrt{N}(\hat{\boldsymbol{\theta}}_{N,M} - \boldsymbol{\theta}^*) \xrightarrow{d} \\ &\mathcal{N}\left(0, \left(\nabla_{\boldsymbol{\theta}}^2 J(\boldsymbol{\theta}^*; p_{\mathbf{v}})\right)^{-1} \left(\sum_{i,j,p,q} \left[\left(1 - \frac{1}{M}\right) \Sigma_{ij} \Sigma_{pq} + \frac{1}{M} \mathfrak{S}_{ijpq} \right] \mathfrak{V}_{ijpq} \right) \left(\nabla_{\boldsymbol{\theta}}^2 J(\boldsymbol{\theta}^*; p_{\mathbf{v}})\right)^{-1} \right). \end{aligned}$$

In particular, if $p_{\mathbf{v}} \sim \mathcal{N}(0, I)$, then the asymptotic variance is

$$\left(\nabla_{\boldsymbol{\theta}}^2 J(\boldsymbol{\theta}^*) \right)^{-1} \left(\sum_{ij} V_{ij} + \frac{2}{M} \sum_i V_{ii} + \frac{2}{M} \sum_{i \neq j} W_{ij} \right) \left(\nabla_{\boldsymbol{\theta}}^2 J(\boldsymbol{\theta}^*) \right)^{-1}.$$

If $p_{\mathbf{v}}$ is the distribution of multivariate Rademacher random variables, the asymptotic variance is

$$\left(\nabla_{\boldsymbol{\theta}}^2 J(\boldsymbol{\theta}^*) \right)^{-1} \left(\sum_{ij} V_{ij} + \frac{2}{M} \sum_{i \neq j} W_{ij} \right) \left(\nabla_{\boldsymbol{\theta}}^2 J(\boldsymbol{\theta}^*) \right)^{-1}.$$

Proof. To simplify notations, we use $P_N h(\mathbf{x}) := \frac{1}{N} \sum_{i=1}^N h(\mathbf{x}_i, \cdot)$, where $h(\mathbf{x}, \cdot)$ is some arbitrary function. For example, $\hat{J}(\boldsymbol{\theta}; \mathbf{x}_1^N, \mathbf{v}_{11}^{NM})$ can be written as $P_N f(\boldsymbol{\theta}; \mathbf{x}, \mathbf{v}_1^M)$. By Taylor expansion, we can approximate $P_N \nabla_{\boldsymbol{\theta}} f(\hat{\boldsymbol{\theta}}_{N,M}; \mathbf{x}, \mathbf{v}_1^M)$ around $\boldsymbol{\theta}^*$:

$$\begin{aligned} 0 &= \nabla_{\boldsymbol{\theta}} P_N f(\hat{\boldsymbol{\theta}}_{N,M}; \mathbf{x}, \mathbf{v}_1^M) \\ &= P_N \nabla_{\boldsymbol{\theta}} f(\boldsymbol{\theta}^*; \mathbf{x}, \mathbf{v}_1^M) + P_N \left(\nabla_{\boldsymbol{\theta}}^2 f(\boldsymbol{\theta}^*; \mathbf{x}, \mathbf{v}_1^M) + E_{\hat{\boldsymbol{\theta}}_{N,M}, \mathbf{x}, \mathbf{v}_1^M} \right) (\hat{\boldsymbol{\theta}}_{N,M} - \boldsymbol{\theta}^*), \end{aligned} \quad (\text{A.14})$$

where $\|E_{\hat{\boldsymbol{\theta}}_{N,M}, \mathbf{x}, \mathbf{v}_1^M}\|_F \leq L(\mathbf{x}, \mathbf{v}_1^M) \|\hat{\boldsymbol{\theta}}_{N,M} - \boldsymbol{\theta}^*\|_2$ from Lemma A.4 and Taylor expansion of vector-valued functions. Combining with the law of large numbers, we have

$$P_N \nabla_{\boldsymbol{\theta}}^2 f(\boldsymbol{\theta}^*; \mathbf{x}, \mathbf{v}_1^M) = \mathbb{E}_{p_d, p_{\mathbf{v}}} [\nabla_{\boldsymbol{\theta}}^2 f(\boldsymbol{\theta}^*; \mathbf{x}, \mathbf{v}_1^M)] + o_p(1)$$

and

$$\left\| P_N E_{\hat{\boldsymbol{\theta}}_{N,M}} \right\|_F \leq \mathbb{E}_{p_d, p_{\mathbf{v}}} [L(\mathbf{x}, \mathbf{v}_1^M)] \left\| \hat{\boldsymbol{\theta}}_{N,M} - \boldsymbol{\theta}^* \right\|_2 + o_p(1) = o_p(1) + o_p(1) = o_p(1),$$

where we used $\mathbb{E}[L(\mathbf{x}, \mathbf{v}_1^M)] \leq \sqrt{\mathbb{E}[L^2(\mathbf{x}, \mathbf{v}_1^M)]} < \infty$ (Lemma A.4) and the consistency of $\hat{\boldsymbol{\theta}}_{N,M}$ (Theorem 3.2). Now returning to Eq. (A.14), we get

$$\begin{aligned} 0 &= P_N \nabla_{\boldsymbol{\theta}} f(\boldsymbol{\theta}^*; \mathbf{x}, \mathbf{v}_1^M) + \left(\mathbb{E}_{p_d, p_{\mathbf{v}}} [\nabla_{\boldsymbol{\theta}}^2 f(\boldsymbol{\theta}^*; \mathbf{x}, \mathbf{v}_1^M)] + o_p(1) \right) (\hat{\boldsymbol{\theta}}_{N,M} - \boldsymbol{\theta}^*) \\ &\Leftrightarrow \left(\nabla_{\boldsymbol{\theta}}^2 J(\boldsymbol{\theta}^*; p_{\mathbf{v}}) + o_p(1) \right) \sqrt{N} (\hat{\boldsymbol{\theta}}_{N,M} - \boldsymbol{\theta}^*) = -\sqrt{N} P_N \nabla_{\boldsymbol{\theta}} f(\boldsymbol{\theta}^*; \mathbf{x}, \mathbf{v}_1^M). \end{aligned}$$

But of course, the central limit theorem and Lemma A.5 yield

$$\begin{aligned} -\sqrt{N}P_N \nabla_{\boldsymbol{\theta}} f(\boldsymbol{\theta}^*; \mathbf{x}, \mathbf{v}_1^M) &\xrightarrow{d} \mathcal{N}(0, \text{Var}_{p_d, p_{\mathbf{v}}} [\nabla_{\boldsymbol{\theta}} f(\boldsymbol{\theta}^*; \mathbf{x}, \mathbf{v}_1^M)]) \\ &= \mathcal{N}\left(0, \sum_{i,j,p,q} \left[\left(1 - \frac{1}{M}\right) \Sigma_{ij} \Sigma_{pq} + \frac{1}{M} \mathfrak{S}_{ijpq} \right] \mathfrak{V}_{ijpq}\right). \end{aligned}$$

Then, Slutsky's theorem gives the desired result

$$\begin{aligned} \sqrt{N}(\hat{\boldsymbol{\theta}}_{N,M} - \boldsymbol{\theta}^*) &\xrightarrow{d} \\ \mathcal{N}\left(0, \left(\nabla_{\boldsymbol{\theta}}^2 J(\boldsymbol{\theta}^*; p_{\mathbf{v}})\right)^{-1} \left(\sum_{i,j,p,q} \left[\left(1 - \frac{1}{M}\right) \Sigma_{ij} \Sigma_{pq} + \frac{1}{M} \mathfrak{S}_{ijpq} \right] \mathfrak{V}_{ijpq} \right) \left(\nabla_{\boldsymbol{\theta}}^2 J(\boldsymbol{\theta}^*; p_{\mathbf{v}})\right)^{-1} \right). \end{aligned}$$

In particular, if $p_{\mathbf{v}} \sim \mathcal{N}(0, I)$ or $p_{\mathbf{v}} \sim \mathcal{U}(\{\pm 1\}^D)$, we have $J(\boldsymbol{\theta}^*; p_{\mathbf{v}}) = J(\boldsymbol{\theta}^*)$, and therefore $\nabla_{\boldsymbol{\theta}}^2 J(\boldsymbol{\theta}^*; p_{\mathbf{v}}) = \nabla_{\boldsymbol{\theta}}^2 J(\boldsymbol{\theta}^*)$. We can apply Lemma A.5 to conclude the simplified expressions for the asymptotic variance.

□

Corollary A.1 (Consistency and asymptotic normality of score matching). *Under similar assumptions used in Theorem 3.2 and Theorem 3.3, we can also conclude that the score matching estimator $\hat{\boldsymbol{\theta}}_N := \arg \min_{\boldsymbol{\theta} \in \Theta} \hat{J}(\boldsymbol{\theta}; \mathbf{x})$ is consistent*

$$\hat{\boldsymbol{\theta}}_N \xrightarrow{p} \boldsymbol{\theta}^*$$

and asymptotically normal

$$\sqrt{N}(\hat{\boldsymbol{\theta}}_N - \boldsymbol{\theta}^*) \xrightarrow{d} \mathcal{N}\left(0, \left(\nabla_{\boldsymbol{\theta}}^2 J(\boldsymbol{\theta}^*)\right)^{-1} \left(\sum_{ij} V_{ij} \right) \left(\nabla_{\boldsymbol{\theta}}^2 J(\boldsymbol{\theta}^*)\right)^{-1} \right).$$

Proof. Note that

$$\begin{aligned} \text{Var}_{p_d}[\nabla_{\boldsymbol{\theta}} f(\boldsymbol{\theta}^*; \mathbf{x})] &= \mathbb{E}_{p_d} \left[\nabla_{\boldsymbol{\theta}} f(\boldsymbol{\theta}^*; \mathbf{x}) \nabla_{\boldsymbol{\theta}} f(\boldsymbol{\theta}^*; \mathbf{x})^T \right] \\ &= \mathbb{E}_{p_d} \left[\left(\sum_i \nabla_{\boldsymbol{\theta}} \partial_i \partial_i l_m + \frac{1}{2} \nabla_{\boldsymbol{\theta}} (\partial_i l_m \partial_i l_m) \right) \left(\sum_j \nabla_{\boldsymbol{\theta}} \partial_j \partial_j l_m + \frac{1}{2} \nabla_{\boldsymbol{\theta}} (\partial_j l_m \partial_j l_m) \right)^T \right] \\ &= \sum_{ij} \mathbb{E}_{p_d} \left[\left(\nabla_{\boldsymbol{\theta}} \partial_i \partial_i l_m + \frac{1}{2} \nabla_{\boldsymbol{\theta}} (\partial_i l_m \partial_i l_m) \right) \left(\nabla_{\boldsymbol{\theta}} \partial_j \partial_j l_m + \frac{1}{2} \nabla_{\boldsymbol{\theta}} (\partial_j l_m \partial_j l_m) \right)^T \right] \end{aligned}$$

$$= \sum_{ij} V_{ij}$$

The other part of the proof is similar to that of Theorem 3.2 and Theorem 3.3 and is thus obmitted. \square

A.1.5 Noise Contrastive Estimation

Proposition A.1. Define

$$J_{NCE}(\boldsymbol{\theta}) := -\mathbb{E}_{p_d}[\log h(\mathbf{x}; \boldsymbol{\theta})] - \mathbb{E}_{p_n}[\log(1 - h(\mathbf{x}; \boldsymbol{\theta}))]$$

where

$$\begin{aligned} h(\mathbf{x}; \boldsymbol{\theta}) &:= \frac{p_m(\mathbf{x}; \boldsymbol{\theta})}{p_m(\mathbf{x}; \boldsymbol{\theta}) + p_m(\mathbf{x} + \mathbf{v}; \boldsymbol{\theta})} \\ p_n(\mathbf{x}) &= p_d(\mathbf{x} + \mathbf{v}). \end{aligned}$$

Then when $\|\mathbf{v}\|_2 \rightarrow 0$, we have

$$J_{NCE}(\boldsymbol{\theta}) = 2 \log 2 + \frac{1}{4} \mathbb{E}_{p_d} \left[\mathbf{v}^\top \nabla^2 \log p_m(\mathbf{x}; \boldsymbol{\theta}) \mathbf{v} + \frac{1}{2} (\nabla \log p_m(\mathbf{x}; \boldsymbol{\theta})^\top \mathbf{v})^2 \right] + o(\|\mathbf{v}\|_2^2)$$

Proof. Using Taylor expansion, we can immediately get

$$\log p_m(\mathbf{x} + \mathbf{v}; \boldsymbol{\theta}) = \log p_m(\mathbf{x}; \boldsymbol{\theta}) + \nabla \log p_m(\mathbf{x}; \boldsymbol{\theta})^\top \mathbf{v} + \frac{1}{2} \mathbf{v}^\top \nabla^2 \log p_m(\mathbf{x}; \boldsymbol{\theta}) \mathbf{v} + o(\|\mathbf{v}\|_2^2).$$

Next, observe that

$$\begin{aligned} \log(p_m(\mathbf{x}; \boldsymbol{\theta}) + p_m(\mathbf{x} + \mathbf{v}; \boldsymbol{\theta})) &= \log p_m(\mathbf{x}; \boldsymbol{\theta}) + \log(1 + \exp\{\log p_m(\mathbf{x} + \mathbf{v}; \boldsymbol{\theta}) - \log p_m(\mathbf{x}; \boldsymbol{\theta})\}) \\ &= \log p_m(\mathbf{x}; \boldsymbol{\theta}) + \log \left(1 + \exp \left\{ \nabla \log p_m(\mathbf{x}; \boldsymbol{\theta})^\top \mathbf{v} + \frac{1}{2} \mathbf{v}^\top \nabla^2 \log p_m(\mathbf{x}; \boldsymbol{\theta}) \mathbf{v} + o(\|\mathbf{v}\|_2^2) \right\} \right) \\ &= \log p_m(\mathbf{x}; \boldsymbol{\theta}) + \log 2 + \frac{1}{2} \left[\nabla \log p_m(\mathbf{x}; \boldsymbol{\theta})^\top \mathbf{v} + \frac{1}{2} \mathbf{v}^\top \nabla^2 \log p_m(\mathbf{x}; \boldsymbol{\theta}) \mathbf{v} \right] \\ &\quad + \frac{1}{8} (\nabla \log p_m(\mathbf{x}; \boldsymbol{\theta})^\top \mathbf{v})^2 + o(\|\mathbf{v}\|_2^2). \end{aligned}$$

Similarly, we have

$$\begin{aligned} & \log(p_m(\mathbf{x}; \boldsymbol{\theta}) + p_m(\mathbf{x} - \mathbf{v}; \boldsymbol{\theta})) \\ &= \log p_m(\mathbf{x}; \boldsymbol{\theta}) + \log 2 + \frac{1}{2} \left[-\nabla \log p_m(\mathbf{x}; \boldsymbol{\theta})^\top \mathbf{v} + \frac{1}{2} \mathbf{v}^\top \nabla^2 \log p_m(\mathbf{x}; \boldsymbol{\theta}) \mathbf{v} \right] \\ &\quad + \frac{1}{8} (\nabla \log p_m(\mathbf{x}; \boldsymbol{\theta})^\top \mathbf{v})^2 + o(\|\mathbf{v}\|_2^2). \end{aligned}$$

Finally, note that

$$\begin{aligned} J_{\text{NCE}}(\boldsymbol{\theta}) &= -\mathbb{E}_{p_d} [\log h(\mathbf{x}; \boldsymbol{\theta}) + \log(1 - h(\mathbf{x} + \mathbf{v}; \boldsymbol{\theta}))] \\ &= -\mathbb{E}_{p_d} [\log p_m(\mathbf{x}; \boldsymbol{\theta}) - \log(p_m(\mathbf{x}; \boldsymbol{\theta}) + p_m(\mathbf{x} - \mathbf{v}; \boldsymbol{\theta}))] \\ &\quad - \mathbb{E}_{p_d} [\log p_m(\mathbf{x}; \boldsymbol{\theta}) - \log(p_m(\mathbf{x}; \boldsymbol{\theta}) + p_m(\mathbf{x} + \mathbf{v}; \boldsymbol{\theta}))] \\ &= 2 \log 2 + \frac{1}{4} \mathbb{E}_{p_d} \left[\mathbf{v}^\top \nabla^2 \log p_m(\mathbf{x}; \boldsymbol{\theta}) \mathbf{v} + \frac{1}{2} (\nabla \log p_m(\mathbf{x}; \boldsymbol{\theta})^\top \mathbf{v})^2 \right] + o(\|\mathbf{v}\|_2^2), \end{aligned}$$

as desired. \square

A.2 Proofs for Chapter 4

In the following, we assume that $q_\sigma(\tilde{\mathbf{x}}|\mathbf{x}) = \mathcal{N}(\tilde{\mathbf{x}}|\mathbf{x}, \sigma^2 I)$. Tweedie's formula can also be derived using the proof for Theorem 4.1.

Theorem 4.1. *Given D-dimensional densities $p(\mathbf{x})$ and $q_\sigma(\tilde{\mathbf{x}}) := \int p(\mathbf{x}) q_\sigma(\tilde{\mathbf{x}}|\mathbf{x}) d\mathbf{x}$, we have*

$$\mathbb{E}[\mathbf{x}\mathbf{x}^\top | \tilde{\mathbf{x}}] = \mathbf{f}(\tilde{\mathbf{x}}, \tilde{\mathbf{s}}_1, \tilde{\mathbf{s}}_2) \tag{A.15}$$

$$\mathbb{E}[\mathbf{x}\mathbf{x}^\top - \mathbf{x}\tilde{\mathbf{x}}^\top - \tilde{\mathbf{x}}\mathbf{x}^\top | \tilde{\mathbf{x}}] = \mathbf{h}(\tilde{\mathbf{x}}, \tilde{\mathbf{s}}_1, \tilde{\mathbf{s}}_2), \tag{A.16}$$

where $\mathbf{f}(\tilde{\mathbf{x}}, \tilde{\mathbf{s}}_1, \tilde{\mathbf{s}}_2)$ and $\mathbf{h}(\tilde{\mathbf{x}}, \tilde{\mathbf{s}}_1, \tilde{\mathbf{s}}_2)$ are polynomials of $\tilde{\mathbf{x}}, \tilde{\mathbf{s}}_1(\tilde{\mathbf{x}}), \tilde{\mathbf{s}}_2(\tilde{\mathbf{x}})$ defined as

$$\mathbf{f}(\tilde{\mathbf{x}}, \tilde{\mathbf{s}}_1, \tilde{\mathbf{s}}_2) = \tilde{\mathbf{x}}\tilde{\mathbf{x}}^\top + \sigma^2 \tilde{\mathbf{x}}\tilde{\mathbf{s}}_1(\tilde{\mathbf{x}})^\top + \sigma^2 \tilde{\mathbf{s}}_1(\tilde{\mathbf{x}})\tilde{\mathbf{x}}^\top + \sigma^4 \tilde{\mathbf{s}}_2(\tilde{\mathbf{x}}) + \sigma^4 \tilde{\mathbf{s}}_1(\tilde{\mathbf{x}})\tilde{\mathbf{s}}_1(\tilde{\mathbf{x}})^\top + \sigma^2 I, \tag{A.17}$$

$$\mathbf{h}(\tilde{\mathbf{x}}, \tilde{\mathbf{s}}_1, \tilde{\mathbf{s}}_2) = -\tilde{\mathbf{x}}\tilde{\mathbf{x}}^\top + \sigma^4 \tilde{\mathbf{s}}_2(\tilde{\mathbf{x}}) + \sigma^4 \tilde{\mathbf{s}}_1(\tilde{\mathbf{x}})\tilde{\mathbf{s}}_1(\tilde{\mathbf{x}})^\top + \sigma^2 I. \tag{A.18}$$

Here $\tilde{\mathbf{s}}_1(\tilde{\mathbf{x}})$ and $\tilde{\mathbf{s}}_2(\tilde{\mathbf{x}})$ denote the first and second order scores of $q_\sigma(\tilde{\mathbf{x}})$.

Proof. We can rewrite $q_\sigma(\tilde{\mathbf{x}}|\mathbf{x})$ in the form of exponential family

$$q_\sigma(\tilde{\mathbf{x}}|\boldsymbol{\eta}) = e^{\boldsymbol{\eta}^\top \tilde{\mathbf{x}} - \psi(\boldsymbol{\eta})} q_0(\tilde{\mathbf{x}}),$$

where $\boldsymbol{\eta} = \frac{\mathbf{x}}{\sigma^2}$ is the natural or canonical parameter of the family, $\psi(\boldsymbol{\eta})$ is the cumulant generating function which makes $q_\sigma(\tilde{\mathbf{x}}|\boldsymbol{\eta})$ normalized and $q_0(\tilde{\mathbf{x}}) = ((2\pi)^d \sigma^{2d})^{-\frac{1}{2}} e^{-\frac{\tilde{\mathbf{x}}^\top \tilde{\mathbf{x}}}{2\sigma^2}}$.

Bayes rule provides the corresponding posterior

$$q(\boldsymbol{\eta}|\tilde{\mathbf{x}}) = \frac{q_\sigma(\tilde{\mathbf{x}}|\boldsymbol{\eta}) p(\boldsymbol{\eta})}{q_\sigma(\tilde{\mathbf{x}})}.$$

Let $\lambda(\tilde{\mathbf{x}}) = \log \frac{q_\sigma(\tilde{\mathbf{x}})}{q_0(\tilde{\mathbf{x}})}$, then we can write posterior as

$$q(\boldsymbol{\eta}|\tilde{\mathbf{x}}) = e^{\boldsymbol{\eta}^\top \tilde{\mathbf{x}} - \psi(\boldsymbol{\eta}) - \lambda(\tilde{\mathbf{x}})} p(\boldsymbol{\eta}).$$

Since the posterior is normalized, we have

$$\int e^{\boldsymbol{\eta}^\top \tilde{\mathbf{x}} - \psi(\boldsymbol{\eta}) - \lambda(\tilde{\mathbf{x}})} p(\boldsymbol{\eta}) d\boldsymbol{\eta} = 1.$$

As a widely used technique in exponential families, we differentiate both sides w.r.t. $\tilde{\mathbf{x}}$

$$\int (\boldsymbol{\eta}^\top - \mathbf{J}_\lambda(\tilde{\mathbf{x}})^\top) q(\boldsymbol{\eta}|\tilde{\mathbf{x}}) d\boldsymbol{\eta} = 0,$$

and the first order posterior moment can be written as

$$\mathbb{E}[\boldsymbol{\eta} | \tilde{\mathbf{x}}] = \mathbf{J}_\lambda(\tilde{\mathbf{x}}) \quad (\text{A.19})$$

$$\mathbb{E}[\boldsymbol{\eta}^\top | \tilde{\mathbf{x}}] = \mathbf{J}_\lambda(\tilde{\mathbf{x}})^\top, \quad (\text{A.20})$$

where $\mathbf{J}_\lambda(\tilde{\mathbf{x}})$ is the Jacobian of $\lambda(\tilde{\mathbf{x}})$ w.r.t. $\tilde{\mathbf{x}}$.

Differentiating both sides w.r.t. $\tilde{\mathbf{x}}$ again

$$\int \boldsymbol{\eta}(\boldsymbol{\eta}^\top - \mathbf{J}_\lambda(\tilde{\mathbf{x}})^\top) q(\boldsymbol{\eta}|\tilde{\mathbf{x}}) d\boldsymbol{\eta} = \mathbf{H}_\lambda(\tilde{\mathbf{x}}),$$

and the second order posterior moment can be written as

$$\mathbb{E}[\boldsymbol{\eta}\boldsymbol{\eta}^\top | \tilde{\mathbf{x}}] = \mathbf{H}_\lambda(\tilde{\mathbf{x}}) + \mathbf{J}_\lambda(\tilde{\mathbf{x}})\mathbf{J}_\lambda(\tilde{\mathbf{x}})^\top, \quad (\text{A.21})$$

where $\mathbf{H}_\lambda(\tilde{\mathbf{x}})$ is the Hessian of $\lambda(\tilde{\mathbf{x}})$ w.r.t. $\tilde{\mathbf{x}}$.

Specifically, for $q_\sigma(\tilde{\mathbf{x}}|\mathbf{x}) = \mathcal{N}(\tilde{\mathbf{x}}|\mathbf{x}, \sigma^2 I)$, we have $\boldsymbol{\eta} = \frac{\mathbf{x}}{\sigma^2}$ and $q_0(\tilde{\mathbf{x}}) = ((2\pi)^d \sigma^{2d})^{-\frac{1}{2}} e^{-\frac{\tilde{\mathbf{x}}^\top \tilde{\mathbf{x}}}{2\sigma^2}}$. Hence we have

$$\begin{aligned}\lambda(\tilde{\mathbf{x}}) &= \log q_\sigma(\tilde{\mathbf{x}}) + \frac{\tilde{\mathbf{x}}^\top \tilde{\mathbf{x}}}{2\sigma^2} + \text{constant} \\ \mathbf{J}_\lambda(\tilde{\mathbf{x}}) &= \tilde{\mathbf{s}}_1(\tilde{\mathbf{x}}) + \frac{\tilde{\mathbf{x}}}{\sigma^2} \\ \mathbf{H}_\lambda(\tilde{\mathbf{x}}) &= \tilde{\mathbf{s}}_2(\tilde{\mathbf{x}}) + \frac{1}{\sigma^2} I.\end{aligned}$$

From Eq. (A.21), we have

$$\mathbb{E}[\mathbf{x}\mathbf{x}^\top | \tilde{\mathbf{x}}] = \tilde{\mathbf{x}}\tilde{\mathbf{x}}^\top + \sigma^2 \tilde{\mathbf{x}}\tilde{\mathbf{s}}_1(\tilde{\mathbf{x}})^\top + \sigma^2 \tilde{\mathbf{s}}_1(\tilde{\mathbf{x}})\tilde{\mathbf{x}}^\top + \sigma^4 \tilde{\mathbf{s}}_2(\tilde{\mathbf{x}}) + \sigma^4 \tilde{\mathbf{s}}_1(\tilde{\mathbf{x}})\tilde{\mathbf{s}}_1(\tilde{\mathbf{x}})^\top + \sigma^2 I = \mathbf{f}(\tilde{\mathbf{x}}, \tilde{\mathbf{s}}_1, \tilde{\mathbf{s}}_2).$$

Combined with Eq. (A.19), Eq. (A.20), and Eq. (A.21), we have

$$\mathbb{E}[\mathbf{x}\mathbf{x}^\top - \mathbf{x}\tilde{\mathbf{x}}^\top - \tilde{\mathbf{x}}\mathbf{x}^\top | \tilde{\mathbf{x}}] = -\tilde{\mathbf{x}}\tilde{\mathbf{x}}^\top + \sigma^4 \tilde{\mathbf{s}}_2(\tilde{\mathbf{x}}) + \sigma^4 \tilde{\mathbf{s}}_1(\tilde{\mathbf{x}})\tilde{\mathbf{s}}_1(\tilde{\mathbf{x}})^\top + \sigma^2 I = \mathbf{h}(\tilde{\mathbf{x}}, \tilde{\mathbf{s}}_1, \tilde{\mathbf{s}}_2).$$

□

Tweedie's formula. Given D -dimensional densities $p(\mathbf{x})$ and $q_\sigma(\tilde{\mathbf{x}}) := \int p(\mathbf{x})q_\sigma(\tilde{\mathbf{x}}|\mathbf{x})d\mathbf{x}$, we have

$$\mathbb{E}[\mathbf{x} | \tilde{\mathbf{x}}] = \tilde{\mathbf{x}} + \sigma^2 \tilde{\mathbf{s}}_1(\tilde{\mathbf{x}}), \quad (\text{A.22})$$

where $\tilde{\mathbf{s}}_1(\tilde{\mathbf{x}}) := \nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}})$.

Proof. Plug in $\boldsymbol{\eta} = \frac{\mathbf{x}}{\sigma^2}$ and $\mathbf{J}_\lambda(\tilde{\mathbf{x}}) = \tilde{\mathbf{s}}_1(\tilde{\mathbf{x}}) + \frac{\tilde{\mathbf{x}}}{\sigma^2}$ in Eq. (A.19), we have

$$\mathbb{E}[\mathbf{x} | \tilde{\mathbf{x}}] = \tilde{\mathbf{x}} + \sigma^2 \tilde{\mathbf{s}}_1(\tilde{\mathbf{x}}), \quad (\text{A.23})$$

which proves Tweedie's formula. □

Theorem 4.2. Suppose the first order score $\tilde{\mathbf{s}}_1(\tilde{\mathbf{x}})$ is given, we can learn a second order score model $\tilde{\mathbf{s}}_2(\tilde{\mathbf{x}}; \boldsymbol{\theta})$ by optimizing the following objectives

$$\begin{aligned}\boldsymbol{\theta}^* &= \arg \min_{\boldsymbol{\theta}} \mathbb{E}_{p_{\text{data}}(\mathbf{x})} \mathbb{E}_{q_\sigma(\tilde{\mathbf{x}}|\mathbf{x})} \left[\left\| \mathbf{x}\mathbf{x}^\top - \mathbf{f}(\tilde{\mathbf{x}}, \tilde{\mathbf{s}}_1(\tilde{\mathbf{x}}), \tilde{\mathbf{s}}_2(\tilde{\mathbf{x}}; \boldsymbol{\theta})) \right\|_2^2 \right], \\ \boldsymbol{\theta}^* &= \arg \min_{\boldsymbol{\theta}} \mathbb{E}_{p_{\text{data}}(\mathbf{x})} \mathbb{E}_{q_\sigma(\tilde{\mathbf{x}}|\mathbf{x})} \left[\left\| \mathbf{x}\mathbf{x}^\top - \mathbf{x}\tilde{\mathbf{x}}^\top - \tilde{\mathbf{x}}\mathbf{x}^\top - \mathbf{h}(\tilde{\mathbf{x}}, \tilde{\mathbf{s}}_1(\tilde{\mathbf{x}}), \tilde{\mathbf{s}}_2(\tilde{\mathbf{x}}; \boldsymbol{\theta})) \right\|_2^2 \right]\end{aligned}$$

where $\mathbf{f}(\cdot)$ and $\mathbf{h}(\cdot)$ are polynomials defined in Eq. (4.9) and Eq. (4.10). Assuming the model has an infinite capacity, then the optimal parameter θ^* satisfies $\tilde{\mathbf{s}}_2(\tilde{\mathbf{x}}; \theta^*) = \tilde{\mathbf{s}}_2(\tilde{\mathbf{x}})$ for almost any $\tilde{\mathbf{x}}$.

Proof. It is well-known that the optimal solution to the least squares regression problems of Eq. (4.11) and Eq. (4.12) are the conditional expectations $\mathbf{f}(\tilde{\mathbf{x}}, \tilde{\mathbf{s}}_1(\tilde{\mathbf{x}}), \tilde{\mathbf{s}}_2(\tilde{\mathbf{x}}; \theta^*)) = \mathbb{E}[\mathbf{x}\mathbf{x}^\top | \tilde{\mathbf{x}}]$ and $\mathbf{h}(\tilde{\mathbf{x}}, \tilde{\mathbf{s}}_1(\tilde{\mathbf{x}}), \tilde{\mathbf{s}}_2(\tilde{\mathbf{x}}; \theta^*)) = \mathbb{E}[\mathbf{x}\mathbf{x}^\top - \mathbf{x}\tilde{\mathbf{x}}^\top - \tilde{\mathbf{x}}\mathbf{x}^\top | \tilde{\mathbf{x}}]$ respectively. According to Theorem 4.1, this implies that the optimal solution satisfies $\tilde{\mathbf{s}}_2(\tilde{\mathbf{x}}; \theta^*) = \tilde{\mathbf{s}}_2(\tilde{\mathbf{x}})$ for almost any $\tilde{\mathbf{x}}$ given the first order score $\tilde{\mathbf{s}}_1(\tilde{\mathbf{x}})$.

Note: Eq. (4.11) and Eq. (4.12) have the same set of solutions assuming sufficient model capacity. However, Eq. (4.12) has a simpler form (e.g., involving fewer terms) than Eq. (4.11) since multiple terms in Eq. (4.12) can be cancelled after expanding the equation by using Eq. (4.4) (Tweedie's formula), resulting in the simplified objective Eq. (4.13). Compared to the expansion of Eq. (4.11), the expansion of Eq. (4.12) (i.e., Eq. (4.13)) is much simpler (i.e., involving fewer terms), which is why we use Eq. (4.12) other than Eq. (4.11) in our experiments. \square

Before proving Theorem 4.3, we first prove the following lemma.

Lemma A.6. *Given a D dimensional distribution $p_{data}(\mathbf{x})$, and $q_\sigma(\tilde{\mathbf{x}}|\mathbf{x}) := \mathcal{N}(\tilde{\mathbf{x}}|\mathbf{x}, \sigma^2 I)$, we have the following for any integer $n \geq 1$:*

$$\mathbb{E}[\otimes^{n+1}\mathbf{x}|\tilde{\mathbf{x}}] = \sigma^2 \frac{\partial}{\partial \tilde{\mathbf{x}}} \mathbb{E}[\otimes^n\mathbf{x}|\tilde{\mathbf{x}}] + \sigma^2 \mathbb{E}[\otimes^n\mathbf{x}|\tilde{\mathbf{x}}] \otimes \left(\tilde{\mathbf{s}}_1(\tilde{\mathbf{x}}) + \frac{\tilde{\mathbf{x}}}{\sigma^2} \right),$$

where $\otimes^n\mathbf{x} \in \mathbb{R}^{D^n}$ denotes n -fold tensor multiplications.

Proof. We follow the notation used in the previous proof. Since

$$\mathbb{E}[\otimes^n\boldsymbol{\eta}|\tilde{\mathbf{x}}] = \int e^{\boldsymbol{\eta}^\top \tilde{\mathbf{x}} - \psi(\boldsymbol{\eta}) - \lambda(\tilde{\mathbf{x}})} p(\boldsymbol{\eta}) \otimes^n \boldsymbol{\eta} d\boldsymbol{\eta},$$

differentiating both sides w.r.t. $\tilde{\mathbf{x}}$

$$\begin{aligned} \frac{\partial}{\partial \tilde{\mathbf{x}}} \mathbb{E}[\otimes^n\boldsymbol{\eta}|\tilde{\mathbf{x}}] &= \int e^{\boldsymbol{\eta}^\top \tilde{\mathbf{x}} - \psi(\boldsymbol{\eta}) - \lambda(\tilde{\mathbf{x}})} p(\boldsymbol{\eta}) \otimes^{n+1} \boldsymbol{\eta} d\boldsymbol{\eta} - \int e^{\boldsymbol{\eta}^\top \tilde{\mathbf{x}} - \psi(\boldsymbol{\eta}) - \lambda(\tilde{\mathbf{x}})} p(\boldsymbol{\eta}) \otimes^n \boldsymbol{\eta} d\boldsymbol{\eta} \otimes \frac{\partial}{\partial \tilde{\mathbf{x}}} \lambda(\tilde{\mathbf{x}}) \\ \frac{\partial}{\partial \tilde{\mathbf{x}}} \mathbb{E}[\otimes^n\boldsymbol{\eta}|\tilde{\mathbf{x}}] &= \mathbb{E}[\otimes^{n+1}\boldsymbol{\eta}|\tilde{\mathbf{x}}] - \mathbb{E}[\otimes^n\boldsymbol{\eta}|\tilde{\mathbf{x}}] \otimes \left(\tilde{\mathbf{s}}_1(\tilde{\mathbf{x}}) + \frac{\tilde{\mathbf{x}}}{\sigma^2} \right). \end{aligned}$$

Thus

$$\mathbb{E}[\otimes^{n+1}\mathbf{x}|\tilde{\mathbf{x}}] = \sigma^2 \frac{\partial}{\partial \tilde{\mathbf{x}}} \mathbb{E}[\otimes^n \mathbf{x}|\tilde{\mathbf{x}}] + \sigma^2 \mathbb{E}[\otimes^n \mathbf{x}|\tilde{\mathbf{x}}] \otimes \left(\tilde{\mathbf{s}}_1(\tilde{\mathbf{x}}) + \frac{\tilde{\mathbf{x}}}{\sigma^2} \right).$$

□

Example When $n = 2$, plug in Eq. (4.4), we have

$$\mathbb{E}[\mathbf{x}\mathbf{x}^\top|\tilde{\mathbf{x}}] = \sigma^2(I + \sigma^2\tilde{\mathbf{s}}_2(\tilde{\mathbf{x}})) + \sigma^2(\tilde{\mathbf{x}} + \sigma^2\tilde{\mathbf{s}}_1(\tilde{\mathbf{x}}))(\tilde{\mathbf{s}}_1(\tilde{\mathbf{x}}) + \frac{\tilde{\mathbf{x}}}{\sigma^2})^\top,$$

which can be simplified as Eq. (4.9).

Lemma A.6 provides a recurrence for obtaining \mathbf{f}_n in closed form. It is further used and discussed in Theorem 4.3.

Theorem 4.3. $\mathbb{E}[\otimes^n \mathbf{x}|\tilde{\mathbf{x}}] = \mathbf{f}_n(\tilde{\mathbf{x}}, \tilde{\mathbf{s}}_1, \dots, \tilde{\mathbf{s}}_n)$, where $\otimes^n \mathbf{x} \in \mathbb{R}^{D^n}$ denotes n -fold tensor multiplications, $\mathbf{f}_n(\tilde{\mathbf{x}}, \tilde{\mathbf{s}}_1, \dots, \tilde{\mathbf{s}}_n)$ is a polynomial of $\{\tilde{\mathbf{x}}, \tilde{\mathbf{s}}_1(\tilde{\mathbf{x}}), \dots, \tilde{\mathbf{s}}_n(\tilde{\mathbf{x}})\}$ and $\tilde{\mathbf{s}}_k(\mathbf{x})$ represents the k -th order score of $q_\sigma(\tilde{\mathbf{x}}) = \int p_{\text{data}}(\mathbf{x}) q_\sigma(\tilde{\mathbf{x}}|\mathbf{x}) d\mathbf{x}$.

Proof. We prove this using induction. When $n = 1$, we have

$$\mathbb{E}[\mathbf{x}|\tilde{\mathbf{x}}] = \sigma^2 \tilde{\mathbf{s}}_1(\tilde{\mathbf{x}}) + \tilde{\mathbf{x}}.$$

Thus, $\mathbb{E}[\mathbf{x}|\tilde{\mathbf{x}}]$ can be written as a polynomial of $\{\tilde{\mathbf{x}}, \tilde{\mathbf{s}}_1(\tilde{\mathbf{x}})\}$. The hypothesis holds.

Assume the hypothesis holds when $n = t$, then

$$\mathbb{E}[\otimes^t \mathbf{x}|\tilde{\mathbf{x}}] = \mathbf{f}_t(\tilde{\mathbf{x}}, \tilde{\mathbf{s}}_1, \dots, \tilde{\mathbf{s}}_t).$$

When $n = t + 1$,

$$\begin{aligned} \mathbb{E}[\otimes^{t+1} \mathbf{x}|\tilde{\mathbf{x}}] &= \sigma^2 \frac{\partial}{\partial \tilde{\mathbf{x}}} \mathbb{E}[\otimes^t \mathbf{x}|\tilde{\mathbf{x}}] + \sigma^2 \mathbb{E}[\otimes^t \mathbf{x}|\tilde{\mathbf{x}}] \otimes \left(\tilde{\mathbf{s}}_1(\tilde{\mathbf{x}}) + \frac{\tilde{\mathbf{x}}}{\sigma^2} \right) \\ &= \sigma^2 \frac{\partial}{\partial \tilde{\mathbf{x}}} \mathbf{f}_t(\tilde{\mathbf{x}}, \tilde{\mathbf{s}}_1, \dots, \tilde{\mathbf{s}}_t) + \sigma^2 \mathbf{f}_t(\tilde{\mathbf{x}}, \tilde{\mathbf{s}}_1, \dots, \tilde{\mathbf{s}}_t) \otimes \left(\tilde{\mathbf{s}}_1(\tilde{\mathbf{x}}) + \frac{\tilde{\mathbf{x}}}{\sigma^2} \right). \end{aligned}$$

Clearly, $\sigma^2 \mathbf{f}_t(\tilde{\mathbf{x}}, \tilde{\mathbf{s}}_1, \dots, \tilde{\mathbf{s}}_t) \otimes \left(\tilde{\mathbf{s}}_1(\tilde{\mathbf{x}}) + \frac{\tilde{\mathbf{x}}}{\sigma^2} \right)$ is a polynomial of $\{\tilde{\mathbf{x}}, \tilde{\mathbf{s}}_1(\tilde{\mathbf{x}}), \dots, \tilde{\mathbf{s}}_t(\tilde{\mathbf{x}})\}$, and $\sigma^2 \frac{\partial}{\partial \tilde{\mathbf{x}}} \mathbf{f}_t(\tilde{\mathbf{x}}, \tilde{\mathbf{s}}_1, \dots, \tilde{\mathbf{s}}_t)$ is a polynomial of $\{\tilde{\mathbf{x}}, \tilde{\mathbf{s}}_1(\tilde{\mathbf{x}}), \dots, \tilde{\mathbf{s}}_{t+1}(\tilde{\mathbf{x}})\}$. This implies $\mathbb{E}[\otimes^{t+1} \mathbf{x}|\tilde{\mathbf{x}}]$ can be written as $\mathbf{f}_{t+1}(\tilde{\mathbf{x}}, \tilde{\mathbf{s}}_1, \dots, \tilde{\mathbf{s}}_{t+1})$, which is a polynomial of $\{\tilde{\mathbf{x}}, \tilde{\mathbf{s}}_1(\tilde{\mathbf{x}}), \dots, \tilde{\mathbf{s}}_{t+1}(\tilde{\mathbf{x}})\}$. Thus, the hypothesis holds when $k = t + 1$, which implies that the hypothesis holds for all integer $n \geq 1$. □

Theorem 4.4. *Given the true score functions $\tilde{\mathbf{s}}_1(\tilde{\mathbf{x}}), \dots, \tilde{\mathbf{s}}_{k-1}(\tilde{\mathbf{x}})$, a k -th order score model $\tilde{\mathbf{s}}_k(\tilde{\mathbf{x}}; \boldsymbol{\theta})$, and*

$$\boldsymbol{\theta}^* = \arg \min_{\boldsymbol{\theta}} \mathbb{E}_{p_{data}(\mathbf{x})} \mathbb{E}_{q_{\sigma}(\tilde{\mathbf{x}}|\mathbf{x})} [\|\otimes^k \mathbf{x} - \mathbf{f}_k(\tilde{\mathbf{x}}, \tilde{\mathbf{s}}_1(\tilde{\mathbf{x}}), \dots, \tilde{\mathbf{s}}_{k-1}(\tilde{\mathbf{x}}), \tilde{\mathbf{s}}_k(\tilde{\mathbf{x}}; \boldsymbol{\theta}))\|^2]. \quad (\text{A.24})$$

Assuming the model has an infinite capacity, we have $\tilde{\mathbf{s}}_k(\tilde{\mathbf{x}}; \boldsymbol{\theta}^*) = \tilde{\mathbf{s}}_k(\tilde{\mathbf{x}})$ for almost all $\tilde{\mathbf{x}}$.

Proof. Similar to the previous case, we can show that the solution to the least squares regression problems of Eq. (A.24) is $\mathbf{f}_k(\tilde{\mathbf{x}}, \tilde{\mathbf{s}}_1(\tilde{\mathbf{x}}), \dots, \tilde{\mathbf{s}}_{k-1}(\tilde{\mathbf{x}}), \tilde{\mathbf{s}}_k(\tilde{\mathbf{x}}; \boldsymbol{\theta})^*) = \mathbb{E}[\otimes^k \mathbf{x} | \tilde{\mathbf{x}}]$. According to Theorem 4.3, this implies $\tilde{\mathbf{s}}_k(\tilde{\mathbf{x}}; \boldsymbol{\theta}^*) = \tilde{\mathbf{s}}_k(\tilde{\mathbf{x}})$ given the score functions $\tilde{\mathbf{s}}_1(\tilde{\mathbf{x}}), \dots, \tilde{\mathbf{s}}_{k-1}(\tilde{\mathbf{x}})$. \square

A.3 Proofs for Chapter 6

Proposition 6.1. *Let $\hat{p}_{\sigma_1}(\mathbf{x}) := \frac{1}{N} \sum_{i=1}^N p^{(i)}(\mathbf{x})$, where $p^{(i)}(\mathbf{x}) := \mathcal{N}(\mathbf{x} \mid \mathbf{x}^{(i)}, \sigma_1^2 I)$. With $r^{(i)}(\mathbf{x}) := \frac{p^{(i)}(\mathbf{x})}{\sum_{k=1}^N p^{(k)}(\mathbf{x})}$, the score function is $\nabla_{\mathbf{x}} \log \hat{p}_{\sigma_1}(\mathbf{x}) = \sum_{i=1}^N r^{(i)}(\mathbf{x}) \nabla_{\mathbf{x}} \log p^{(i)}(\mathbf{x})$. Moreover,*

$$\mathbb{E}_{p^{(i)}(\mathbf{x})} [r^{(j)}(\mathbf{x})] \leq \frac{1}{2} \exp \left(- \frac{\|\mathbf{x}^{(i)} - \mathbf{x}^{(j)}\|_2^2}{8\sigma_1^2} \right). \quad (\text{A.25})$$

Proof. According to the definition of $p_{\sigma_1}(\mathbf{x})$ and $r(\mathbf{x})$, we have

$$\begin{aligned} \nabla_{\mathbf{x}} \log \hat{p}_{\sigma_1}(\mathbf{x}) &= \nabla_{\mathbf{x}} \log \left(\frac{1}{N} \sum_{i=1}^N p^{(i)}(\mathbf{x}) \right) = \sum_{i=1}^N \frac{\nabla_{\mathbf{x}} p^{(i)}(\mathbf{x})}{\sum_{j=1}^N p^{(j)}(\mathbf{x})} \\ &= \sum_{i=1}^N \frac{p^{(i)}(\mathbf{x}) \nabla_{\mathbf{x}} \log p^{(i)}(\mathbf{x})}{\sum_{j=1}^N p^{(j)}(\mathbf{x})} \\ &= \sum_{i=1}^N r^{(i)}(\mathbf{x}) \nabla_{\mathbf{x}} \log p^{(i)}(\mathbf{x}). \end{aligned}$$

Next, assuming $\mathbf{x} \in \mathbb{R}^D$, we have

$$\begin{aligned} \mathbb{E}_{p^{(i)}(\mathbf{x})} [r^{(j)}(\mathbf{x})] &= \int \frac{p^{(i)}(\mathbf{x}) p^{(j)}(\mathbf{x})}{\sum_{k=1}^N p^{(k)}(\mathbf{x})} d\mathbf{x} \leq \int \frac{p^{(i)}(\mathbf{x}) p^{(j)}(\mathbf{x})}{p^{(i)}(\mathbf{x}) + p^{(j)}(\mathbf{x})} d\mathbf{x} \\ &= \frac{1}{2} \int \frac{2}{\frac{1}{p^{(i)}(\mathbf{x})} + \frac{1}{p^{(j)}(\mathbf{x})}} d\mathbf{x} \stackrel{(1)}{\leq} \frac{1}{2} \int \sqrt{p^{(i)}(\mathbf{x}) p^{(j)}(\mathbf{x})} d\mathbf{x} \end{aligned}$$

$$\begin{aligned}
&= \frac{1}{2(2\pi\sigma_1^2)^{D/2}} \int \exp \left(-\frac{1}{4\sigma_1^2} \left(\|\mathbf{x} - \mathbf{x}^{(i)}\|_2^2 + \|\mathbf{x} - \mathbf{x}^{(j)}\|_2^2 \right) \right) d\mathbf{x} \\
&= \frac{1}{2(2\pi\sigma_1^2)^{D/2}} \int \exp \left(-\frac{1}{4\sigma_1^2} \left(\|\mathbf{x} - \mathbf{x}^{(i)}\|_2^2 + \|\mathbf{x} - \mathbf{x}^{(i)} + \mathbf{x}^{(i)} - \mathbf{x}^{(j)}\|_2^2 \right) \right) d\mathbf{x} \\
&= \frac{1}{2(2\pi\sigma_1^2)^{D/2}} \int \exp \left\{ -\frac{1}{2\sigma_1^2} \left(\|\mathbf{x} - \mathbf{x}^{(i)}\|_2^2 + (\mathbf{x} - \mathbf{x}^{(i)})^\top (\mathbf{x}^{(i)} - \mathbf{x}^{(j)}) + \frac{\|\mathbf{x}^{(i)} - \mathbf{x}^{(j)}\|_2^2}{2} \right) \right\} d\mathbf{x} \\
&= \frac{1}{2(2\pi\sigma_1^2)^{D/2}} \int \exp \left\{ -\frac{1}{2\sigma_1^2} \left(\left\| \mathbf{x} - \mathbf{x}^{(i)} + \frac{\mathbf{x}^{(i)} - \mathbf{x}^{(j)}}{2} \right\|_2^2 + \frac{\|\mathbf{x}^{(i)} - \mathbf{x}^{(j)}\|_2^2}{4} \right) \right\} d\mathbf{x} \\
&= \frac{1}{2} \exp \left(-\frac{\|\mathbf{x}^{(i)} - \mathbf{x}^{(j)}\|_2^2}{8\sigma_1^2} \right) \frac{1}{(2\pi\sigma_1^2)^{D/2}} \int \exp \left\{ -\frac{1}{2\sigma_1^2} \left(\left\| \mathbf{x} - \mathbf{x}^{(i)} + \frac{\mathbf{x}^{(i)} - \mathbf{x}^{(j)}}{2} \right\|_2^2 \right) \right\} d\mathbf{x} \\
&= \frac{1}{2} \exp \left(-\frac{\|\mathbf{x}^{(i)} - \mathbf{x}^{(j)}\|_2^2}{8\sigma_1^2} \right),
\end{aligned}$$

where (1) is due to the geometric mean–harmonic mean inequality. \square

Proposition 6.2. *Let $\mathbf{x} \in \mathbb{R}^D \sim \mathcal{N}(\mathbf{0}, \sigma^2 I)$, and $r = \|\mathbf{x}\|_2$. We have*

$$p(r) = \frac{1}{2^{D/2-1}\Gamma(D/2)} \frac{r^{D-1}}{\sigma^D} \exp \left(-\frac{r^2}{2\sigma^2} \right) \quad \text{and} \quad r - \sqrt{D}\sigma \xrightarrow{d} \mathcal{N}(0, \sigma^2/2) \quad \text{when } D \rightarrow \infty.$$

Proof. Since $\mathbf{x} \sim \mathcal{N}(\mathbf{0}, \sigma^2 I)$, we have $s := \|\mathbf{x}\|_2^2 / \sigma^2 \sim \chi_D^2$, i.e.,

$$p_s(s) = \frac{1}{2^{D/2}\Gamma(D/2)} s^{D/2-1} e^{-s/2}.$$

Because $r = \|\mathbf{x}\|_2 = \sigma\sqrt{s}$, we can use the change of variables formula to get

$$p(r) = \frac{2r}{\sigma^2} p_s(s) = \frac{2r}{\sigma^2} p_s \left(\frac{r^2}{\sigma^2} \right) = \frac{1}{2^{D/2-1}\Gamma(D/2)} \frac{r^{D-1}}{\sigma^D} \exp \left(-\frac{r^2}{2\sigma^2} \right),$$

which proves our first result. Next, we notice that if $x \sim \mathcal{N}(0, \sigma^2)$, we have $x^2/\sigma^2 \sim \chi_1^2$ and thus $\mathbb{E}[x] = \sigma^2$, $\text{Var}[x] = 2\sigma^4$. As a result, if $x_1, x_2, \dots, x_D \stackrel{\text{i.i.d.}}{\sim} \mathcal{N}(0, \sigma^2)$, the law of large numbers and the central limit theorem will imply that as $D \rightarrow \infty$, both of the following hold:

$$\begin{aligned}
&\frac{x_1^2 + x_2^2 + \dots + x_D^2}{D} \xrightarrow{p} \sigma^2 \\
&\sqrt{D} \left(\frac{x_1^2 + x_2^2 + \dots + x_D^2}{D} - \sigma^2 \right) \xrightarrow{d} \mathcal{N}(0, 2\sigma^4).
\end{aligned}$$

Equivalently,

$$\sqrt{D} \left(\frac{r^2}{D} - \sigma^2 \right) \xrightarrow{d} \mathcal{N}(0, 2\sigma^4).$$

Applying the delta method, we obtain

$$\sqrt{D} \left(\frac{r}{\sqrt{D}} - \sigma \right) \xrightarrow{d} \mathcal{N}(0, \sigma^2/2),$$

and therefore $r - \sqrt{D}\sigma \xrightarrow{d} \mathcal{N}(0, \sigma^2/2)$. \square

Proposition 6.3. Let $\gamma = \frac{\sigma_{i-1}}{\sigma_i}$. For $\alpha = \epsilon \cdot \frac{\sigma_i^2}{\sigma_L^2}$ (as in Algorithm 6.1), we have $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, s_T^2 I)$, where

$$\frac{s_T^2}{\sigma_i^2} = \left(1 - \frac{\epsilon}{\sigma_L^2}\right)^{2T} \left(\gamma^2 - \frac{2\epsilon}{\sigma_L^2 - \sigma_L^2 \left(1 - \frac{\epsilon}{\sigma_L^2}\right)^2} \right) + \frac{2\epsilon}{\sigma_L^2 - \sigma_L^2 \left(1 - \frac{\epsilon}{\sigma_L^2}\right)^2}. \quad (\text{A.26})$$

Proof. First, the conditions we know are

$$\begin{aligned} \mathbf{x}_0 &\sim p_{\sigma_{i-1}}(\mathbf{x}) = \mathcal{N}(\mathbf{0}, \sigma_{i-1}^2 I), \\ \mathbf{x}_{t+1} &\leftarrow \mathbf{x}_t + \alpha \nabla_{\mathbf{x}} \log p_{\sigma_i}(\mathbf{x}_t) + \sqrt{2\alpha} \mathbf{z}_t = \mathbf{x}_t - \alpha \frac{\mathbf{x}_t}{\sigma_i^2} + \sqrt{2\alpha} \mathbf{z}_t, \end{aligned}$$

where $\mathbf{z}_t \sim \mathcal{N}(\mathbf{0}, I)$. Therefore, the variance of \mathbf{x}_t satisfies

$$\text{Var}[\mathbf{x}_t] = \begin{cases} \sigma_{i-1}^2 I & \text{if } t = 0 \\ \left(1 - \frac{\alpha}{\sigma_i^2}\right)^2 \text{Var}[\mathbf{x}_{t-1}] + 2\alpha I & \text{otherwise.} \end{cases}$$

Now let $\mathbf{v} := \frac{2\alpha}{1 - \left(1 - \frac{\alpha}{\sigma_i^2}\right)^2} I$, we have

$$\text{Var}[\mathbf{x}_t] - \mathbf{v} = \left(1 - \frac{\alpha}{\sigma_i^2}\right)^2 (\text{Var}[\mathbf{x}_{t-1}] - \mathbf{v}).$$

Therefore,

$$\text{Var}[\mathbf{x}_T] - \mathbf{v} = \left(1 - \frac{\alpha}{\sigma_i^2}\right)^{2T} (\text{Var}[\mathbf{x}_0] - \mathbf{v})$$

$$\begin{aligned} \implies \text{Var}[\mathbf{x}_T] &= \left(1 - \frac{\alpha}{\sigma_i^2}\right)^{2T} (\text{Var}[\mathbf{x}_0] - \mathbf{v}) + \mathbf{v} \\ \implies s_T^2 &= \left(1 - \frac{\alpha}{\sigma_i^2}\right)^{2T} \left(\sigma_{i-1}^2 - \frac{2\alpha}{1 - (1 - \frac{\alpha}{\sigma_i^2})^2}\right) + \frac{2\alpha}{1 - (1 - \frac{\alpha}{\sigma_i^2})^2}. \end{aligned} \quad (\text{A.27})$$

Substituting $\epsilon \sigma_i^2 / \sigma_L^2$ for α in Eq. (A.27), we immediately obtain Eq. (A.26). \square

A.4 Proofs for Chapter 8

Proposition 8.1. *If $\text{rank}(\mathbf{A}) = m$, then there exist an invertible matrix $\mathbf{T} \in \mathbb{R}^{n \times n}$, and a diagonal matrix $\mathbf{\Lambda} \in \{0, 1\}^{n \times n}$ with $\text{tr}(\mathbf{\Lambda}) = m$, such that $\mathbf{A} = \mathcal{P}(\mathbf{\Lambda})\mathbf{T}$. Here $\mathcal{P}(\mathbf{\Lambda}) \in \{0, 1\}^{m \times n}$ is an operator that, when multiplied with any vector $\mathbf{a} \in \mathbb{R}^n$, reduces its dimensionality to m by removing each i -th element of \mathbf{a} for $i = 1, 2, \dots, n$ if $\mathbf{\Lambda}_{ii} = 0$.*

Proof. Let $\mathbf{A} = (\mathbf{a}_1^\top, \mathbf{a}_2^\top, \dots, \mathbf{a}_m^\top) \in \mathbb{R}^{m \times n}$. Since \mathbf{A} has full rank, the row vectors $\{\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_m\}$ are linearly independent. We can therefore extend them to a total of n linearly independent vectors, i.e., $\{\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_m, \mathbf{b}_1, \dots, \mathbf{b}_{n-m}\}$. Due to the linear independence, we know $\mathbf{T} = (\mathbf{a}_1^\top, \mathbf{a}_2^\top, \dots, \mathbf{a}_m^\top, \mathbf{b}_1^\top, \dots, \mathbf{b}_{n-m}^\top) \in \mathbb{R}^{n \times n}$ has full rank and is invertible. Next, we define

$$\mathbf{\Lambda} = \text{diag}(\underbrace{1, 1, \dots, 1}_m, \underbrace{0, 0, \dots, 0}_{n-m}),$$

where diag converts a vector to a diagonal matrix. Clearly $\text{tr}(\mathbf{\Lambda}) = m$ and $\mathbf{A} = \mathcal{P}(\mathbf{\Lambda})\mathbf{T}$, which completes the proof. \square

Lemma A.7. *Let $\mathcal{P}^{-1}(\mathbf{\Lambda}) : \mathbb{R}^m \rightarrow \mathbb{R}^n$ be any right inverse of $\mathcal{P}(\mathbf{\Lambda}) : \mathbb{R}^n \rightarrow \mathbb{R}^m$. For any $\mathbf{u} \in \mathbb{R}^n$ and $\hat{\mathbf{y}}_t \in \mathbb{R}^m$, we have*

$$\mathcal{P}(\mathbf{\Lambda})\mathbf{T}\mathbf{u} = \hat{\mathbf{y}}_t \iff \mathbf{\Lambda}\mathbf{T}\mathbf{u} = \mathbf{\Lambda}\mathcal{P}^{-1}(\mathbf{\Lambda})\hat{\mathbf{y}}_t$$

Proof. By the definition of $\mathcal{P}(\mathbf{\Lambda})$, we have $\mathcal{P}(\mathbf{\Lambda}) = \mathcal{P}(\mathbf{\Lambda})\mathbf{\Lambda}$, and

$$\forall \mathbf{a} \in \mathbb{R}^n, \mathbf{b} \in \mathbb{R}^n : \quad \mathcal{P}(\mathbf{\Lambda})\mathbf{a} = \mathcal{P}(\mathbf{\Lambda})\mathbf{b} \iff \mathbf{\Lambda}\mathbf{a} = \mathbf{\Lambda}\mathbf{b}. \quad (\text{A.28})$$

To prove the “if” direction, we note that

$$\mathbf{\Lambda}\mathbf{T}\mathbf{u} = \mathbf{\Lambda}\mathcal{P}^{-1}(\mathbf{\Lambda})\hat{\mathbf{y}}_t \implies \mathcal{P}(\mathbf{\Lambda})\mathbf{\Lambda}\mathbf{T}\mathbf{u} = \mathcal{P}(\mathbf{\Lambda})\mathbf{\Lambda}\mathcal{P}^{-1}(\mathbf{\Lambda})\hat{\mathbf{y}}_t$$

$$\begin{aligned} &\implies \mathcal{P}(\boldsymbol{\Lambda})\mathbf{T}\mathbf{u} = \mathcal{P}(\boldsymbol{\Lambda})\mathcal{P}^{-1}(\boldsymbol{\Lambda})\hat{\mathbf{y}}_t \\ &\implies \mathcal{P}(\boldsymbol{\Lambda})\mathbf{T}\mathbf{u} = \hat{\mathbf{y}}_t. \end{aligned}$$

To prove the “only if” direction, we have

$$\begin{aligned} \mathcal{P}(\boldsymbol{\Lambda})\mathbf{T}\mathbf{u} = \hat{\mathbf{y}}_t &\implies \mathcal{P}(\boldsymbol{\Lambda})\mathbf{T}\mathbf{u} = \mathcal{P}(\boldsymbol{\Lambda})\mathcal{P}^{-1}(\boldsymbol{\Lambda})\hat{\mathbf{y}}_t \\ &\stackrel{(i)}{\implies} \boldsymbol{\Lambda}\mathbf{T}\mathbf{u} = \boldsymbol{\Lambda}\mathcal{P}^{-1}(\boldsymbol{\Lambda})\hat{\mathbf{y}}_t, \end{aligned}$$

where (i) is due to the property in Eq. (A.28). This completes the proof for both directions. \square

Theorem 9.2. Suppose $p(\mathbf{x})$ and $q(\mathbf{x})$ have continuous second-order derivatives and finite second moments. Let $\{\mathbf{x}(t)\}_{t \in [0, T]}$ be the diffusion process defined by the SDE in Eq. (9.1). We use p_t and q_t to denote the distributions of $\mathbf{x}(t)$ when $\mathbf{x}(0) \sim p$ and $\mathbf{x}(0) \sim q$, and assume they satisfy the same assumptions in Appendix A.5. Under the conditions $q_T = \pi$ and $s_{\boldsymbol{\theta}}(\mathbf{x}, t) \equiv \nabla_{\mathbf{x}} \log q_t(\mathbf{x})$ for all $t \in [0, T]$, we have the following equivalence in distributions

$$p_{\boldsymbol{\theta}}^{\text{SDE}} = p_{\boldsymbol{\theta}}^{\text{ODE}} = q. \quad (9.9)$$

Moreover, we have

$$D_{\text{KL}}(p \parallel p_{\boldsymbol{\theta}}^{\text{SDE}}) = \mathcal{J}_{\text{SM}}(\boldsymbol{\theta}; g(\cdot)^2) + D_{\text{KL}}(p_T \parallel \pi). \quad (9.10)$$

Proof. The optimization objective function in Eq. (8.8) can be written as

$$\begin{aligned} &(1 - \lambda) \|\mathbf{z} - \hat{\mathbf{x}}_t\|_{\mathbf{T}}^2 + \lambda \|\mathbf{z} - \mathbf{u}\|_{\mathbf{T}}^2 \\ &= (1 - \lambda) \|\mathbf{T}\mathbf{z} - \mathbf{T}\hat{\mathbf{x}}_t\|_2^2 + \lambda \|\mathbf{T}\mathbf{z} - \mathbf{T}\mathbf{u}\|_2^2 \\ &= (1 - \lambda) \|\mathbf{T}\mathbf{z} - \mathbf{T}\hat{\mathbf{x}}_t\|_2^2 + \lambda \|\boldsymbol{\Lambda}\mathbf{T}(\mathbf{z} - \mathbf{u}) + (\mathbf{I} - \boldsymbol{\Lambda})\mathbf{T}(\mathbf{z} - \mathbf{u})\|_2^2 \\ &= (1 - \lambda) \|\mathbf{T}\mathbf{z} - \mathbf{T}\hat{\mathbf{x}}_t\|_2^2 + \lambda \|\boldsymbol{\Lambda}\mathbf{T}(\mathbf{z} - \mathbf{u})\|_2^2 + \lambda \|(\mathbf{I} - \boldsymbol{\Lambda})\mathbf{T}(\mathbf{z} - \mathbf{u})\|_2^2 \\ &= (1 - \lambda) \|\mathbf{T}\mathbf{z} - \mathbf{T}\hat{\mathbf{x}}_t\|_2^2 + \lambda \|\boldsymbol{\Lambda}\mathbf{T}\mathbf{z} - \boldsymbol{\Lambda}\mathcal{P}^{-1}(\boldsymbol{\Lambda})\hat{\mathbf{y}}_t\|_2^2 + \lambda \|(\mathbf{I} - \boldsymbol{\Lambda})\mathbf{T}(\mathbf{z} - \mathbf{u})\|_2^2 \end{aligned}$$

Since $\mathbf{A}\mathbf{u} = \hat{\mathbf{y}}_t$, we have $\mathcal{P}(\boldsymbol{\Lambda})\mathbf{T}\mathbf{u} = \hat{\mathbf{y}}_t$ and equivalently $\boldsymbol{\Lambda}\mathbf{T}\mathbf{u} = \boldsymbol{\Lambda}\mathcal{P}^{-1}(\boldsymbol{\Lambda})\hat{\mathbf{y}}_t$ due to Lemma A.7. This constraint does not restrict the value of $(\mathbf{I} - \boldsymbol{\Lambda})\mathbf{T}\mathbf{u}$. Therefore, when

$\mathbf{A}\mathbf{u} = \hat{\mathbf{y}}_t$, we have

$$\begin{aligned}
& \|z - \hat{x}_t\|_{\mathbf{T}}^2 + \min_{\mathbf{u}} (1 - \lambda) \lambda \|z - \mathbf{u}\|_{\mathbf{T}}^2 \\
&= (1 - \lambda) \|\mathbf{T}z - \mathbf{T}\hat{x}_t\|_2^2 + \min_{\mathbf{u}} \lambda \|\Lambda \mathbf{T}z - \Lambda \mathcal{P}^{-1}(\Lambda)\hat{y}_t\|_2^2 + \lambda \|(I - \Lambda)\mathbf{T}(z - \mathbf{u})\|_2^2 \\
&= (1 - \lambda) \|\mathbf{T}z - \mathbf{T}\hat{x}_t\|_2^2 + \lambda \|\Lambda \mathbf{T}z - \Lambda \mathcal{P}^{-1}(\Lambda)\hat{y}_t\|_2^2 \\
&= (1 - \lambda) \|\Lambda \mathbf{T}z - \Lambda \mathbf{T}\hat{x}_t\|_2^2 + \lambda \|\Lambda \mathbf{T}z - \Lambda \mathcal{P}^{-1}(\Lambda)\hat{y}_t\|_2^2 \\
&\quad + (1 - \lambda) \|(I - \Lambda)\mathbf{T}z - (I - \Lambda)\mathbf{T}\hat{x}_t\|_2^2.
\end{aligned}$$

This simplifies the optimization problem in Eq. (8.8) to

$$\begin{aligned}
& \min_{\mathbf{z}} (1 - \lambda) \|\Lambda \mathbf{T}z - \Lambda \mathbf{T}\hat{x}_t\|_2^2 + \lambda \|\Lambda \mathbf{T}z - \Lambda \mathcal{P}^{-1}(\Lambda)\hat{y}_t\|_2^2 \\
&\quad + (1 - \lambda) \|(I - \Lambda)\mathbf{T}z - (I - \Lambda)\mathbf{T}\hat{x}_t\|_2^2,
\end{aligned}$$

which is minimizing a quadratic function of \mathbf{z} . The optimal solution \mathbf{z}^* is thus in closed form:

$$\mathbf{z}^* = \mathbf{T}^{-1}[(I - \Lambda)\mathbf{T}\hat{x}_t + (1 - \lambda)\Lambda \mathbf{T}\hat{x}_t + \lambda \Lambda \mathcal{P}^{-1}(\Lambda)\hat{y}_t].$$

According to the definition, $\hat{x}'_t = \mathbf{z}^*$, whereby the proof is completed. \square

A.5 Proofs for Chapter 9

We first summarize the notations and assumptions used in our theorems.

Notations The drift and diffusion coefficients of the SDE in Eq. (9.1) are denoted as $\mathbf{f} : \mathbb{R}^D \times [0, T] \rightarrow \mathbb{R}^D$ and $\mathbf{g} : [0, T] \rightarrow \mathbb{R}$ respectively, where $[0, T]$ represents a fixed time horizon, and \times denotes the Cartesian product. The solution to Eq. (9.1) is a stochastic process $\{\mathbf{x}(t)\}_{t \in [0, T]}$. We use p_t to represent the marginal distribution of $\mathbf{x}(t)$, and $p_{0t}(\mathbf{x}' | \mathbf{x})$ to denote the transition distribution from $\mathbf{x}(0)$ to $\mathbf{x}(t)$. The data distribution and prior distribution are given by p and π . We use \mathcal{C} to denote all continuous functions, and let \mathcal{C}^k denote the family of functions with continuous k -th order derivatives. For any vector-valued function $\mathbf{h} : \mathbb{R}^D \times [0, T] \rightarrow \mathbb{R}^D$, we use $\nabla \cdot \mathbf{h}(\mathbf{x}, t)$ to represent its divergence with respect to the first input variable.

Assumptions We make the following assumptions throughout the paper:

- (i) $p(\mathbf{x}) \in \mathcal{C}^2$ and $\mathbb{E}_{\mathbf{x} \sim p} [\|\mathbf{x}\|_2^2] < \infty$.
- (ii) $\pi(\mathbf{x}) \in \mathcal{C}^2$ and $\mathbb{E}_{\mathbf{x} \sim \pi} [\|\mathbf{x}\|_2^2] < \infty$.
- (iii) $\forall t \in [0, T] : \mathbf{f}(\cdot, t) \in \mathcal{C}^1, \exists C > 0 \forall \mathbf{x} \in \mathbb{R}^D, t \in [0, T] : \|\mathbf{f}(\mathbf{x}, t)\|_2 \leq C(1 + \|\mathbf{x}\|_2)$.
- (iv) $\exists C > 0, \forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^D : \|\mathbf{f}(\mathbf{x}, t) - \mathbf{f}(\mathbf{y}, t)\|_2 \leq C \|\mathbf{x} - \mathbf{y}\|_2$.
- (v) $g \in \mathcal{C}$ and $\forall t \in [0, T], |g(t)| > 0$.
- (vi) For any open bounded set \mathcal{O} , $\int_0^T \int_{\mathcal{O}} \|p_t(\mathbf{x})\|_2^2 + Dg(t)^2 \|\nabla_{\mathbf{x}} p_t(\mathbf{x})\|_2^2 d\mathbf{x} dt < \infty$.
- (vii) $\exists C > 0 \forall \mathbf{x} \in \mathbb{R}^D, t \in [0, T] : \|\nabla_{\mathbf{x}} \log p_t(\mathbf{x})\|_2 \leq C(1 + \|\mathbf{x}\|_2)$.
- (viii) $\exists C > 0, \forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^D : \|\nabla_{\mathbf{x}} \log p_t(\mathbf{x}) - \nabla_{\mathbf{y}} \log p_t(\mathbf{y})\|_2 \leq C \|\mathbf{x} - \mathbf{y}\|_2$.
- (ix) $\exists C > 0 \forall \mathbf{x} \in \mathbb{R}^D, t \in [0, T] : \|\mathbf{s}_{\theta}(\mathbf{x}, t)\|_2 \leq C(1 + \|\mathbf{x}\|_2)$.
- (x) $\exists C > 0, \forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^D : \|\mathbf{s}_{\theta}(\mathbf{x}, t) - \mathbf{s}_{\theta}(\mathbf{y}, t)\|_2 \leq C \|\mathbf{x} - \mathbf{y}\|_2$.
- (xi) Novikov's condition: $\mathbb{E} \left[\exp \left(\frac{1}{2} \int_0^T \|\nabla_{\mathbf{x}} \log p_t(\mathbf{x}) - \mathbf{s}_{\theta}(\mathbf{x}, t)\|_2^2 dt \right) \right] < \infty$.
- (xii) $\forall t \in [0, T] \exists k > 0 : p_t(\mathbf{x}) = O(e^{-\|\mathbf{x}\|_2^k})$ as $\|\mathbf{x}\|_2 \rightarrow \infty$.

Below we provide all proofs for our theorems.

Theorem 9.1. *Let $p(\mathbf{x})$ be the data distribution, $\pi(\mathbf{x})$ be a known prior distribution, and p_{θ}^{SDE} be defined as in Section 9.3. Suppose $\{\mathbf{x}(t)\}_{t \in [0, T]}$ is a stochastic process defined by the SDE in Eq. (9.1) with $\mathbf{x}(0) \sim p$, where the marginal distribution of $\mathbf{x}(t)$ is denoted as p_t . Under some regularity conditions detailed in Appendix A.5, we have*

$$D_{\text{KL}}(p \parallel p_{\theta}^{\text{SDE}}) \leq \mathcal{J}_{\text{SM}}(\boldsymbol{\theta}; g(\cdot)^2) + D_{\text{KL}}(p_T \parallel \pi). \quad (9.8)$$

Proof. We denote the path measure of $\{\mathbf{x}(t)\}_{t \in [0, T]}$ and $\{\hat{\mathbf{x}}_{\theta}(t)\}_{t \in [0, T]}$ as $\boldsymbol{\mu}$ and $\boldsymbol{\nu}$ respectively. Due to assumptions (i) (ii) (iii) (iv) (v) (ix) and (x), both $\boldsymbol{\mu}$ and $\boldsymbol{\nu}$ are uniquely given by the corresponding SDEs. Consider a Markov kernel $K(\{\mathbf{z}(t)\}_{t \in [0, t]}, \mathbf{y}) := \delta(\mathbf{z}(0) = \mathbf{y})$. Since $\mathbf{x}(0) \sim p_0$ and $\hat{\mathbf{x}}_{\theta}(0) \sim p_{\theta}$, we have the following result

$$\int K(\{\mathbf{x}(t)\}_{t \in [0, T]}, \mathbf{x}) d\boldsymbol{\mu}(\{\mathbf{x}(t)\}_{t \in [0, T]}) = p_0(\mathbf{x})$$

$$\int K(\{\hat{\mathbf{x}}_{\theta}(t)\}_{t \in [0,T]}, \mathbf{x}) d\nu(\{\hat{\mathbf{x}}_{\theta}(t)\}_{t \in [0,T]}) = p_{\theta}(\mathbf{x}).$$

Here the Markov kernel K essentially performs marginalization of path measures to obtain “sliced” distributions at $t = 0$. We can use the data processing inequality with this Markov kernel to obtain

$$\begin{aligned} D_{\text{KL}}(p \parallel p_{\theta}) &= D_{\text{KL}}(p_0 \parallel p_{\theta}) \\ &= D_{\text{KL}}\left(\int K(\{\mathbf{x}(t)\}_{t \in [0,T]}, \mathbf{x}) d\mu(\{\mathbf{x}(t)\}_{t \in [0,T]}) \middle\| \int K(\{\hat{\mathbf{x}}_{\theta}(t)\}_{t \in [0,T]}, \mathbf{x}) d\nu(\{\hat{\mathbf{x}}_{\theta}(t)\}_{t \in [0,T]})\right) \\ &\leq D_{\text{KL}}(\mu \parallel \nu). \end{aligned} \quad (\text{A.29})$$

Recall that by definition $\mathbf{x}(T) \sim p_T$ and $\hat{\mathbf{x}}_{\theta}(T) \sim \pi$. Leveraging the chain rule of KL divergences (see, for example, Theorem 2.4 in [Léo14]), we have

$$D_{\text{KL}}(\mu \parallel \nu) = D_{\text{KL}}(p_T \parallel \pi) + \mathbb{E}_{\mathbf{z} \sim p_T}[D_{\text{KL}}(\mu(\cdot | \mathbf{x}(T) = \mathbf{z}) \parallel \nu(\cdot | \hat{\mathbf{x}}_{\theta}(T) = \mathbf{z}))]. \quad (\text{A.30})$$

Under assumptions (i) (iii) (iv) (v) (vi) (vii) (viii), the SDE in Eq. (9.1) has a corresponding reverse-time SDE given by

$$d\mathbf{x} = [\mathbf{f}(\mathbf{x}, t) - g(t)^2 \nabla_{\mathbf{x}} \log p_t(\mathbf{x})] dt + g(t) d\bar{\mathbf{w}}. \quad (\text{A.31})$$

Since Eq. (A.31) is the time reversal of Eq. (9.1), it induces the same path measure μ . As a result, $D_{\text{KL}}(\mu(\cdot | \mathbf{x}(T) = \mathbf{z}) \parallel \nu(\cdot | \hat{\mathbf{x}}_{\theta}(T) = \mathbf{z}))$ can be viewed as the KL divergence between the path measures induced by the following two (reverse-time) SDEs:

$$\begin{aligned} d\mathbf{x} &= [\mathbf{f}(\mathbf{x}, t) - g(t)^2 \nabla_{\mathbf{x}} \log p_t(\mathbf{x})] dt + g(t) d\bar{\mathbf{w}}, \quad \mathbf{x}(T) = \mathbf{x} \\ d\hat{\mathbf{x}} &= [\mathbf{f}(\hat{\mathbf{x}}, t) - g(t)^2 s_{\theta}(\hat{\mathbf{x}}, t)] dt + g(t) d\bar{\mathbf{w}}, \quad \hat{\mathbf{x}}_{\theta}(T) = \mathbf{x}. \end{aligned}$$

The KL divergence between two SDEs with shared diffusion coefficients and starting points exists under assumptions (vii) (viii) (ix) (x) (xi) (see, e.g., [TR19; Li+20]), and can be computed via the Girsanov theorem [Oks13]:

$$\begin{aligned} D_{\text{KL}}(\mu(\cdot | \mathbf{x}(T) = \mathbf{z}) \parallel \nu(\cdot | \hat{\mathbf{x}}_{\theta}(T) = \mathbf{z})) \\ = -\mathbb{E}_{\mu}\left[\log \frac{d\nu}{d\mu}\right] \end{aligned} \quad (\text{A.32})$$

$$\begin{aligned}
&\stackrel{(i)}{=} \mathbb{E}_{\boldsymbol{\mu}} \left[\int_0^T g(t) (\nabla_{\mathbf{x}} \log p_t(\mathbf{x}) - \mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}, t)) d\bar{\mathbf{w}}_t + \frac{1}{2} \int_0^T g(t)^2 \|\nabla_{\mathbf{x}} \log p_t(\mathbf{x}) - \mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}, t)\|_2^2 dt \right] \\
&\stackrel{(ii)}{=} \mathbb{E}_{\boldsymbol{\mu}} \left[\frac{1}{2} \int_0^T g(t)^2 \|\nabla_{\mathbf{x}} \log p_t(\mathbf{x}) - \mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}, t)\|_2^2 dt \right] \\
&= \frac{1}{2} \int_0^T \mathbb{E}_{p_t(\mathbf{x})} [g(t)^2 \|\nabla_{\mathbf{x}} \log p_t(\mathbf{x}) - \mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}, t)\|_2^2] dt \\
&= \mathcal{J}_{\text{SM}}(\boldsymbol{\theta}; g(\cdot)^2),
\end{aligned} \tag{A.33}$$

where (i) is due to Girsanov Theorem II [Oks13, Theorem 8.6.6], and (ii) is due to the martingale property of Itô integrals. Combining Eqs. (A.29), (A.30) and (A.33) completes the proof. \square

Theorem 9.2. Suppose $p(\mathbf{x})$ and $q(\mathbf{x})$ have continuous second-order derivatives and finite second moments. Let $\{\mathbf{x}(t)\}_{t \in [0, T]}$ be the diffusion process defined by the SDE in Eq. (9.1). We use p_t and q_t to denote the distributions of $\mathbf{x}(t)$ when $\mathbf{x}(0) \sim p$ and $\mathbf{x}(0) \sim q$, and assume they satisfy the same assumptions in Appendix A.5. Under the conditions $q_T = \pi$ and $\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}, t) \equiv \nabla_{\mathbf{x}} \log q_t(\mathbf{x})$ for all $t \in [0, T]$, we have the following equivalence in distributions

$$p_{\boldsymbol{\theta}}^{\text{SDE}} = p_{\boldsymbol{\theta}}^{\text{ODE}} = q. \tag{9.9}$$

Moreover, we have

$$D_{\text{KL}}(p \parallel p_{\boldsymbol{\theta}}^{\text{SDE}}) = \mathcal{J}_{\text{SM}}(\boldsymbol{\theta}; g(\cdot)^2) + D_{\text{KL}}(p_T \parallel \pi). \tag{9.10}$$

Proof. When $\pi = q_T$ and $\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}, t) \equiv \nabla_{\mathbf{x}} \log q_t(\mathbf{x})$, the reverse-time SDE that defines $p_{\boldsymbol{\theta}}^{\text{SDE}}$, i.e.,

$$d\hat{\mathbf{x}} = [\mathbf{f}(\hat{\mathbf{x}}, t) - g(t)^2 \mathbf{s}_{\boldsymbol{\theta}}(\hat{\mathbf{x}}, t)] dt + g(t) d\bar{\mathbf{w}}, \quad \hat{\mathbf{x}}_{\boldsymbol{\theta}}(T) \sim \pi, \tag{A.34}$$

becomes equivalent to

$$d\hat{\mathbf{x}} = [\mathbf{f}(\hat{\mathbf{x}}, t) - g(t)^2 \nabla_{\hat{\mathbf{x}}} \log q_t(\hat{\mathbf{x}})] dt + g(t) d\bar{\mathbf{w}}, \quad \hat{\mathbf{x}}_{\boldsymbol{\theta}}(T) \sim q_T, \tag{A.35}$$

which yields the same stochastic process as the following forward-time SDE

$$d\hat{\mathbf{x}} = \mathbf{f}(\hat{\mathbf{x}}, t) dt + g(t) d\mathbf{w}, \quad \hat{\mathbf{x}}_{\boldsymbol{\theta}}(0) \sim q. \tag{A.36}$$

Since $\hat{\mathbf{x}}_{\theta}(0) \sim p_{\theta}^{\text{SDE}}$ by definition, we immediately have $p_{\theta}^{\text{SDE}} = q$. Similarly, the ODE that defines p_{θ}^{ODE} is

$$\frac{d\tilde{\mathbf{x}}}{dt} = \mathbf{f}_{\theta}(\tilde{\mathbf{x}}, t) - \frac{1}{2}g(t)^2\mathbf{s}_{\theta}(\tilde{\mathbf{x}}, t), \quad \tilde{\mathbf{x}}_{\theta}(T) \sim \pi, \quad (\text{A.37})$$

which is equivalent to the following when $q_T = \pi$ and $\mathbf{s}_{\theta}(\mathbf{x}, t) \equiv \nabla_{\mathbf{x}} \log q_t(\mathbf{x})$,

$$\frac{d\tilde{\mathbf{x}}}{dt} = \mathbf{f}_{\theta}(\tilde{\mathbf{x}}, t) - \frac{1}{2}g(t)^2\nabla_{\tilde{\mathbf{x}}} \log q_t(\tilde{\mathbf{x}}, t), \quad \tilde{\mathbf{x}}_{\theta}(T) \sim q_T. \quad (\text{A.38})$$

The theory of probability flow ODEs [Son+21b] (Chapter 7) guarantees that Eq. (A.36) and Eq. (A.38) share the same set of marginal distributions, $\{q_t\}_{t \in [0, T]}$, which implies that $\tilde{\mathbf{x}}_{\theta}(0) \sim q$. Since by definition $\tilde{\mathbf{x}}_{\theta}(0) \sim p_{\theta}^{\text{ODE}}$, we have $p_{\theta}^{\text{ODE}} = q$.

The next part of the theorem can be proved by first rewriting the KL divergence from p to q in an integral form:

$$\begin{aligned} D_{\text{KL}}(p(\mathbf{x}) \parallel q(\mathbf{x})) &\stackrel{(i)}{=} D_{\text{KL}}(p_0(\mathbf{x}) \parallel q_0(\mathbf{x})) - D_{\text{KL}}(p_T(\mathbf{x}) \parallel q_T(\mathbf{x})) + D_{\text{KL}}(p_T(\mathbf{x}) \parallel q_T(\mathbf{x})) \\ &\stackrel{(ii)}{=} \int_T^0 \frac{\partial D_{\text{KL}}(p_t(\mathbf{x}) \parallel q_t(\mathbf{x}))}{\partial t} dt + D_{\text{KL}}(p_T(\mathbf{x}) \parallel q_T(\mathbf{x})), \end{aligned} \quad (\text{A.39})$$

where (i) holds due to our definition $p_0(\mathbf{x}) \equiv p(\mathbf{x})$ and $q_0(\mathbf{x}) \equiv q(\mathbf{x})$; (ii) is due to the fundamental theorem of calculus.

Next, we show how to rewrite Eq. (A.39) as a mixture of score matching losses. The Fokker–Planck equation for the SDE in Eq. (9.1) describes the time-evolution of the stochastic process's associated probability density function, and is given by

$$\frac{\partial p_t(\mathbf{x})}{\partial t} = \nabla_{\mathbf{x}} \cdot \left(\frac{1}{2}g^2(t)p_t(\mathbf{x})\nabla_{\mathbf{x}} \log p_t(\mathbf{x}) - \mathbf{f}(\mathbf{x}, t)p_t(\mathbf{x}) \right) = \nabla_{\mathbf{x}} \cdot (\mathbf{h}_p(\mathbf{x}, t)p_t(\mathbf{x})),$$

where for simplified notations we define $\mathbf{h}_p(\mathbf{x}, t) := \frac{1}{2}g^2(t)\nabla_{\mathbf{x}} \log p_t(\mathbf{x}) - \mathbf{f}(\mathbf{x}, t)$. Similarly, $\frac{\partial q_t(\mathbf{x})}{\partial t} = \nabla_{\mathbf{x}} \cdot (\mathbf{h}_q(\mathbf{x}, t)q_t(\mathbf{x}))$. Since we assume $\log p_t(\mathbf{x})$ and $\log q_t(\mathbf{x})$ are smooth functions with at most polynomial growth at infinity (assumption (xii)), we have $\lim_{\mathbf{x} \rightarrow \infty} \mathbf{h}_p(\mathbf{x}, t)p_t(\mathbf{x}) = \mathbf{0}$ and $\lim_{\mathbf{x} \rightarrow \infty} \mathbf{h}_q(\mathbf{x}, t)q_t(\mathbf{x}) = \mathbf{0}$ for all t . Then, the time-derivative of $D_{\text{KL}}(p_t \parallel q_t)$ can be rewritten in the following way:

$$\frac{\partial D_{\text{KL}}(p_t(\mathbf{x}) \parallel q_t(\mathbf{x}))}{\partial t} = \frac{\partial}{\partial t} \int p_t(\mathbf{x}) \log \frac{p_t(\mathbf{x})}{q_t(\mathbf{x})} d\mathbf{x}$$

$$\begin{aligned}
&= \int \frac{\partial p_t(\mathbf{x})}{\partial t} \log \frac{p_t(\mathbf{x})}{q_t(\mathbf{x})} d\mathbf{x} + \underbrace{\int \frac{\partial p_t(\mathbf{x})}{\partial t} d\mathbf{x}}_{=0} - \int \frac{p_t(\mathbf{x})}{q_t(\mathbf{x})} \frac{\partial q_t(\mathbf{x})}{\partial t} d\mathbf{x} \\
&= \int \nabla_{\mathbf{x}} \cdot (\mathbf{h}_p(\mathbf{x}, t) p_t(\mathbf{x})) \log \frac{p_t(\mathbf{x})}{q_t(\mathbf{x})} d\mathbf{x} - \int \frac{p_t(\mathbf{x})}{q_t(\mathbf{x})} \nabla_{\mathbf{x}} \cdot (\mathbf{h}_q(\mathbf{x}, t) q_t(\mathbf{x})) d\mathbf{x} \\
&\stackrel{(i)}{=} - \int p_t(\mathbf{x}) [\mathbf{h}_p^T(\mathbf{x}, t) - \mathbf{h}_q^T(\mathbf{x}, t)] [\nabla_{\mathbf{x}} \log p_t(\mathbf{x}) - \nabla_{\mathbf{x}} \log q_t(\mathbf{x})] d\mathbf{x} \\
&= - \frac{1}{2} \int p_t(\mathbf{x}) g(t)^2 \|\nabla_{\mathbf{x}} \log p_t(\mathbf{x}) - \nabla_{\mathbf{x}} \log q_t(\mathbf{x})\|_2^2 d\mathbf{x},
\end{aligned}$$

where (i) is due to integration by parts. Combining with Eq. (A.39), we can conclude that

$$D_{\text{KL}}(p \parallel q) = \frac{1}{2} \int_0^T \mathbb{E}_{\mathbf{x} \sim p_t(\mathbf{x})} [g(t)^2 \|\nabla_{\mathbf{x}} \log p_t(\mathbf{x}) - \nabla_{\mathbf{x}} \log q_t(\mathbf{x})\|_2^2] dt + D_{\text{KL}}(p_T \parallel q_T). \quad (\text{A.40})$$

Since $p_{\boldsymbol{\theta}}^{\text{SDE}} = q$ and $q_T = \pi$, we also have

$$\begin{aligned}
D_{\text{KL}}(p \parallel p_{\boldsymbol{\theta}}^{\text{SDE}}) &= \frac{1}{2} \int_0^T \mathbb{E}_{\mathbf{x} \sim p_t(\mathbf{x})} [g(t)^2 \|\nabla_{\mathbf{x}} \log p_t(\mathbf{x}) - \nabla_{\mathbf{x}} \log q_t(\mathbf{x})\|_2^2] dt + D_{\text{KL}}(p_T \parallel q_T) \\
&= \mathcal{J}_{\text{SM}}(\boldsymbol{\theta}; g(\cdot)^2) + D_{\text{KL}}(p_T \parallel q_T),
\end{aligned} \quad (\text{A.41})$$

which completes the proof. \square

Using a similar technique to Theorem 9.2, we can express the entropy of a distribution in terms of a time-dependent score function, as detailed in the following theorem.

Theorem A.1. *Let $\mathcal{H}(p(\mathbf{x}))$ be the differential entropy of the initial probability density $p(\mathbf{x})$. Under the same conditions in Theorem 9.2, we have*

$$\begin{aligned}
\mathcal{H}(p(\mathbf{x})) &= \mathcal{H}(p_T(\mathbf{x})) + \frac{1}{2} \int_0^T \mathbb{E}_{\mathbf{x} \sim p_t(\mathbf{x})} \left[2\mathbf{f}(\mathbf{x}, t)^T \nabla_{\mathbf{x}} \log p_t(\mathbf{x}) - g(t)^2 \|\nabla_{\mathbf{x}} \log p_t(\mathbf{x})\|_2^2 \right] dt. \\
&= \mathcal{H}(p_T(\mathbf{x})) - \frac{1}{2} \int_0^T \mathbb{E}_{\mathbf{x} \sim p_t(\mathbf{x})} \left[2\nabla \cdot \mathbf{f}(\mathbf{x}, t) + g(t)^2 \|\nabla_{\mathbf{x}} \log p_t(\mathbf{x})\|_2^2 \right] dt. \quad (\text{A.43})
\end{aligned}$$

Proof. Once more we proceed analogously to the proofs of Theorem 9.2. We have

$$\mathcal{H}(p(\mathbf{x})) - \mathcal{H}(p_T(\mathbf{x})) = \int_T^0 \frac{\partial}{\partial t} \mathcal{H}(p_t(\mathbf{x})) dt. \quad (\text{A.44})$$

Expanding the integrand, we have

$$\begin{aligned}
\frac{\partial}{\partial t} \mathcal{H}(p_t(\mathbf{x})) &= -\frac{\partial}{\partial t} \int p_t(\mathbf{x}) \log p_t(\mathbf{x}) d\mathbf{x} \\
&= -\int \frac{\partial p_t(\mathbf{x})}{\partial t} \log p_t(\mathbf{x}) + \frac{\partial p_t(\mathbf{x})}{\partial t} d\mathbf{x} \\
&= -\int \frac{\partial p_t(\mathbf{x})}{\partial t} \log p_t(\mathbf{x}) d\mathbf{x} - \frac{\partial}{\partial t} \underbrace{\int p_t(\mathbf{x}) d\mathbf{x}}_{=1} \\
&= -\int \nabla_{\mathbf{x}} \cdot (\mathbf{h}_p(\mathbf{x}, t) p_t(\mathbf{x})) \log p_t(\mathbf{x}) d\mathbf{x} \\
&\stackrel{(i)}{=} \int p_t(\mathbf{x}) \mathbf{h}_p^T(\mathbf{x}, t) \nabla_{\mathbf{x}} \log p_t(\mathbf{x}) d\mathbf{x} \\
&= \frac{1}{2} \mathbb{E}_{\mathbf{x} \sim p_t(\mathbf{x})} [g(t)^2 \|\nabla_{\mathbf{x}} \log p_t(\mathbf{x})\|_2^2 - 2\mathbf{f}(\mathbf{x}, t)^T \nabla_{\mathbf{x}} \log p_t(\mathbf{x})],
\end{aligned}$$

where again (i) follows from integration by parts and the limiting behaviour of \mathbf{h}_p given by assumption (xii). Plugging this expression in for the integrand in Eq. (A.44) then completes the proof for Eq. (A.42). For Eq. (A.43), we can once again perform integration by parts and leverage the limiting behavior of $p_t(\mathbf{x})$ in assumption (xii) to get

$$\mathbb{E}_{p_t(\mathbf{x})} [\mathbf{f}(\mathbf{x}, t)^T \nabla_{\mathbf{x}} \log p_t(\mathbf{x})] = \int \mathbf{f}(\mathbf{x}, t)^T \nabla_{\mathbf{x}} p_t(\mathbf{x}) d\mathbf{x} = - \int p_t(\mathbf{x}) \nabla \cdot \mathbf{f}(\mathbf{x}, t) d\mathbf{x},$$

which establishes the equivalence between Eq. (A.43) and Eq. (A.42). \square

Remark The formula in Theorem A.1 provides a new way to estimate the entropy of a data distribution from i.i.d. samples. Specifically, given $\{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\} \stackrel{\text{i.i.d.}}{\sim} p(\mathbf{x})$ and an SDE like Eq. (9.1), we can first apply score matching to train a time-dependent score-based model such that $\mathbf{s}_{\theta}(\mathbf{x}, t) \approx \nabla_{\mathbf{x}} \log p_t(\mathbf{x})$, and then plug $\mathbf{s}_{\theta}(\mathbf{x}, t)$ into Eq. (A.42) to obtain the following estimator of $\mathcal{H}(p(\mathbf{x}))$:

$$\mathcal{H}(p_T(\mathbf{x})) + \frac{1}{2N} \sum_{i=1}^N \int_0^T \left[2\mathbf{f}(\mathbf{x}_i, t)^T \mathbf{s}_{\theta}(\mathbf{x}_i, t) - g(t)^2 \|\mathbf{s}_{\theta}(\mathbf{x}_i, t)\|_2^2 \right] dt,$$

or plug it into Eq. (A.43) to obtain the following alternative estimator

$$\mathcal{H}(p_T(\mathbf{x})) - \frac{1}{2N} \sum_{i=1}^N \int_0^T \left[2\nabla \cdot \mathbf{f}(\mathbf{x}_i, t) + g(t)^2 \|\mathbf{s}_{\theta}(\mathbf{x}_i, t)\|_2^2 \right] dt.$$

Both estimators can be computed from a score-based model alone, and do not require training a density model.

Theorem A.2. *Let $p_{0t}(\mathbf{x}' \mid \mathbf{x})$ denote the transition kernel from $p_0(\mathbf{x})$ to $p_t(\mathbf{x})$ for any $t \in (0, T]$. With the same conditions and notations in Theorem 9.1, we have*

$$\begin{aligned} -\mathbb{E}_{p(\mathbf{x})}[\log p_{\boldsymbol{\theta}}^{\text{SDE}}(\mathbf{x})] &\leq -\mathbb{E}_{p_T(\mathbf{x})}[\log \pi(\mathbf{x})] + \frac{1}{2} \int_0^T \mathbb{E}_{\mathbf{x} \sim p_t(\mathbf{x})}[2g(t)^2 \nabla \cdot \mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}, t) \\ &\quad + g(t)^2 \|\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}, t)\|_2^2 - 2\nabla \cdot \mathbf{f}(\mathbf{x}, t)] dt. \end{aligned} \quad (\text{A.45})$$

$$\begin{aligned} &= -\mathbb{E}_{p_T(\mathbf{x})}[\log \pi(\mathbf{x})] \\ &\quad + \frac{1}{2} \int_0^T \mathbb{E}_{p_{0t}(\mathbf{x}' \mid \mathbf{x})p(\mathbf{x})}[g(t)^2 \|\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}', t) - \nabla_{\mathbf{x}'} \log p_{0t}(\mathbf{x}' \mid \mathbf{x})\|_2^2 \\ &\quad - g(t)^2 \|\nabla_{\mathbf{x}'} \log p_{0t}(\mathbf{x}' \mid \mathbf{x})\|_2^2 - 2\nabla \cdot \mathbf{f}(\mathbf{x}', t)] dt. \end{aligned} \quad (\text{A.46})$$

Proof. Since $-\mathbb{E}_{p(\mathbf{x})}[\log p_{\boldsymbol{\theta}}^{\text{SDE}}(\mathbf{x})] = D_{\text{KL}}(p \parallel p_{\boldsymbol{\theta}}^{\text{SDE}}) + \mathcal{H}(p)$, we can combine Theorem 9.1 and Theorem A.1 to obtain

$$\begin{aligned} &- \mathbb{E}_{p(\mathbf{x})}[\log p_{\boldsymbol{\theta}}^{\text{SDE}}(\mathbf{x})] \\ &\leq \frac{1}{2} \int_0^T \mathbb{E}_{p_t(\mathbf{x})}[g(t)^2 \|\nabla_{\mathbf{x}} \log p_t(\mathbf{x}) - \mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}, t)\|_2^2] dt + D_{\text{KL}}(p_T \parallel \pi) \end{aligned} \quad (\text{A.47})$$

$$\begin{aligned} &\quad + \mathcal{H}(p_T(\mathbf{x})) - \frac{1}{2} \int_0^T \mathbb{E}_{p_t(\mathbf{x})}[2\nabla \cdot \mathbf{f}(\mathbf{x}, t) + g(t)^2 \|\nabla_{\mathbf{x}} \log p_t(\mathbf{x})\|_2^2] dt \\ &= -\mathbb{E}_{p_T(\mathbf{x})}[\log \pi(\mathbf{x})] \\ &\quad + \frac{1}{2} \int_0^T \mathbb{E}_{p_t(\mathbf{x})}[g(t)^2 \|\nabla_{\mathbf{x}} \log p_t(\mathbf{x}) - \mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}, t)\|_2^2 - g(t)^2 \|\nabla_{\mathbf{x}} \log p_t(\mathbf{x})\|_2^2] dt \\ &\quad - \int_0^T \mathbb{E}_{p_t(\mathbf{x})}[\nabla \cdot \mathbf{f}(\mathbf{x}, t)] dt. \end{aligned} \quad (\text{A.48})$$

The second term of Eq. (A.48) can be simplified via integration by parts

$$\begin{aligned} &\frac{1}{2} \int_0^T \mathbb{E}_{p_t(\mathbf{x})}[g(t)^2 \|\nabla_{\mathbf{x}} \log p_t(\mathbf{x}) - \mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}, t)\|_2^2 - g(t)^2 \|\nabla_{\mathbf{x}} \log p_t(\mathbf{x})\|_2^2] dt \\ &= \frac{1}{2} \int_0^T \mathbb{E}_{p_t(\mathbf{x})}[g(t)^2 \|\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}, t)\|_2^2 - 2g(t)^2 \mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}, t)^T \nabla_{\mathbf{x}} \log p_t(\mathbf{x})] dt \\ &= \frac{1}{2} \int_0^T \mathbb{E}_{p_t(\mathbf{x})}[g(t)^2 \|\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}, t)\|_2^2] dt - \int_0^T \mathbb{E}_{p_t(\mathbf{x})}[g(t)^2 \mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}, t)^T \nabla_{\mathbf{x}} \log p_t(\mathbf{x})] dt \end{aligned}$$

$$\begin{aligned}
&= \frac{1}{2} \int_0^T \mathbb{E}_{p_t(\mathbf{x})} [g(t)^2 \|s_{\theta}(\mathbf{x}, t)\|_2^2] dt - \int_0^T g(t)^2 \int p_t(\mathbf{x}) s_{\theta}(\mathbf{x}, t)^T \nabla_{\mathbf{x}} \log p_t(\mathbf{x}) d\mathbf{x} dt \\
&= \frac{1}{2} \int_0^T \mathbb{E}_{p_t(\mathbf{x})} [g(t)^2 \|s_{\theta}(\mathbf{x}, t)\|_2^2] dt - \int_0^T g(t)^2 \int s_{\theta}(\mathbf{x}, t)^T \nabla_{\mathbf{x}} p_t(\mathbf{x}) d\mathbf{x} dt \\
&\stackrel{(i)}{=} \frac{1}{2} \int_0^T \mathbb{E}_{p_t(\mathbf{x})} [g(t)^2 \|s_{\theta}(\mathbf{x}, t)\|_2^2] dt + \int_0^T g(t)^2 \int p_t(\mathbf{x}) \nabla \cdot s_{\theta}(\mathbf{x}, t) d\mathbf{x} dt \\
&= \frac{1}{2} \int_0^T \mathbb{E}_{p_t(\mathbf{x})} [g(t)^2 \|s_{\theta}(\mathbf{x}, t)\|_2^2 + 2g(t)^2 \nabla \cdot s_{\theta}(\mathbf{x}, t)] dt,
\end{aligned} \tag{A.49}$$

where (i) is due to integration by parts and the limiting behavior of $p_t(\mathbf{x})$ given by assumption (xii). Combining Eq. (A.49) and Eq. (A.48) completes the proof for Eq. (A.45).

The proof for Eq. (A.46) parallels that of denoising score matching [Vin11]. Observe that $p_t(\mathbf{x}) = \int p(\mathbf{x}') p_{0t}(\mathbf{x} \mid \mathbf{x}') d\mathbf{x}'$. As a result,

$$\begin{aligned}
&\int_0^T \mathbb{E}_{p_t(\mathbf{x})} [g(t)^2 s_{\theta}(\mathbf{x}, t)^T \nabla_{\mathbf{x}} \log p_t(\mathbf{x})] dt = \int_0^T g(t)^2 \int s_{\theta}(\mathbf{x}, t)^T \nabla_{\mathbf{x}} p_t(\mathbf{x}) d\mathbf{x} dt \\
&= \int_0^T g(t)^2 \int s_{\theta}(\mathbf{x}, t)^T \nabla_{\mathbf{x}} \int p(\mathbf{x}') p_{0t}(\mathbf{x} \mid \mathbf{x}') d\mathbf{x}' d\mathbf{x} dt \\
&= \int_0^T g(t)^2 \int s_{\theta}(\mathbf{x}, t)^T \int p(\mathbf{x}') \nabla_{\mathbf{x}} p_{0t}(\mathbf{x} \mid \mathbf{x}') d\mathbf{x}' d\mathbf{x} dt \\
&= \int_0^T g(t)^2 \int s_{\theta}(\mathbf{x}, t)^T \int p(\mathbf{x}') p_{0t}(\mathbf{x} \mid \mathbf{x}') \nabla_{\mathbf{x}} \log p_{0t}(\mathbf{x} \mid \mathbf{x}') d\mathbf{x}' d\mathbf{x} dt \\
&= \int_0^T \mathbb{E}_{p(\mathbf{x}) p_{0t}(\mathbf{x}' \mid \mathbf{x})} [g(t)^2 s_{\theta}(\mathbf{x}', t)^T \nabla_{\mathbf{x}'} \log p_{0t}(\mathbf{x}' \mid \mathbf{x})] dt.
\end{aligned} \tag{A.50}$$

Substituting Eq. (A.50) into the second term of Eq. (A.48), we have

$$\begin{aligned}
&\frac{1}{2} \int_0^T \mathbb{E}_{p_t(\mathbf{x})} [g(t)^2 \|\nabla_{\mathbf{x}} \log p_t(\mathbf{x}) - s_{\theta}(\mathbf{x}, t)\|_2^2 - g(t)^2 \|\nabla_{\mathbf{x}} \log p_t(\mathbf{x})\|_2^2] dt \\
&= \frac{1}{2} \int_0^T \mathbb{E}_{p_t(\mathbf{x})} [g(t)^2 \|s_{\theta}(\mathbf{x}, t)\|_2^2 - 2g(t)^2 s_{\theta}(\mathbf{x}, t)^T \nabla_{\mathbf{x}} \log p_t(\mathbf{x})] dt \\
&= \frac{1}{2} \int_0^T \mathbb{E}_{p_t(\mathbf{x})} [g(t)^2 \|s_{\theta}(\mathbf{x}, t)\|_2^2 - 2g(t)^2 s_{\theta}(\mathbf{x}, t)^T \nabla_{\mathbf{x}} \log p_t(\mathbf{x})] dt \\
&= \frac{1}{2} \int_0^T \mathbb{E}_{p(\mathbf{x}) p_{0t}(\mathbf{x}' \mid \mathbf{x})} [g(t)^2 \|s_{\theta}(\mathbf{x}', t)\|_2^2 - 2g(t)^2 s_{\theta}(\mathbf{x}', t)^T \nabla_{\mathbf{x}'} \log p_{0t}(\mathbf{x}' \mid \mathbf{x})] dt \\
&= \frac{1}{2} \int_0^T \mathbb{E}_{p(\mathbf{x}) p_{0t}(\mathbf{x}' \mid \mathbf{x})} [g(t)^2 \|s_{\theta}(\mathbf{x}', t) - \nabla_{\mathbf{x}'} \log p_{0t}(\mathbf{x}' \mid \mathbf{x})\|_2^2 - g(t)^2 \|\nabla_{\mathbf{x}'} \log p_{0t}(\mathbf{x}' \mid \mathbf{x})\|_2^2] dt.
\end{aligned} \tag{A.51}$$

We can now complete the proof for Eq. (A.46) by combining Eq. (A.51) and Eq. (A.48). \square

Theorem 9.3. *Let $p_{0t}(\mathbf{x}' | \mathbf{x})$ denote the transition distribution from $p_0(\mathbf{x})$ to $p_t(\mathbf{x})$ for the SDE in Eq. (9.1). With the same notations and conditions in Theorem 9.1, we have*

$$-\log p_{\boldsymbol{\theta}}^{\text{SDE}}(\mathbf{x}) \leq \mathcal{L}_{\boldsymbol{\theta}}^{\text{SM}}(\mathbf{x}) = \mathcal{L}_{\boldsymbol{\theta}}^{\text{DSM}}(\mathbf{x}), \quad (9.11)$$

where $\mathcal{L}_{\boldsymbol{\theta}}^{\text{SM}}(\mathbf{x})$ is defined as

$$\begin{aligned} & -\mathbb{E}_{p_{0T}(\mathbf{x}'|\mathbf{x})}[\log \pi(\mathbf{x}')] + \frac{1}{2} \int_0^T \mathbb{E}_{p_{0t}(\mathbf{x}'|\mathbf{x})} \left[2g(t)^2 \nabla_{\mathbf{x}'} \cdot \mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}', t) \right. \\ & \quad \left. + g(t)^2 \|\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}', t)\|_2^2 - 2\nabla_{\mathbf{x}'} \cdot \mathbf{f}(\mathbf{x}', t) \right] dt, \end{aligned}$$

and $\mathcal{L}_{\boldsymbol{\theta}}^{\text{DSM}}(\mathbf{x})$ is given by

$$\begin{aligned} & -\mathbb{E}_{p_{0T}(\mathbf{x}'|\mathbf{x})}[\log \pi(\mathbf{x}')] + \frac{1}{2} \int_0^T \mathbb{E}_{p_{0t}(\mathbf{x}'|\mathbf{x})} \left[g(t)^2 \|\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}', t) - \nabla_{\mathbf{x}'} \log p_{0t}(\mathbf{x}' | \mathbf{x})\|_2^2 \right] dt \\ & \quad - \frac{1}{2} \int_0^T \mathbb{E}_{p_{0t}(\mathbf{x}'|\mathbf{x})} \left[g(t)^2 \|\nabla_{\mathbf{x}'} \log p_{0t}(\mathbf{x}' | \mathbf{x})\|_2^2 + 2\nabla_{\mathbf{x}'} \cdot \mathbf{f}(\mathbf{x}', t) \right] dt. \end{aligned}$$

Proof. The result in Theorem A.2 can be re-written as

$$\begin{aligned} -\mathbb{E}_{p(\mathbf{x})}[\log p_{\boldsymbol{\theta}}^{\text{SDE}}(\mathbf{x})] & \leq -\mathbb{E}_{p(\mathbf{x})p_{0T}(\mathbf{x}'|\mathbf{x})}[\log \pi(\mathbf{x}')] + \frac{1}{2} \int_0^T \mathbb{E}_{p(\mathbf{x})p_{0t}(\mathbf{x}'|\mathbf{x})} [2g(t)^2 \nabla \cdot \mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}', t) \\ & \quad + g(t)^2 \|\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}', t)\|_2^2 - 2\nabla \cdot \mathbf{f}(\mathbf{x}', t)] dt. \\ & = -\mathbb{E}_{p(\mathbf{x})p_{0T}(\mathbf{x}'|\mathbf{x})}[\log \pi(\mathbf{x}')] \\ & \quad + \frac{1}{2} \int_0^T \mathbb{E}_{p(\mathbf{x})p_{0t}(\mathbf{x}'|\mathbf{x})} [g(t)^2 \|\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}', t) - \nabla_{\mathbf{x}'} \log p_{0t}(\mathbf{x}' | \mathbf{x})\|_2^2 \\ & \quad - g(t)^2 \|\nabla_{\mathbf{x}'} \log p_{0t}(\mathbf{x}' | \mathbf{x})\|_2^2 - 2\nabla \cdot \mathbf{f}(\mathbf{x}', t)] dt. \end{aligned}$$

Given a fixed SDE (and its transition kernel $p_{0t}(\mathbf{x}' | \mathbf{x})$), Theorem A.2 holds for any data distribution p that satisfies our assumptions. Leveraging proof by contradiction, we can easily see that $\mathbb{E}_{p(\mathbf{x})}$ in both sides of Eqs. (A.45) and (A.46) can be cancelled to get

$$-\log p_{\boldsymbol{\theta}}^{\text{SDE}}(\mathbf{x}) \leq \mathcal{L}_{\boldsymbol{\theta}}^{\text{SM}}(\mathbf{x}) = \mathcal{L}_{\boldsymbol{\theta}}^{\text{DSM}}(\mathbf{x}),$$

which finishes the proof. \square

Appendix B

Additional Details and Results

B.1 Chapter 3

B.1.1 Samples

We provide uncurated samples from various models in Tables B.1 to B.4.

B.1.2 Additional Details of Experiments

Kernel Exponential Families

Models The kernel exponential family is a class of densities with unnormalized log density given by $\log \tilde{p}_f(\mathbf{x}) = f(\mathbf{x}) + \log q_0(\mathbf{x})$. Here, q_0 is a fixed function and f belongs to a reproducing kernel Hilbert space \mathcal{H} , with kernel k [CS06; Sri+17]. We see this is a member of the exponential family by using reproducing property, $f(\mathbf{x}) = \langle f, k(x, \cdot) \rangle_{\mathcal{H}}$. Rewriting the density, the model has natural parameter f and sufficient statistic $k(\mathbf{x}, \cdot)$:

$$\tilde{p}_f(\mathbf{x}) = \exp(f(\mathbf{x}))q_0(\mathbf{x}) = \exp(\langle f, k(x, \cdot) \rangle_{\mathcal{H}})q_0(\mathbf{x})$$

	Latent Dim 8	Latent Dim 32
ELBO	<pre> 0 9 2 7 4 0 7 3 9 0 0 7 5 4 5 6 8 8 9 4 9 6 4 9 9 2 2 8 6 0 7 3 8 2 0 0 9 0 3 8 1 5 8 3 4 9 6 9 8 6 8 4 2 8 8 1 8 7 5 9 5 6 0 9 0 8 1 1 4 3 6 9 0 9 0 1 7 0 7 3 9 8 2 9 9 1 9 3 6 4 9 8 9 2 0 8 7 9 5 4 </pre>	<pre> 4 5 9 6 8 7 4 4 8 2 2 1 8 0 8 3 1 3 9 7 4 8 9 3 9 3 8 4 8 7 9 9 6 4 7 5 6 0 1 7 6 2 5 7 5 5 2 4 9 9 3 8 2 4 3 4 9 1 8 0 3 9 8 9 8 3 6 7 6 0 0 3 0 3 4 9 8 3 3 3 6 4 1 2 8 7 4 6 4 8 8 7 2 3 4 7 9 6 4 9 </pre>
SSM	<pre> 2 0 9 0 5 6 0 4 0 4 9 5 2 8 3 0 8 1 3 8 3 8 5 1 1 8 8 8 6 2 0 7 4 7 0 0 4 0 2 8 3 2 4 6 8 8 0 8 2 8 1 3 1 4 9 9 8 0 0 0 2 5 1 8 1 2 3 1 3 4 1 9 3 0 7 1 9 6 0 9 3 8 4 0 3 3 0 0 0 3 5 1 0 2 0 8 3 0 4 1 </pre>	<pre> 7 4 3 0 5 9 5 3 2 8 0 1 2 8 8 7 9 9 7 4 8 6 1 0 7 3 5 1 0 9 1 8 3 9 7 8 0 9 7 4 8 9 7 5 3 9 7 4 6 3 5 6 0 5 4 3 8 3 2 7 9 9 6 1 2 5 4 4 6 3 0 8 0 3 1 8 6 9 9 5 0 7 0 8 8 1 3 0 2 2 8 9 0 3 7 1 0 5 8 7 </pre>
Stein	<pre> 5 8 9 1 1 3 0 1 9 3 0 4 2 6 2 6 1 1 9 4 9 2 0 0 0 9 9 1 0 6 4 8 2 7 2 2 8 8 8 9 3 2 4 4 4 0 6 2 0 8 4 9 8 8 1 8 4 2 8 8 7 0 9 9 2 9 9 0 3 5 8 0 4 9 1 1 2 5 4 6 8 0 8 1 0 4 0 6 9 9 3 6 0 6 1 0 0 3 5 </pre>	<pre> 2 2 2 4 4 7 2 2 3 7 2 8 5 0 3 7 6 3 0 5 3 1 0 6 3 4 2 2 2 3 2 3 0 6 5 4 0 8 4 0 9 5 7 2 5 1 0 2 3 3 1 6 4 5 0 8 3 0 5 4 6 0 3 5 5 1 1 2 1 5 2 3 8 6 8 5 8 2 8 5 4 2 4 4 9 2 0 6 7 3 9 3 6 2 8 3 3 5 6 5 </pre>
Spectral	<pre> 6 4 0 9 1 6 2 3 4 9 0 4 1 2 2 6 2 8 8 9 4 2 6 4 9 0 0 3 9 6 0 8 3 9 3 5 3 8 2 8 0 8 3 7 9 2 9 4 2 8 8 2 8 9 8 9 8 3 2 8 2 0 9 3 9 2 2 9 0 3 6 8 2 9 0 9 8 2 8 3 4 8 7 6 1 2 7 0 6 0 0 9 8 0 1 3 4 0 8 9 </pre>	<pre> 2 3 8 1 2 1 6 4 1 7 2 0 5 0 3 6 7 4 5 3 3 3 1 8 0 3 6 6 0 0 2 6 3 6 8 3 8 4 3 6 3 9 7 9 8 5 7 8 0 2 3 4 0 4 6 0 2 3 1 0 8 2 9 6 3 6 5 0 3 0 3 2 2 8 0 2 5 0 2 1 5 2 0 3 8 4 6 3 2 4 5 1 2 4 0 6 9 1 3 5 1 3 </pre>

Table B.1: VAE samples on MNIST.



Table B.2: VAE samples on CelebA.

	Latent Dim 8	Latent Dim 32
SSM	<pre> 2 9 2 8 8 6 9 6 0 0 8 7 3 7 3 8 0 9 8 5 1 8 6 9 1 1 0 2 8 9 2 6 2 7 2 7 7 1 3 0 2 5 8 8 0 3 1 1 9 5 9 2 2 6 9 7 3 7 4 1 9 1 6 5 9 1 8 6 6 4 7 0 1 6 4 3 4 2 9 9 0 4 4 7 3 7 3 1 1 0 8 9 8 8 9 8 6 4 </pre>	<pre> 6 8 1 1 4 2 5 2 5 6 4 1 9 8 9 1 2 7 6 8 7 9 2 8 0 8 6 1 2 0 2 3 1 4 1 8 7 3 7 0 5 1 0 9 6 6 4 2 7 9 0 1 7 4 9 3 8 5 0 3 1 8 9 2 0 8 3 2 0 9 9 8 9 6 3 9 5 0 5 3 1 6 2 1 3 6 9 2 1 6 3 2 3 8 3 5 0 9 3 2 </pre>
Stein	<pre> 1 1 4 8 5 8 8 3 7 9 1 9 6 0 8 8 8 5 5 0 7 8 9 7 4 4 8 6 7 4 8 8 3 7 1 2 1 6 6 8 9 6 2 7 0 0 9 9 5 8 5 9 9 6 8 0 6 8 1 8 3 9 9 8 2 8 9 3 0 3 5 8 6 9 8 0 0 1 8 8 9 0 6 8 5 6 9 7 8 9 4 1 4 6 6 3 9 9 1 0 </pre>	<pre> 2 2 8 3 8 9 6 0 1 9 6 7 4 8 3 4 9 3 4 9 8 3 2 8 9 6 6 7 6 5 3 3 5 8 7 2 3 9 6 9 8 8 0 9 8 8 0 9 7 3 5 9 0 8 4 1 3 0 5 1 4 4 3 3 3 2 2 9 4 9 5 6 5 0 9 8 9 8 9 8 4 4 2 2 0 0 4 0 2 8 3 9 0 9 7 0 0 1 3 5 </pre>
Spectral	<pre> 4 3 9 3 8 1 3 2 7 9 2 8 4 4 5 4 8 1 3 6 5 2 4 1 4 0 0 6 7 4 8 0 2 7 0 0 8 0 9 6 3 7 8 3 6 5 9 9 3 6 0 5 0 6 2 3 0 8 1 8 3 9 9 0 9 9 3 7 9 3 7 3 9 8 0 7 8 7 2 3 9 6 4 8 3 5 9 7 7 4 8 7 9 4 6 3 5 3 7 6 </pre>	<pre> 3 8 3 2 8 9 0 2 1 3 5 5 0 8 9 8 9 5 4 9 8 3 2 8 9 6 7 6 3 3 2 4 9 7 6 2 9 4 9 8 5 0 8 8 0 0 9 3 9 0 3 2 3 4 0 3 5 0 3 5 8 3 5 3 2 0 9 4 6 4 5 3 3 3 8 9 3 4 1 4 2 8 8 0 5 0 2 2 8 2 4 0 9 3 7 0 8 3 3 </pre>

Table B.3: WAE samples on MNIST.

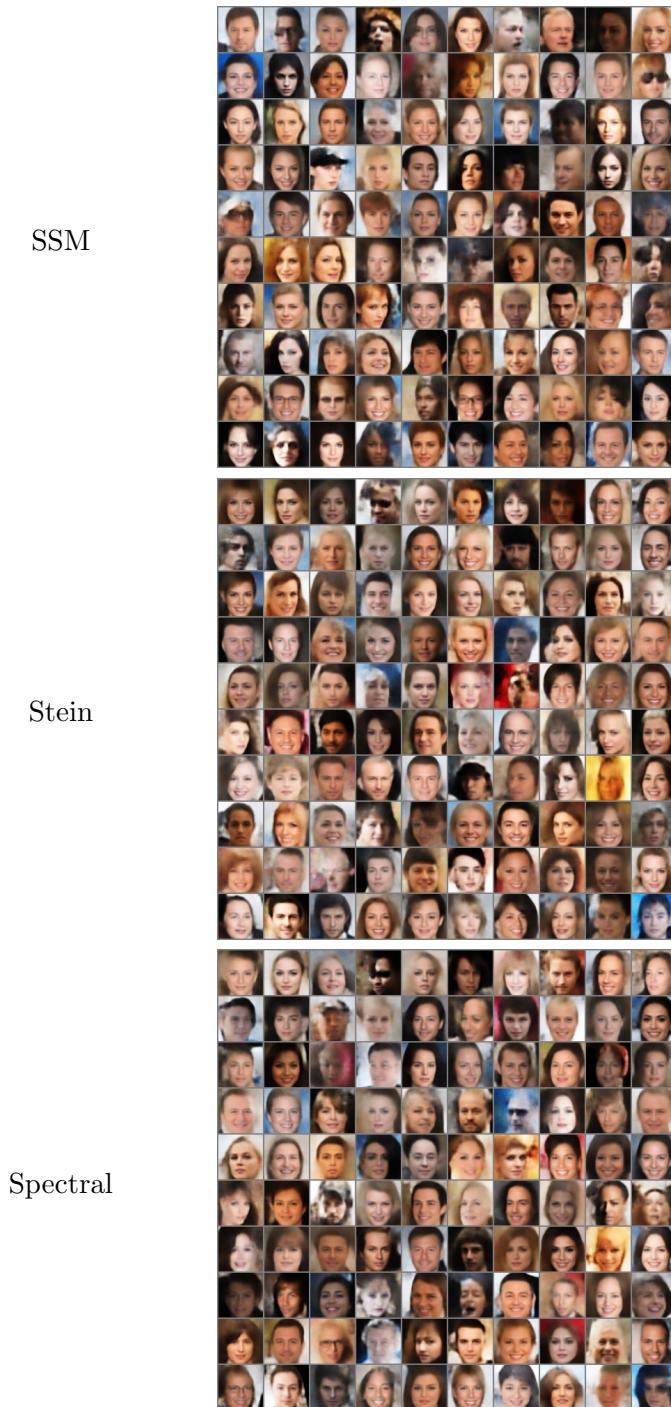


Table B.4: WAE samples on CelebA.

To improve computational cost of learning f , [Sut+18] use a Nyström-type lite approximation, selecting L inducing points \mathbf{z}_l , and f of the form:

$$f(\mathbf{x}) = \sum_{l=1}^L \alpha_l k(\mathbf{x}, \mathbf{z}_l)$$

We compare training using our objective against deep kernel exponential family (DKEF) models trained using exact score matching in [Wen+19].

The kernel $k(\mathbf{x}, \mathbf{y})$ is a mixture of $R = 3$ Gaussian kernels, with features extracted by a neural network, $\phi_{w_r}(\cdot)$, length scales σ_r , and nonnegative mixture coefficients ρ_r . The neural network is a three layer fully connected network with a skip connection from the input to output layers and softplus nonlinearities. Each hidden layer has 30 neurons. We have:

$$k_w(\mathbf{x}, \mathbf{y}) = \sum_{r=1}^R \rho_r \exp\left(-\frac{1}{2\sigma_r^2} \|\phi_{w_r}(\mathbf{x}) - \phi_{w_r}(\mathbf{y})\|^2\right).$$

When training DKEF models, [Wen+19] note that it is possible to analytically minimize the score matching loss over α because the objective is quadratic in α . As a result, models are trained in a two step procedure: α is analytically minimized over a training minibatch, then the loss is computed over a validation minibatch. When training models, the analytically minimized α is treated as function of the other parameters. By doing this, [Wen+19] can also directly optimize the coefficient λ_α of a ℓ_2 regularization loss on α . This regularization coefficient is initialized to 0.01, and is trained. (More details about the two-step optimization procedure can be found in [Wen+19], which also includes a finalization stage for α).

A similar closed form for sliced score matching, denoising score matching, approximate backpropogation, and curvature propagation can be derived. The derivation for sliced score matching is presented below for completeness.

Proposition B.1. *Consider the loss*

$$\hat{J}(\boldsymbol{\theta}, \lambda_\alpha; \mathbf{x}_1^N, \mathbf{v}_{11}^{NM}) = \hat{J}(\boldsymbol{\theta}; \mathbf{x}_1^N, \mathbf{v}_{11}^{NM}) + \frac{1}{2} \lambda_\alpha \|\alpha\|_2^2$$

where

$$\hat{J}(\boldsymbol{\theta}; \mathbf{x}_1^N, \mathbf{v}_{11}^{NM}) = \frac{1}{N} \frac{1}{M} \sum_{i=1}^N \sum_{j=1}^M \left[\mathbf{v}_{ij}^\top \nabla^2 \log p_m(\mathbf{x}_i) \mathbf{v}_{ij} + \frac{1}{2} \left(\mathbf{v}_{ij}^\top \nabla \log p_m(\mathbf{x}_i) \right)^2 \right].$$

For fixed k , \mathbf{z} , and λ_{α} , as long as $\lambda_{\alpha} > 0$ then the optimal $\boldsymbol{\alpha}$ is

$$\begin{aligned}\boldsymbol{\alpha}(\lambda_{\alpha}, k, \mathbf{z}, \mathbf{x}_1^N, \mathbf{v}_{11}^{NM}) &= \arg \min_{\boldsymbol{\alpha}} \hat{J}(\boldsymbol{\theta}, \lambda_{\alpha}; \mathbf{x}_1^N, \mathbf{v}_{11}^{NM}) = -(\mathbf{G} + \lambda_{\alpha} \mathbf{I})^{-1} \mathbf{b} \\ G_{l,l'} &= \frac{1}{N} \frac{1}{M} \sum_{i=1}^N \sum_{j=1}^M \left(\mathbf{v}_{ij}^T \nabla k(\mathbf{x}_i, \mathbf{z}_l) \right) \left(\mathbf{v}_{ij}^T \nabla k(\mathbf{x}_i, \mathbf{z}_{l'}) \right) \\ b_l &= \frac{1}{N} \frac{1}{M} \sum_{i=1}^N \sum_{j=1}^M \mathbf{v}_{ij}^T \nabla^2 k(\mathbf{x}_i, \mathbf{z}_l) \mathbf{v}_{ij} + \left(\mathbf{v}_{ij}^T \nabla \log q_0(\mathbf{x}_i) \right) \left(\mathbf{v}_{ij}^T \nabla k(\mathbf{x}_i, \mathbf{z}_l) \right).\end{aligned}$$

Proof. The derivation follows very similarly to Proposition 3 in [Wen+19]. We will show that the loss is quadratic in $\boldsymbol{\alpha}$. Note that

$$\begin{aligned}\frac{1}{N} \frac{1}{M} \sum_{i=1}^N \sum_{j=1}^M \mathbf{v}_{ij}^T \nabla^2 \log p_m(\mathbf{x}_i) \mathbf{v}_{ij} &= \frac{1}{N} \frac{1}{M} \sum_{i=1}^N \sum_{j=1}^M \left[\sum_{m=1}^M \alpha_l \mathbf{v}_{ij}^T \nabla^2 k(\mathbf{x}_i, \mathbf{z}_l) \mathbf{v}_{ij} \right] + C \\ &= \boldsymbol{\alpha}^T \left[\frac{1}{N} \frac{1}{M} \sum_{i=1}^N \sum_{j=1}^M \mathbf{v}_{ij}^T \nabla^2 k(\mathbf{x}_i, \mathbf{z}_l) \mathbf{v}_{ij} \right]_l + C\end{aligned}$$

and

$$\begin{aligned}\frac{1}{N} \frac{1}{M} \sum_{i=1}^N \sum_{j=1}^M \frac{1}{2} \left(\mathbf{v}_{ij}^T \nabla \log p_m(\mathbf{x}_i) \right)^2 &= \frac{1}{N} \frac{1}{M} \sum_{i=1}^N \sum_{j=1}^M \frac{1}{2} \left(\sum_{m,m'=1}^M \alpha_l \alpha_{m'} \left(\mathbf{v}_{ij}^T \nabla k(\mathbf{x}_i, \mathbf{z}_l) \right) \left(\mathbf{v}_{ij}^T \nabla k(\mathbf{x}_i, \mathbf{z}_{l'}) \right) \right. \\ &\quad \left. + 2 \sum_{m=1}^M \alpha_l \left(\mathbf{v}_{ij}^T \nabla \log q_0(\mathbf{x}_i) \right) \left(\mathbf{v}_{ij}^T \nabla k(\mathbf{x}_i, \mathbf{z}_l) \right) + \left(\mathbf{v}_{ij}^T \nabla \log q_0(\mathbf{x}_i) \right)^2 \right) \\ &= \frac{1}{2} \boldsymbol{\alpha}^T \mathbf{G} \boldsymbol{\alpha} + \boldsymbol{\alpha}^T \left[\frac{1}{N} \frac{1}{M} \sum_{i=1}^N \sum_{j=1}^M \left(\mathbf{v}_{ij}^T \nabla \log q_0(\mathbf{x}_i) \right) \left(\mathbf{v}_{ij}^T \nabla k(\mathbf{x}_i, \mathbf{z}_l) \right) \right] + C.\end{aligned}$$

Thus the overall optimization problem is

$$\begin{aligned}\boldsymbol{\alpha}(\lambda_{\alpha}, k, \mathbf{z}, \mathbf{x}_1^N, \mathbf{v}_{11}^{NM}) &= \arg \min_{\boldsymbol{\alpha}} \hat{J}(\boldsymbol{\theta}, \lambda_{\alpha}; \mathbf{x}_1^N, \mathbf{v}_{11}^{NM}) \\ &= \arg \min_{\boldsymbol{\alpha}} \frac{1}{2} \boldsymbol{\alpha}^T (\mathbf{G} + \lambda_{\alpha} \mathbf{I}) \boldsymbol{\alpha} + \boldsymbol{\alpha}^T \mathbf{b}.\end{aligned}$$

Because $\lambda_{\alpha} > 0$ and \mathbf{G} is positive semidefinite, the matrix in parentheses is strictly positive

definite, and the claimed result follows directly from standard vector calculus. \square

Hyperparameters RedWine and WhiteWine are dequantized by adding uniform noise to each dimension in the range $[-d, d]$ where d is the median distance between two values for that dimension. For each dataset, 10% of the entire data was used as testing, and 10% of the remaining was used for validation. PCA whitening is applied to the data. Noise of standard deviation 0.05 is added as part of preprocessing.

The DKEF models have $R = 3$ Gaussian kernels. Each feature extractor is a 3-layer neural network with a skip connection from the input to output, with 30 hidden neurons per layer. Weights were initialized from a Gaussian distribution with standard deviation equal to $\frac{1}{\sqrt{30}}$. Length scales σ_r were initialized to 1.0, 3.3 and 10.0. We use $L = 200$ trainable inducing points, which were initialized from training data.

Models are trained using an Adam optimizer, with learning rate 10^{-2} . A batch size of 200 is used, with 100 points for computing α , and 100 for computing the loss. Models are stopped after validation loss does not improve for 200 steps.

For denoising score matching, we perform a grid search with values [0.02, 0.04, 0.06, 0.08, 0.10, 0.12, 0.14, 0.16, 0.20, 0.24, 0.28, 0.32, 0.40, 0.48, 0.56, 0.64, 1.28]. We train models for each value of σ using two random seeds, and pick the σ with the best average validation score matching loss. For curvature propagation, one noise sample is used to match the performance of sliced score matching.

Log-likelihoods Log-likelihoods are presented below. They are estimated using AIS, using a proposal distribution $\mathcal{N}(0, 2I)$, using 1,000,000 samples.

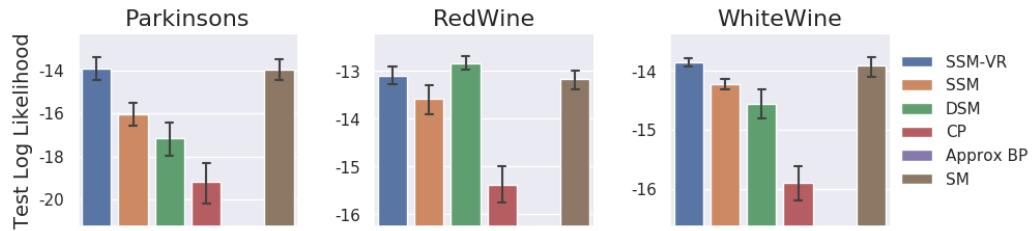


Figure B.1: Log-likelihoods after training DKEF models on UCI datasets with different loss functions; higher is better. Results for approximate backpropagation are not shown because log-likelihoods were smaller than -10^6 .

NICE

Hyperparameters and model architecture The model has four coupling layers, each with five hidden layers, for a total of 20 hidden layers, as well as a final scale layer [DKB15]. Softplus nonlinearities are used between hidden layers.

Models are trained using the Adam optimizer with learning rate 10^{-3} for 100 epochs. The best checkpoint on exact score matching loss, evaluation every 100 iterations, is used to report test set performance. We use a batch size of 128.

Data are dequantized by adding uniform noise in the range $[-\frac{1}{512}, \frac{1}{512}]$, clipped to be in the range $[-0.001, 0.001]$, and then transformed using a logit transformation $\log(x) - \log(1 - x)$. 90% of the training set is used for training, and 10% for validation, and the standard test set is used.

For grid search for the optimal value of σ , eight values are used: [0.01, 0.05, 0.10, 0.20, 0.28, 0.50, 1.00, 1.50]. We also evaluate $\sigma = 1.74$, chosen by the heuristic in [Sar+18]. The model with the best performance on validation score matching loss is used. Only nine values of σ are evaluated because training each model takes approximately two hours.

Score Estimation for Implicit Distributions

Architectures We put the architectures of all networks used in the MNIST and CelebA experiments in Table B.5 and Table B.6 respectively.

Training For MNIST experiments, we use RMSProp optimizer with a learning rate of 0.001 for all methods. On CelebA, the learning rate is changed to 0.0001. All algorithms are trained for 100000 iterations with a batch size of 128.

Samples All samples are generated after 100000 training iterations.

Variance Reduction

Below we discuss approaches to reduce the variance of $\hat{J}(\boldsymbol{\theta}; \mathbf{x}_1^N, \mathbf{v}_{11}^{NM})$, which can lead to better performance in practice. The most naïve approach, of course, is using a larger M to compute $\hat{J}(\boldsymbol{\theta}; \mathbf{x}_1^N, \mathbf{v}_{11}^{NM})$. However, this requires more computation and when M is close to the data dimension, sliced score matching will lose its computational advantage over score matching.

Table B.5: Architectures on MNIST. In our models, $D_\epsilon = D_z$. D_z takes the values 8 and 32 in different experiments.

Name	Configuration	Algorithm
Encoder	Linear(784, 256), Tanh Linear(256, 256), Tanh Linear(256, D_z)	ELBO VAE, WAE
Implicit Encoder	Linear($784 + D_\epsilon$, 256), Tanh Linear(256, 256), Tanh Linear(256, D_z)	Implicit VAE
Decoder	Linear(D_z , 256), Tanh Linear(256, 256), Tanh Linear(256, 784), Sigmoid	All
Score Estimator	Linear($784 + D_z$, 256), Tanh Linear(256, 256), Tanh Linear(256, D_z)	Implicit VAE
Score Estimator	Linear(D_z , 256), Tanh Linear(256, 256), Tanh Linear(256, D_z)	WAE

An alternative approach is to leverage control variates [Owe13]. A control variate is a random variable whose expectation is tractable, and is highly correlated with another random variable without a tractable expectation. Define $c(\boldsymbol{\theta}; \mathbf{x}, \mathbf{v}) := \frac{1}{2}(\mathbf{v}^\top \mathbf{s}_m(\mathbf{x}; \boldsymbol{\theta}))^2$. Note that when $p_{\mathbf{v}}$ is a multivariate standard normal or multivariate Rademacher distribution, $c(\boldsymbol{\theta}; \mathbf{x}, \mathbf{v})$ will have a tractable expectation, *i.e.*,

$$\mathbb{E}_{p_{\mathbf{v}}}[c(\boldsymbol{\theta}; \mathbf{x}, \mathbf{v})] = \frac{1}{2} \|\mathbf{s}_m(\mathbf{x}; \boldsymbol{\theta})\|_2^2,$$

which is easily computable. Now let $\beta(\mathbf{x})c(\boldsymbol{\theta}; \mathbf{x}, \mathbf{v})$ be our control variate, where $\beta(\mathbf{x})$ is a function to be determined. Due to the structural similarity between $\beta(\mathbf{x})c(\boldsymbol{\theta}; \mathbf{x}, \mathbf{v})$ and $\hat{J}(\boldsymbol{\theta}; \mathbf{x}_1^N, \mathbf{v}_{11}^{NM})$, it is easy to believe that $\beta(\mathbf{x})c(\boldsymbol{\theta}; \mathbf{x}, \mathbf{v})$ can be a correlated control variate with an appropriate $\beta(\mathbf{x})$. We thus consider the following objective

$$\hat{J}_{\text{vr}}(\boldsymbol{\theta}; \mathbf{x}_1^N, \mathbf{v}_{11}^{NM}) := \hat{J}(\boldsymbol{\theta}; \mathbf{x}_1^N, \mathbf{v}_{11}^{NM}) - \frac{1}{N} \sum_{i=1}^N \beta(\mathbf{x}_i) \left(\frac{1}{M} \sum_{j=1}^M c(\boldsymbol{\theta}; \mathbf{x}_i, \mathbf{v}_{ij}) - \frac{1}{2} \|\mathbf{s}_m(\mathbf{x}_i; \boldsymbol{\theta})\|_2^2 \right).$$

Note that $\mathbb{E}[\hat{J}_{\text{vr}}(\boldsymbol{\theta}; \mathbf{x}_1^N, \mathbf{v}_{11}^{NM})] = J(\boldsymbol{\theta}; p_{\mathbf{v}})$. The theory of control variates guarantees the existence of $\beta(\mathbf{x})$ that can reduce the variance. In practice, there can be many heuristics of

Table B.6: Architectures on CelebA. In our models, $D_\epsilon = D_z$. D_z takes the values 8 or 32 in different experiments.

Name	Configuration	Algorithm
Encoder	5×5 conv; m maps; stride 2×2 ; padding 2, ReLU 5×5 conv; $2m$ maps; stride 2×2 ; padding 2, ReLU 5×5 conv; $4m$ maps; stride 2×2 ; padding 2, ReLU 5×5 conv; $8m$ maps; stride 2×2 ; padding 2, ReLU 512 Dense, ReLU D_z Dense	ELBO VAE, WAE
Implicit Encoder	concat $[\mathbf{x}, \text{ReLU}(\text{Dense}(\epsilon))]$ along channels 5×5 conv; m maps; stride 2×2 ; padding 2, ReLU 5×5 conv; $2m$ maps; stride 2×2 ; padding 2, ReLU 5×5 conv; $4m$ maps; stride 2×2 ; padding 2, ReLU 5×5 conv; $8m$ maps; stride 2×2 ; padding 2, ReLU 512 Dense, ReLU D_z Dense	Implicit VAE
Decoder	Dense, ReLU 5×5 conv $^\top$; $4m$ maps; stride 2×2 ; padding 2; out padding 1, ReLU 5×5 conv $^\top$; $2m$ maps; stride 2×2 ; padding 2; out padding 1, ReLU 5×5 conv $^\top$; $1m$ maps; stride 2×2 ; padding 2; out padding 1, ReLU 5×5 conv $^\top$; c maps; stride 2×2 ; padding 2; out padding 1, Tanh	All
Score Estimator	concat $[\mathbf{x}, \text{ReLU}(\text{Dense}(\mathbf{z}))]$ along channels 5×5 conv; m maps; stride 2×2 ; padding 2, ReLU 5×5 conv; $2m$ maps; stride 2×2 ; padding 2, ReLU 5×5 conv; $4m$ maps; stride 2×2 ; padding 2, ReLU 5×5 conv; $8m$ maps; stride 2×2 ; padding 2, ReLU 512 Dense, ReLU D_z Dense	Implicit VAE
Score Estimator	Reshape($\text{ReLU}(\text{Dense}(\mathbf{z}))$) to 1 channel 5×5 conv; m maps; stride 2×2 ; padding 2, ReLU 5×5 conv; $2m$ maps; stride 2×2 ; padding 2, ReLU 5×5 conv; $4m$ maps; stride 2×2 ; padding 2, ReLU 5×5 conv; $8m$ maps; stride 2×2 ; padding 2, ReLU 512 Dense, ReLU D_z Dense	WAE

choosing $\beta(\mathbf{x})$, and we found that $\beta(\mathbf{x}) \equiv 1$ can often be a good choice in our experiments.

B.1.3 Pesudocode

Algorithm B.1 Score Matching

Require: $\tilde{p}_m(\cdot; \boldsymbol{\theta}), \mathbf{x}$

```

1:  $s_m(\mathbf{x}; \boldsymbol{\theta}) \leftarrow \text{grad}(\log \tilde{p}_m(\mathbf{x}; \boldsymbol{\theta}), \mathbf{x})$ 
2:  $J \leftarrow \frac{1}{2} \|s_m(\mathbf{x}; \boldsymbol{\theta})\|_2^2$ 
3: for  $d \leftarrow 1$  to  $D$  do ▷ For each diagonal entry
4:    $(\nabla_{\mathbf{x}} s_m(\mathbf{x}; \boldsymbol{\theta}))_d \leftarrow \text{grad}((s_m(\mathbf{x}; \boldsymbol{\theta}))_d, \mathbf{x})_d$ 
5:    $J \leftarrow J + (\nabla_{\mathbf{x}} s_m(\mathbf{x}; \boldsymbol{\theta}))_d$ 
6: end for
    return  $J$ 

```

B.2 Chapter 4

B.2.1 Analysis on Second Order Score Models

Variance Reduction

If we want to match the score of true distribution $p_{\text{data}}(\mathbf{x})$, σ should be approximately zero for both DSM and $D_2\text{SM}$ so that $q_\sigma(\tilde{\mathbf{x}})$ is close to $p_{\text{data}}(\mathbf{x})$. However, when $\sigma \rightarrow 0$, both DSM and $D_2\text{SM}$ can be unstable to train and might not converge, which calls for variance reduction techniques. In this section, we show that we can introduce a control variate to improve the empirical performance of DSM and $D_2\text{SM}$ when σ tends to zero. Our variance control method can be derived from expanding the original training objective function using Taylor expansion.

DSM with varaince reduction Expand the objective using Taylor expansion

$$\begin{aligned}
\mathcal{L}_{DSM}(\boldsymbol{\theta}) &= \frac{1}{2} \mathbb{E}_{p_{\text{data}}(\mathbf{x})} \mathbb{E}_{q_\sigma(\tilde{\mathbf{x}}|\mathbf{x})} \left[\left\| \tilde{\mathbf{s}}_1(\tilde{\mathbf{x}}; \boldsymbol{\theta}) + \frac{1}{\sigma^2} (\tilde{\mathbf{x}} - \mathbf{x}) \right\|_2^2 \right] \\
&= \frac{1}{2} \mathbb{E}_{p_{\text{data}}(\mathbf{x})} \mathbb{E}_{\mathbf{z} \sim \mathcal{N}(0, I)} \left[\left\| \tilde{\mathbf{s}}_1(\mathbf{x} + \sigma \mathbf{z}; \boldsymbol{\theta}) + \frac{\mathbf{z}}{\sigma} \right\|_2^2 \right] \\
&= \frac{1}{2} \mathbb{E}_{p_{\text{data}}(\mathbf{x})} \mathbb{E}_{\mathbf{z} \sim \mathcal{N}(0, I)} \left[\left\| \tilde{\mathbf{s}}_1(\mathbf{x} + \sigma \mathbf{z}; \boldsymbol{\theta}) \right\|_2^2 + \frac{2}{\sigma} \tilde{\mathbf{s}}_1(\mathbf{x} + \sigma \mathbf{z}; \boldsymbol{\theta})^T \mathbf{z} + \frac{\|\mathbf{z}\|_2^2}{\sigma^2} \right] \\
&= \frac{1}{2} \mathbb{E}_{p_{\text{data}}(\mathbf{x})} \mathbb{E}_{\mathbf{z} \sim \mathcal{N}(0, I)} \left[\left\| \tilde{\mathbf{s}}_1(\mathbf{x}; \boldsymbol{\theta}) \right\|_2^2 + \frac{2}{\sigma} \tilde{\mathbf{s}}_1(\mathbf{x}; \boldsymbol{\theta})^T \mathbf{z} + \frac{\|\mathbf{z}\|_2^2}{\sigma^2} \right] + \mathcal{O}(1),
\end{aligned}$$

where $\mathcal{O}(1)$ is bounded when $\sigma \rightarrow 0$. Since

$$\mathbb{E}_{\mathbf{z} \sim \mathcal{N}(0, I)} \left[\frac{2}{\sigma} \tilde{\mathbf{s}}_1(\mathbf{x}; \boldsymbol{\theta})^T \mathbf{z} + \frac{\|\mathbf{z}\|_2^2}{\sigma^2} \right] = \frac{D}{\sigma^2}, \quad (\text{B.1})$$

where D is the dimension of $p_{\text{data}}(\mathbf{x})$, we can use Eq. (B.1) as a control variate and define DSM with variance reduction as

$$\mathcal{L}_{DSM-VR} = \mathcal{L}_{DSM} - \mathbb{E}_{p_{\text{data}}(\mathbf{x})} \mathbb{E}_{\mathbf{z} \sim \mathcal{N}(0, I)} \left[\frac{2}{\sigma} \tilde{\mathbf{s}}_1(\mathbf{x}; \boldsymbol{\theta})^T \mathbf{z} + \frac{\|\mathbf{z}\|_2^2}{\sigma^2} \right] + \frac{D}{\sigma^2} \quad (\text{B.2})$$

An equivalent version of Eq. (B.2) is first proposed in [Wan+20].

D_2 SM with variance reduction We now derive the variance reduction objective for D_2 SM. Let us first consider the ij -th term of $\mathcal{L}_{D_2\text{SM}}(\boldsymbol{\theta})$. We denote $\psi(\tilde{\mathbf{x}}; \boldsymbol{\theta}) = \tilde{\mathbf{s}}_2(\tilde{\mathbf{x}}; \boldsymbol{\theta}) + \tilde{\mathbf{s}}_1(\tilde{\mathbf{x}}; \boldsymbol{\theta})\tilde{\mathbf{s}}_1(\tilde{\mathbf{x}}; \boldsymbol{\theta})^T$ and $\psi_{ij}(\tilde{\mathbf{x}}; \boldsymbol{\theta})$ the ij -th term of $\psi(\tilde{\mathbf{x}}; \boldsymbol{\theta})$. Similar as the variance reduction method for DSM [Wan+20], we expand the objective of D_2 SM (Eq. (4.13)) using Taylor expansion

$$\begin{aligned} \mathcal{L}_{D_2\text{SM}}(\boldsymbol{\theta})_{ij} &= \frac{1}{2} \mathbb{E}_{p_{\text{data}}(\mathbf{x})} \mathbb{E}_{\mathbf{z} \sim \mathcal{N}(0, I)} [\psi_{ij}(\mathbf{x} + \sigma \mathbf{z}; \boldsymbol{\theta}) + \frac{\mathbf{I}_{ij} - z_i z_j}{\sigma^2}]^2 \\ &= \frac{1}{2} \mathbb{E}_{p_{\text{data}}(\mathbf{x})} \mathbb{E}_{\mathbf{z} \sim \mathcal{N}(0, I)} [\psi_{ij}(\mathbf{x} + \sigma \mathbf{z}; \boldsymbol{\theta})^2 + 2 \frac{\mathbf{I}_{ij} - z_i z_j}{\sigma^2} \psi_{ij}(\mathbf{x} + \sigma \mathbf{z}; \boldsymbol{\theta}) + \frac{(\mathbf{I}_{ij} - z_i z_j)^2}{\sigma^4}] \\ &= \frac{1}{2} \mathbb{E}_{p_{\text{data}}(\mathbf{x})} \mathbb{E}_{\mathbf{z} \sim \mathcal{N}(0, I)} [\psi_{ij}(\mathbf{x}; \boldsymbol{\theta})^2 + 2 \frac{\mathbf{I}_{ij} - z_i z_j}{\sigma^2} \psi_{ij}(\mathbf{x}; \boldsymbol{\theta}) + 2 \frac{\mathbf{I}_{ij} - z_i z_j}{\sigma} \mathbf{J}_{\psi_{ij}} \mathbf{z} + \frac{(\mathbf{I}_{ij} - z_i z_j)^2}{\sigma^4}] + \mathcal{O}(1), \end{aligned}$$

where $\mathcal{O}(1)$ is bounded when $\sigma \rightarrow 0$. It is clear to see that the term $\frac{(\mathbf{I}_{ij} - z_i z_j)^2}{\sigma^4}$ is a constant w.r.t. optimization. When σ approximates zero, both $\frac{\mathbf{I}_{ij} - z_i z_j}{\sigma^2}$ and $\frac{\mathbf{I}_{ij} - z_i z_j}{\sigma}$ would be very large, making the training process unstable and hard to converge. Thus we aim at designing a control variate to cancel out these two terms. To do this, we propose to use antithetic sampling. Instead of using independent noise samples, we use two correlated (opposite) noise vectors centered at \mathbf{x} defined as $\tilde{\mathbf{x}}_+ = \tilde{\mathbf{x}} + \sigma \mathbf{z}$ and $\tilde{\mathbf{x}}_- = \tilde{\mathbf{x}} - \sigma \mathbf{z}$. We propose the following objective function to reduce variance

$$\mathcal{L}_{D_2\text{SM-VR}} = \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x})} \mathbb{E}_{\mathbf{z} \sim \mathcal{N}(0, I)} \left[\psi(\tilde{\mathbf{x}}_+; \boldsymbol{\theta})^2 + \psi(\tilde{\mathbf{x}}_-; \boldsymbol{\theta})^2 + 2 \frac{\mathbf{I} - \mathbf{z} \mathbf{z}^T}{\sigma} \odot (\psi(\tilde{\mathbf{x}}_+; \boldsymbol{\theta}) + \psi(\tilde{\mathbf{x}}_-; \boldsymbol{\theta}) - 2\psi(\mathbf{x}; \boldsymbol{\theta})) \right]. \quad (\text{B.3})$$

Similarly, we can show easily by using Taylor expansion that optimizing Eq. (B.3) is equivalent to optimizing Eq. (4.13) up to a control variate. On the other hand, Eq. (B.3) is more stable to optimize than Eq. (4.13) when σ approximates zero since the unstable terms $\frac{\mathbf{I}_{ij} - z_i z_j}{\sigma^2}$ and

$\frac{\mathbf{I}_{ij} - z_i z_j}{\sigma}$ are both cancelled by the introduced control variate.

Learning Accuracy

This section provides more experimental details on Section 4.4.4. We use a 3-layer MLP model with latent size 128 and Tanh activation function for $\tilde{\mathbf{s}}_1(\tilde{\mathbf{x}}; \boldsymbol{\theta})$. As discussed in Section 4.4.2, our $\tilde{\mathbf{s}}_2(\tilde{\mathbf{x}}; \boldsymbol{\theta})$ model consists of two parts $\alpha(\tilde{\mathbf{x}}; \boldsymbol{\theta})$ and $\beta(\tilde{\mathbf{x}}; \boldsymbol{\theta})$. We also use a 3-layer MLP model with latent size 32 and Tanh activation function to parameterize $\alpha(\tilde{\mathbf{x}}; \boldsymbol{\theta})$ and $\beta(\tilde{\mathbf{x}}; \boldsymbol{\theta})$. For the mean squared error diagonal comparison experiments, we only parameterize the diagonal component $\alpha(\tilde{\mathbf{x}}; \boldsymbol{\theta})$. We use a 3-layer MLP model with latent size 32, and Tanh activation function to parameterize $\alpha(\tilde{\mathbf{x}}; \boldsymbol{\theta})$. We use learning rate 0.001, and train the models using Adam optimizer until convergence. We use noise scale $\sigma = 0.01, 0.05, 0.1$ during training so that the noise perturbed distribution $q_\sigma(\tilde{\mathbf{x}})$ is close to $p_{\text{data}}(\mathbf{x})$. All the mean squared error results in Table 4.1 are computed w.r.t. to the ground truth second order score of the clean data $p_{\text{data}}(\mathbf{x})$. The experiments are performed on 1 GPU.

Computational Efficiency

This section provides more experimental details on the computational efficiency experiments in Section 4.4.4. In the experiment, we consider two types of models.

MLP model We use a 3-layer MLP model to parameterize $\tilde{\mathbf{s}}_1(\tilde{\mathbf{x}}; \boldsymbol{\theta})$ for a 100 dimensional data distribution. As discussed in Section 4.4.2, our $\tilde{\mathbf{s}}_2(\tilde{\mathbf{x}}; \boldsymbol{\theta})$ model consists of two parts $\alpha(\tilde{\mathbf{x}}; \boldsymbol{\theta})$ and $\beta(\tilde{\mathbf{x}}; \boldsymbol{\theta})$. We use a 3-layer MLP model with comparable amount of parameters as $\tilde{\mathbf{s}}_1(\tilde{\mathbf{x}}; \boldsymbol{\theta})$ to parameterize $\alpha(\tilde{\mathbf{x}}; \boldsymbol{\theta})$ and $\beta(\tilde{\mathbf{x}}; \boldsymbol{\theta})$. We consider rank $r = 20, 50, 200$ and 1000 for $\beta(\tilde{\mathbf{x}}; \boldsymbol{\theta})$ in the experiment as reported in Table 4.2.

U-Net model We use a U-Net model to parameterize $\tilde{\mathbf{s}}_1(\tilde{\mathbf{x}}; \boldsymbol{\theta})$ for the 784 dimensional data distribution. We use a similar U-Net architecture as $\tilde{\mathbf{s}}_1(\tilde{\mathbf{x}}; \boldsymbol{\theta})$ for parameterizing $\alpha(\tilde{\mathbf{x}}; \boldsymbol{\theta})$ and $\beta(\tilde{\mathbf{x}}; \boldsymbol{\theta})$, except that we modify the output channel size to match the rank r of $\beta(\tilde{\mathbf{x}}; \boldsymbol{\theta})$. We consider rank $r = 20, 50, 200$ and 1000 for $\beta(\tilde{\mathbf{x}}; \boldsymbol{\theta})$ in the experiment as reported in Table 4.2. All the experiments are performed on the same TITAN Xp GPU using exactly the same computational setting. We use the implementation of U-Net from this repository <https://github.com/ermongroup/ncls>.

B.2.2 Uncertainty Quantification

This section provides more experimental details on Section 4.5.

Synthetic Experiments

This section provides more details on the synthetic data experiments. We use a 3-layer MLP model for both $\tilde{\mathbf{s}}_1(\tilde{\mathbf{x}}; \boldsymbol{\theta})$ and $\tilde{\mathbf{s}}_2(\tilde{\mathbf{x}}; \boldsymbol{\theta})$. We train $\tilde{\mathbf{s}}_1(\tilde{\mathbf{x}}; \boldsymbol{\theta})$ and $\tilde{\mathbf{s}}_2(\tilde{\mathbf{x}}; \boldsymbol{\theta})$ jointly with Eq. (4.15). We use $\sigma = 0.15$ for $q_\sigma(\tilde{\mathbf{x}}|\mathbf{x})$, and train the models using Adam optimizer until convergence. We observe that training $\tilde{\mathbf{s}}_1(\tilde{\mathbf{x}}; \boldsymbol{\theta})$ directly with DSM and training $\tilde{\mathbf{s}}_1(\tilde{\mathbf{x}}; \boldsymbol{\theta})$ jointly with Eq. (4.15) have the same empirical performance in terms of estimating $\tilde{\mathbf{s}}_1$. Thus, we train $\tilde{\mathbf{s}}_1(\tilde{\mathbf{x}}; \boldsymbol{\theta})$ jointly with $\tilde{\mathbf{s}}_2(\tilde{\mathbf{x}}; \boldsymbol{\theta})$ in our experiments.

Convariance Diagonal Visualizations

For both the MNIST and CIFAR-10 models, we use U-Net architectures to parameterize $\tilde{\mathbf{s}}_1(\tilde{\mathbf{x}}; \boldsymbol{\theta})$. We also use a similar U-Net architecture to parameterize $\tilde{\mathbf{s}}_2(\tilde{\mathbf{x}}; \boldsymbol{\theta})$, except that we modify the output channel size to match the rank r of $\beta(\tilde{\mathbf{x}}; \boldsymbol{\theta})$. We use $r = 50$ for $\beta(\tilde{\mathbf{x}}; \boldsymbol{\theta})$ for both MNIST and CIFAR-10 models. We use the U-Net model implementation from this repository <https://github.com/ermongroup/ncls>. We consider noise scales $\sigma = 0.3, 0.5, 0.8, 1.0$ for MNIST and $\sigma = 0.3, 0.5, 0.8$ for CIFAR-10. We train the models jointly until convergence with Eq. (4.15), using learning rate 0.0002 with Adam optimizer. The models are trained on the corresponding training sets on 2 GPUs.

Full Convariance Visualizations

We use U-Net architectures to parameterize $\tilde{\mathbf{s}}_1(\tilde{\mathbf{x}}; \boldsymbol{\theta})$. We also use a similar U-Net architecture to parameterize $\tilde{\mathbf{s}}_2(\tilde{\mathbf{x}}; \boldsymbol{\theta})$, except that we modify the output channel size to match the rank r of $\beta(\tilde{\mathbf{x}}; \boldsymbol{\theta})$. We use $r = 50$ for $\beta(\tilde{\mathbf{x}}; \boldsymbol{\theta})$ for this experiment. We use the U-Net model implementation from this repository <https://github.com/ermongroup/ncls>. We train the models until convergence, using learning rate 0.0002 with Adam optimizer. The models are trained on the corresponding training set on 2 GPUs. We provide extra eigenvector visualizations for Fig. 4.4 in Figs. B.2 and B.3.

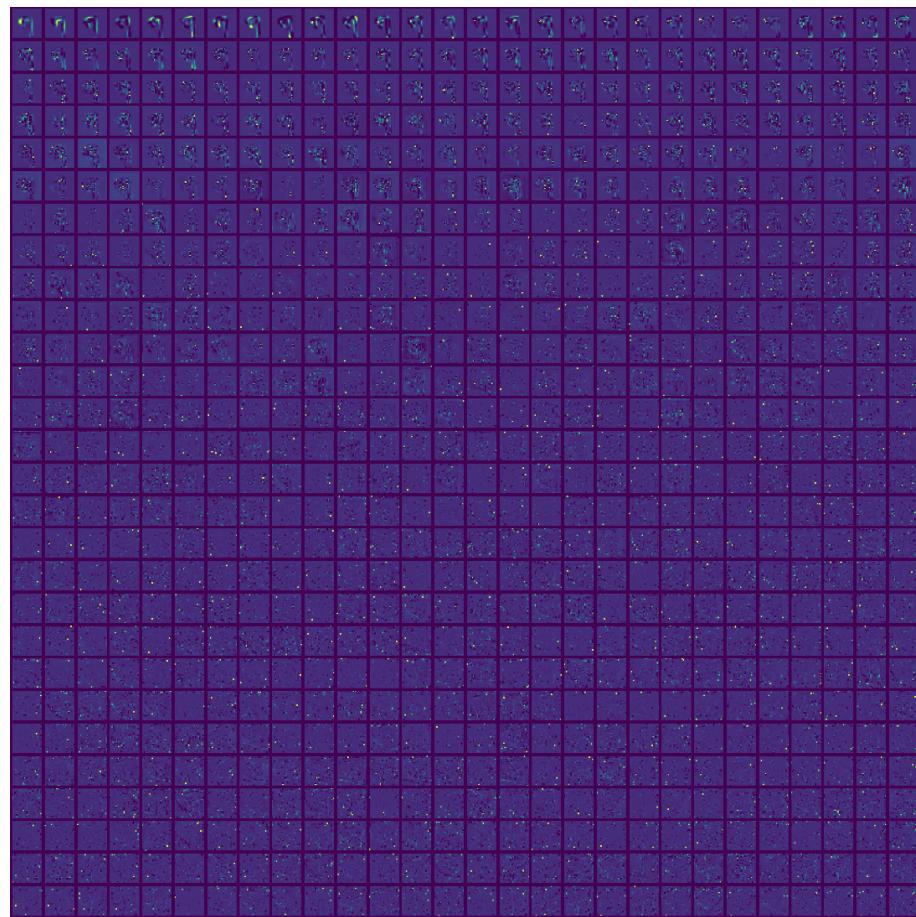


Figure B.2: Eigenvectors (sorted by eigenvalues) of $\text{Cov}[\mathbf{x} \mid \tilde{\mathbf{x}}]$ estimated by $\tilde{\mathbf{s}}_2(\tilde{\mathbf{x}}; \boldsymbol{\theta})$ on MNIST (more details in Section 4.5).



Figure B.3: Eigenvectors (sorted by eigenvalues) of $\text{Cov}[\mathbf{x} | \tilde{\mathbf{x}}]$ estimated by $\tilde{s}_2(\tilde{\mathbf{x}}; \theta)$ on MNIST (more details in Section 4.5).

B.2.3 Ozaki Sampling

This section provides more details on Section 4.6.

Synthetic Datasets

This section provides more details on Section 4.6.2. We use a 3-layer MLP model for both $\tilde{s}_1(\tilde{\mathbf{x}}; \theta)$ and $\tilde{s}_2(\tilde{\mathbf{x}}; \theta)$. Since we only need the diagonal of the second order score, we parameterize $\tilde{s}_2(\tilde{\mathbf{x}}; \theta)$ with a diagonal model (*i.e.* with only $\alpha(\tilde{\mathbf{x}}; \theta)$) and optimize the models jointly using Eq. (4.16). We use $\sigma = 0.1$ during training so that the noise perturbed distribution $q_\sigma(\tilde{\mathbf{x}})$ is close to $p_{\text{data}}(\mathbf{x})$. The models are trained with Adam optimizer with

learning rate 0.001.

Given the trained models, we perform a parameter search to find the optimal step size for both Langevin dynamics and Ozaki sampling. We also observe that Ozaki sampling can use a larger step size than Langevin dynamics, which is also discussed in [DK19]. We observe that the optimal step size for Ozaki sampling is $\epsilon = 5$ on Dataset 1 and $\epsilon = 6$ on Dataset 2, while the optimal step size for Langevin dynamics is $\epsilon = 0.5$ on Dataset 1 and $\epsilon = 2$ on Dataset 2. We also explore using the same setting of Ozaki sampling for Langevin dynamics (*i.e.* we use the optimal step size of Ozaki sampling and the same number of iterations). We present the results in Fig. B.4. We observe that the optimal step size for Ozaki sampling is too large for Langevin dynamics, and does not allow Langevin dynamics to generate reasonable samples. We also find that Ozaki sampling can converge using fewer iterations than Langevin dynamics even when using the same step size (see Fig. 4.6). All the experiments in this section are performed using 1 GPU.

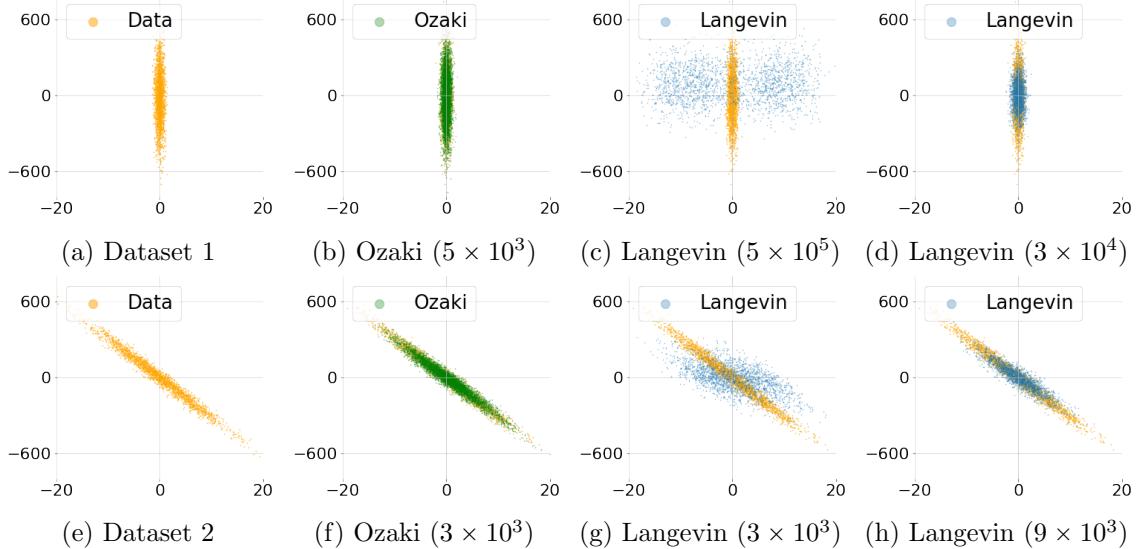


Figure B.4: Sampling 2-D synthetic data with score functions. The number in the parenthesis stands for the number of iterations used for sampling. We observe that Ozaki obtains more reasonable samples using 1/6 or 1/3 iterations compared to Langevin dynamics. The second column uses the optimal step size for Ozaki, and the third column uses the same step size and setting for Langevin dynamics. The fourth column uses the optimal step size for Langevin dynamics.

MNIST

We use the U-Net implementation from this repository <https://github.com/ermongroup/ncls>. We train the models until convergence on the corresponding MNIST training set using learning rate 0.0002 and Adam optimizer. We use 2 GPUs during training. As shown in [SE19b] (see also Chapter 5), sampling images from score-based models trained with DSM is challenging when σ is small due to the ill-conditioned estimated scores in the low density data region. In our experiments, we use a slightly larger $\sigma = 0.5$ to avoid the issues of training and sampling from $\tilde{\mathbf{s}}_1(\tilde{\mathbf{x}}; \boldsymbol{\theta})$ as discussed in [SE19b] (see also Chapter 5). We train the $\tilde{\mathbf{s}}_1(\tilde{\mathbf{x}}; \boldsymbol{\theta})$ and $\tilde{\mathbf{s}}_2(\tilde{\mathbf{x}}; \boldsymbol{\theta})$ jointly with Eq. (4.14).

For experiments on class label changes, we select 10 images with different class labels from the MNIST test set. For each of the image, we initialize 1000 chains using it as the initialization for sampling. We consider two sampling methods Langevin dynamics and Ozaki method in this section. For the generated images, we first denoise the sampled results with Eq. (4.4) and then use a pretrained classifier, which has 99.5% accuracy on MNIST test set classification, to classify the labels of the generated images in Fig. 4.8a.

B.3 Chapter 5

B.3.1 Architectures

The architecture of our NCSNs used in the experiments has three important components: instance normalization, dilated convolutions and U-Net-type architectures. Below we introduce more details about them and discuss how we modify them to suit our purpose. Our score networks are implemented in PyTorch.

Instance Normalization

We use conditional instance normalization [DSK17] so that $s_{\boldsymbol{\theta}}(\mathbf{x}, \sigma)$ takes account of σ when predicting the scores. In conditional instance normalization, a different set of scales and biases is used for different $\sigma \in \{\sigma_i\}_{i=1}^L$. More specifically, suppose \mathbf{x} is an input with C feature maps. Let μ_k and s_k denote the mean and standard deviation of the k -th feature map of \mathbf{x} , taken along the spatial axes. Conditional instance normalization is achieved by

$$\mathbf{z}_k = \gamma[i, k] \frac{\mathbf{x}_k - \mu_k}{s_k} + \beta[i, k],$$

where $\gamma \in \mathbb{R}^{L \times C}$ and $\beta \in \mathbb{R}^{L \times C}$ are learnable parameters, k denotes the index of feature maps, and i denotes the index of σ in $\{\sigma_i\}_{i=1}^L$.

However, one downside of instance normalization is that it completely removes the information of μ_k for different feature maps. This can lead to shifted colors in the generated images. To fix this issue, we propose a simple modification to conditional instance normalization. First, we compute the mean and standard deviation of μ_k 's and denote them as m and v respectively. Then, we add another learnable parameter $\alpha \in \mathbb{R}^{L \times C}$. The modified conditional instance normalization is defined as

$$\mathbf{z}_k = \gamma[i, k] \frac{\mathbf{x}_k - \mu_k}{s_k} + \beta[i, k] + \alpha[i, k] \frac{\mu_k - m}{v}.$$

We abbreviate this modification of conditional instance normalization as CondInstanceNorm++. In our architecture, we add CondInstanceNorm++ before every convolutional layer and pooling layer.

Dilated Convolutions

Dilated convolutions can be used to increase the size of receptive field while maintaining the resolution of feature maps. It has been shown very effective in semantic segmentation because they preserve the location information better using feature maps of larger resolutions. In our architecture design of NCSNs, we use it to replace all the subsampling layers except the first one.

U-Net Architecture

U-Net is an architecture with special skip connections. These skip connections help transfer lower level information in shallow layers to deeper layers of the network. Since the shallower layers often contain low level information such as location and shape, these skip connections help improve the result of semantic segmentation. For building $s_\theta(\mathbf{x}, \sigma)$, we use the architecture of RefineNet [Lin+17], a modern variant of U-Net that also incorporates ResNet designs. We refer the readers to [Lin+17] for a detailed description of the RefineNet architecture.

In our experiments, we use a 4-cascaded RefineNet. We use pre-activation residual blocks. We remove all batch normalizations in the RefineNet architecture, and replace them with CondInstanceNorm++. We replace the max pooling layers in Refine Blocks with average pooling, as average pooling is reported to produce smoother images for image generation

tasks such as style transfer. In addition, we also add CondInstanceNorm++ before each convolution and average pooling in the Refine Blocks, although no normalization is used in the original Refine Blocks. All activation functions are chosen to be ELU. As mentioned previously, we use dilated convolutions to replace the subsampling layers in residual blocks, except the first one. Following the common practice, we increase the dilation by a factor of 2 when proceeding to the next cascade. For CelebA and CIFAR-10 experiments, the number of filters for layers corresponding to the first cascade is 128, while the number of filters for other cascades are doubled. For MNIST experiments, the number of filters are halved.

B.3.2 Additional Experimental Details

Toy Experiments

For the results in Fig. 5.1, we train a ResNet with sliced score matching on CIFAR-10. We use pre-activation residual blocks, and the ResNet is structured as an auto-encoder, where the encoder contains 5 residual blocks and the decoder mirrors the architecture of the encoder. The number of filters for each residual block of the encoder part is respectively 32, 64, 64, 128 and 128. The 2nd and 4th residual block of the encoder subsamples the feature maps by a factor of two. We use ELU activations throughout the network. We train the network with 50000 iterations using Adam optimizer and a batch size of 128 and learning rate of 0.001. The experiment was run on one Titan XP GPU.

For the results in Fig. 5.2, we choose $p_{\text{data}} = \frac{1}{5}\mathcal{N}((-5, -5), \mathbf{I}) + \frac{4}{5}\mathcal{N}((5, 5), \mathbf{I})$. The score network is a 3-layer MLP with 128 hidden units and softplus activation functions. We train the score network with sliced score matching for 10000 iterations with Adam optimizer. The learning rate is 0.001, and the batch size is 128. The experiment was run on an Intel Core i7 GPU with 2.7GHz.

For the results in Fig. 5.3, we use the same toy distribution $p_{\text{data}} = \frac{1}{5}\mathcal{N}((-5, -5), \mathbf{I}) + \frac{4}{5}\mathcal{N}((5, 5), \mathbf{I})$. We generate 1280 samples for each subfigure of Fig. 5.3. The initial samples are all uniformly chosen in the square $[-8, 8] \times [-8, 8]$. For Langevin dynamics, we use $T = 1000$ and $\epsilon = 0.1$. For annealed Langevin dynamics, we use $T = 100$, $L = 10$ and $\epsilon = 0.1$. We choose $\{\sigma_i\}_{i=1}^L$ to be a geometric progression, with $L = 10$, $\sigma_1 = 10$ and $\sigma_{10} = 0.1$. Both Langevin methods use the ground-truth data score for sampling. The experiment was run on an Intel Core i7 GPU with 2.7GHz.

Image Generation

During training, we randomly flip the images in CelebA and CIFAR-10. All models are optimized by Adam with learning rate 0.001 for a total of 200000 iterations. The batch size is fixed to 128. We save one checkpoint every 5000 iterations. For MNIST, we choose the last checkpoint at the 200000-th training iteration. For selecting our CIFAR-10 and CelebA models, we generate 1000 images for each checkpoint and choose the one with the smallest FID score computed on these 1000 images. Our image samples and results in Table 5.1 are from these checkpoints. This model selection procedure is very similar to the approach used in ProgressiveGAN [Kar+18].

The inception and FID scores are computed using the official code from OpenAI¹ [Sal+16] and TTUR [Heu+17] authors² respectively. The architectures are described in Appendix B.3.1. When reporting the numbers in Table 5.1, we compute inception and FID scores based on a total of 50000 samples.

The baseline model uses the same score network. The only difference is that the score network is only conditioned on one noise level $\{\sigma_1 = 0.01\}$. When sampling using Langevin dynamics, we use $\epsilon = 2 \times 10^{-5}$ and $T = 1000$.

The models on MNIST were run with one Titan XP GPU, while the models on CelebA and CIFAR-10 used two Titan XP GPUs.

Image Inpainting

We use Algorithm B.2 for image inpainting. The hyperparameters are the same as those of the annealed Langevin dynamics used for image generation.

¹https://github.com/openai/improved-gan/tree/master/inception_score

²<https://github.com/bioinf-jku/TTUR>

Algorithm B.2 Inpainting with annealed Langevin dynamics.

Require: $\{\sigma_i\}_{i=1}^L, \epsilon, T$ $\triangleright \epsilon$ is the smallest step size; T is the number of iteration for each noise level.

Require: \mathbf{m}, \mathbf{x} $\triangleright \mathbf{m}$ is a mask to indicate regions not occluded; \mathbf{x} is the given image.

- 1: Initialize $\tilde{\mathbf{x}}_0$
- 2: **for** $i \leftarrow 1$ to L **do**
- 3: $\alpha_i \leftarrow \epsilon \cdot \sigma_i^2 / \sigma_L^2$ $\triangleright \alpha_i$ is the step size.
- 4: Draw $\tilde{\mathbf{z}} \sim \mathcal{N}(0, \sigma_i^2 \mathbf{I})$
- 5: $\mathbf{y} \leftarrow \mathbf{x} + \tilde{\mathbf{z}}$
- 6: **for** $t \leftarrow 1$ to T **do**
- 7: Draw $\mathbf{z}_t \sim \mathcal{N}(0, \mathbf{I})$
- 8: $\tilde{\mathbf{x}}_t \leftarrow \tilde{\mathbf{x}}_{t-1} + \frac{\alpha_i}{2} s_{\theta}(\tilde{\mathbf{x}}_{t-1}, \sigma_i) + \sqrt{\alpha_i} \mathbf{z}_t$
- 9: $\tilde{\mathbf{x}}_t \leftarrow \tilde{\mathbf{x}}_t \odot (1 - \mathbf{m}) + \mathbf{y} \odot \mathbf{m}$
- 10: **end for**
- 11: $\tilde{\mathbf{x}}_0 \leftarrow \tilde{\mathbf{x}}_T$
- 12: **end for**
- return $\tilde{\mathbf{x}}_T$

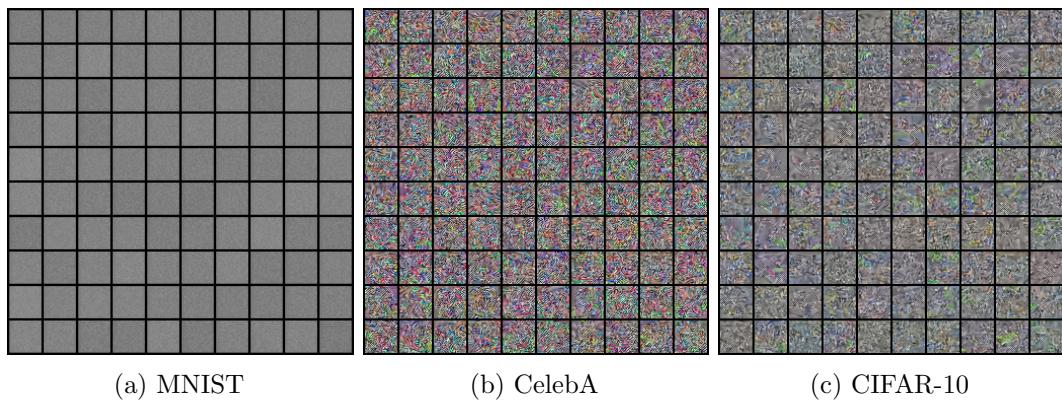
B.3.3 Samples**Samples from the Baseline Models**

Figure B.5: Uncurated samples on MNIST, CelebA, and CIFAR-10 datasets from the baseline model.

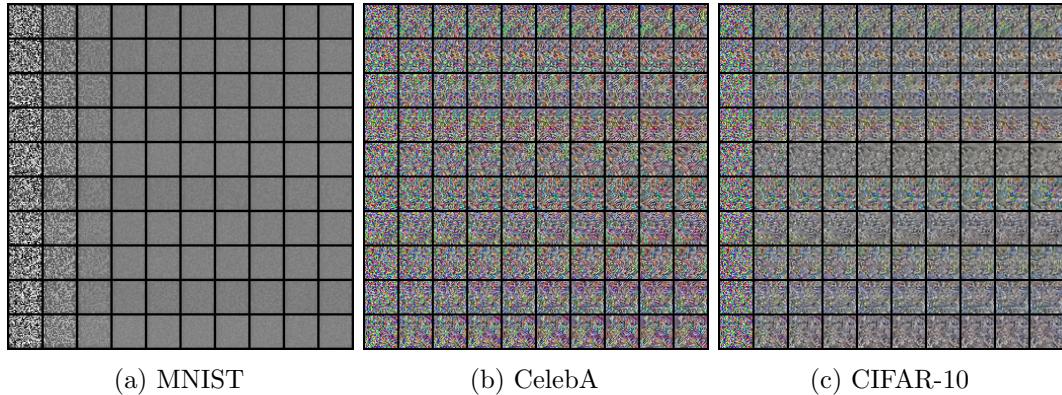


Figure B.6: Intermediate samples from Langevin dynamics for the baseline model.

Nearest Neighbors



Figure B.7: Nearest neighbors measured by the ℓ_2 distance between images. Images on the left of the red vertical line are samples from NCSN. Images on the right are nearest neighbors in the training dataset.

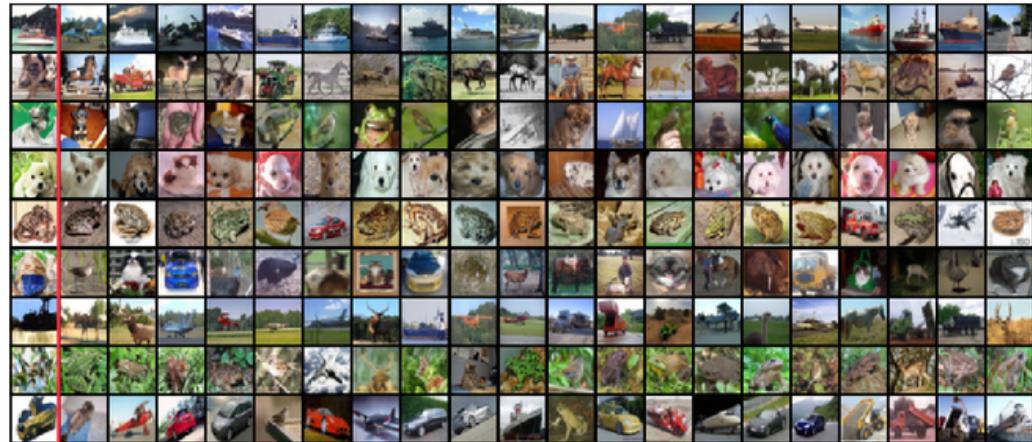


Figure B.8: Nearest neighbors measured by the ℓ_2 distance in the feature space of an Inception V3 network pretrained on ImageNet. Images on the left of the red vertical line are samples from NCSN. Images on the right are nearest neighbors in the training dataset.

Extended Samples

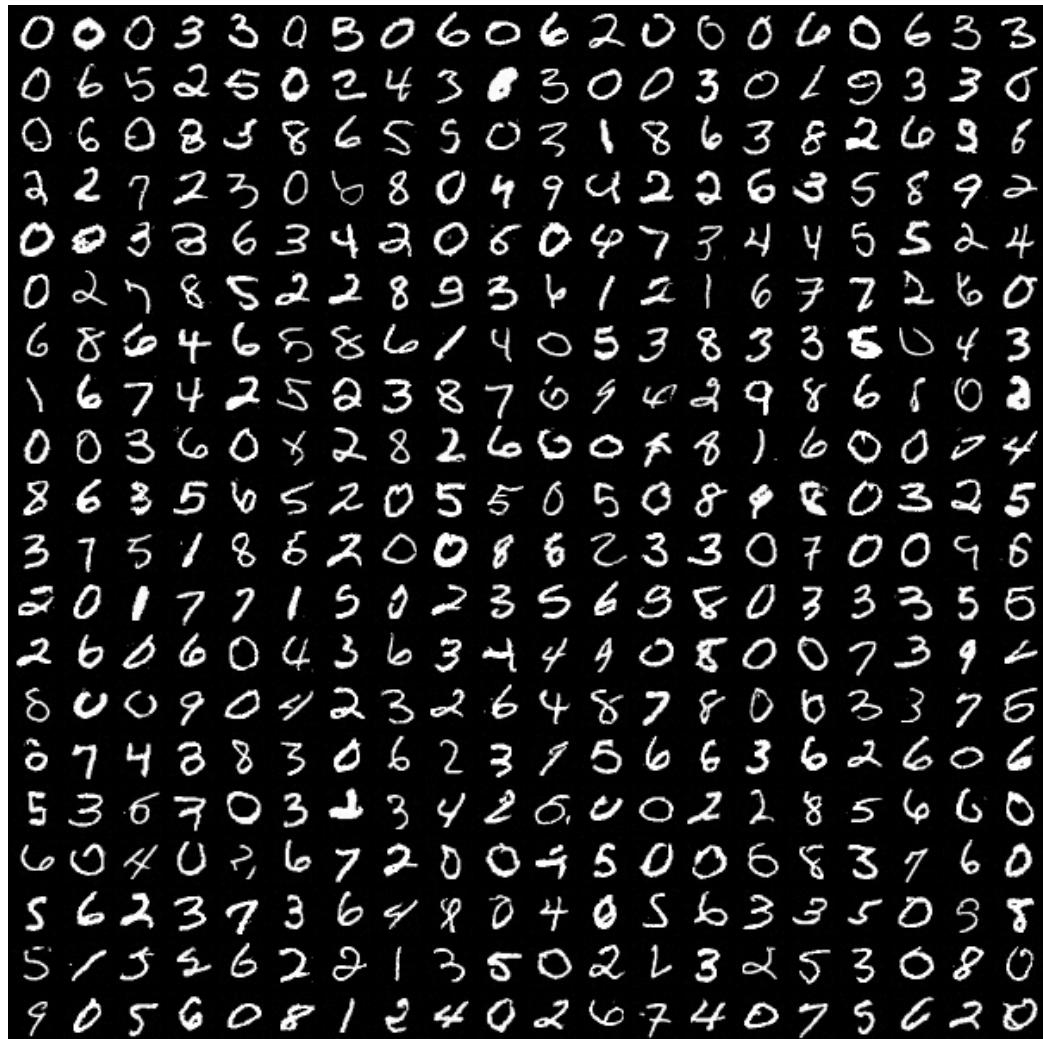


Figure B.9: Extended MNIST samples

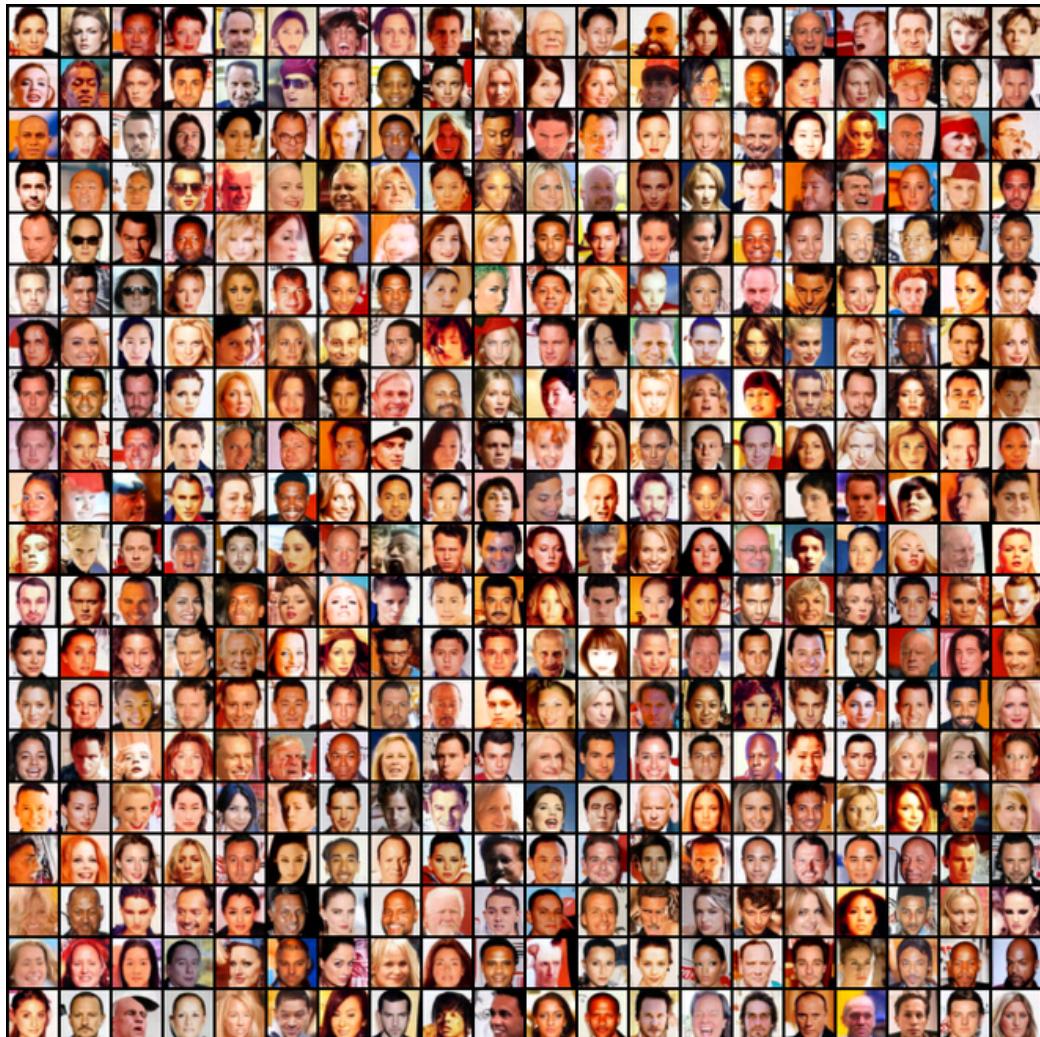


Figure B.10: Extended CelebA samples

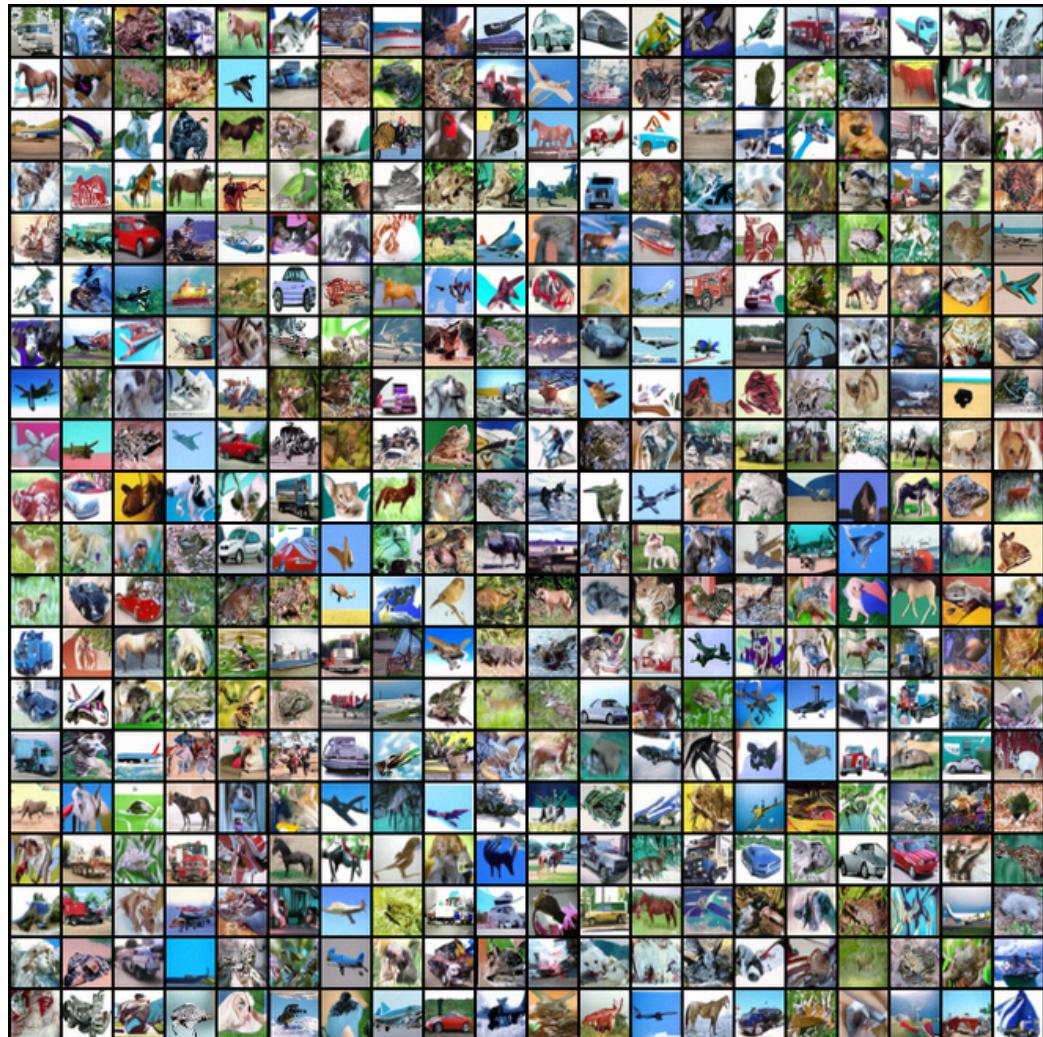


Figure B.11: Extended CIFAR-10 samples

Extended Intermediate Samples from Annealed Langevin Dynamics

Figure B.12: Extended intermediate samples from annealed Langevin dynamics for CelebA.



Figure B.13: Extended intermediate samples from annealed Langevin dynamics for CelebA.

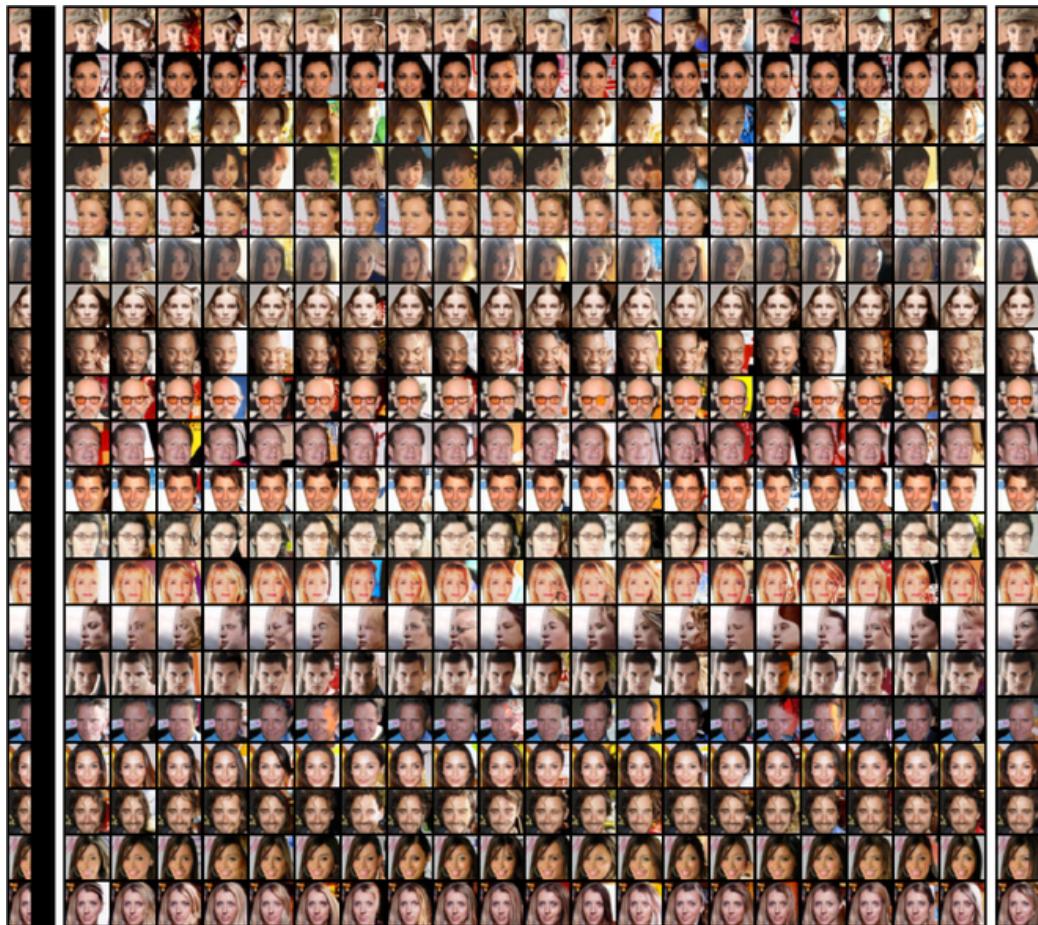
Extended Image Inpainting Results

Figure B.14: Extended image inpainting results for CelebA. The leftmost column of each figure shows the occluded images, while the rightmost column shows the original images.

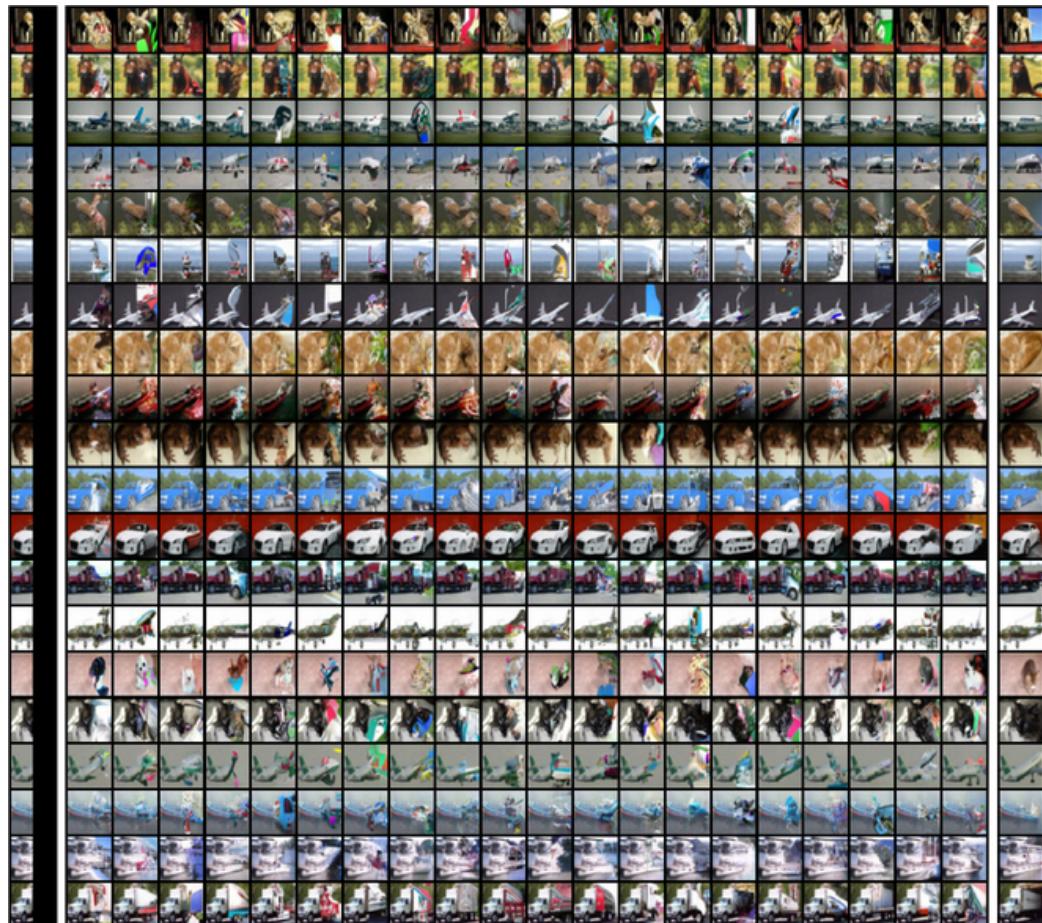


Figure B.15: Extended image inpainting results for CIFAR-10. The leftmost column of each figure shows the occluded images, while the rightmost column shows the original images.

B.4 Chapter 6

B.4.1 Experimental Details

Network Architectures and Hyperparameters

The original NCSN in Chapter 5 uses a network structure based on RefineNet [Lin+17]—a classical architecture for semantic segmentation. There are three major modifications to the original RefineNet in NCSN: (i) adding an enhanced version of conditional instance normalization (designed in Chapter 5 and named CondInstanceNorm++) for every convolutional layer; (ii) replacing max pooling with average pooling in RefineNet blocks; and (iii) using dilated convolutions in the ResNet backend of RefineNet. We use exactly the same architecture for NCSN experiments, but for NCSNv2 or any other architecture implementing Technique 6.3, we apply the following modifications: (i) setting the number of classes in CondInstanceNorm++ to 1 (which we name as InstanceNorm++); (ii) changing average pooling back to max pooling; and (iii) removing all normalization layers in RefineNet blocks. Here (ii) and (iii) do not affect the results much, but they are included because we hope to minimize the number of unnecessary changes to the standard RefineNet architecture (the original RefineNet blocks in [Lin+17] use max pooling and have no normalization layers). We name a ResNet block (with InstanceNorm++ instead of BatchNorm) “ResBlock”, and a RefineNet block “RefineBlock”. When CondInstanceNorm++ is added, we name them “CondResBlock” and “CondRefineBlock” respectively. We use the ELU activation function [CUH15] throughout all architectures.

To ensure sufficient capacity and receptive fields, the network structures for images of different resolutions have different numbers of layers and filters. We summarize the architectures in Table B.7 and Table B.8.

Table B.7: The architectures of NCSN for images of various resolutions.

(a) NCSN 32^2 - 64^2	(b) NCSN 96^2 - 128^2
3x3 Conv2D, 128	3x3 Conv2D, 128
CondResBlock, 128	CondResBlock, 128
CondResBlock, 128	CondResBlock, 128
CondResBlock down, 256	CondResBlock down, 256
CondResBlock, 256	CondResBlock, 256
CondResBlock down, 256 dilation 2	CondResBlock down, 256
CondResBlock, 256 dilation 2	CondResBlock, 256
CondResBlock down, 256 dilation 4	CondResBlock down, 512 dilation 2
CondResBlock, 256 dilation 4	CondResBlock, 512 dilation 2
CondRefineBlock, 256	CondResBlock down, 512 dilation 4
CondRefineBlock, 256	CondResBlock, 512 dilation 4
CondRefineBlock, 128	CondRefineBlock, 512
CondRefineBlock, 128	CondRefineBlock, 256
3x3 Conv2D, 3	CondRefineBlock, 256
	CondRefineBlock, 128
	CondRefineBlock, 128
	3x3 Conv2D, 3

Table B.8: The architectures of NCSNv2 for images of various resolutions.

(a) NCSNv2 32^2 – 64^2	(b) NCSNv2 96^2 – 128^2	(c) NCSNv2 256^2
3x3 Conv2D, 128	3x3 Conv2D, 128	3x3 Conv2D, 128
ResBlock, 128	ResBlock, 128	ResBlock, 128
ResBlock, 128	ResBlock, 128	ResBlock, 128
ResBlock down, 256	ResBlock down, 256	ResBlock down, 256
ResBlock, 256	ResBlock, 256	ResBlock, 256
ResBlock down, 256 dilation 2	ResBlock down, 256	ResBlock down, 256
ResBlock, 256 dilation 2	ResBlock, 256	ResBlock, 256
ResBlock down, 256 dilation 4	ResBlock down, 512 dilation 2	ResBlock down, 512 dilation 2
ResBlock, 256 dilation 4	ResBlock, 512 dilation 2	ResBlock, 512 dilation 2
RefineBlock, 256	ResBlock down, 512 dilation 4	ResBlock down, 512 dilation 4
RefineBlock, 256	ResBlock, 512 dilation 4	ResBlock, 512 dilation 4
RefineBlock, 128	RefineBlock, 512	RefineBlock, 512
RefineBlock, 128	RefineBlock, 256	RefineBlock, 256
3x3 Conv2D, 3	RefineBlock, 256	RefineBlock, 256
	RefineBlock, 128	RefineBlock, 128
		3x3 Conv2D, 3

We use the Adam optimizer [KB14] for all models. When Technique 6.3 is not in effect, we choose the learning rate 0.001; otherwise we use a learning rate 0.0001 to avoid loss explosion. We set the ϵ parameter of Adam to 10^{-3} for FFHQ and 10^{-8} otherwise. We provide other hyperparameters in Table B.9, where σ_1 , L , T , and ϵ of NCSNv2 are all chosen in accordance with our proposed techniques. When the number of training data is larger than 60000, we randomly sample 10000 of them and compute the maximum pairwise distance, which is set as σ_1 for NCSNv2.

Table B.9: Hyperparameters of NCSN/NCSNv2. The latter is configured according to Techniques 6.1 to 6.4. σ_1 and L determine the set of noise levels. T and ϵ are parameters of annealed Langevin dynamics.

Model	Dataset	σ_1	L	T	ϵ	Batch size	Training iterations
NCSN	CIFAR-10 32 ²	1	10	100	2e-5	128	300k
NCSN	CelebA 64 ²	1	10	100	2e-5	128	210k
NCSN	LSUN church_outdoor 96 ²	1	10	100	2e-5	128	200k
NCSN	LSUN bedroom 128 ²	1	10	100	2e-5	64	150k
NCSNv2	CIFAR-10 32 ²	50	232	5	6.2e-6	128	300k
NCSNv2	CelebA 64 ²	90	500	5	3.3e-6	128	210k
NCSNv2	LSUN church_outdoor 96 ²	140	788	4	4.9e-6	128	200k
NCSNv2	LSUN bedroom/tower 128 ²	190	1086	3	1.8e-6	128	150k
NCSNv2	FFHQ 256 ²	348	2311	3	0.9e-7	32	80k

Additional Settings

Datasets: We use the following datasets in our experiments: CIFAR-10 [KH+09], CelebA [Liu+15], LSUN [Yu+15], and FFHQ [KLA19]. CIFAR-10 contains 50000 training images and 10000 test images, all of resolution 32×32 . CelebA contains 162770 training images and 19962 test images with various resolutions. For preprocessing, we first center crop them to size 140×140 , and then resize them to 64×64 . We choose the church_outdoor, bedroom and tower categories in the LSUN dataset. They contain 126227, 3033042, and 708264 training images respectively, and all have 300 validation images. For preprocessing, we first resize them so that the smallest dimension of images is 96 (for church_outdoor) or 128 (for bedroom and tower), and then center crop them to equalize their lengths and heights. Finally, the

FFHQ dataset consists of 70000 high-quality facial images at resolution 1024×1024 . We resize them to 256×256 in our experiments. Because FFHQ does not have an official test dataset, we randomly select 63000 images for training and the remaining 7000 as the test dataset. In addition, we apply random horizontal flip as data augmentation in all cases.

Metrics: We use FID [Heu+17] and HYPE $_{\infty}$ [Zho+19] scores for quantitative comparison of results. When computing FIDs on CIFAR-10 32×32 , we measure the distance between the statistics of samples and training data. When computing FIDs on CelebA 64×64 , we follow the settings in [SE19a] where the distance is measured between 10000 samples and the test dataset. We use the official website <https://hype.stanford.edu> for computing HYPE $_{\infty}$ scores. Regarding model selection, we follow the settings in Chapter 5, where we compute FID scores on 1000 samples every 5000 training iterations and choose the checkpoint with the smallest FID for computing both full FID scores (with more samples from it) and the HYPE $_{\infty}$ scores.

Training: We use the Adam [KB14] optimizer with default hyperparameters. The learning rates and batch sizes are provided in Appendix B.4.1 and Table B.9. We observe that for images at resolution 128×128 or 256×256 , training can be unstable when the loss is near convergence. We note, however, this is a well-known problem of the Adam optimizer, and can be mitigated by techniques such as AMSGrad [RKK18]. We trained all models on Nvidia Tesla V100 GPUs.

Settings for Section 6.3.3: The loss curves in Fig. 6.3 are results of two settings: (i) Techniques 6.1, 6.2, 6.4 and 6.5 are in effect, but the model architecture is the same as the original NCSN (*i.e.*, Table B.7a); and (ii) all techniques are in effect, *i.e.*, the model is the same as NCSNv2 depicted in Table B.8a. We apply EMA with momentum 0.9 to smooth the curves in Fig. 6.3. We observe that despite being simpler to implement, the new noise conditioning method proposed in Technique 6.3 performs as well as the original and arguably more complex one in Chapter 5 in terms of the training loss. See the ablation studies in Section 6.6 and Appendix B.4.2 for additional results.

Interpolation: We can interpolate between two different samples from NCSN/NCSNv2 via interpolating the Gaussian random noise injected by annealed Langevin dynamics. Specifically, suppose we have a total of L noise levels, and for each noise level we run T steps of Langevin dynamics. Let

$$\{\mathbf{z}_{ij}\}_{1 \leq i \leq L, 1 \leq j \leq T} := \{\mathbf{z}_{11}, \mathbf{z}_{12}, \dots, \mathbf{z}_{1T}, \mathbf{z}_{21}, \mathbf{z}_{22}, \dots, \mathbf{z}_{2T}, \dots, \mathbf{z}_{L1}, \mathbf{z}_{L2}, \dots, \mathbf{z}_{LT}\}$$

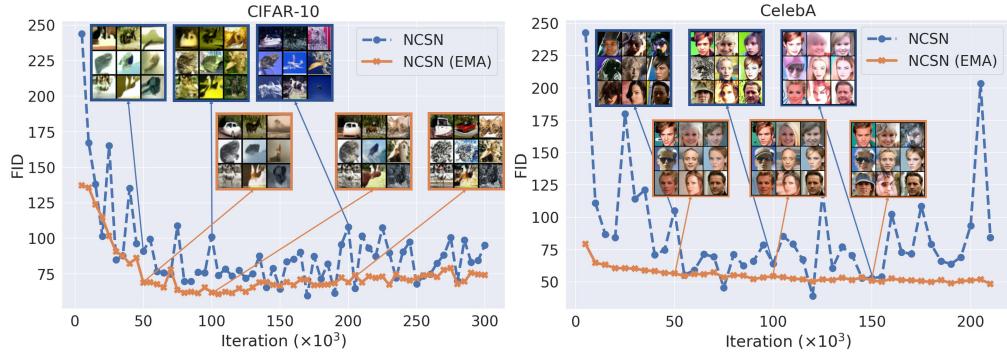


Figure B.16: FIDs and color artifacts over the course of training (best viewed in color). The FIDs of NCSN have much higher volatility compared to NCSN with EMA. Samples from the vanilla NCSN often have obvious color shifts. All FIDs are computed without the denoising step.

denote the set of all Gaussian noise used in this procedure, where \mathbf{z}_{ij} is the noise injected at the j -th iteration of Langevin dynamics corresponding to the i -th noise level. Next, suppose we have two samples $\mathbf{x}^{(1)}$ and $\mathbf{x}^{(2)}$ with the same initialization \mathbf{x}_0 , and denote the corresponding set of Gaussian noise as $\{\mathbf{z}_{ij}^{(1)}\}_{1 \leq i \leq L, 1 \leq j \leq T}$ and $\{\mathbf{z}_{ij}^{(2)}\}_{1 \leq i \leq L, 1 \leq j \leq T}$ respectively. We can generate N interpolated samples between $\mathbf{x}^{(1)}$ and $\mathbf{x}^{(2)}$, where for the k -th interpolated sample we use Gaussian noise $\{\cos(\frac{k\pi}{2(N+1)})\mathbf{z}_{ij}^{(1)} + \sin(\frac{k\pi}{2(N+1)})\mathbf{z}_{ij}^{(2)}\}_{1 \leq i \leq L, 1 \leq j \leq T}$ and initialization \mathbf{x}_0 .

B.4.2 Additional Experimental Results

Additional Results without the Denoising Step

We further demonstrate the stabilizing effect of EMA in Fig. B.16, where FIDs are computed without the denoising step. As indicated by Figs. 6.4 and B.16, EMA can stabilize training and remove sample artifacts regardless of whether denoising is used or not.

FID scores should be interpreted with caution because they may not align well with human judgement. For example, the samples from NCSNv2 as demonstrated in Fig. B.18b have an FID score of 28.9 (without denoising), worse than NCSN (Fig. B.18a) whose FID is 26.9 (without denoising), but arguably produce much more visually appealing samples. To investigate whether FID scores align well with human ratings, we use the HYPE_∞ [Zho+19] score (higher is better), a metric of sample quality based on human evaluation, to compare the two models that generated samples in Figs. B.18a and B.18b. We provide full results in Table B.10, where all numbers except those for NCSN and NCSNv2 are directly taken from

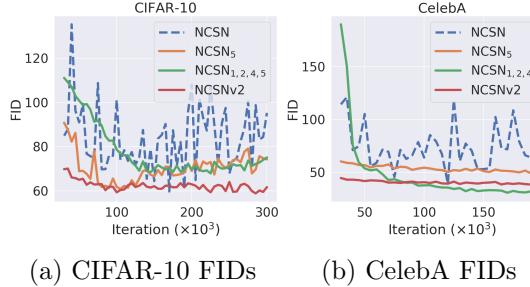


Figure B.17: FIDs for different groups of techniques. Subscripts of “NCSN” are IDs of techniques in effect. “NCSNV2” uses all techniques. Results are computed without the denoising step.

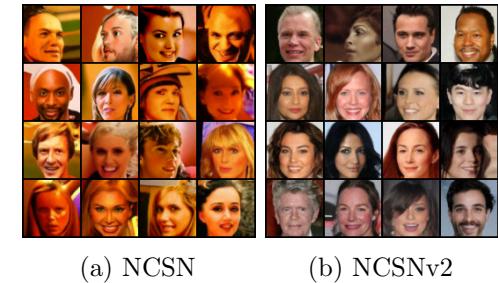


Figure B.18: Uncurated samples from NCSN (a) and NCSNV2 (b) on CelebA 64×64 .

[Zho+19]. As Table B.10 shows, our NCSNV2 achieves 37.3 on CelebA 64×64 which is comparable to ProgressiveGAN [Kar+18], whereas NCSN achieves 19.8. This is completely different from the ranking indicated by FIDs.

Table B.10: HYPE_∞ scores on CelebA 64×64 . *With truncation tricks.

Model	$\text{HYPE}_\infty(\%)$	Fakes Error(%)	Reals Error(%)	Std.
StyleGAN* [KLA19]	50.7	62.2	39.3	1.3
ProgressiveGAN [Kar+18]	40.3	46.2	34.4	0.9
BEGAN [BSM17]	10	6.2	13.8	1.6
WGAN-GP [Gul+17]	3.8	1.7	5.9	0.6
NCSN	19.8	22.3	17.3	0.4
NCSNV2	37.3	49.8	24.8	0.5

* with truncation tricks

Finally, we provide ablation results without the denoising step in Fig. B.17. It is qualitatively similar to Fig. 6.5 where results are computed with denoising.

Training and sampling speed

In Table B.11, we provide the time cost for training and sampling from NCSNv2 models on various datasets considered in our experiments.

Table B.11: Training and sampling speed of NCSNv2 on various datasets.

Dataset	Device	Sampling time	Training time
CIFAR-10	2x V100	2 min	22 h
CelebA	4x V100	7 min	29 h
Church	8x V100	17 min	52 h
Bedroom	8x V100	19 min	52 h
Tower	8x V100	19 min	52 h
FFHQ	8x V100	50 min	41 h

Color Shifts

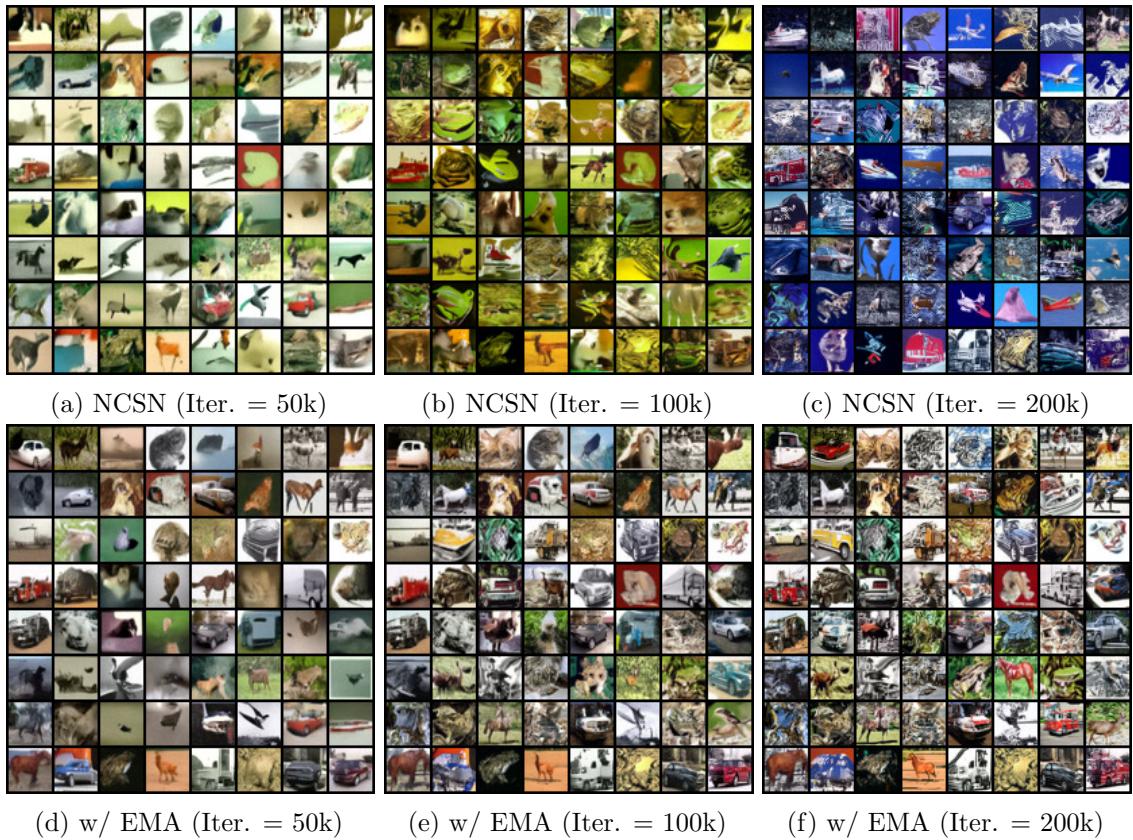


Figure B.19: EMA reduces undesirable color shifts on CIFAR-10. We show samples from NCSN and NCSN with EMA at the 50k/100k/200k-th training iteration.

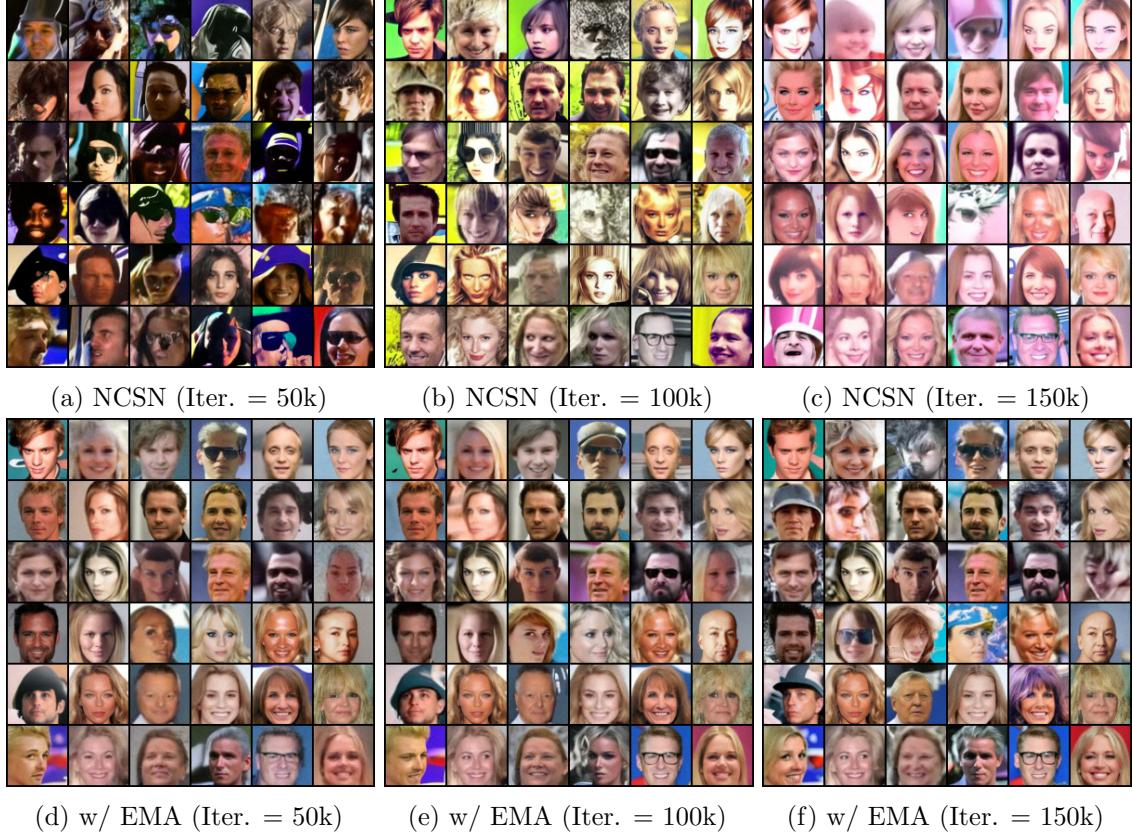


Figure B.20: EMA reduces undesirable color shifts on CelebA-10. We show samples from NCSN and NCSN with EMA at the 50k/100k/150k-th training iteration.

Additional Results on Ablation Studies

As discussed in Section 6.6, we partition all techniques into three groups: (i) Technique 6.5, (ii) Techniques 6.1, 6.2 and 6.4, and (iii) Technique 6.3, and investigate the performance of models after successively removing (iii), (ii), and (i) from NCSNv2. Aside from the FID curves in Figs. 6.5 and B.17, we also provide samples from different models for visual inspection in Figs. B.21 and B.22. To generate these samples, we compute the FID scores on 1000 samples every 5000 training iterations for each considered model, and sample from the checkpoint of the smallest FID (the same setting as in Chapter 5. From samples in Figs. B.21 and B.22, we easily observe that removing any group of techniques leads to worse samples.

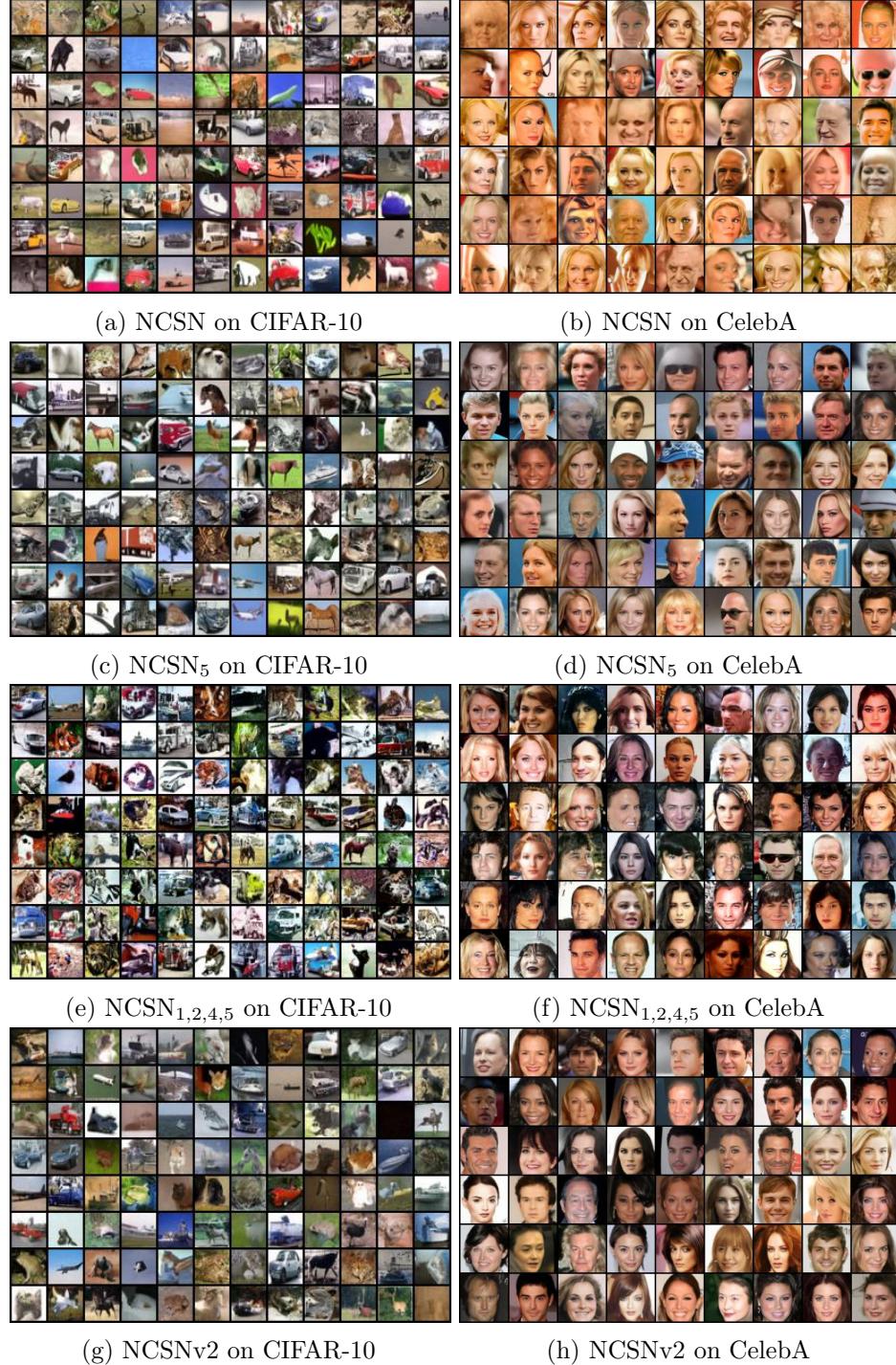


Figure B.21: Samples from models with different groups of techniques applied. NCSN is the original model in Chapter 5 and does not use any of the newly proposed techniques. Subscripts of “NCSN” denote the IDs of techniques in effect. NCSN₅ only applies EMA. NCSN_{1,2,4,5} applies both EMA and technique group (ii). NCSNv2 is the result of all techniques combined. Checkpoints are selected according to the lowest FID (with denoising) over the course of training.

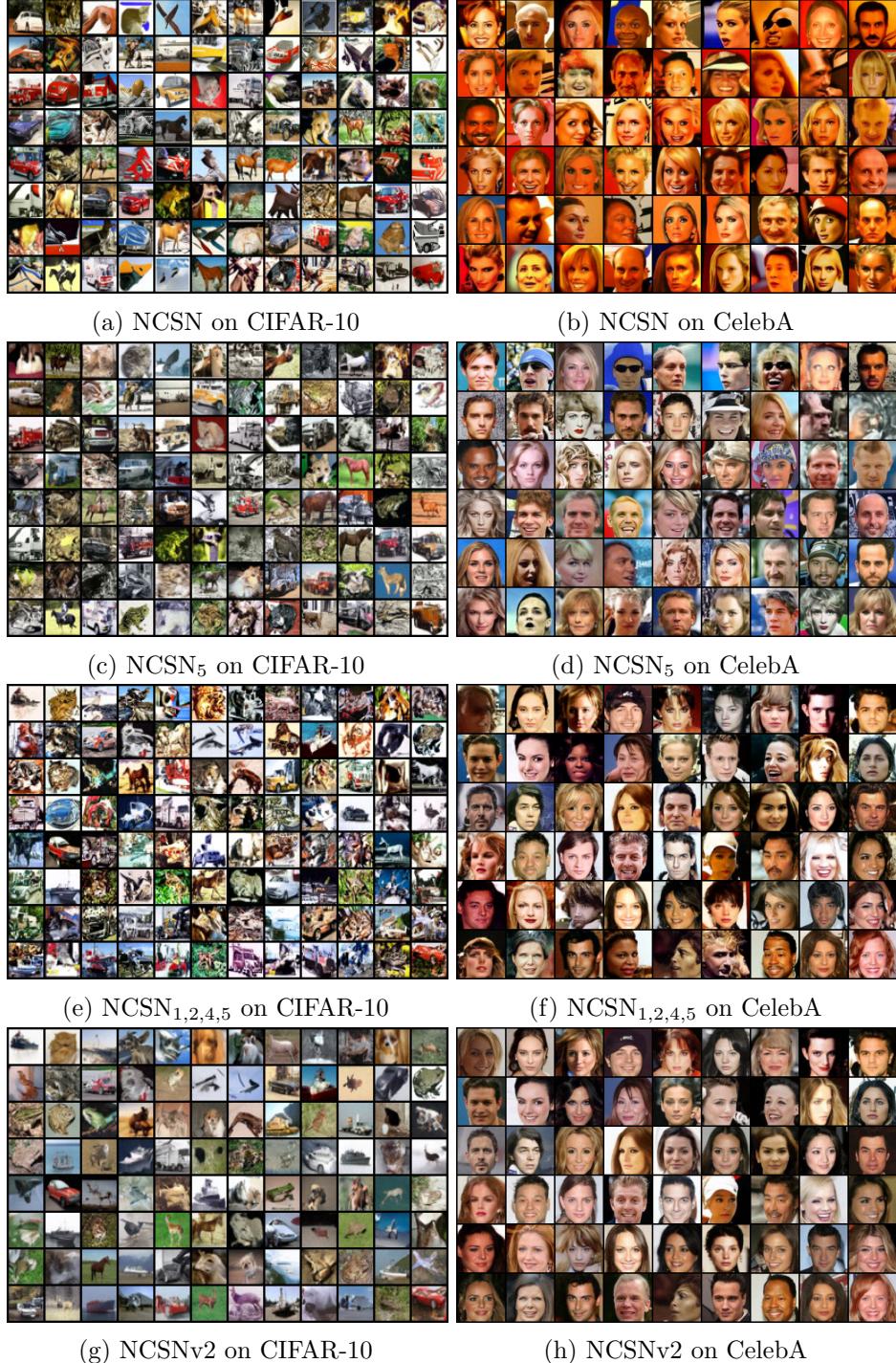


Figure B.22: Samples from models with different groups of techniques applied. NCSN is the original model in Chapter 5 and does not use any of the newly proposed techniques. Subscripts of “NCSN” denote the IDs of techniques in effect. NCSN₅ only applies EMA. NCSN_{1,2,4,5} applies both EMA and technique group (ii). NCSNv2 is the result of all techniques combined. Checkpoints are selected according to the lowest FID (without denoising) over the course of training.

Generalization

Loss curves First, we demonstrate that our NCSNv2 does not overfit to the training dataset by showing the curves of training/test loss in Fig. B.23. Since the loss on the test dataset is always close to the loss on the training dataset during the course of training, this indicates that our model does not simply memorize training data.

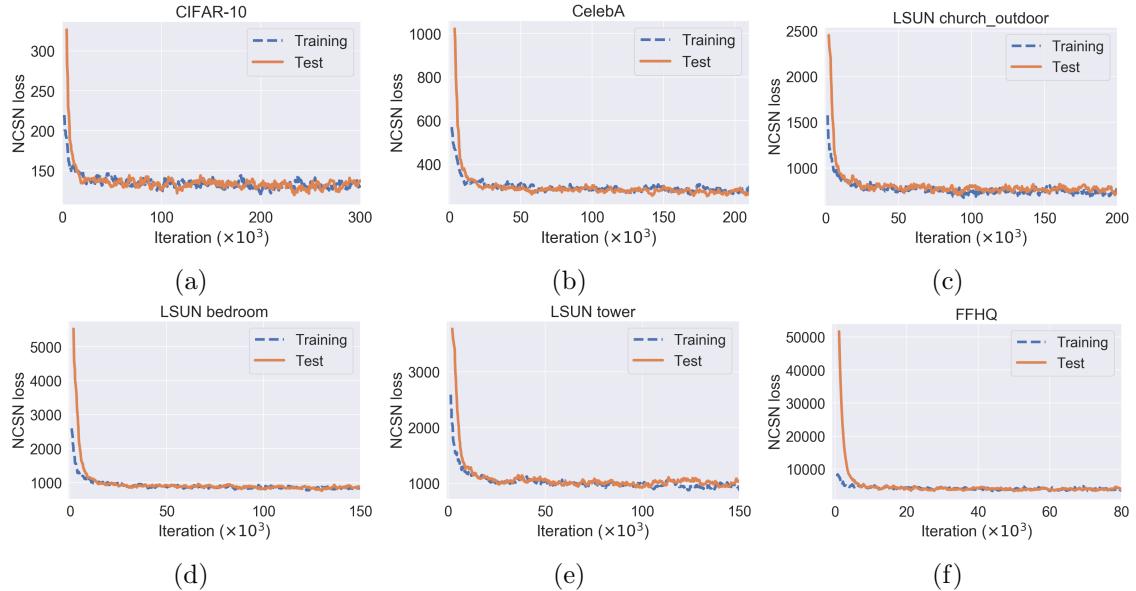


Figure B.23: Training vs. test loss curves of NCSNv2.

Nearest neighbors Starting from this section, all samples are from NCSNv2 at the last training iteration. For each generated sample, we show the nearest neighbors from the training dataset, measured by ℓ_2 distance in the feature space of a pre-trained InceptionV3 network. Since we apply random horizontal flip when training, we also take this into consideration when computing nearest neighbors, so that we can detect cases in which NCSNv2 memorizes a flipped training data point.



Figure B.24: Nearest neighbors on CIFAR-10. NCSNv2 samples are on the left side of the red vertical line. Corresponding nearest neighbors are on the right side in the same row.

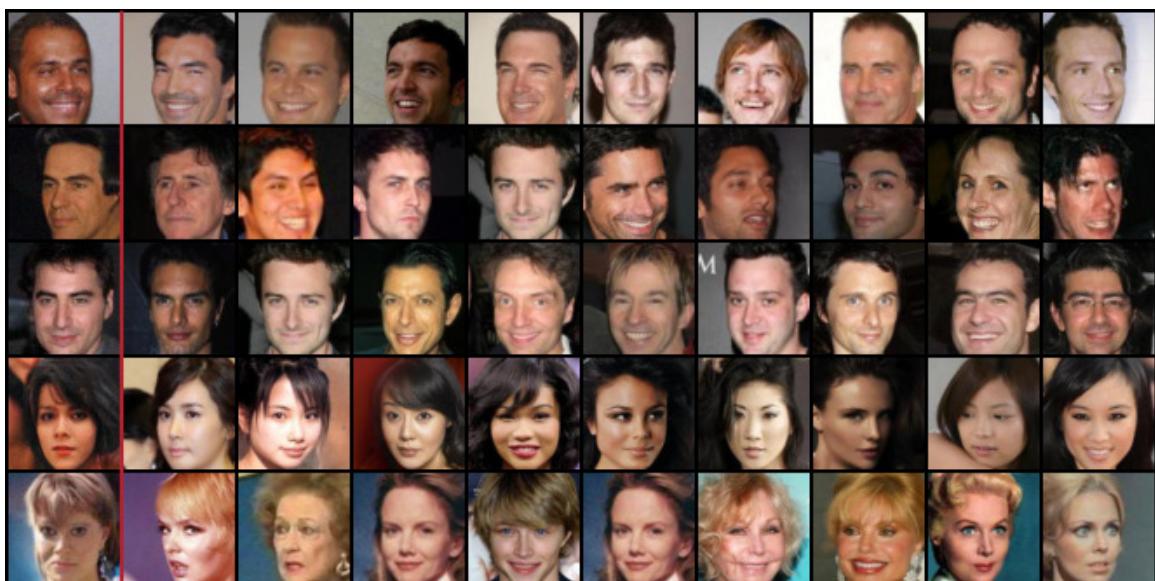
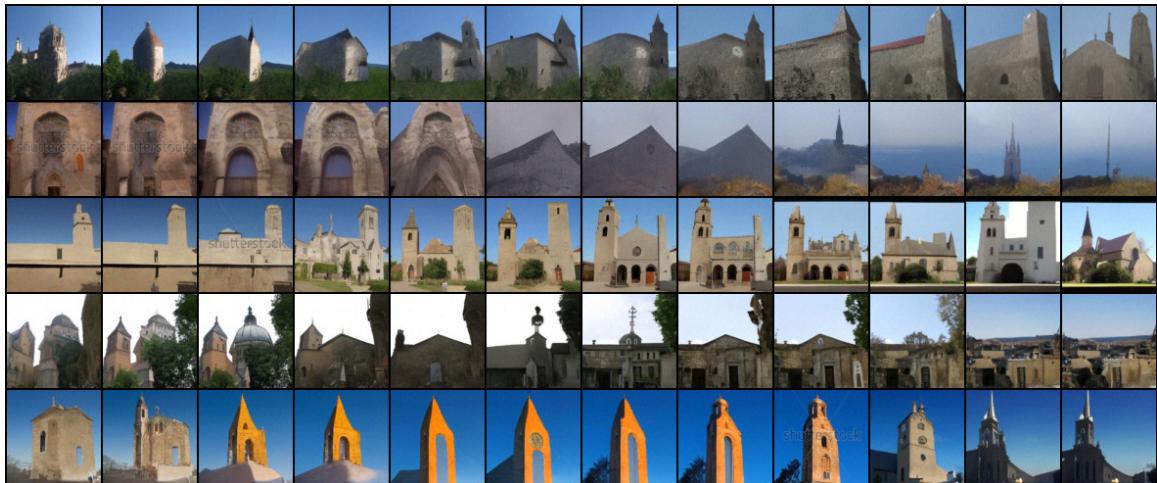


Figure B.25: Nearest neighbors on CelebA 64 × 64.

Figure B.26: Nearest neighbors on LSUN church_outdoor 96×96 .Figure B.27: Nearest neighbors on FFHQ 256×256 .

Additional interpolation results We generate samples from NCSNv2 and interpolate between them using the method described in Appendix B.4.1.

Figure B.28: NCSNv2 interpolation results on CelebA 64×64 .Figure B.29: NCSNv2 interpolation results on LSUN church_outdoor 96×96 .

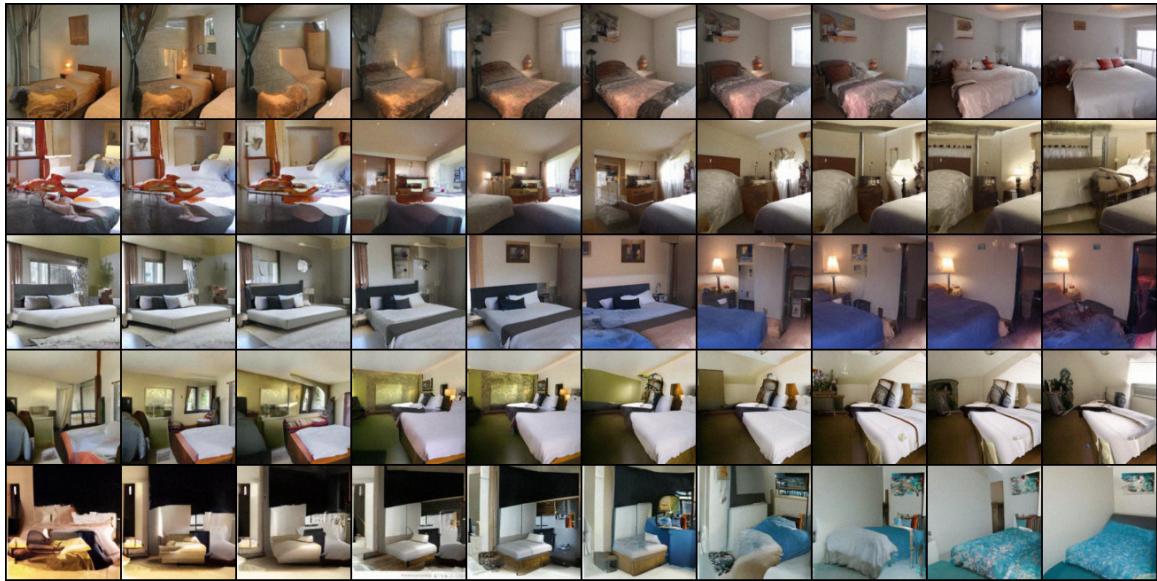


Figure B.30: NCSNv2 interpolation results on LSUN bedroom 128×128 .

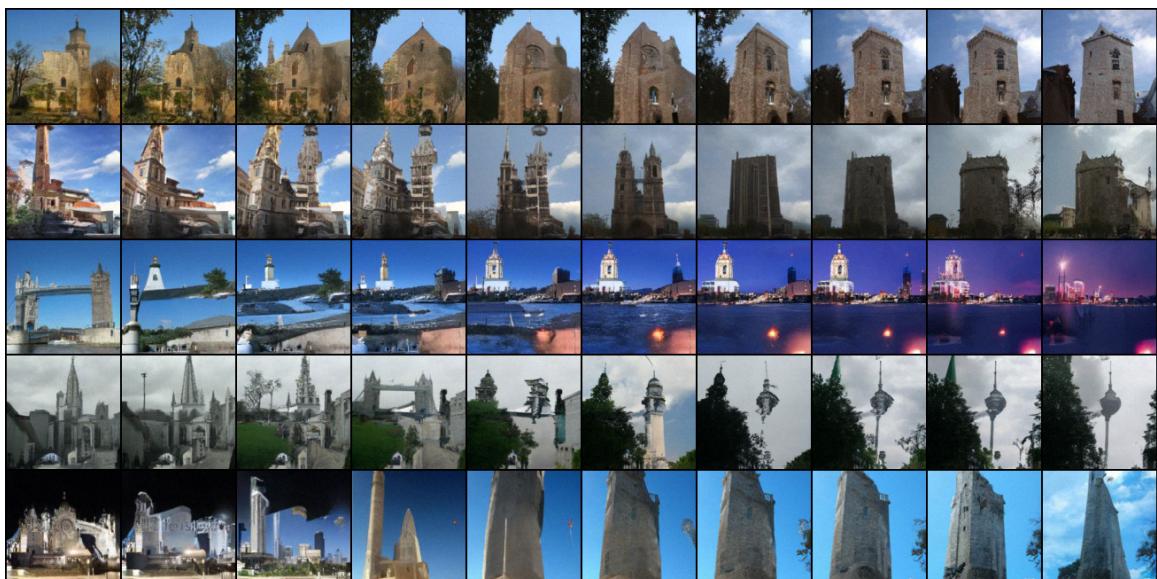


Figure B.31: NCSNv2 interpolation results on LSUN tower 128×128 .



Figure B.32: NCSNv2 interpolation results on FFHQ 256×256 .

Additional Uncurated Samples

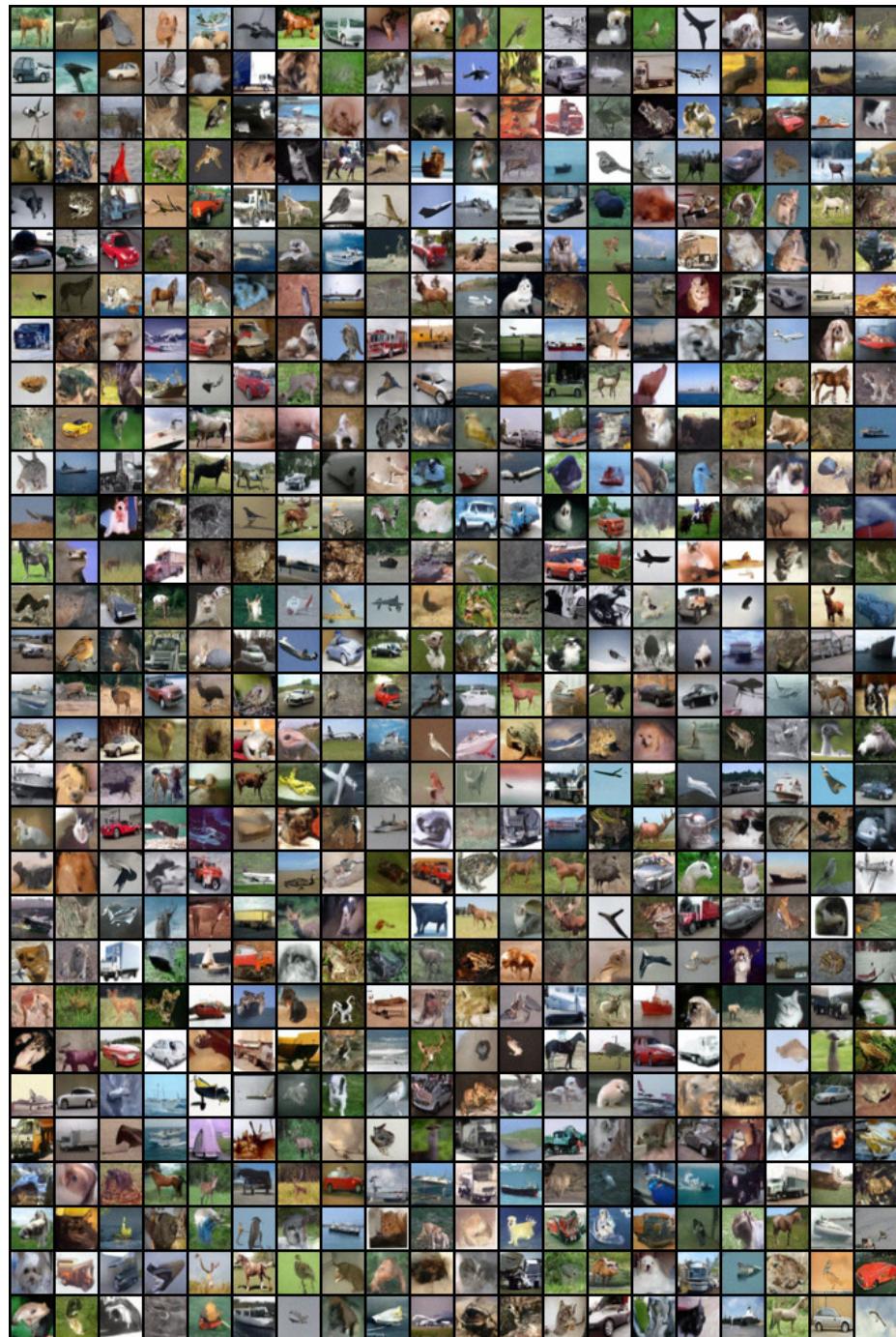


Figure B.33: Uncurred CIFAR-10 32×32 samples from NCSNv2.

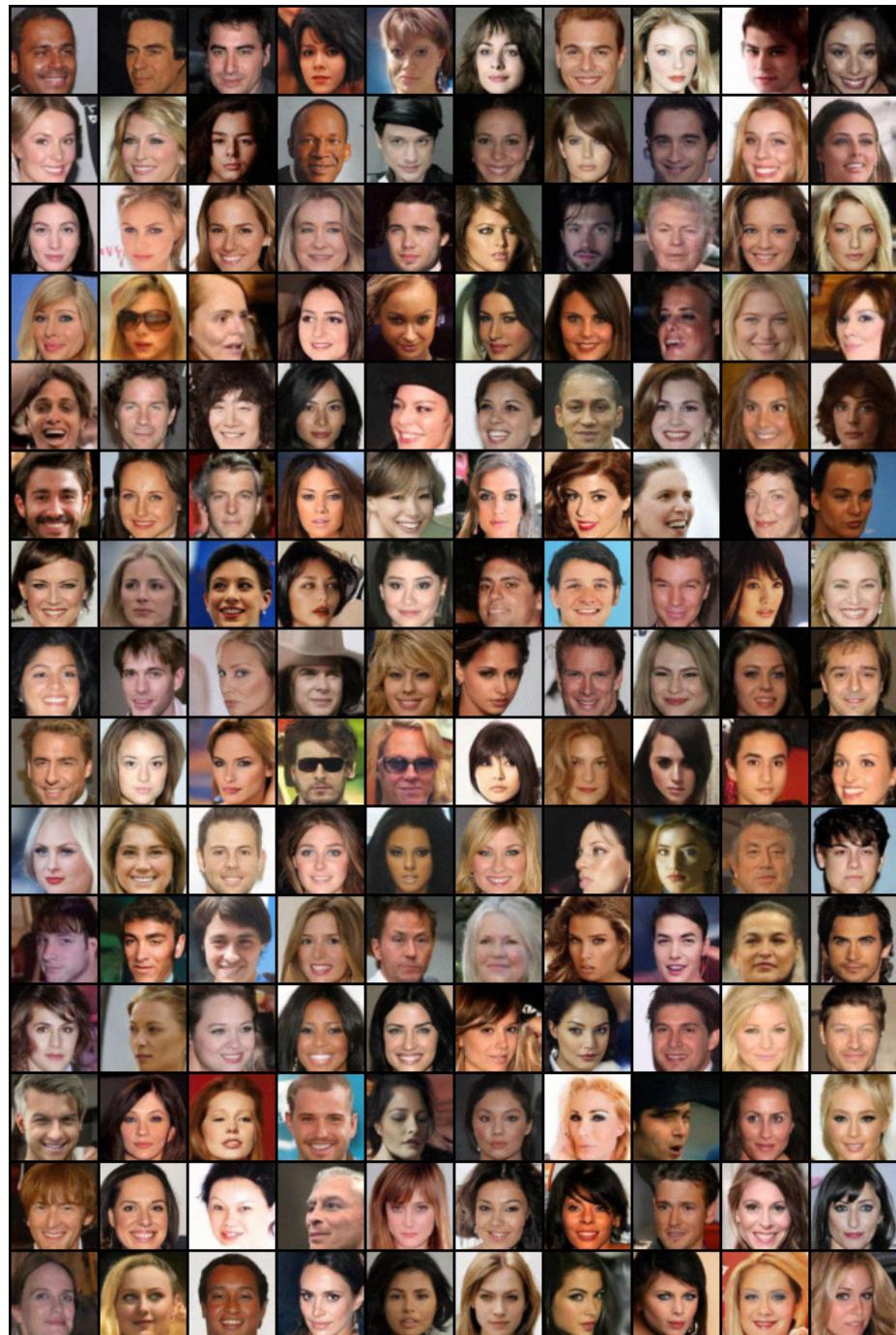
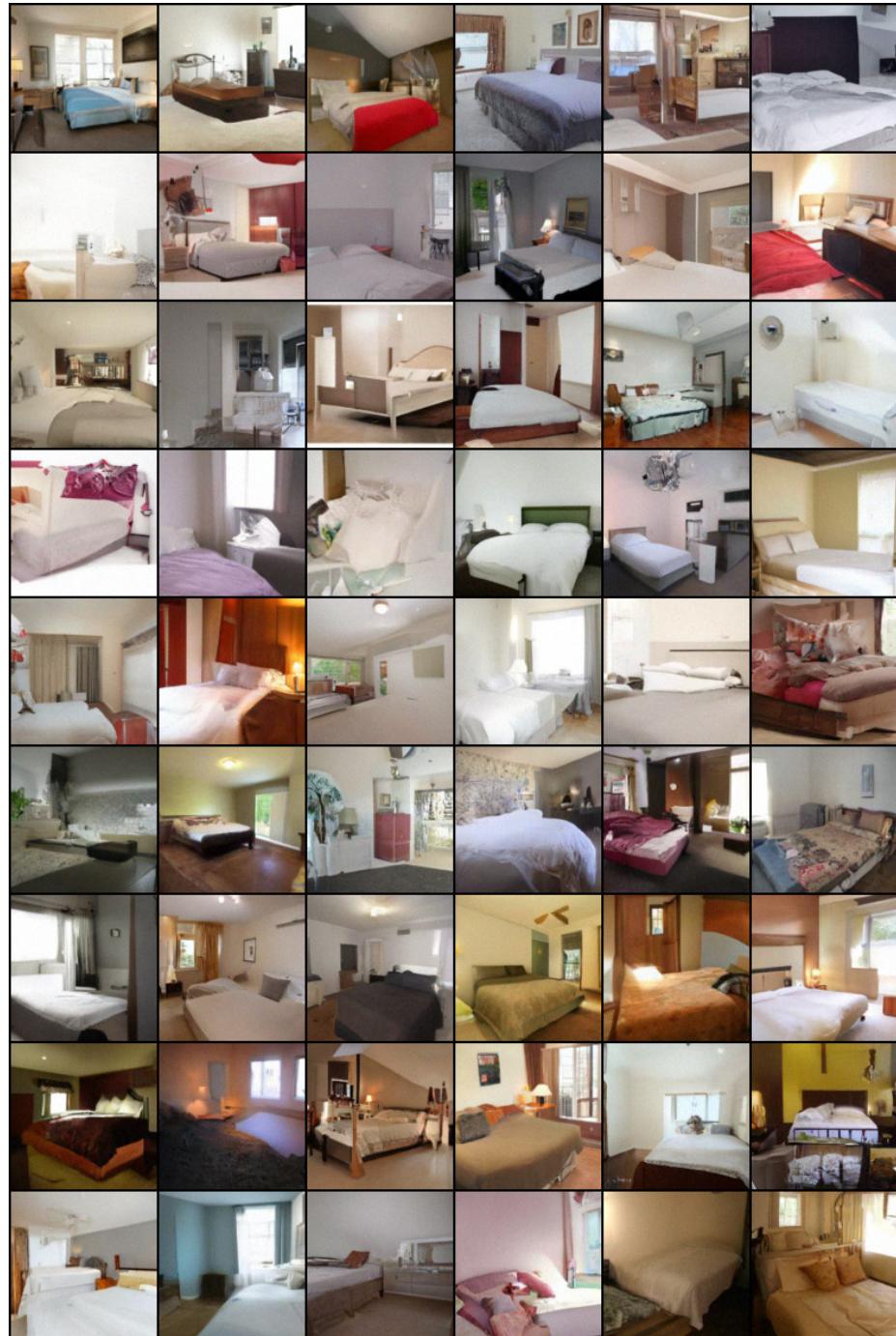


Figure B.34: Uncurated CelebA 64 × 64 samples from NCSNv2.



Figure B.35: Uncurated LSUN church_outdoor 96 × 96 samples from NCSNv2.

Figure B.36: Uncurated LSUN bedroom 128×128 samples from NCSNv2.

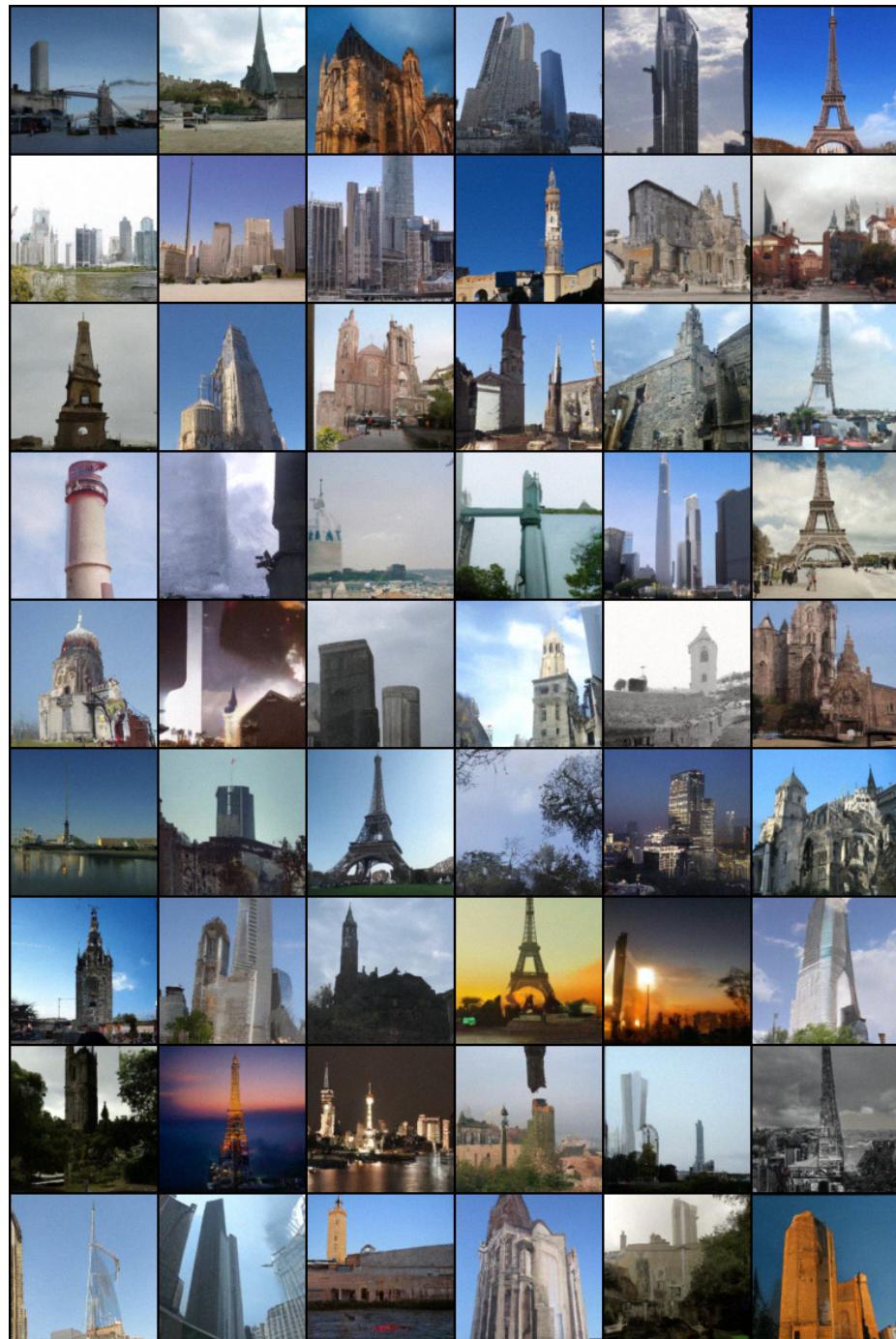


Figure B.37: Uncurated LSUN tower 128×128 samples from NCSNv2.

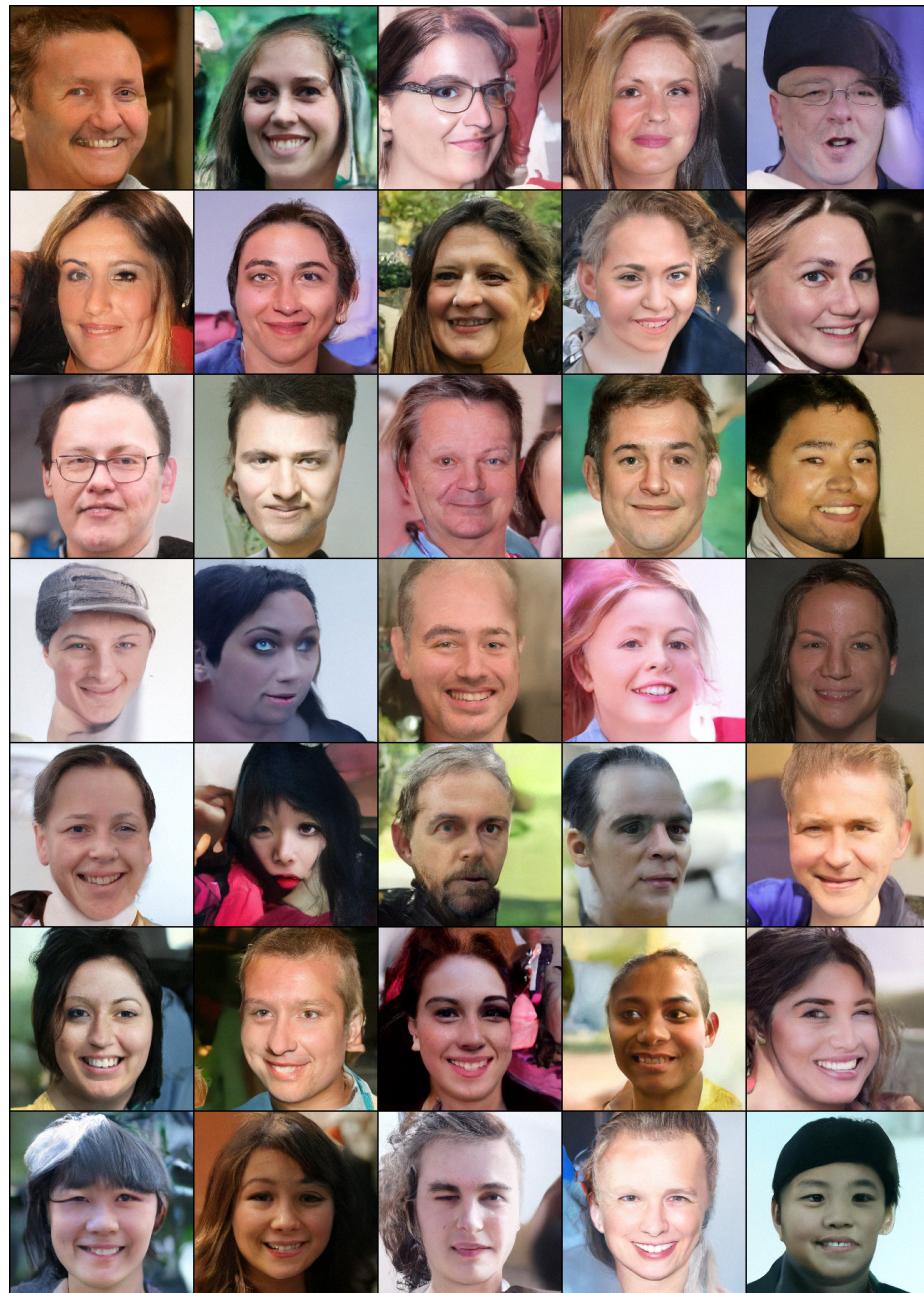


Figure B.38: Uncurated FFHQ 256 × 256 samples from NCSNv2.

B.5 Chapter 7

We include several appendices with additional details, derivations, and results. Our framework allows general SDEs with matrix-valued diffusion coefficients that depend on the state, for which we provide a detailed discussion in Appendix B.5.1. We give a full derivation of VE, VP and sub-VP SDEs in Appendix B.5.2, and discuss how to use them from a practitioner’s perspective in Appendix B.5.3. We elaborate on the probability flow formulation of our framework in Appendix B.5.4, including a derivation of the probability flow ODE (Appendix B.5.4), probability flow sampling with a fixed discretization strategy (Appendix B.5.4), sampling with black-box ODE solvers (Appendix B.5.4), and experimental verification on uniquely identifiable encoding (Appendix B.5.4). We give a full description of the reverse diffusion sampler in Appendix B.5.5, the DDPM-type ancestral sampler for SMLD models in Appendix B.5.6, and Predictor-Corrector samplers in Appendix B.5.7. We explain our model architectures and detailed experimental settings in Appendix B.5.8, with 1024×1024 CelebA-HQ samples therein. Finally, we detail on the algorithms for controllable generation in Appendix B.5.9, and include extended results for class-conditional generation (Appendix B.5.9), image inpainting (Appendix B.5.9), colorization (Appendix B.5.9), and a strategy for solving general inverse problems (Appendix B.5.9).

B.5.1 The Framework for More General SDEs

In the main text, we introduced our framework based on a simplified SDE Eq. (7.5) where the diffusion coefficient is independent of $\mathbf{x}(t)$. It turns out that our framework can be extended to hold for more general diffusion coefficients. We can consider SDEs in the following form:

$$d\mathbf{x} = \mathbf{f}(\mathbf{x}, t) dt + \mathbf{G}(\mathbf{x}, t) d\mathbf{w}, \quad (\text{B.4})$$

where $\mathbf{f}(\cdot, t) : \mathbb{R}^d \rightarrow \mathbb{R}^d$ and $\mathbf{G}(\cdot, t) : \mathbb{R}^d \rightarrow \mathbb{R}^{d \times d}$. We follow the Itô interpretation of SDEs throughout this paper.

According to [And82], the reverse-time SDE is given by (*cf.*, Eq. (7.6))

$$d\mathbf{x} = \{\mathbf{f}(\mathbf{x}, t) - \nabla \cdot [\mathbf{G}(\mathbf{x}, t)\mathbf{G}(\mathbf{x}, t)^\top] - \mathbf{G}(\mathbf{x}, t)\mathbf{G}(\mathbf{x}, t)^\top \nabla_{\mathbf{x}} \log p_t(\mathbf{x})\} dt + \mathbf{G}(\mathbf{x}, t) d\bar{\mathbf{w}}, \quad (\text{B.5})$$

where we define $\nabla \cdot \mathbf{F}(\mathbf{x}) := (\nabla \cdot \mathbf{f}^1(\mathbf{x}), \nabla \cdot \mathbf{f}^2(\mathbf{x}), \dots, \nabla \cdot \mathbf{f}^d(\mathbf{x}))^\top$ for a matrix-valued function

$\mathbf{F}(\mathbf{x}) := (\mathbf{f}^1(\mathbf{x}), \mathbf{f}^2(\mathbf{x}), \dots, \mathbf{f}^d(\mathbf{x}))^\top$ throughout the paper.

The probability flow ODE corresponding to Eq. (B.4) has the following form (*cf.*, Eq. (7.13), see a detailed derivation in Appendix B.5.4):

$$d\mathbf{x} = \left\{ \mathbf{f}(\mathbf{x}, t) - \frac{1}{2} \nabla \cdot [\mathbf{G}(\mathbf{x}, t) \mathbf{G}(\mathbf{x}, t)^\top] - \frac{1}{2} \mathbf{G}(\mathbf{x}, t) \mathbf{G}(\mathbf{x}, t)^\top \nabla_{\mathbf{x}} \log p_t(\mathbf{x}) \right\} dt. \quad (\text{B.6})$$

Finally for conditional generation with the general SDE Eq. (B.4), we can solve the conditional reverse-time SDE below (*cf.*, Eq. (7.14), details in Appendix B.5.9):

$$\begin{aligned} d\mathbf{x} = & \{ \mathbf{f}(\mathbf{x}, t) - \nabla \cdot [\mathbf{G}(\mathbf{x}, t) \mathbf{G}(\mathbf{x}, t)^\top] - \mathbf{G}(\mathbf{x}, t) \mathbf{G}(\mathbf{x}, t)^\top \nabla_{\mathbf{x}} \log p_t(\mathbf{x}) \\ & - \mathbf{G}(\mathbf{x}, t) \mathbf{G}(\mathbf{x}, t)^\top \nabla_{\mathbf{x}} \log p_t(\mathbf{y} | \mathbf{x}) \} dt + \mathbf{G}(\mathbf{x}, t) d\bar{\mathbf{w}}. \end{aligned} \quad (\text{B.7})$$

When the drift and diffusion coefficient of an SDE are not affine, it can be difficult to compute the transition kernel $p_{0t}(\mathbf{x}(t) | \mathbf{x}(0))$ in closed form. This hinders the training of score-based models, because Eq. (7.7) requires knowing $\nabla_{\mathbf{x}(t)} \log p_{0t}(\mathbf{x}(t) | \mathbf{x}(0))$. To overcome this difficulty, we can replace denoising score matching in Eq. (7.7) with other efficient variants of score matching that do not require computing $\nabla_{\mathbf{x}(t)} \log p_{0t}(\mathbf{x}(t) | \mathbf{x}(0))$. For example, when using sliced score matching (*cf.*, Chapter 3), our training objective Eq. (7.7) becomes

$$\boldsymbol{\theta}^* = \arg \min_{\boldsymbol{\theta}} \mathbb{E}_t \left\{ \lambda(t) \mathbb{E}_{\mathbf{x}(0)} \mathbb{E}_{\mathbf{x}(t)} \mathbb{E}_{\mathbf{v} \sim p_{\mathbf{v}}} \left[\frac{1}{2} \| \mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}(t), t) \|_2^2 + \mathbf{v}^\top \mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}(t), t) \mathbf{v} \right] \right\}, \quad (\text{B.8})$$

where $\lambda : [0, T] \rightarrow \mathbb{R}^+$ is a positive weighting function, $t \sim \mathcal{U}(0, T)$, $\mathbb{E}[\mathbf{v}] = \mathbf{0}$, and $\text{Cov}[\mathbf{v}] = \mathbf{I}$. We can always simulate the SDE to sample from $p_{0t}(\mathbf{x}(t) | \mathbf{x}(0))$, and solve Eq. (B.8) to train the time-dependent score-based model $\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}, t)$.

B.5.2 VE, VP and sub-VP SDEs

Below we provide detailed derivations to show that the noise perturbations of SMLD and DDPM are discretizations of the Variance Exploding (VE) and Variance Preserving (VP) SDEs respectively. We additionally introduce sub-VP SDEs, a modification to VP SDEs that often achieves better performance in sample quality.

First, when using a total of N noise scales, each perturbation kernel $p_{\sigma_i}(\mathbf{x} | \mathbf{x}_0)$ of SMLD

can be derived from the following Markov chain:

$$\mathbf{x}_i = \mathbf{x}_{i-1} + \sqrt{\sigma_i^2 - \sigma_{i-1}^2} \mathbf{z}_{i-1}, \quad i = 1, \dots, N, \quad (\text{B.9})$$

where $\mathbf{z}_{i-1} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, $\mathbf{x}_0 \sim p_{\text{data}}$, and we have introduced $\sigma_0 = 0$ to simplify the notation. In the limit of $N \rightarrow \infty$, the Markov chain $\{\mathbf{x}_i\}_{i=1}^N$ becomes a continuous stochastic process $\{\mathbf{x}(t)\}_{t=0}^1$, $\{\sigma_i\}_{i=1}^N$ becomes a function $\sigma(t)$, and \mathbf{z}_i becomes $\mathbf{z}(t)$, where we have used a continuous time variable $t \in [0, 1]$ for indexing, rather than an integer $i \in \{1, 2, \dots, N\}$. Let $\mathbf{x}\left(\frac{i}{N}\right) = \mathbf{x}_i$, $\sigma\left(\frac{i}{N}\right) = \sigma_i$, and $\mathbf{z}\left(\frac{i}{N}\right) = \mathbf{z}_i$ for $i = 1, 2, \dots, N$. We can rewrite Eq. (B.9) as follows with $\Delta t = \frac{1}{N}$ and $t \in \{0, \frac{1}{N}, \dots, \frac{N-1}{N}\}$:

$$\mathbf{x}(t + \Delta t) = \mathbf{x}(t) + \sqrt{\sigma^2(t + \Delta t) - \sigma^2(t)} \mathbf{z}(t) \approx \mathbf{x}(t) + \sqrt{\frac{d[\sigma^2(t)]}{dt} \Delta t} \mathbf{z}(t),$$

where the approximate equality holds when $\Delta t \ll 1$. In the limit of $\Delta t \rightarrow 0$, this converges to

$$d\mathbf{x} = \sqrt{\frac{d[\sigma^2(t)]}{dt}} dw, \quad (\text{B.10})$$

which is the VE SDE.

For the perturbation kernels $\{p_{\alpha_i}(\mathbf{x} | \mathbf{x}_0)\}_{i=1}^N$ used in DDPM, the discrete Markov chain is

$$\mathbf{x}_i = \sqrt{1 - \beta_i} \mathbf{x}_{i-1} + \sqrt{\beta_i} \mathbf{z}_{i-1}, \quad i = 1, \dots, N, \quad (\text{B.11})$$

where $\mathbf{z}_{i-1} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. To obtain the limit of this Markov chain when $N \rightarrow \infty$, we define an auxiliary set of noise scales $\{\bar{\beta}_i = N\beta_i\}_{i=1}^N$, and re-write Eq. (B.11) as below

$$\mathbf{x}_i = \sqrt{1 - \frac{\bar{\beta}_i}{N}} \mathbf{x}_{i-1} + \sqrt{\frac{\bar{\beta}_i}{N}} \mathbf{z}_{i-1}, \quad i = 1, \dots, N. \quad (\text{B.12})$$

In the limit of $N \rightarrow \infty$, $\{\bar{\beta}_i\}_{i=1}^N$ becomes a function $\beta(t)$ indexed by $t \in [0, 1]$. Let $\beta\left(\frac{i}{N}\right) = \bar{\beta}_i$, $\mathbf{x}\left(\frac{i}{N}\right) = \mathbf{x}_i$, $\mathbf{z}\left(\frac{i}{N}\right) = \mathbf{z}_i$. We can rewrite the Markov chain Eq. (B.12) as the following with $\Delta t = \frac{1}{N}$ and $t \in \{0, 1, \dots, \frac{N-1}{N}\}$:

$$\mathbf{x}(t + \Delta t) = \sqrt{1 - \beta(t + \Delta t) \Delta t} \mathbf{x}(t) + \sqrt{\beta(t + \Delta t) \Delta t} \mathbf{z}(t)$$

$$\begin{aligned} &\approx \mathbf{x}(t) - \frac{1}{2}\beta(t + \Delta t)\Delta t \mathbf{x}(t) + \sqrt{\beta(t + \Delta t)\Delta t} \mathbf{z}(t) \\ &\approx \mathbf{x}(t) - \frac{1}{2}\beta(t)\Delta t \mathbf{x}(t) + \sqrt{\beta(t)\Delta t} \mathbf{z}(t), \end{aligned} \quad (\text{B.13})$$

where the approximate equality holds when $\Delta t \ll 1$. Therefore, in the limit of $\Delta t \rightarrow 0$, Eq. (B.13) converges to the following VP SDE:

$$d\mathbf{x} = -\frac{1}{2}\beta(t)\mathbf{x} dt + \sqrt{\beta(t)} dw. \quad (\text{B.14})$$

So far, we have demonstrated that the noise perturbations used in SMLD and DDPM correspond to discretizations of VE and VP SDEs respectively. The VE SDE always yields a process with exploding variance when $t \rightarrow \infty$. In contrast, the VP SDE yields a process with bounded variance. In addition, the process has a constant unit variance for all $t \in [0, \infty)$ when $p(\mathbf{x}(0))$ has a unit variance. Since the VP SDE has affine drift and diffusion coefficients, we can use Eq. (5.51) in [SS19] to obtain an ODE that governs the evolution of variance

$$\frac{d\Sigma_{\text{VP}}(t)}{dt} = \beta(t)(\mathbf{I} - \Sigma_{\text{VP}}(t)),$$

where $\Sigma_{\text{VP}}(t) := \text{Cov}[\mathbf{x}(t)]$ for $\{\mathbf{x}(t)\}_{t=0}^1$ obeying a VP SDE. Solving this ODE, we obtain

$$\Sigma_{\text{VP}}(t) = \mathbf{I} + e^{\int_0^t -\beta(s) ds}(\Sigma_{\text{VP}}(0) - \mathbf{I}), \quad (\text{B.15})$$

from which it is clear that the variance $\Sigma_{\text{VP}}(t)$ is always bounded given $\Sigma_{\text{VP}}(0)$. Moreover, $\Sigma_{\text{VP}}(t) = \mathbf{I}$ if $\Sigma_{\text{VP}}(0) = \mathbf{I}$. Due to this difference, we name Eq. (7.9) as the *Variance Exploding (VE) SDE*, and Eq. (7.11) the *Variance Preserving (VP) SDE*.

Inspired by the VP SDE, we propose a new SDE called the *sub-VP SDE*, namely

$$d\mathbf{x} = -\frac{1}{2}\beta(t)\mathbf{x} dt + \sqrt{\beta(t)(1 - e^{-2\int_0^t \beta(s) ds})} dw. \quad (\text{B.16})$$

Following standard derivations, it is straightforward to show that $\mathbb{E}[\mathbf{x}(t)]$ is the same for both VP and sub-VP SDEs; the variance function of sub-VP SDEs is different, given by

$$\Sigma_{\text{sub-VP}}(t) = \mathbf{I} + e^{-2\int_0^t \beta(s) ds} \mathbf{I} + e^{-\int_0^t \beta(s) ds}(\Sigma_{\text{sub-VP}}(0) - 2\mathbf{I}), \quad (\text{B.17})$$

where $\Sigma_{\text{sub-VP}}(t) := \text{Cov}[\mathbf{x}(t)]$ for a process $\{\mathbf{x}(t)\}_{t=0}^1$ obtained by solving Eq. (B.16). In

addition, we observe that (i) $\Sigma_{\text{sub-VP}}(t) \leq \Sigma_{\text{VP}}(t)$ for all $t \geq 0$ with $\Sigma_{\text{sub-VP}}(0) = \Sigma_{\text{VP}}(0)$ and shared $\beta(s)$; and (ii) $\lim_{t \rightarrow \infty} \Sigma_{\text{sub-VP}}(t) = \lim_{t \rightarrow \infty} \Sigma_{\text{VP}}(t) = \mathbf{I}$ if $\lim_{t \rightarrow \infty} \int_0^t \beta(s) ds = \infty$. The former is why we name Eq. (B.16) the sub-VP SDE—its variance is always upper bounded by the corresponding VP SDE. The latter justifies the use of sub-VP SDEs for score-based generative modeling, since they can perturb any data distribution to standard Gaussian under suitable conditions, just like VP SDEs.

VE, VP and sub-VP SDEs all have affine drift coefficients. Therefore, their perturbation kernels $p_{0t}(\mathbf{x}(t) | \mathbf{x}(0))$ are all Gaussian and can be computed with Eqs. (5.50) and (5.51) in [SS19]:

$$p_{0t}(\mathbf{x}(t) | \mathbf{x}(0)) = \begin{cases} \mathcal{N}(\mathbf{x}(t); \mathbf{x}(0), [\sigma^2(t) - \sigma^2(0)]\mathbf{I}), & (\text{VE SDE}) \\ \mathcal{N}(\mathbf{x}(t); \mathbf{x}(0)e^{-\frac{1}{2}\int_0^t \beta(s) ds}, \mathbf{I} - \mathbf{I}e^{-\int_0^t \beta(s) ds}) & (\text{VP SDE}) \\ \mathcal{N}(\mathbf{x}(t); \mathbf{x}(0)e^{-\frac{1}{2}\int_0^t \beta(s) ds}, [1 - e^{-\int_0^t \beta(s) ds}]^2\mathbf{I}) & (\text{sub-VP SDE}) \end{cases}. \quad (\text{B.18})$$

As a result, all SDEs introduced here can be efficiently trained with the objective in Eq. (7.7).

B.5.3 SDEs in the Wild

Below we discuss concrete instantiations of VE and VP SDEs whose discretizations yield SMLD and DDPM models, and the specific sub-VP SDE used in our experiments. In SMLD, the noise scales $\{\sigma_i\}_{i=1}^N$ is typically a geometric sequence where σ_{\min} is fixed to 0.01 and σ_{\max} is chosen according to Technique 1 in [SE20]. Usually, SMLD models normalize image inputs to the range $[0, 1]$. Since $\{\sigma_i\}_{i=1}^N$ is a geometric sequence, we have $\sigma(\frac{i}{N}) = \sigma_i = \sigma_{\min} \left(\frac{\sigma_{\max}}{\sigma_{\min}} \right)^{\frac{i-1}{N-1}}$ for $i = 1, 2, \dots, N$. In the limit of $N \rightarrow \infty$, we have $\sigma(t) = \sigma_{\min} \left(\frac{\sigma_{\max}}{\sigma_{\min}} \right)^t$ for $t \in (0, 1]$. The corresponding VE SDE is

$$d\mathbf{x} = \sigma_{\min} \left(\frac{\sigma_{\max}}{\sigma_{\min}} \right)^t \sqrt{2 \log \frac{\sigma_{\max}}{\sigma_{\min}}} d\mathbf{w}, \quad t \in (0, 1], \quad (\text{B.19})$$

and the perturbation kernel can be derived via Eq. (B.18):

$$p_{0t}(\mathbf{x}(t) | \mathbf{x}(0)) = \mathcal{N} \left(\mathbf{x}(t); \mathbf{x}(0), \sigma_{\min}^2 \left(\frac{\sigma_{\max}}{\sigma_{\min}} \right)^{2t} \mathbf{I} \right), \quad t \in (0, 1]. \quad (\text{B.20})$$

There is one subtlety when $t = 0$: by definition, $\sigma(0) = \sigma_0 = 0$ (following the convention in Eq. (B.9)), but $\sigma(0^+) := \lim_{t \rightarrow 0^+} \sigma(t) = \sigma_{\min} \neq 0$. In other words, $\sigma(t)$ for SMLD is not differentiable since $\sigma(0) \neq \sigma(0^+)$, causing the VE SDE in Eq. (B.10) undefined for $t = 0$. In practice, we bypass this issue by always solving the SDE and its associated probability flow ODE in the range $t \in [\epsilon, 1]$ for some small constant $\epsilon > 0$, and we use $\epsilon = 10^{-5}$ in our VE SDE experiments.

For DDPM models, $\{\beta_i\}_{i=1}^N$ is typically an arithmetic sequence where $\beta_i = \frac{\bar{\beta}_{\min}}{N} + \frac{i-1}{N(N-1)}(\bar{\beta}_{\max} - \bar{\beta}_{\min})$ for $i = 1, 2, \dots, N$. Therefore, $\beta(t) = \bar{\beta}_{\min} + t(\bar{\beta}_{\max} - \bar{\beta}_{\min})$ for $t \in [0, 1]$ in the limit of $N \rightarrow \infty$. This corresponds to the following instantiation of the VP SDE:

$$d\mathbf{x} = -\frac{1}{2}(\bar{\beta}_{\min} + t(\bar{\beta}_{\max} - \bar{\beta}_{\min}))\mathbf{x} dt + \sqrt{\bar{\beta}_{\min} + t(\bar{\beta}_{\max} - \bar{\beta}_{\min})} d\mathbf{w}, \quad t \in [0, 1], \quad (\text{B.21})$$

where $\mathbf{x}(0) \sim p_{\text{data}}(\mathbf{x})$. In our experiments, we let $\bar{\beta}_{\min} = 0.1$ and $\bar{\beta}_{\max} = 20$ to match the settings in [HJA20]. The perturbation kernel is given by

$$\begin{aligned} p_{0t}(\mathbf{x}(t) | \mathbf{x}(0)) \\ = \mathcal{N}\left(\mathbf{x}(t); e^{-\frac{1}{4}t^2(\bar{\beta}_{\max}-\bar{\beta}_{\min})-\frac{1}{2}t\bar{\beta}_{\min}} \mathbf{x}(0), \mathbf{I} - \mathbf{I} e^{-\frac{1}{2}t^2(\bar{\beta}_{\max}-\bar{\beta}_{\min})-t\bar{\beta}_{\min}}\right), \quad t \in [0, 1]. \end{aligned} \quad (\text{B.22})$$

For DDPM, there is no discontinuity issue with the corresponding VP SDE; yet, there are numerical instability issues for training and sampling at $t = 0$, due to the vanishing variance of $\mathbf{x}(t)$ as $t \rightarrow 0$. Therefore, same as the VE SDE, we restrict computation to $t \in [\epsilon, 1]$ for a small $\epsilon > 0$. For sampling, we choose $\epsilon = 10^{-3}$ so that the variance of $\mathbf{x}(\epsilon)$ in VP SDE matches the variance of \mathbf{x}_1 in DDPM; for training, we adopt $\epsilon = 10^{-5}$ which empirically gives better results.

As a sanity check for our SDE generalizations to SMLD and DDPM, we compare the perturbation kernels of SDEs and original discrete Markov chains in Fig. B.39. The SMLD and DDPM models both use $N = 1000$ noise scales. For SMLD, we only need to compare the variances of perturbation kernels since means are the same by definition. For DDPM, we compare the scaling factors of means and the variances. As demonstrated in Fig. B.39, the discrete perturbation kernels of original SMLD and DDPM models align well with perturbation kernels derived from VE and VP SDEs.

For sub-VP SDEs, we use exactly the same $\beta(t)$ as VP SDEs. This leads to the following

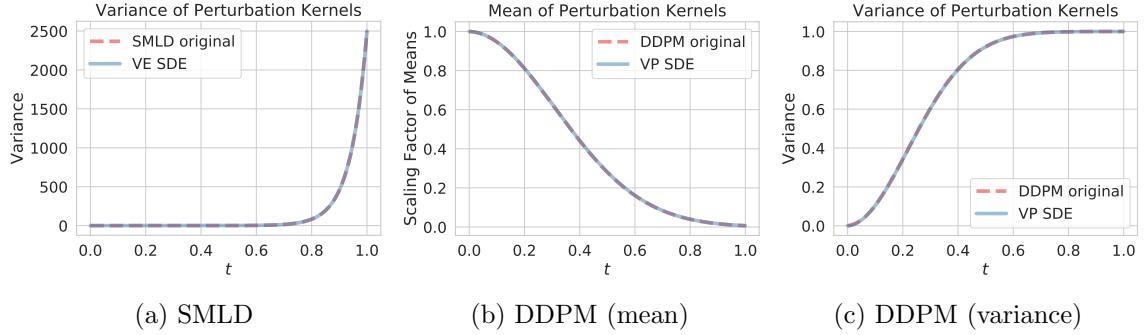


Figure B.39: Discrete-time perturbation kernels and our continuous generalizations match each other almost exactly. (a) compares the variance of perturbation kernels for SMLD and VE SDE; (b) compares the scaling factors of means of perturbation kernels for DDPM and VP SDE; and (c) compares the variance of perturbation kernels for DDPM and VP SDE.

perturbation kernel

$$\begin{aligned} p_{0t}(\mathbf{x}(t) \mid \mathbf{x}(0)) \\ = \mathcal{N}\left(\mathbf{x}(t); e^{-\frac{1}{4}t^2(\bar{\beta}_{\max}-\bar{\beta}_{\min})-\frac{1}{2}t\bar{\beta}_{\min}}\mathbf{x}(0), [1 - e^{-\frac{1}{2}t^2(\bar{\beta}_{\max}-\bar{\beta}_{\min})-t\bar{\beta}_{\min}}]^2 \mathbf{I}\right), \quad t \in [0, 1]. \end{aligned} \quad (\text{B.23})$$

We also restrict numerical computation to the same interval of $[\epsilon, 1]$ as VP SDEs.

For sampling, it is important to use an appropriate ϵ for better Inception scores and FIDs, although samples across different ϵ look visually the same to human eyes.

B.5.4 Probability Flow ODE

Derivation

The idea of probability flow ODE is inspired by [MRO20], and one can find the derivation of a simplified case therein. Below we provide a derivation for the fully general ODE in Eq. (B.6). We consider the SDE in Eq. (B.4), which possesses the following form:

$$d\mathbf{x} = \mathbf{f}(\mathbf{x}, t) dt + \mathbf{G}(\mathbf{x}, t) d\mathbf{w},$$

where $\mathbf{f}(\cdot, t) : \mathbb{R}^d \rightarrow \mathbb{R}^d$ and $\mathbf{G}(\cdot, t) : \mathbb{R}^d \rightarrow \mathbb{R}^{d \times d}$. The marginal probability density $p_t(\mathbf{x}(t))$ evolves according to Kolmogorov's forward equation (Fokker-Planck equation) [Oks13]

$$\frac{\partial p_t(\mathbf{x})}{\partial t} = - \sum_{i=1}^d \frac{\partial}{\partial x_i} [f_i(\mathbf{x}, t)p_t(\mathbf{x})] + \frac{1}{2} \sum_{i=1}^d \sum_{j=1}^d \frac{\partial^2}{\partial x_i \partial x_j} \left[\sum_{k=1}^d G_{ik}(\mathbf{x}, t)G_{jk}(\mathbf{x}, t)p_t(\mathbf{x}) \right]. \quad (\text{B.24})$$

We can easily rewrite Eq. (B.24) to obtain

$$\begin{aligned} \frac{\partial p_t(\mathbf{x})}{\partial t} &= - \sum_{i=1}^d \frac{\partial}{\partial x_i} [f_i(\mathbf{x}, t)p_t(\mathbf{x})] + \frac{1}{2} \sum_{i=1}^d \sum_{j=1}^d \frac{\partial^2}{\partial x_i \partial x_j} \left[\sum_{k=1}^d G_{ik}(\mathbf{x}, t)G_{jk}(\mathbf{x}, t)p_t(\mathbf{x}) \right] \\ &= - \sum_{i=1}^d \frac{\partial}{\partial x_i} [f_i(\mathbf{x}, t)p_t(\mathbf{x})] + \frac{1}{2} \sum_{i=1}^d \frac{\partial}{\partial x_i} \left[\sum_{j=1}^d \frac{\partial}{\partial x_j} \left[\sum_{k=1}^d G_{ik}(\mathbf{x}, t)G_{jk}(\mathbf{x}, t)p_t(\mathbf{x}) \right] \right]. \end{aligned} \quad (\text{B.25})$$

Note that

$$\begin{aligned} &\sum_{j=1}^d \frac{\partial}{\partial x_j} \left[\sum_{k=1}^d G_{ik}(\mathbf{x}, t)G_{jk}(\mathbf{x}, t)p_t(\mathbf{x}) \right] \\ &= \sum_{j=1}^d \frac{\partial}{\partial x_j} \left[\sum_{k=1}^d G_{ik}(\mathbf{x}, t)G_{jk}(\mathbf{x}, t) \right] p_t(\mathbf{x}) + \sum_{j=1}^d \sum_{k=1}^d G_{ik}(\mathbf{x}, t)G_{jk}(\mathbf{x}, t)p_t(\mathbf{x}) \frac{\partial}{\partial x_j} \log p_t(\mathbf{x}) \\ &= p_t(\mathbf{x}) \nabla \cdot [\mathbf{G}(\mathbf{x}, t) \mathbf{G}(\mathbf{x}, t)^\top] + p_t(\mathbf{x}) \mathbf{G}(\mathbf{x}, t) \mathbf{G}(\mathbf{x}, t)^\top \nabla_{\mathbf{x}} \log p_t(\mathbf{x}), \end{aligned}$$

based on which we can continue the rewriting of Eq. (B.25) to obtain

$$\begin{aligned} \frac{\partial p_t(\mathbf{x})}{\partial t} &= - \sum_{i=1}^d \frac{\partial}{\partial x_i} [f_i(\mathbf{x}, t)p_t(\mathbf{x})] + \frac{1}{2} \sum_{i=1}^d \frac{\partial}{\partial x_i} \left[\sum_{j=1}^d \frac{\partial}{\partial x_j} \left[\sum_{k=1}^d G_{ik}(\mathbf{x}, t)G_{jk}(\mathbf{x}, t)p_t(\mathbf{x}) \right] \right] \\ &= - \sum_{i=1}^d \frac{\partial}{\partial x_i} [f_i(\mathbf{x}, t)p_t(\mathbf{x})] \\ &\quad + \frac{1}{2} \sum_{i=1}^d \frac{\partial}{\partial x_i} \left[p_t(\mathbf{x}) \nabla \cdot [\mathbf{G}(\mathbf{x}, t) \mathbf{G}(\mathbf{x}, t)^\top] + p_t(\mathbf{x}) \mathbf{G}(\mathbf{x}, t) \mathbf{G}(\mathbf{x}, t)^\top \nabla_{\mathbf{x}} \log p_t(\mathbf{x}) \right] \\ &= - \sum_{i=1}^d \frac{\partial}{\partial x_i} \left\{ f_i(\mathbf{x}, t)p_t(\mathbf{x}) \right. \\ &\quad \left. - \frac{1}{2} \left[\nabla \cdot [\mathbf{G}(\mathbf{x}, t) \mathbf{G}(\mathbf{x}, t)^\top] + \mathbf{G}(\mathbf{x}, t) \mathbf{G}(\mathbf{x}, t)^\top \nabla_{\mathbf{x}} \log p_t(\mathbf{x}) \right] p_t(\mathbf{x}) \right\} \end{aligned}$$

$$= - \sum_{i=1}^d \frac{\partial}{\partial x_i} [\tilde{f}_i(\mathbf{x}, t) p_t(\mathbf{x})], \quad (\text{B.26})$$

where we define

$$\tilde{\mathbf{f}}(\mathbf{x}, t) := \mathbf{f}(\mathbf{x}, t) - \frac{1}{2} \nabla \cdot [\mathbf{G}(\mathbf{x}, t) \mathbf{G}(\mathbf{x}, t)^\top] - \frac{1}{2} \mathbf{G}(\mathbf{x}, t) \mathbf{G}(\mathbf{x}, t)^\top \nabla_{\mathbf{x}} \log p_t(\mathbf{x}).$$

Inspecting Eq. (B.26), we observe that it equals Kolmogorov's forward equation of the following SDE with $\tilde{\mathbf{G}}(\mathbf{x}, t) := \mathbf{0}$ (Kolmogorov's forward equation in this case is also known as the Liouville equation.)

$$d\mathbf{x} = \tilde{\mathbf{f}}(\mathbf{x}, t) dt + \tilde{\mathbf{G}}(\mathbf{x}, t) d\mathbf{w},$$

which is essentially an ODE:

$$d\mathbf{x} = \tilde{\mathbf{f}}(\mathbf{x}, t) dt,$$

same as the probability flow ODE given by Eq. (B.6). Therefore, we have shown that the probability flow ODE Eq. (B.6) induces the same marginal probability density $p_t(\mathbf{x})$ as the SDE in Eq. (B.4).

Probability Flow Sampling

Suppose we have a forward SDE

$$d\mathbf{x} = \mathbf{f}(\mathbf{x}, t) dt + \mathbf{G}(t) d\mathbf{w},$$

and one of its discretization

$$\mathbf{x}_{i+1} = \mathbf{x}_i + \mathbf{f}_i(\mathbf{x}_i) + \mathbf{G}_i \mathbf{z}_i, \quad i = 0, 1, \dots, N-1, \quad (\text{B.27})$$

where $\mathbf{z}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. We assume the discretization schedule of time is fixed beforehand, and thus we absorb the dependency on Δt into the notations of \mathbf{f}_i and \mathbf{G}_i . Using Eq. (B.6), we can obtain the following probability flow ODE:

$$d\mathbf{x} = \left\{ \mathbf{f}(\mathbf{x}, t) - \frac{1}{2} \mathbf{G}(t) \mathbf{G}(t)^\top \nabla_{\mathbf{x}} \log p_t(\mathbf{x}) \right\} dt. \quad (\text{B.28})$$

We may employ any numerical method to integrate the probability flow ODE backwards in time for sample generation. In particular, we propose a discretization in a similar functional form to Eq. (B.27):

$$\mathbf{x}_i = \mathbf{x}_{i+1} - \mathbf{f}_{i+1}(\mathbf{x}_{i+1}) + \frac{1}{2} \mathbf{G}_{i+1} \mathbf{G}_{i+1}^T \mathbf{s}_{\theta^*}(\mathbf{x}_{i+1}, i+1), \quad i = 0, 1, \dots, N-1,$$

where the score-based model $\mathbf{s}_{\theta^*}(\mathbf{x}_i, i)$ is conditioned on the iteration number i . This is a deterministic iteration rule. Unlike reverse diffusion samplers or ancestral sampling, there is no additional randomness once the initial sample \mathbf{x}_N is obtained from the prior distribution. When applied to SMLD models, we can get the following iteration rule for probability flow sampling:

$$\mathbf{x}_i = \mathbf{x}_{i+1} + \frac{1}{2} (\sigma_{i+1}^2 - \sigma_i^2) \mathbf{s}_{\theta^*}(\mathbf{x}_{i+1}, \sigma_{i+1}), \quad i = 0, 1, \dots, N-1. \quad (\text{B.29})$$

Similarly, for DDPM models, we have

$$\mathbf{x}_i = (2 - \sqrt{1 - \beta_{i+1}}) \mathbf{x}_{i+1} + \frac{1}{2} \beta_{i+1} \mathbf{s}_{\theta^*}(\mathbf{x}_{i+1}, i+1), \quad i = 0, 1, \dots, N-1. \quad (\text{B.30})$$

Sampling with Black-Box ODE Solvers

For producing figures in Fig. 7.3, we use a DDPM model trained on 256×256 CelebA-HQ with the same settings in [HJA20].

Aside from the interpolation results in Fig. 7.3, we demonstrate more examples of latent space manipulation in Fig. B.40, including interpolation and temperature scaling. The model tested here is a DDPM model trained with the same settings in [HJA20].

Although solvers for the probability flow ODE allow fast sampling, their samples typically have higher (worse) FID scores than those from SDE solvers if no corrector is used. We have this empirical observation for both the discretization strategy in Appendix B.5.4, and black-box ODE solvers introduced above. Moreover, the performance of probability flow ODE samplers depends on the choice of the SDE—their sample quality for VE SDEs is much worse than VP SDEs especially for high-dimensional data.

Uniquely Identifiable Encoding

As a sanity check, we train two models (denoted as “Model A” and “Model B”) with different architectures using the VE SDE on CIFAR-10. Here Model A is an NCSN++ model with 4



Figure B.40: Samples from the probability flow ODE for VP SDE on 256×256 CelebA-HQ. Top: spherical interpolations between random samples. Bottom: temperature rescaling (reducing norm of embedding).

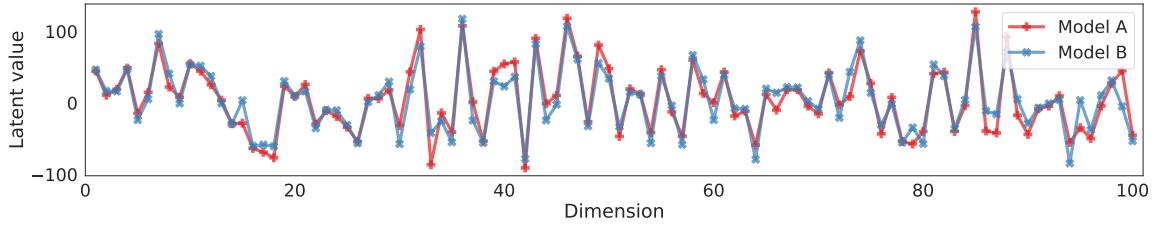


Figure B.41: Comparing the first 100 dimensions of the latent code obtained for a random CIFAR-10 image. “Model A” and “Model B” are separately trained with different architectures.

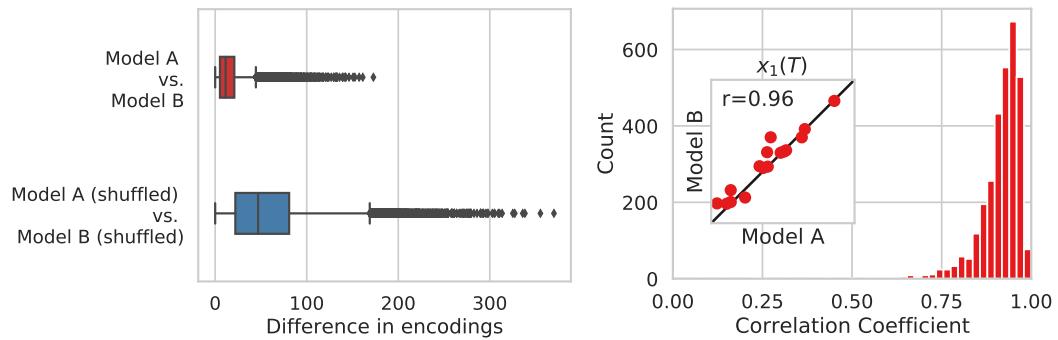


Figure B.42: *Left:* The dimension-wise difference between encodings obtained by Model A and B. As a baseline, we also report the difference between shuffled representations of these two models. *Right:* The dimension-wise correlation coefficients of encodings obtained by Model A and Model B.

layers per resolution trained using the continuous objective in Eq. (7.7), and Model B is all the same except that it uses 8 layers per resolution. Model definitions are in Appendix B.5.8.

We report the latent codes obtained by Model A and Model B for a random CIFAR-10 image in Fig. B.41. In Fig. B.42, we show the dimension-wise differences and correlation coefficients between latent encodings on a total of 16 CIFAR-10 images. Our results demonstrate that for the same inputs, Model A and Model B provide encodings that are close in every dimension, despite having different model architectures and training runs.

B.5.5 Reverse Diffusion Sampling

Given a forward SDE

$$d\mathbf{x} = \mathbf{f}(\mathbf{x}, t) dt + \mathbf{G}(t) d\mathbf{w},$$

and suppose the following iteration rule is a discretization of it:

$$\mathbf{x}_{i+1} = \mathbf{x}_i + \mathbf{f}_i(\mathbf{x}_i) + \mathbf{G}_i \mathbf{z}_i, \quad i = 0, 1, \dots, N-1 \quad (\text{B.31})$$

where $\mathbf{z}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. Here we assume the discretization schedule of time is fixed beforehand, and thus we can absorb it into the notations of \mathbf{f}_i and \mathbf{G}_i .

Based on Eq. (B.31), we propose to discretize the reverse-time SDE

$$d\mathbf{x} = [\mathbf{f}(\mathbf{x}, t) - \mathbf{G}(t) \mathbf{G}(t)^\top \nabla_{\mathbf{x}} \log p_t(\mathbf{x})] dt + \mathbf{G}(t) d\bar{\mathbf{w}},$$

with a similar functional form, which gives the following iteration rule for $i \in \{0, 1, \dots, N-1\}$:

$$\mathbf{x}_i = \mathbf{x}_{i+1} - \mathbf{f}_{i+1}(\mathbf{x}_{i+1}) + \mathbf{G}_{i+1} \mathbf{G}_{i+1}^\top s_{\theta^*}(\mathbf{x}_{i+1}, i+1) + \mathbf{G}_{i+1} \mathbf{z}_{i+1}, \quad (\text{B.32})$$

where our trained score-based model $s_{\theta^*}(\mathbf{x}_i, i)$ is conditioned on iteration number i .

When applying Eq. (B.32) to Eqs. (7.10) and (B.9), we obtain a new set of numerical solvers for the reverse-time VE and VP SDEs, resulting in sampling algorithms as shown in the “predictor” part of Algorithms B.4 and B.5. We name these sampling methods (that are based on the discretization strategy in Eq. (B.32)) *reverse diffusion samplers*.

As expected, the ancestral sampling of DDPM [HJA20] (Eq. (7.4)) matches its reverse diffusion counterpart when $\beta_i \rightarrow 0$ for all i (which happens when $\Delta t \rightarrow 0$ since $\beta_i = \bar{\beta}_i \Delta t$,

see Appendix B.5.2), because

$$\begin{aligned}
\mathbf{x}_i &= \frac{1}{\sqrt{1 - \beta_{i+1}}} (\mathbf{x}_{i+1} + \beta_{i+1} \mathbf{s}_{\theta^*}(\mathbf{x}_{i+1}, i+1)) + \sqrt{\beta_{i+1}} \mathbf{z}_{i+1} \\
&= \left(1 + \frac{1}{2}\beta_{i+1} + o(\beta_{i+1})\right) (\mathbf{x}_{i+1} + \beta_{i+1} \mathbf{s}_{\theta^*}(\mathbf{x}_{i+1}, i+1)) + \sqrt{\beta_{i+1}} \mathbf{z}_{i+1} \\
&\approx \left(1 + \frac{1}{2}\beta_{i+1}\right) (\mathbf{x}_{i+1} + \beta_{i+1} \mathbf{s}_{\theta^*}(\mathbf{x}_{i+1}, i+1)) + \sqrt{\beta_{i+1}} \mathbf{z}_{i+1} \\
&= \left(1 + \frac{1}{2}\beta_{i+1}\right) \mathbf{x}_{i+1} + \beta_{i+1} \mathbf{s}_{\theta^*}(\mathbf{x}_{i+1}, i+1) + \frac{1}{2}\beta_{i+1}^2 \mathbf{s}_{\theta^*}(\mathbf{x}_{i+1}, i+1) + \sqrt{\beta_{i+1}} \mathbf{z}_{i+1} \\
&\approx \left(1 + \frac{1}{2}\beta_{i+1}\right) \mathbf{x}_{i+1} + \beta_{i+1} \mathbf{s}_{\theta^*}(\mathbf{x}_{i+1}, i+1) + \sqrt{\beta_{i+1}} \mathbf{z}_{i+1} \\
&= \left[2 - \left(1 - \frac{1}{2}\beta_{i+1}\right)\right] \mathbf{x}_{i+1} + \beta_{i+1} \mathbf{s}_{\theta^*}(\mathbf{x}_{i+1}, i+1) + \sqrt{\beta_{i+1}} \mathbf{z}_{i+1} \\
&\approx \left[2 - \left(1 - \frac{1}{2}\beta_{i+1}\right) + o(\beta_{i+1})\right] \mathbf{x}_{i+1} + \beta_{i+1} \mathbf{s}_{\theta^*}(\mathbf{x}_{i+1}, i+1) + \sqrt{\beta_{i+1}} \mathbf{z}_{i+1} \\
&= (2 - \sqrt{1 - \beta_{i+1}}) \mathbf{x}_{i+1} + \beta_{i+1} \mathbf{s}_{\theta^*}(\mathbf{x}_{i+1}, i+1) + \sqrt{\beta_{i+1}} \mathbf{z}_{i+1}.
\end{aligned}$$

Therefore, the original ancestral sampler of Eq. (7.4) is essentially a different discretization to the same reverse-time SDE. This unifies the sampling method in [HJA20] as a numerical solver to the reverse-time VP SDE in our continuous framework.

B.5.6 Ancestral Sampling for SMLD Models

The ancestral sampling method for DDPM models can also be adapted to SMLD models. Consider a sequence of noise scales $\sigma_1 < \sigma_2 < \dots < \sigma_N$ as in SMLD. By perturbing a data point \mathbf{x}_0 with these noise scales sequentially, we obtain a Markov chain $\mathbf{x}_0 \rightarrow \mathbf{x}_1 \rightarrow \dots \rightarrow \mathbf{x}_N$, where

$$p(\mathbf{x}_i \mid \mathbf{x}_{i-1}) = \mathcal{N}(\mathbf{x}_i; \mathbf{x}_{i-1}, (\sigma_i^2 - \sigma_{i-1}^2) \mathbf{I}), \quad i = 1, 2, \dots, N.$$

Here we assume $\sigma_0 = 0$ to simplify notations. Following [HJA20], we can compute

$$q(\mathbf{x}_{i-1} \mid \mathbf{x}_i, \mathbf{x}_0) = \mathcal{N}\left(\mathbf{x}_{i-1}; \frac{\sigma_{i-1}^2}{\sigma_i^2} \mathbf{x}_i + \left(1 - \frac{\sigma_{i-1}^2}{\sigma_i^2}\right) \mathbf{x}_0, \frac{\sigma_{i-1}^2(\sigma_i^2 - \sigma_{i-1}^2)}{\sigma_i^2} \mathbf{I}\right).$$

If we parameterize the reverse transition kernel as $p_{\theta}(\mathbf{x}_{i-1} \mid \mathbf{x}_i) = \mathcal{N}(\mathbf{x}_{i-1}; \boldsymbol{\mu}_{\theta}(\mathbf{x}_i, i), \tau_i^2 \mathbf{I})$, then

$$\begin{aligned} L_{t-1} &= \mathbb{E}_q[D_{\text{KL}}(q(\mathbf{x}_{i-1} \mid \mathbf{x}_i, \mathbf{x}_0)) \parallel p_{\theta}(\mathbf{x}_{i-1} \mid \mathbf{x}_i)] \\ &= \mathbb{E}_q \left[\frac{1}{2\tau_i^2} \left\| \frac{\sigma_{i-1}^2}{\sigma_i^2} \mathbf{x}_i + \left(1 - \frac{\sigma_{i-1}^2}{\sigma_i^2}\right) \mathbf{x}_0 - \boldsymbol{\mu}_{\theta}(\mathbf{x}_i, i) \right\|_2^2 \right] + C \\ &= \mathbb{E}_{\mathbf{x}_0, \mathbf{z}} \left[\frac{1}{2\tau_i^2} \left\| \mathbf{x}_i(\mathbf{x}_0, \mathbf{z}) - \frac{\sigma_i^2 - \sigma_{i-1}^2}{\sigma_i} \mathbf{z} - \boldsymbol{\mu}_{\theta}(\mathbf{x}_i(\mathbf{x}_0, \mathbf{z}), i) \right\|_2^2 \right] + C, \end{aligned}$$

where L_{t-1} is one representative term in the ELBO objective (see Eq. (8) in [HJA20]), C is a constant that does not depend on θ , $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, and $\mathbf{x}_i(\mathbf{x}_0, \mathbf{z}) = \mathbf{x}_0 + \sigma_i \mathbf{z}$. We can therefore parameterize $\boldsymbol{\mu}_{\theta}(\mathbf{x}_i, i)$ via

$$\boldsymbol{\mu}_{\theta}(\mathbf{x}_i, i) = \mathbf{x}_i + (\sigma_i^2 - \sigma_{i-1}^2) \mathbf{s}_{\theta}(\mathbf{x}_i, i),$$

where $\mathbf{s}_{\theta}(\mathbf{x}_i, i)$ is to estimate \mathbf{z}/σ_i . As in [HJA20], we let $\tau_i = \sqrt{\frac{\sigma_{i-1}^2(\sigma_i^2 - \sigma_{i-1}^2)}{\sigma_i^2}}$. Through ancestral sampling on $\prod_{i=1}^N p_{\theta}(\mathbf{x}_{i-1} \mid \mathbf{x}_i)$, we obtain the following iteration rule

$$\mathbf{x}_{i-1} = \mathbf{x}_i + (\sigma_i^2 - \sigma_{i-1}^2) \mathbf{s}_{\theta^*}(\mathbf{x}_i, i) + \sqrt{\frac{\sigma_{i-1}^2(\sigma_i^2 - \sigma_{i-1}^2)}{\sigma_i^2}} \mathbf{z}_i, i = 1, 2, \dots, N, \quad (\text{B.33})$$

where $\mathbf{x}_N \sim \mathcal{N}(\mathbf{0}, \sigma_N^2 \mathbf{I})$, θ^* denotes the optimal parameter of \mathbf{s}_{θ} , and $\mathbf{z}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. We call Eq. (B.33) the ancestral sampling method for SMLD models.

B.5.7 Predictor-Corrector Samplers

Predictor-Corrector (PC) sampling The predictor can be any numerical solver for the reverse-time SDE with a fixed discretization strategy. The corrector can be any score-based MCMC approach. In PC sampling, we alternate between the predictor and corrector, as described in Algorithm B.3. For example, when using the reverse diffusion SDE solver (Appendix B.5.5) as the predictor, and annealed Langevin dynamics (*cf.*, Chapters 5 and 6) as the corrector, we have Algorithms B.4 and B.5 for VE and VP SDEs respectively, where $\{\epsilon_i\}_{i=0}^{N-1}$ are step sizes for Langevin dynamics as specified below.

Algorithm B.3 Predictor-Corrector (PC) sampling**Require:**

N: Number of discretization steps for the reverse-time SDE

M: Number of corrector steps

```

1: Initialize  $\mathbf{x}_N \sim p_T(\mathbf{x})$ 
2: for  $i = N - 1$  to 0 do
3:    $\mathbf{x}_i \leftarrow \text{Predictor}(\mathbf{x}_{i+1})$ 
4:   for  $j = 1$  to M do
5:      $\mathbf{x}_i \leftarrow \text{Corrector}(\mathbf{x}_i)$ 
6:   end for
7: end for
8: return  $\mathbf{x}_0$ 

```

Algorithm B.4 PC sampling (VE SDE)

```

1:  $\mathbf{x}_N \sim \mathcal{N}(\mathbf{0}, \sigma_{\max}^2 \mathbf{I})$ 
2: for  $i = N - 1$  to 0 do
3:    $\mathbf{x}'_i \leftarrow \mathbf{x}_{i+1} + (\sigma_{i+1}^2 - \sigma_i^2) \mathbf{s}_{\theta^*}(\mathbf{x}_{i+1}, \sigma_{i+1})$ 
4:    $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ 
5:    $\mathbf{x}_i \leftarrow \mathbf{x}'_i + \sqrt{\sigma_{i+1}^2 - \sigma_i^2} \mathbf{z}$ 
6:   for  $j = 1$  to M do
7:      $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ 
8:      $\mathbf{x}_i \leftarrow \mathbf{x}_i + \epsilon_i \mathbf{s}_{\theta^*}(\mathbf{x}_i, \sigma_i) + \sqrt{2\epsilon_i} \mathbf{z}$ 
9:   end for
10: end for
11: return  $\mathbf{x}_0$ 

```

Algorithm B.5 PC sampling (VP SDE)

<pre> 1: $\mathbf{x}_N \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ 2: for $i = N - 1$ to 0 do 3: $\mathbf{x}'_i \leftarrow (2 - \sqrt{1 - \beta_{i+1}}) \mathbf{x}_{i+1} + \beta_{i+1} \mathbf{s}_{\theta^*}(\mathbf{x}_{i+1}, i + 1)$ 4: $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ 5: $\mathbf{x}_i \leftarrow \mathbf{x}'_i + \sqrt{\beta_{i+1}} \mathbf{z}$ </pre>	Predictor
<pre> 6: for $j = 1$ to M do 7: $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ 8: $\mathbf{x}_i \leftarrow \mathbf{x}_i + \epsilon_i \mathbf{s}_{\theta^*}(\mathbf{x}_i, i) + \sqrt{2\epsilon_i} \mathbf{z}$ 9: end for 10: end for 11: return \mathbf{x}_0 </pre>	Corrector

The corrector algorithms We take the schedule of annealed Langevin dynamics in [SE19b], but re-frame it with slight modifications in order to get better interpretability and empirical performance. We provide the corrector algorithms in Algorithms B.6 and B.7 respectively, where we call r the “signal-to-noise” ratio. We determine the step size ϵ using the norm of the Gaussian noise $\|\mathbf{z}\|_2$, norm of the score-based model $\|\mathbf{s}_{\theta^*}\|_2$ and the signal-to-noise ratio r . When sampling a large batch of samples together, we replace the norm $\|\cdot\|_2$ with the average norm across the mini-batch. When the batch size is small, we suggest replacing $\|\mathbf{z}\|_2$ with \sqrt{d} , where d is the dimensionality of \mathbf{z} .

Algorithm B.6 Corrector algorithm (VE).

Require: $\{\sigma_i\}_{i=1}^N, r, N, M$.

- 1: $\mathbf{x}_N^0 \sim \mathcal{N}(\mathbf{0}, \sigma_{\max}^2 \mathbf{I})$
- 2: **for** $i \leftarrow N$ to 1 **do**
- 3: **for** $j \leftarrow 1$ to M **do**
- 4: $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
- 5: $\mathbf{G} \leftarrow \mathbf{s}_{\theta*}(\mathbf{x}_i^{j-1}, \sigma_i)$
- 6: $\epsilon \leftarrow 2(r \|\mathbf{z}\|_2 / \|\mathbf{G}\|_2)^2$
- 7: $\mathbf{x}_i^j \leftarrow \mathbf{x}_i^{j-1} + \epsilon \mathbf{G} + \sqrt{2\epsilon} \mathbf{z}$
- 8: **end for**
- 9: $\mathbf{x}_{i-1}^0 \leftarrow \mathbf{x}_i^M$
- 10: **end for**

return \mathbf{x}_0^0

Algorithm B.7 Corrector algorithm (VP).

Require: $\{\beta_i\}_{i=1}^N, \{\alpha_i\}_{i=1}^N, r, N, M$.

- 1: $\mathbf{x}_N^0 \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
- 2: **for** $i \leftarrow N$ to 1 **do**
- 3: **for** $j \leftarrow 1$ to M **do**
- 4: $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
- 5: $\mathbf{G} \leftarrow \mathbf{s}_{\theta*}(\mathbf{x}_i^{j-1}, i)$
- 6: $\epsilon \leftarrow 2\alpha_i(r \|\mathbf{z}\|_2 / \|\mathbf{G}\|_2)^2$
- 7: $\mathbf{x}_i^j \leftarrow \mathbf{x}_i^{j-1} + \epsilon \mathbf{G} + \sqrt{2\epsilon} \mathbf{z}$
- 8: **end for**
- 9: $\mathbf{x}_{i-1}^0 \leftarrow \mathbf{x}_i^M$
- 10: **end for**

return \mathbf{x}_0^0

Denoising For both SMLD and DDPM models, the generated samples typically contain small noise that is hard to detect by humans. As noted by [Jol+21], FIDs can be significantly worse without removing this noise. This unfortunate sensitivity to noise is also part of the reason why NCSN models trained with SMLD has been performing worse than DDPM models in terms of FID, because the former does not use a denoising step at the end of sampling, while the latter does. In all experiments of this paper we ensure there is a single denoising step at the end of sampling, using Tweedie’s formula [Efr11].

Training We use the same architecture in [HJA20] for our score-based models. For the VE SDE, we train a model with the original SMLD objective in Eq. (7.1); similarly for the VP SDE, we use the original DDPM objective in Eq. (7.3). We apply a total number of 1000 noise scales for training both models. For results in Fig. B.43, we train an NCSN++ model (definition in Appendix B.5.8) on 256×256 LSUN bedroom and church_outdoor [Yu+15] datasets with the VE SDE and our continuous objective Eq. (7.7). The batch size is fixed to 128 on CIFAR-10 and 64 on LSUN.

Ad-hoc interpolation methods for noise scales Models in this experiment are all trained with 1000 noise scales. To get results for P2000 (predictor-only sampler using 2000

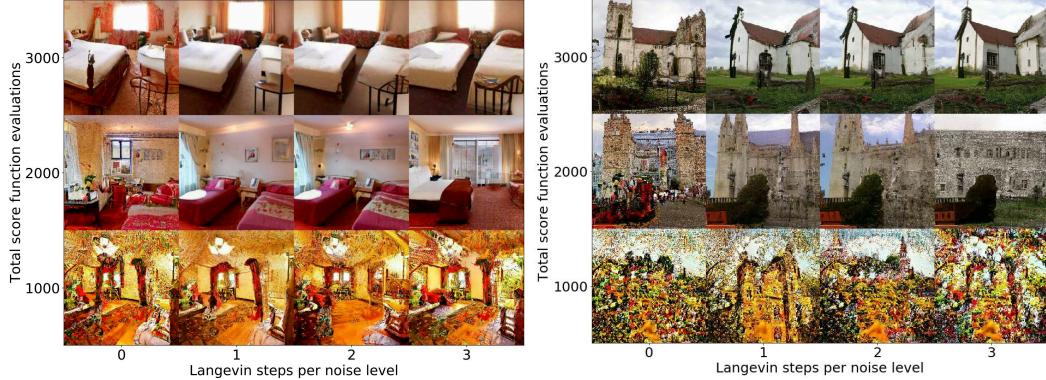


Figure B.43: PC sampling for LSUN bedroom and church. The vertical axis corresponds to the total computation, and the horizontal axis represents the amount of computation allocated to the corrector. Samples are the best when computation is split between the predictor and corrector.

steps) which requires 2000 noise scales, we need to interpolate between 1000 noise scales at test time. The specific architecture of the noise-conditional score-based model in [HJA20] uses sinusoidal positional embeddings for conditioning on integer time steps. This allows us to interpolate between noise scales at test time in an ad-hoc way (while it is hard to do so for other architectures like the one in Chapters 5 and 6). Specifically, for SMLD models, we keep σ_{\min} and σ_{\max} fixed and double the number of time steps. For DDPM models, we halve β_{\min} and β_{\max} before doubling the number of time steps. Suppose $\{s_{\theta}(\mathbf{x}, i)\}_{i=0}^{N-1}$ is a score-based model trained on N time steps, and let $\{s'_{\theta}(\mathbf{x}, i)\}_{i=0}^{2N-1}$ denote the corresponding interpolated score-based model at $2N$ time steps. We test two different interpolation strategies for time steps: linear interpolation where $s'_{\theta}(\mathbf{x}, i) = s_{\theta}(\mathbf{x}, i/2)$ and rounding interpolation where $s'_{\theta}(\mathbf{x}, i) = s_{\theta}(\mathbf{x}, \lfloor i/2 \rfloor)$. We provide results with linear interpolation in Table 7.1, and give results of rounding interpolation in Table B.12. We observe that different interpolation methods result in performance differences but maintain the general trend of predictor-corrector methods performing on par or better than predictor-only or corrector-only samplers.

Hyper-parameters of the samplers For Predictor-Corrector and corrector-only samplers on CIFAR-10, we search for the best signal-to-noise ratio (r) over a grid that increments at 0.01. We report the best r in Table B.13. For LSUN bedroom/church_outdoor, we fix r to 0.075. Unless otherwise noted, we use one corrector step per noise scale for all

Table B.12: Comparing different samplers on CIFAR-10, where “P2000” uses the rounding interpolation between noise scales. Shaded regions are obtained with the same computation (number of score function evaluations). Mean and standard deviation are reported over five sampling runs.

FID↓\ Sampler Predictor	Variance Exploding SDE (SMLD)				Variance Preserving SDE (DDPM)			
	P1000	P2000	C2000	PC1000	P1000	P2000	C2000	PC1000
ancestral sampling	$4.98 \pm .06$	$4.92 \pm .02$		$3.62 \pm .03$	$3.24 \pm .02$	$3.11 \pm .03$		$3.21 \pm .02$
reverse diffusion	$4.79 \pm .07$	$4.72 \pm .07$	$20.43 \pm .07$	$3.60 \pm .02$	$3.21 \pm .02$	$3.10 \pm .03$	$19.06 \pm .06$	$3.18 \pm .01$
probability flow	$15.41 \pm .15$	$12.87 \pm .09$		$3.51 \pm .04$	$3.59 \pm .04$	$3.25 \pm .04$		$3.06 \pm .03$

Table B.13: Optimal signal-to-noise ratios of different samplers. “P1000” or “P2000”: predictor-only samplers using 1000 or 2000 steps. “C2000”: corrector-only samplers using 2000 steps. “PC1000”: PC samplers using 1000 predictor and 1000 corrector steps.

r\ Sampler Predictor	VE SDE (SMLD)				VP SDE (DDPM)			
	P1000	P2000	C2000	PC1000	P1000	P2000	C2000	PC1000
ancestral sampling	-	-		0.17	-	-		0.01
reverse diffusion	-	-	0.22	0.16	-	-	0.27	0.01
probability flow	-	-		0.17	-	-		0.04

PC samplers. We use two corrector steps per noise scale for corrector-only samplers on CIFAR-10. For sample generation, the batch size is 1024 on CIFAR-10 and 8 on LSUN bedroom/church_outdoor.

B.5.8 Architecture Improvements

We explored several architecture designs to improve score-based models for both VE and VP SDEs. Our endeavor gives rise to new state-of-the-art sample quality on CIFAR-10, and enables the first high-fidelity image samples of resolution 1024×1024 from score-based generative models.

Settings for Architecture Exploration

Unless otherwise noted, all models are trained for 1.3M iterations, and we save one checkpoint per 50k iterations. For VE SDEs, we consider two datasets: 32×32 CIFAR-10 [KH+09] and

64×64 CelebA [Liu+15], pre-processed following [SE20]. We compare different configurations based on their FID scores averaged over checkpoints after 0.5M iterations. For VP SDEs, we only consider the CIFAR-10 dataset to save computation, and compare models based on the average FID scores over checkpoints obtained between 0.25M and 0.5M iterations, because FIDs turn to increase after 0.5M iterations for VP SDEs.

All FIDs are computed on 50k samples with `tensorflow_gan`. For sampling, we use the PC sampler discretized at 1000 time steps. We choose reverse diffusion (see Appendix B.5.5) as the predictor. We use one corrector step per update of the predictor for VE SDEs with a signal-to-noise ratio of 0.16, but save the corrector step for VP SDEs since correctors there only give slightly better results but require double computation. We follow [HJA20] for optimization, including the learning rate, gradient clipping, and learning rate warm-up schedules. Unless otherwise noted, models are trained with the original discrete SMLD and DDPM objectives in Eqs. (7.1) and (7.3) and use a batch size of 128. The optimal architectures found under these settings are subsequently transferred to continuous objectives and deeper models. We also directly transfer the best architecture for VP SDEs to sub-VP SDEs, given the similarity of these two SDEs.

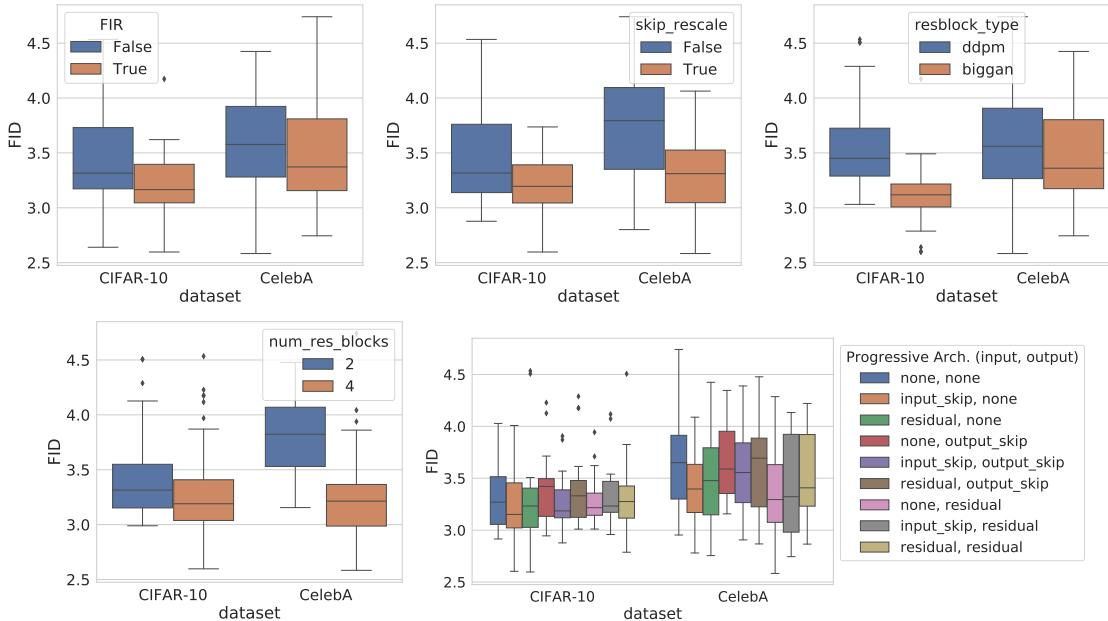


Figure B.44: The effects of different architecture components for score-based models trained with VE perturbations.

Our architecture is mostly based on [HJA20]. We additionally introduce the following components to maximize the potential improvement of score-based models.

1. Upsampling and downsampling images with anti-aliasing based on Finite Impulse Response (FIR) [Zha19]. We follow the same implementation and hyper-parameters in StyleGAN-2 [Kar+20b].
2. Rescaling all skip connections by $1/\sqrt{2}$. This has been demonstrated effective in several best-in-class GAN models, including ProgressiveGAN [Kar+18], StyleGAN [KLA19] and StyleGAN-2 [Kar+20b].
3. Replacing the original residual blocks in DDPM with residual blocks from BigGAN [BDS19].
4. Increasing the number of residual blocks per resolution from 2 to 4.
5. Incorporating progressive growing architectures. We consider two progressive architectures for input: “input skip” and “residual”, and two progressive architectures for output: “output skip” and “residual”. These progressive architectures are defined and implemented according to StyleGAN-2.

We also tested equalized learning rates, a trick used in very successful models like ProgressiveGAN [Kar+18] and StyleGAN [KLA19]. However, we found it harmful at an early stage of our experiments, and therefore decided not to explore more on it.

The exponential moving average (EMA) rate has a significant impact on performance. For models trained with VE perturbations, we notice that 0.999 works better than 0.9999, whereas for models trained with VP perturbations it is the opposite. We therefore use an EMA rate of 0.999 and 0.9999 for VE and VP models respectively.

Results on CIFAR-10

All architecture components introduced above can improve the performance of score-based models trained with VE SDEs, as shown in Fig. B.44. The box plots demonstrate the importance of each component when other components can vary freely. On both CIFAR-10 and CelebA, the additional components that we explored always improve the performance on average for VE SDEs. For progressive growing, it is not clear which combination of configurations consistently performs the best, but the results are typically better than when

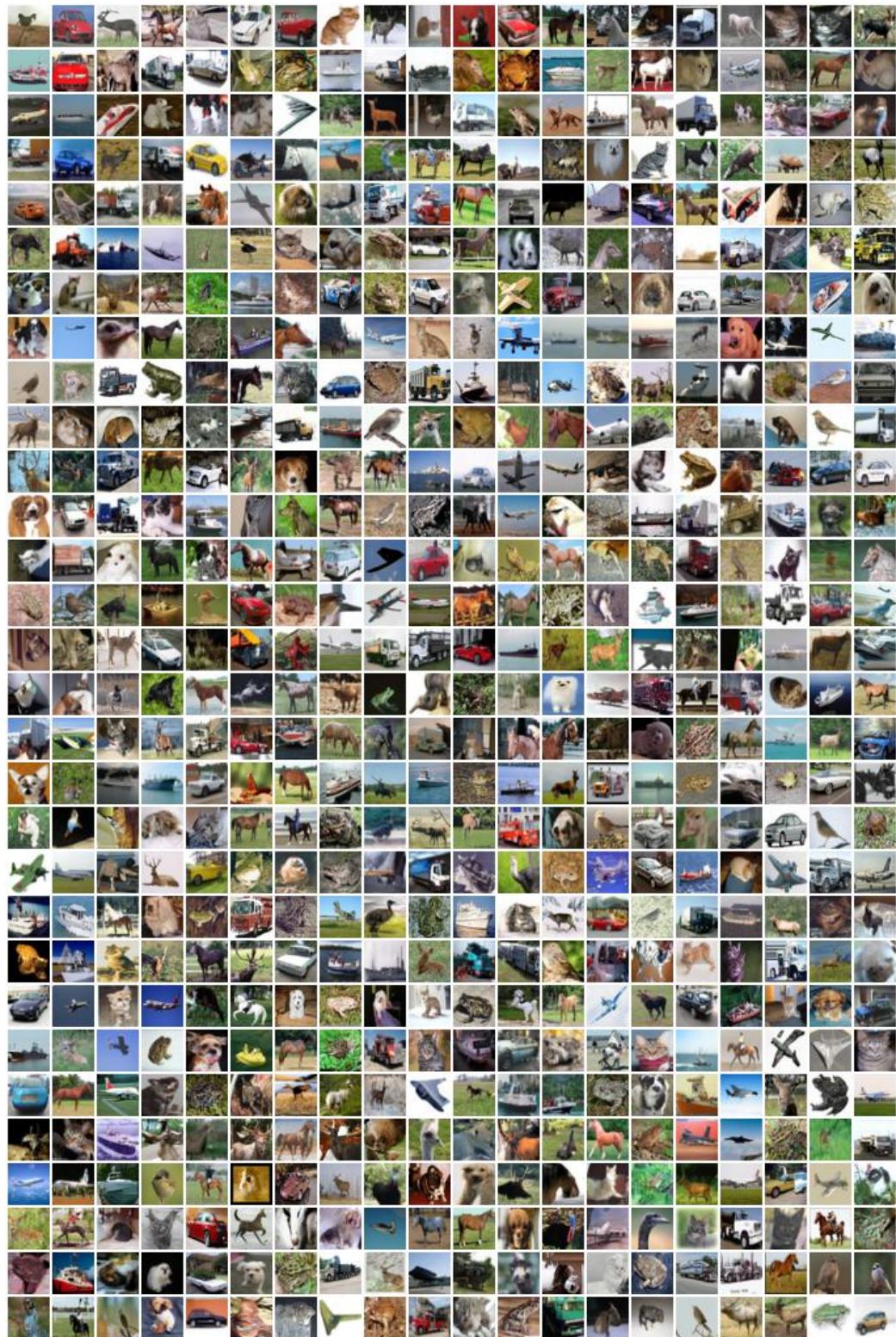


Figure B.45: unconditional cifar-10 samples from ncsn++ cont. (deep, ve).



Figure B.46: samples on 1024×1024 celeba-hq from a modified ncsn++ model trained with the ve sde.

no progressive growing architecture is used. Our best score-based model for VE SDEs 1) uses FIR upsampling/downsampling, 2) rescales skip connections, 3) employs BigGAN-type residual blocks, 4) uses 4 residual blocks per resolution instead of 2, and 5) uses “residual” for input and no progressive growing architecture for output. We name this model “NCSN++”, following the naming convention of previous SMLD models [SE19b; SE20].

We followed a similar procedure to examine these architecture components for VP SDEs, except that we skipped experiments on CelebA due to limited computing resources. The NCSN++ architecture worked decently well for VP SDEs, ranked 4th place over all 144 possible configurations. The top configuration, however, has a slightly different structure, which uses no FIR upsampling/downsampling and no progressive growing architecture compared to NCSN++. We name this model “DDPM++”, following the naming convention of [HJA20].

The basic NCSN++ model with 4 residual blocks per resolution achieves an FID of 2.45 on CIFAR-10, whereas the basic DDPM++ model achieves an FID of 2.78. Here in order to match the convention used in [Kar+18; SE19b] and [HJA20], we report the lowest FID value over the course of training, rather than the average FID value over checkpoints after 0.5M iterations (used for comparing different models of VE SDEs) or between 0.25M and 0.5M iterations (used for comparing VP SDE models) in our architecture exploration.

Switching from discrete training objectives to continuous ones in Eq. (7.7) further improves the FID values for all SDEs. To condition the NCSN++ model on continuous time variables, we change positional embeddings, the layers in [HJA20] for conditioning on discrete time steps, to random Fourier feature embeddings [Tan+20]. The scale parameter of these random Fourier feature embeddings is fixed to 16. We also reduce the number of training iterations to 0.95M to suppress overfitting. These changes improve the FID on CIFAR-10 from 2.45 to 2.38 for NCSN++ trained with the VE SDE, resulting in a model called “NCSN++ cont.”. In addition, we can further improve the FID from 2.38 to 2.20 by doubling the number of residual blocks per resolution for NCSN++ cont., resulting in the model denoted as “NCSN++ cont. (deep)”. All quantitative results are summarized in Table 7.2, and we provide random samples from our best model in Fig. B.45.

Similarly, we can also condition the DDPM++ model on continuous time steps, resulting in a model “DDPM++ cont.”. When trained with the VP SDE, it improves the FID of 2.78 from DDPM++ to 2.55. When trained with the sub-VP SDE, it achieves an FID of 2.61. To get better performance, we used the Euler-Maruyama solver as the predictor

for continuously-trained models, instead of the ancestral sampling predictor or the reverse diffusion predictor. This is because the discretization strategy of the original DDPM method does not match the variance of the continuous process well when $t \rightarrow 0$, which significantly hurts FID scores. Doubling the depth, and training with 0.95M iterations, we can improve FIDs for both VP and sub-VP SDEs, leading to a model “DDPM++ cont. (deep)”. Its FID score is 2.41, same for both VP and sub-VP SDEs.

High Resolution Images

Encouraged by the success of NCSN++ on CIFAR-10, we proceed to test it on 1024×1024 CelebA-HQ [Kar+18], a task that was previously only achievable by some GAN models and VQ-VAE-2 [ROV19]. We used a batch size of 8, increased the EMA rate to 0.9999, and trained a model similar to NCSN++ with the continuous objective (Eq. (7.7)) for around 2.4M iterations (please find the detailed architecture in our code release.) We use the PC sampler discretized at 2000 steps with the reverse diffusion predictor, one Langevin step per predictor update and a signal-to-noise ratio of 0.15. The scale parameter for the random Fourier feature embeddings is fixed to 16. We use the “input skip” progressive architecture for the input, and “output skip” progressive architecture for the output. We provide samples in Fig. B.46. Although these samples are not perfect (*e.g.*, there are visible flaws on facial symmetry), we believe these results are encouraging and can demonstrate the scalability of our approach. Future work on more effective architectures are likely to significantly advance the performance of score-based generative models on this task.

B.5.9 Controllable Generation

Consider a forward SDE with the following general form

$$d\mathbf{x} = \mathbf{f}(\mathbf{x}, t) dt + \mathbf{G}(\mathbf{x}, t) d\mathbf{w},$$

and suppose the initial state distribution is $p_0(\mathbf{x}(0) | \mathbf{y})$. The density at time t is $p_t(\mathbf{x}(t) | \mathbf{y})$ when conditioned on \mathbf{y} . Therefore, using [And82], the reverse-time SDE is given by

$$d\mathbf{x} = \{\mathbf{f}(\mathbf{x}, t) - \nabla \cdot [\mathbf{G}(\mathbf{x}, t)\mathbf{G}(\mathbf{x}, t)^T] - \mathbf{G}(\mathbf{x}, t)\mathbf{G}(\mathbf{x}, t)^T \nabla_{\mathbf{x}} \log p_t(\mathbf{x} | \mathbf{y})\} dt + \mathbf{G}(\mathbf{x}, t) d\bar{\mathbf{w}}. \quad (\text{B.34})$$

Since $p_t(\mathbf{x}(t) \mid \mathbf{y}) \propto p_t(\mathbf{x}(t))p(\mathbf{y} \mid \mathbf{x}(t))$, the score $\nabla_{\mathbf{x}} \log p_t(\mathbf{x}(t) \mid \mathbf{y})$ can be computed easily by

$$\nabla_{\mathbf{x}} \log p_t(\mathbf{x}(t) \mid \mathbf{y}) = \nabla_{\mathbf{x}} \log p_t(\mathbf{x}(t)) + \nabla_{\mathbf{x}} \log p(\mathbf{y} \mid \mathbf{x}(t)). \quad (\text{B.35})$$

This subsumes the conditional reverse-time SDE in Eq. (7.14) as a special case. All sampling methods we have discussed so far can be applied to the conditional reverse-time SDE for sample generation.

Class-Conditional Sampling

When \mathbf{y} represents class labels, we can train a time-dependent classifier $p_t(\mathbf{y} \mid \mathbf{x}(t))$ for class-conditional sampling. Since the forward SDE is tractable, we can easily create a pair of training data $(\mathbf{x}(t), \mathbf{y})$ by first sampling $(\mathbf{x}(0), \mathbf{y})$ from a dataset and then obtaining $\mathbf{x}(t) \sim p_{0t}(\mathbf{x}(t) \mid \mathbf{x}(0))$. Afterwards, we may employ a mixture of cross-entropy losses over different time steps, like Eq. (7.7), to train the time-dependent classifier $p_t(\mathbf{y} \mid \mathbf{x}(t))$.

To test this idea, we trained a Wide ResNet [ZK16] (Wide-ResNet-28-10) on CIFAR-10 with VE perturbations. The classifier is conditioned on $\log \sigma_i$ using random Fourier features [Tan+20], and the training objective is a simple sum of cross-entropy losses sampled at different scales. We provide a plot to show the accuracy of this classifier over noise scales in Fig. B.47. The score-based model is an unconditional NCSN++ (4 blocks/resolution) in Table 7.2, and we generate samples using the PC algorithm with 2000 discretization steps. The class-conditional samples are provided in Fig. 7.4, and an extended set of conditional samples is given in Fig. B.47.

Imputation

Imputation is a special case of conditional sampling. Denote by $\Omega(\mathbf{x})$ and $\bar{\Omega}(\mathbf{x})$ the known and unknown dimensions of \mathbf{x} respectively, and let $\mathbf{f}_{\bar{\Omega}}(\cdot, t)$ and $\mathbf{G}_{\bar{\Omega}}(\cdot, t)$ denote $\mathbf{f}(\cdot, t)$ and $\mathbf{G}(\cdot, t)$ restricted to the unknown dimensions. For VE/VP SDEs, the drift coefficient $\mathbf{f}(\cdot, t)$ is element-wise, and the diffusion coefficient $\mathbf{G}(\cdot, t)$ is diagonal. When $\mathbf{f}(\cdot, t)$ is element-wise, $\mathbf{f}_{\bar{\Omega}}(\cdot, t)$ denotes the same element-wise function applied only to the unknown dimensions. When $\mathbf{G}(\cdot, t)$ is diagonal, $\mathbf{G}_{\bar{\Omega}}(\cdot, t)$ denotes the sub-matrix restricted to unknown dimensions.

For imputation, our goal is to sample from $p(\bar{\Omega}(\mathbf{x}(0)) \mid \Omega(\mathbf{x}(0)) = \mathbf{y})$. Define a new

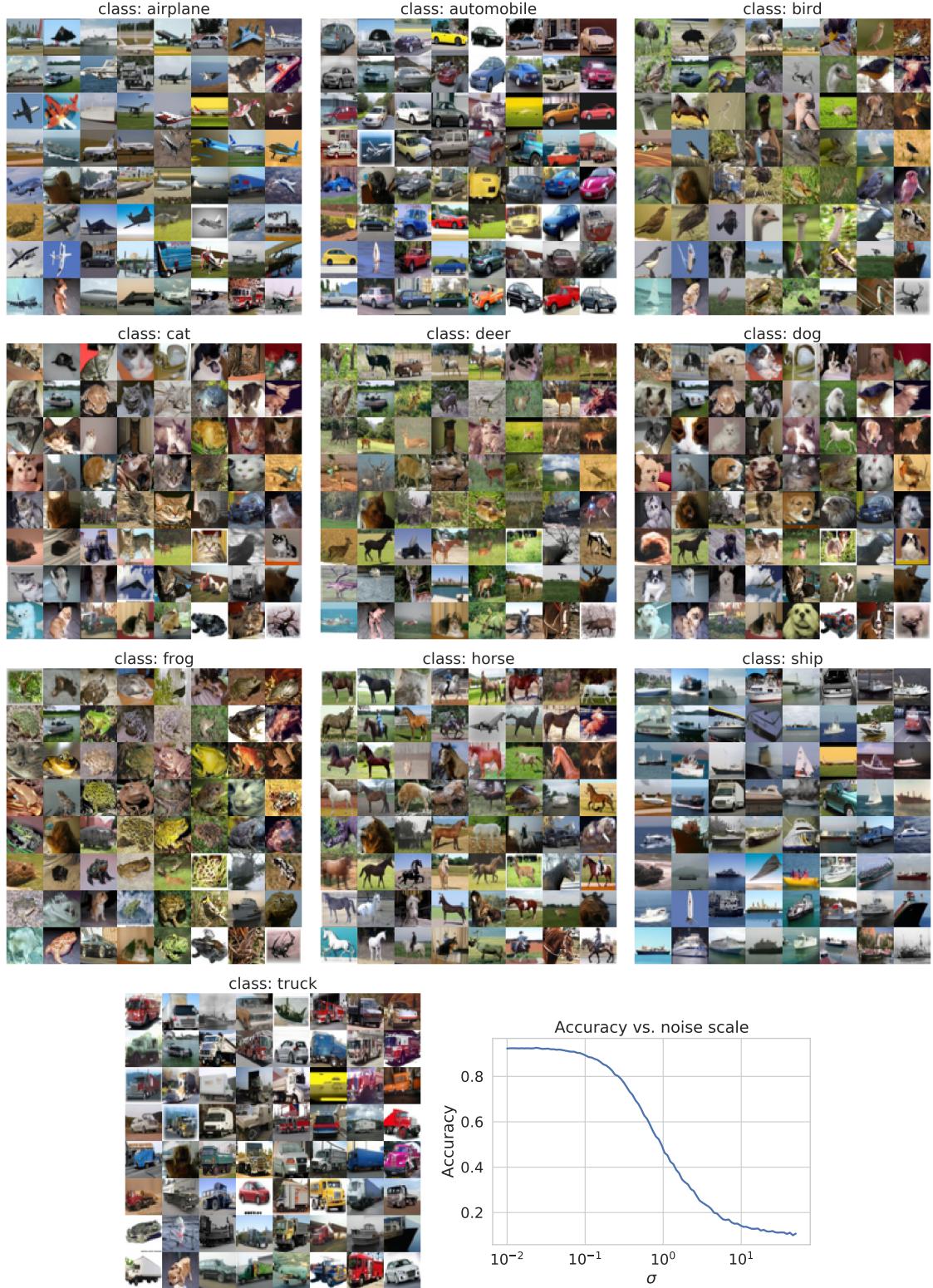


Figure B.47: Class-conditional image generation by solving the conditional reverse-time SDE with PC. The curve shows the accuracy of our noise-conditional classifier over different noise scales.

diffusion process $\mathbf{z}(t) = \bar{\Omega}(\mathbf{x}(t))$, and note that the SDE for $\mathbf{z}(t)$ can be written as

$$d\mathbf{z} = \mathbf{f}_{\bar{\Omega}}(\mathbf{z}, t) dt + \mathbf{G}_{\bar{\Omega}}(\mathbf{z}, t) d\mathbf{w}.$$

The reverse-time SDE, conditioned on $\Omega(\mathbf{x}(0)) = \mathbf{y}$, is given by

$$\begin{aligned} d\mathbf{z} = & \left\{ \mathbf{f}_{\bar{\Omega}}(\mathbf{z}, t) - \nabla \cdot [\mathbf{G}_{\bar{\Omega}}(\mathbf{z}, t) \mathbf{G}_{\bar{\Omega}}(\mathbf{z}, t)^T] \right. \\ & \left. - \mathbf{G}_{\bar{\Omega}}(\mathbf{z}, t) \mathbf{G}_{\bar{\Omega}}(\mathbf{z}, t)^T \nabla_{\mathbf{z}} \log p_t(\mathbf{z} \mid \Omega(\mathbf{x}(0)) = \mathbf{y}) \right\} dt + \mathbf{G}_{\bar{\Omega}}(\mathbf{z}, t) d\bar{\mathbf{w}}. \end{aligned}$$

Although $p_t(\mathbf{z}(t) \mid \Omega(\mathbf{x}(0)) = \mathbf{y})$ is in general intractable, it can be approximated. Let A denote the event $\Omega(\mathbf{x}(0)) = \mathbf{y}$. We have

$$\begin{aligned} p_t(\mathbf{z}(t) \mid \Omega(\mathbf{x}(0)) = \mathbf{y}) &= p_t(\mathbf{z}(t) \mid A) = \int p_t(\mathbf{z}(t) \mid \Omega(\mathbf{x}(t)), A) p_t(\Omega(\mathbf{x}(t)) \mid A) d\Omega(\mathbf{x}(t)) \\ &= \mathbb{E}_{p_t(\Omega(\mathbf{x}(t)) \mid A)} [p_t(\mathbf{z}(t) \mid \Omega(\mathbf{x}(t)), A)] \\ &\approx \mathbb{E}_{p_t(\Omega(\mathbf{x}(t)) \mid A)} [p_t(\mathbf{z}(t) \mid \hat{\Omega}(\mathbf{x}(t)))] \\ &\approx p_t(\mathbf{z}(t) \mid \hat{\Omega}(\mathbf{x}(t))), \end{aligned}$$

where $\hat{\Omega}(\mathbf{x}(t))$ is a random sample from $p_t(\Omega(\mathbf{x}(t)) \mid A)$, which is typically a tractable distribution. Therefore,

$$\begin{aligned} \nabla_{\mathbf{z}} \log p_t(\mathbf{z}(t) \mid \Omega(\mathbf{x}(0)) = \mathbf{y}) &\approx \nabla_{\mathbf{z}} \log p_t(\mathbf{z}(t) \mid \hat{\Omega}(\mathbf{x}(t))) \\ &= \nabla_{\mathbf{z}} \log p_t([\mathbf{z}(t); \hat{\Omega}(\mathbf{x}(t))]), \end{aligned}$$

where $[\mathbf{z}(t); \hat{\Omega}(\mathbf{x}(t))]$ denotes a vector $\mathbf{u}(t)$ such that $\Omega(\mathbf{u}(t)) = \hat{\Omega}(\mathbf{x}(t))$ and $\bar{\Omega}(\mathbf{u}(t)) = \mathbf{z}(t)$, and the identity holds because $\nabla_{\mathbf{z}} \log p_t([\mathbf{z}(t); \hat{\Omega}(\mathbf{x}(t))]) = \nabla_{\mathbf{z}} \log p_t(\mathbf{z}(t) \mid \hat{\Omega}(\mathbf{x}(t))) + \nabla_{\mathbf{z}} \log p_t(\hat{\Omega}(\mathbf{x}(t))) = \nabla_{\mathbf{z}} \log p_t(\mathbf{z}(t) \mid \hat{\Omega}(\mathbf{x}(t)))$.

We provided an extended set of inpainting results in Figs. B.48 and B.49.

Colorization

Colorization is a special case of imputation, except that the known data dimensions are coupled. We can decouple these data dimensions by using an orthogonal linear transformation to map the gray-scale image to a separate channel in a different space, and then perform imputation to complete the other channels before transforming everything back to the

original image space. The orthogonal matrix we used to decouple color channels is

$$\begin{pmatrix} 0.577 & -0.816 & 0 \\ 0.577 & 0.408 & 0.707 \\ 0.577 & 0.408 & -0.707 \end{pmatrix}.$$

Because the transformations are all orthogonal matrices, the standard Wiener process $\mathbf{w}(t)$ will still be a standard Wiener process in the transformed space, allowing us to build an SDE and use the same imputation method in Appendix B.5.9. We provide an extended set of colorization results in Figs. B.50 and B.51.

Solving General Inverse Problems

Suppose we have two random variables \mathbf{x} and \mathbf{y} , and we know the forward process of generating \mathbf{y} from \mathbf{x} , given by $p(\mathbf{y} | \mathbf{x})$. The inverse problem is to obtain \mathbf{x} from \mathbf{y} , that is, generating samples from $p(\mathbf{x} | \mathbf{y})$. In principle, we can estimate the prior distribution $p(\mathbf{x})$ and obtain $p(\mathbf{x} | \mathbf{y})$ using Bayes' rule: $p(\mathbf{x} | \mathbf{y}) = p(\mathbf{x})p(\mathbf{y} | \mathbf{x})/p(\mathbf{y})$. In practice, however, both estimating the prior and performing Bayesian inference are non-trivial.

Leveraging Eq. (B.34), score-based generative models provide one way to solve the inverse problem. Suppose we have a diffusion process $\{\mathbf{x}(t)\}_{t=0}^T$ generated by perturbing \mathbf{x} with an SDE, and a time-dependent score-based model $s_{\theta*}(\mathbf{x}(t), t)$ trained to approximate $\nabla_{\mathbf{x}} \log p_t(\mathbf{x}(t))$. Once we have an estimate of $\nabla_{\mathbf{x}} \log p_t(\mathbf{x}(t) | \mathbf{y})$, we can simulate the reverse-time SDE in Eq. (B.34) to sample from $p_0(\mathbf{x}(0) | \mathbf{y}) = p(\mathbf{x} | \mathbf{y})$. To obtain this estimate, we first observe that

$$\nabla_{\mathbf{x}} \log p_t(\mathbf{x}(t) | \mathbf{y}) = \nabla_{\mathbf{x}} \log \int p_t(\mathbf{x}(t) | \mathbf{y}(t), \mathbf{y}) p(\mathbf{y}(t) | \mathbf{y}) d\mathbf{y}(t),$$

where $\mathbf{y}(t)$ is defined via $\mathbf{x}(t)$ and the forward process $p(\mathbf{y}(t) | \mathbf{x}(t))$. Now assume two conditions:

- $p(\mathbf{y}(t) | \mathbf{y})$ is tractable. We can often derive this distribution from the interaction between the forward process and the SDE, like in the case of image imputation and colorization.
- $p_t(\mathbf{x}(t) | \mathbf{y}(t), \mathbf{y}) \approx p_t(\mathbf{x}(t) | \mathbf{y}(t))$. For small t , $\mathbf{y}(t)$ is almost the same as \mathbf{y} so the approximation holds. For large t , \mathbf{y} becomes further away from $\mathbf{x}(t)$ in the Markov

chain, and thus have smaller impact on $\mathbf{x}(t)$. Moreover, the approximation error for large t matter less for the final sample, since it is used early in the sampling process.

Given these two assumptions, we have

$$\begin{aligned}
 \nabla_{\mathbf{x}} \log p_t(\mathbf{x}(t) \mid \mathbf{y}) &\approx \nabla_{\mathbf{x}} \log \int p_t(\mathbf{x}(t) \mid \mathbf{y}(t)) p(\mathbf{y}(t) \mid \mathbf{y}) d\mathbf{y} \\
 &\approx \nabla_{\mathbf{x}} \log p_t(\mathbf{x}(t) \mid \hat{\mathbf{y}}(t)) \\
 &= \nabla_{\mathbf{x}} \log p_t(\mathbf{x}(t)) + \nabla_{\mathbf{x}} \log p_t(\hat{\mathbf{y}}(t) \mid \mathbf{x}(t)) \\
 &\approx s_{\theta^*}(\mathbf{x}(t), t) + \nabla_{\mathbf{x}} \log p_t(\hat{\mathbf{y}}(t) \mid \mathbf{x}(t)),
 \end{aligned} \tag{B.36}$$

where $\hat{\mathbf{y}}(t)$ is a sample from $p(\mathbf{y}(t) \mid \mathbf{y})$. Now we can plug Eq. (B.36) into Eq. (B.34) and solve the resulting reverse-time SDE to generate samples from $p(\mathbf{x} \mid \mathbf{y})$.

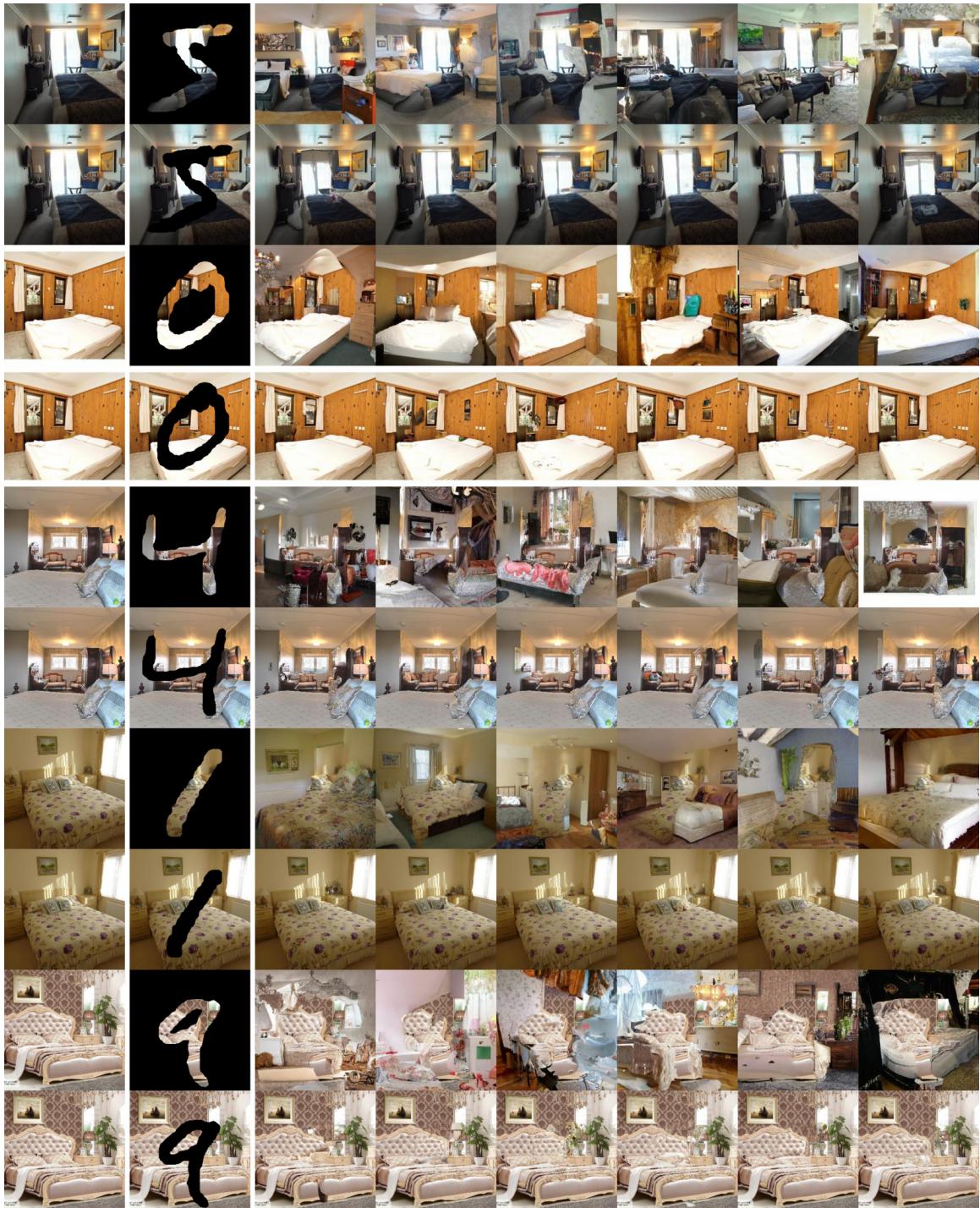
B.6 Chapter 8

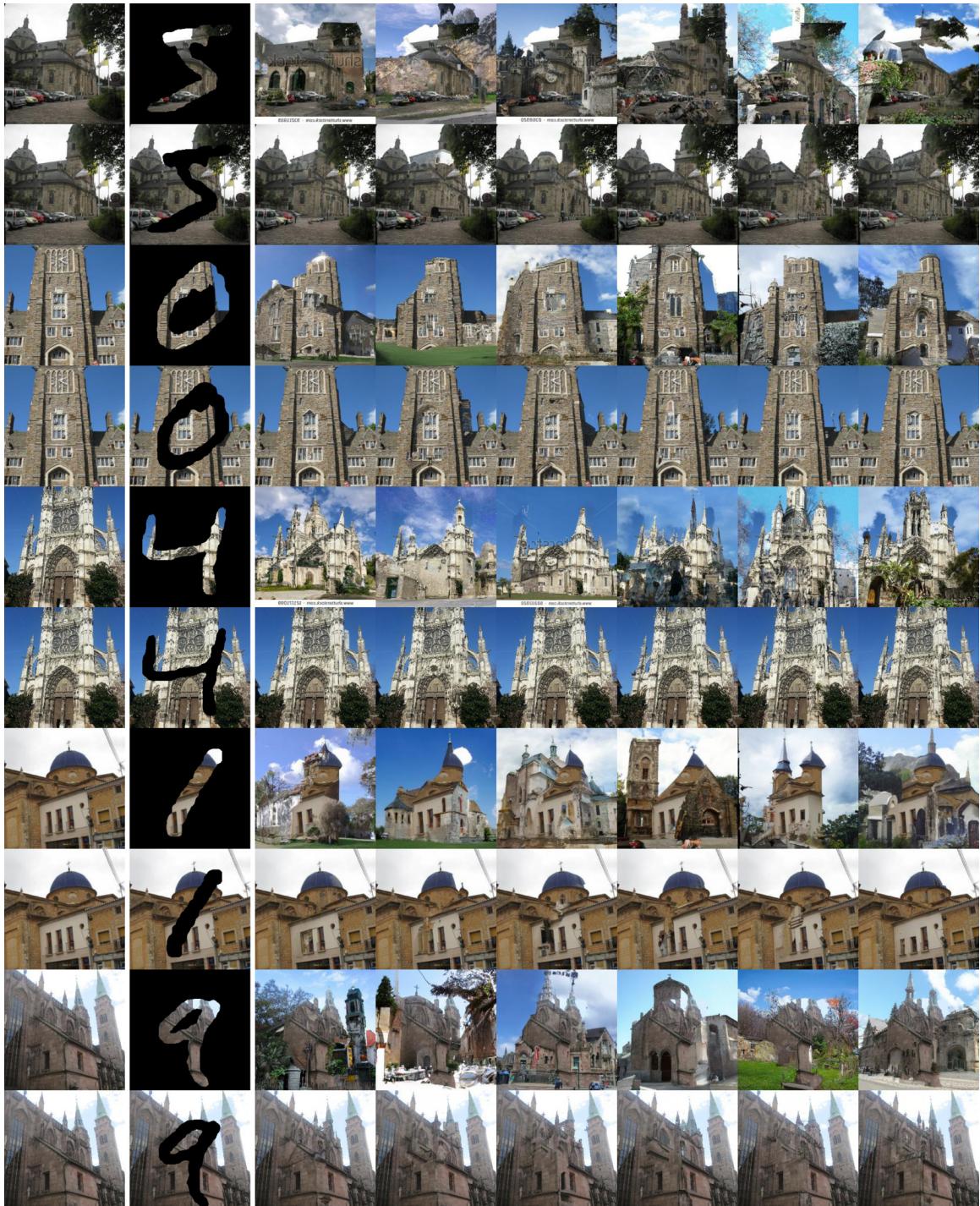
B.6.1 Additional Results

In Fig. B.52, we provide SSIM results versus the number of measurements for multiple methods and tasks. In general, the SSIM curves have very similar trends to the PSNR curves in Fig. 8.5. We additionally provide a visualization of metal artifact removal results in Fig. B.53.

B.6.2 The Task of Metal Artifact Removal

Metallic implants in an object can cause strong metal artifacts in CT imaging. As shown in Fig. B.54, the source of artifacts come from extremely bright regions in the sinogram, called metal traces. To reduce or ideally remove metal artifacts from a CT image, we remove metal traces from the sinogram and leverage the data prior to complete the sinogram. As a result, metal artifact removal can be viewed as an inverse problem, where the measurement process gives the full sinogram except for the metal trace region, and our goal is to reconstruct the full CT image using this partially known sinogram, which will be artifact-free assuming perfect inpainting of the sinogram.

Figure B.48: Extended inpainting results for 256×256 bedroom images.

Figure B.49: Extended inpainting results for 256×256 church images.

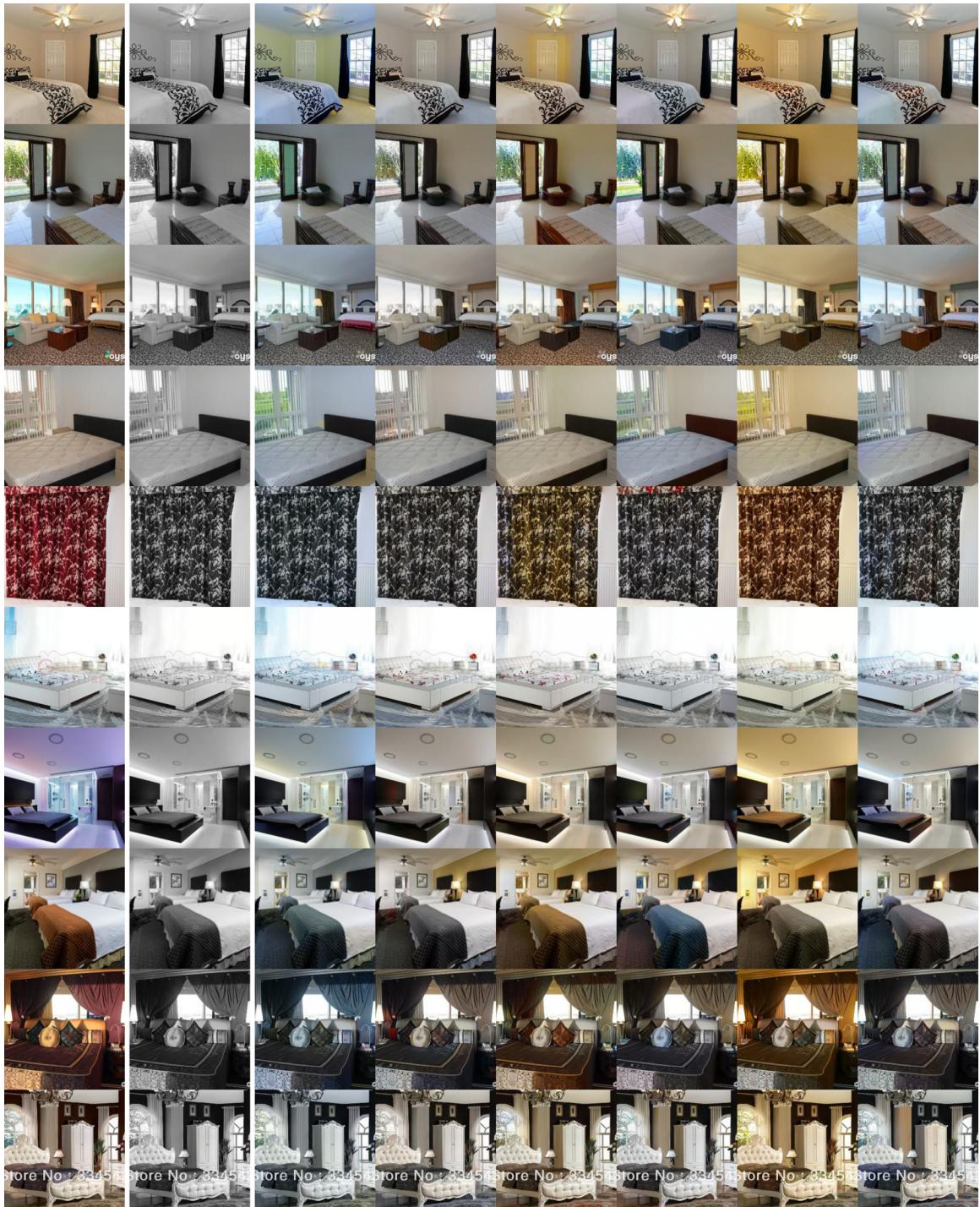
Figure B.50: Extended colorization results for 256×256 bedroom images.

Figure B.51: Extended colorization results for 256×256 church images.

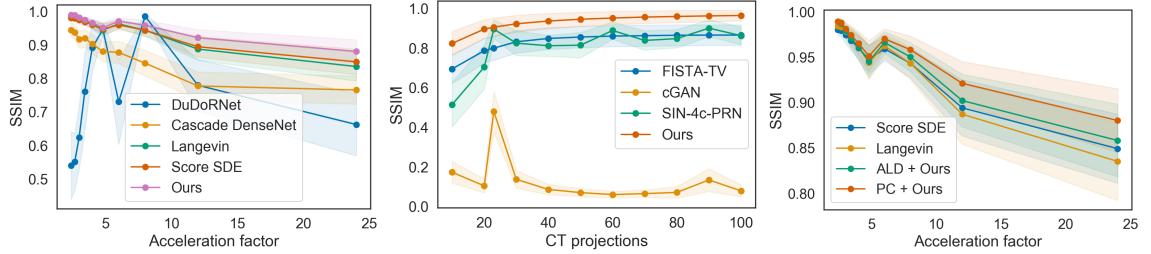


Figure B.52: SSIM vs. numbers of measurements. Shaded areas represent standard deviation. (Left) MRI on BraTS. (Center) CT on LIDC. (Right) Comparing score-based generative models for undersampled MRI reconstruction on BraTS.

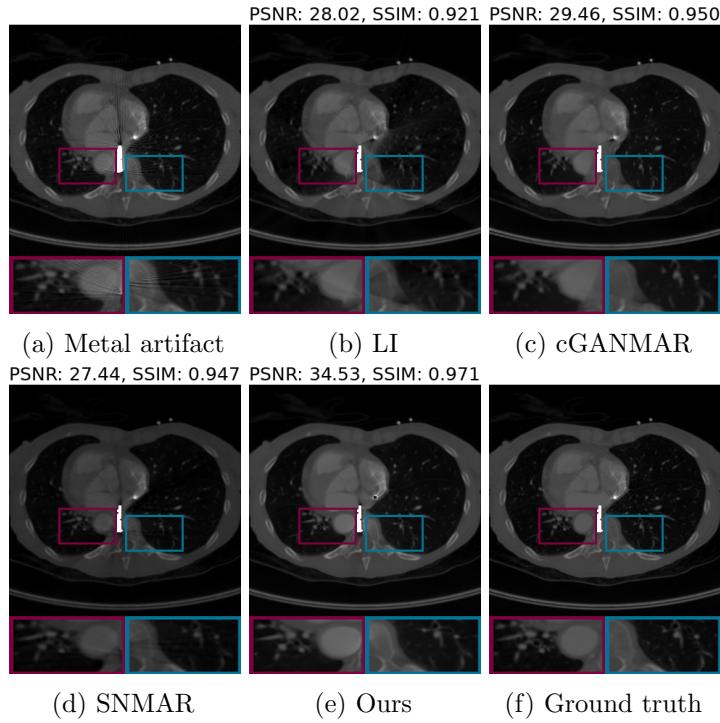


Figure B.53: Examples of metal artifact removal on LIDC. You may zoom in to view more details.

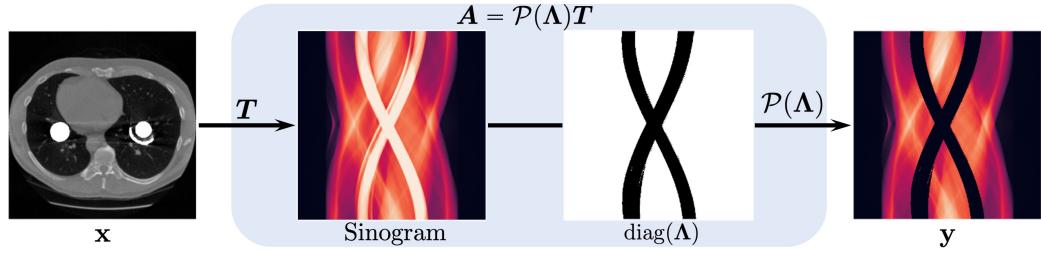


Figure B.54: The linear measurement process of metal artifact removal.

B.6.3 Details of datasets

CT datasets We conduct experiments of 2D CT image reconstruction on two datasets. First, the Lung Image Database Consortium image collection (LIDC) [Arm+11; Cla+13] consists of diagnostic and lung cancer screening thoracic computed tomography (CT) scans for lung cancer detection and diagnosis, which contains 1018 cases. Second, the Low Dose CT Image and Projection dataset (LDCT) [Cla+13; Moe+21] involves CT images of multiple anatomic sites, including 99 head CT scans, 100 chest CT scans, and 100 abdomen CT scans. Note that for the LDCT dataset, we only use the full-dose CT images in our experiments. In CT image processing, we convert the Hounsfield units from dicom files to the attenuation coefficients and set the background pixels to zero. Then, 2D CT images are sliced from 3D CT volumes. The sinograms are simulated from 2D CT images based on parallel-beam geometry with different number of projection angles that are equally distributed across 180 degrees.

MRI dataset The Brain Tumor Segmentation (BraTS) 2021 dataset [Men+14; Bak+17] collected for the image segmentation challenge contains 2000 cases (8000 MRI scans), where each case has four different MR contrasts: native (T1), post-contrast T1-weighted (T1Gd), T2-weighted (T2), and T2 Fluid Attenuated Inversion Recovery (T2-FLAIR). For each 3D MR volume, we extract 2D slices from 3D volumes and simulate k-space data by Fast Fourier Transform. To reconstruct MR images, we follow [Kno+20; Zbo+18] to undersample k-space data with an equispaced Cartesian mask, where the center k-space is fully sampled while the left k-space is under-sampled by equispaced columns.

B.6.4 Details of Score-Based Generative Models

We use the NCSN++ model architecture in Chapter 7, and perturb the data with the Variance Exploding (VE) SDE. Our training procedure follows that of Chapter 7. Instead of generating samples according to the numerical SDE solver in Algorithm 8.1, we use the Predictor-Corrector (PC) sampler as described in Chapter 7 since it generally has better performance for VE SDEs. In PC samplers, the predictor refers to a numerical solver for the reverse-time SDE while the corrector can be any Markov chain Monte Carlo (MCMC) method that only depends on the scores. One such MCMC method considered in this chapter is Langevin dynamics, whereby we transform any initial sample $\mathbf{x}^{(0)}$ to an approximate sample from $p_t(\mathbf{x})$ via the following procedure:

$$\mathbf{x}^{(i+1)} \leftarrow \mathbf{x}^{(i)} + \epsilon \nabla_{\mathbf{x}} \log p_t(\mathbf{x}^{(i)}) + \sqrt{2\epsilon} \mathbf{z}^{(i)}, \quad i = 0, 1, \dots, N-1. \quad (\text{B.37})$$

Here $N \in \mathbb{N}_{>0}$, $\epsilon > 0$, and $\mathbf{z}^{(i)} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. The theory of Langevin dynamics guarantees that in the limit of $N \rightarrow \infty$ and $\epsilon \rightarrow 0$, $\mathbf{x}^{(N)}$ is a sample from $p_t(\mathbf{x})$ under some regularity conditions. Note that Langevin dynamics only requires the knowledge of $\nabla_{\mathbf{x}} \log p_t(\mathbf{x})$, which can be approximated using the time-dependent score model $s_{\theta^*}(\mathbf{x}, t)$. In PC samplers, each predictor step immediately follows multiple consecutive corrector steps, all using the same $s_{\theta^*}(\mathbf{x}, t)$ evaluated at the same t . This jointly ensures that our intermediate sample at t is approximately distributed according to $p_t(\mathbf{x})$. As shown in Chapter 7, PC sampling often outperforms numerical solvers for the reverse-time SDE, especially when the forward SDE in Eq. (8.1) is a VE SDE. In order to use PC samplers for inverse problem solving, our modification is similar to the change made in Algorithm 8.2 for Algorithm 8.1. Specifically, we run line 4 & 5 in Algorithm 8.2 before every corrector or predictor step.

When comparing our approach to previous methods with score-based generative models, we use the same score model to isolate the confounding factors in model training and architecture design. Moreover, we make sure the total cost of sampling is comparable across different methods. For the ALD sampler used in [Jal+21], we use 700 noise scales with 3 steps of Langevin dynamics per noise scale, resulting in a total of $700 \times 3 = 2100$ steps that require score function evaluation. For the PC sampler, we use 1000 noise scales and 1 step of Langevin dynamics per noise scale, totalling $1000 + 1000 = 2000$ steps of score model evaluation.

For PC samplers, the step size ϵ in Langevin dynamics is determined by a signal-to-noise ratio η . For all methods, we tune η and λ in Eq. (8.8) with 100 steps of Bayesian optimization on a validation dataset, and report the results on the test dataset with the optimal parameters. We use the [ax-platform](#) toolkit for Bayesian optimization. The optimal parameters in our experiments are given by

- Sparse-view CT on LIDC 320×320 : $\eta = 0.246$, $\lambda = 0.841$.
- Metal artifact removal on LIDC 320×320 : $\eta = 0.209$, $\lambda = 0.227$.
- Sparse-view CT on LDCT 512×512 : $\eta = 0.4$, $\lambda = 0.72$.
- Accelerated MRI on BraTS 240×240 : $\eta = 0.577$, $\lambda = 0.982$.

B.6.5 Training Details of Baseline Models

Baseline Models for Sparse-View CT Reconstruction

FBP Filtered back projection (FBP) is a standard way for CT image reconstruction, which simply put the projections (sinogram) back to the image space based on the corresponding projection angles and geometry to get an approximated estimation of the unknown image. Usually, a high-pass filter, ramp filter is used to eliminate the blurring during this process. In our experiments, we conduct FBP on sparse-view sinograms using the torch radon toolbox [Ron20].

FISTA-TV FISTA-TV is a fast iterative shrinkage-thresholding algorithm (FISTA) for solving linear inverse problems in image processing [BT09]. It adopts a total variation (TV) term as the regularization in the optimization procedure. Each optimization iteration involves a matrix-vector multiplication followed by a shrinkage-threshold step. In experiments, FISTA is implemented using the tomobar toolbox [KW20] with the regularization using the CCPi regularisation toolkit [Kaz+19]. We run 300 iterations for reconstructing each CT image with regularization parameter 0.001. Considering the nature of iterative reconstruction in FISTA, it is quite natural to generalize this method to different number of projections for reconstructing CT images. In experiments of generalizing to different number of measurements, FISTA method takes as input the sinogram with different numbers of projections and the corresponding angles for these input projections for the iterative procedure.

cGAN Conventional iterative CT reconstruction algorithms like FISTA are typically slow due to their iterative nature. Ghani and Karl [GK18] proposed to cast sparse-view CT reconstruction as a sinogram inpainting problem. Specifically, it used a conditional generative adversarial network (cGAN) to first complete the sinogram data prior to reconstructing CT images, thereby avoiding the costly iterative tomographic processing. However, the imperfect sinogram inpainting may further cause image artifacts. Specifically, cGAN model takes zero-padded sparse-view sinogram with 23 projections as input and generates the completed full-angle sinogram with 180 projections. The cGAN model was implemented using PyTorch [Pas+19] and trained using a batchsize of 64 and learning rate of 0.0001 with 50 epochs in total. In experiments of generalizing to different number of measurements, we deployed the trained cGAN model by zero-padding sparse-view sinogram with different numbers of projections to full-view sinogram as the input. After obtaining the output inpainted sinogram, we replace the corresponding projections in the output based on the ground truth projections in the input. Finally, the images were reconstructed from the overlayed sinogram. Note that we trained the model using 23 projections and tested it on other projection settings to evaluate the generalization.

SIN-4c-PRN To further reduce the artifacts in both sinogram and image space, SIN-4c-PRN [Wei+20] proposed a two-step sparse-view CT reconstruction model. It involves a sinogram inpainting network (SIN) to generate super-resolved sinograms with different number of projections, and then a post-processing refining network (PRN) to further remove image artifacts. Both networks are connected through a filtered back-projection operation (FBP). Specifically, SIN model takes 23-view sinogram as input to firstly upsample to full-view sinogram and then generate sinograms through network for 23, 45, 90, 180 projections respectively. FBP transforms these generated sinograms to image space, which was then concatenated and feed into PRN model for refinement. The framework was implemented using PyTorch [Pas+19] while FBP operation was implemented using . SIN model was trained using a batchsize of 20 and learning rate of 0.0001, while PRN model was trained using a batchsize of 15 and learning rate of 0.0001. Considering that LIDC dataset is much larger than LDCT dataset, the SIN-4c-PRN model was trained for 30 epochs on LIDC dataset and 50 epochs on LDCT dataset. To deploy the trained SIN model to different numbers of measurements, the sinograms with various number of projections are taken as the input for SIN model to generate multi-view sinograms, which were also overlayed with

corresponding ground truth projections in inputs. The generated multi-view sinograms are then used for PRN model inference. Since SIN-4c-PRN model involves the dual-domain learning in both sinogram and image spaces to remove artifacts, and generates multi-scale sinograms during sinogram inpainting, it shows a better generalization to different numbers of measurements compared with cGAN model as shown in Fig. 8.5 and Fig. B.52.

Neumann Meanwhile, in another parallel direction, researchers proposed to learn the regularizer used in optimization from training data, outperforming traditional regularizers. Specifically, Gilton et al. [GOW19] presented an end-to-end, data-driven method for learning a nonlinear regularizer for solving inverse problems inspired by the Neumann series, called Neumann network. Neumann network was implemented using PyTorch [Pas+19]. Due to GPU memory constraints, the model training used the batchsize of 5 on LIDC dataset and the batchsize of 2 on LDCT dataset. The initial learning rate was 0.00001 with an exponential learning rate decay. The network was trained with 15 training epochs on both datasets.

Baseline Models for Undersampled MRI Reconstruction

DuDoRNet Zhou and Zhou [ZZ20] proposed a dual domain recurrent network (DuDoRNet) to simultaneously recover k-space data and images for MRI reconstruction, in order to address aliasing artifacts in both frequency and image domains. The original model in [ZZ20] also embedded a deep T1 prior to make use of fully-sampled short protocol (T1) as complementary information. For a fair comparison with other supervised learning approaches, in our experiments, we do not include this additional information but train the DuDoRNet model without T1 prior. The DuDoRNet was trained using a batchsize of 6 and a learning rate of 0.0005 with 5 training epochs. In experiments of generalizing to different number of measurements, we trained the model with an acceleration factor of 8 and deployed the trained model to other acceleration factors during testing. Specifically, for inference, we use different Cartesian masking function corresponding to different acceleration factors or down-sampling ratios to sub-sample the k-space data for the network input with the corresponding initial reconstructed image with zero-padding k-space.

Cascade DenseNet To reconstruct de-aliased MR images from under-sampled k-space data, Zheng et al. [ZFZ19] proposed a cascaded dilated dense network (CDDN) for MRI

reconstruction, based on stacked dense blocks with residual connections while using the zero-filled MR image as inputs. Specifically, they used a two-step data consistency layer for k-space correction, and replaced corresponding phase-coding lines of the generated image with the original sampled k-space data after each block. In experiments, we trained the model using a batchsize of 8 and a learning rate of 0.0001, with 5 epochs on BraTS dataset. In experiments of generalizing to different number of measurements, we trained the model with an acceleration factor of 8 and deployed the trained model to other acceleration factors during testing. Similarly, different masking functions corresponding to different acceleration factors were used to sub-sample k-space data to get network inputs. From results, we observe that Cascaded DenseNet generalizes better to more measurements than DuDoRNet as shown in Fig. 8.5 and Fig. B.52.

Baseline Models for Metal Artifact Removal

LI One straightforward way for reducing metal artifacts is to complete or inpaint the metal-affected missing regions in sinogram directly through linear interpolation [KHE87]. This method does not need any network training. However, the imperfect completion of sinogram may introduce secondary artifacts to the reconstructed image. In our experiments setting, to fit for the practical applications in real world, we assume the ground truth metal trace and mask information are unknown, which can only be estimated by a rough thresholding in artifacts-affected images. We use the estimated metal mask and metal trace for linear interpolation baseline.

cGANMAR Wang et al. [Wan+18] proposed a conditional generative adversarial network (cGAN)-based approach for metal artifacts reduction (MAR) in CT. Specifically, cGANMAR network learns the mapping directly from the artifacts-affected CTs to artifacts-free CTs through refinement in image space. The cGANMAR model was implemented using PyTorch [Pas+19] and was trained with the batchsize of 64 and the learning rate of 0.0001. The network was trained with 400 epochs.

SNMAR Yu et al. [Yu+20] proposed a sinogram completion neural network (SinoNet) to recover the metal-affected projections. Especially, it leveraged the learning in both sinogram domain and image domain by using a prior network to generate a good prior image to guide sinogram learning. Note that in original setting, SNMAR required linear

interpolated sinogram and CT as inputs and used ground truth metal trace and mask information to generated them. But in our method, we assume the ground truth metal trace and mask information are unknown according to practical scenario and estimate it by a rough thresholding, which will introduce estimation errors. In SNMAR experiments, we still follow the original setting to guarantee the best performance of this baseline method for a strong comparison. We trained the SNMAR using the batchsize of 64 and the learning rate of 0.0001, with a total of 100 training epochs.

B.7 Chapter 9

B.7.1 Experimental Details

Datasets All our experiments are performed on two image datasets: CIFAR-10 [KNH14] and down-sampled ImageNet [VKK16]. Both contain images of resolution 32×32 . CIFAR-10 has 50000 images as the training set and 10000 images as the test set. Down-sampled ImageNet has 1281149 training images and 49999 test images. It is well-known that ImageNet contains some personal sensitive information and may cause privacy concern [Yan+21]. We minimize this risk by using the dataset with a small resolution (32×32).

Model architectures Our variational dequantization model, $q_\phi(\mathbf{u} \mid \mathbf{x})$, follows the same architecture of Flow++ [Ho+19]. We do not use dropout for score-based models trained on ImageNet. We did not tune model architectures or training hyper-parameters specifically for maximizing likelihoods. All likelihood values were reported using the last checkpoint of each setting.

Training We follow the same training procedure for score-based models in Chapter 7. We also use the same hyperparameters for training the variational dequantization model, except that we train it for only 300000 iterations while fixing the score-based model. All models are trained on Cloud TPU v3-8 (roughly equivalent to 4 Tesla V100 GPUs). The baseline DDPM++ model requires around 33 hours to finish training, while the deep DDPM++ model requires around 44 hours. The variational dequantization model for the former requires around 7 hours to train, and for the latter it requires around 9.5 hours.

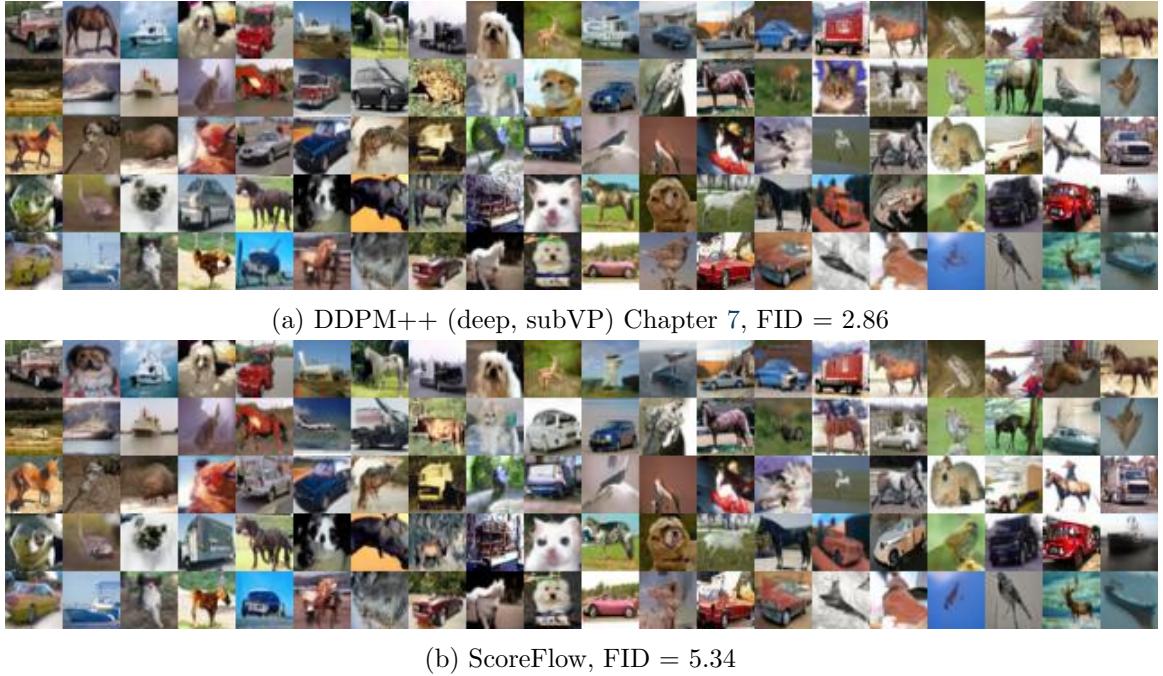


Figure B.55: Samples on CIFAR-10. (a) Model with the best FID. (b) ScoreFlow trained with likelihood weighting + importance sampling + VP SDE. Samples of both models are generated with the same random seed.

Confidence intervals All likelihood values are obtained by averaging the results on around 50000 data points, sampled with replacement from the test dataset. We can compute the confidence intervals with Student’s t-test. On CIFAR-10, the radius of 95% confidence intervals is typically around 0.006 bits/dim, while on ImageNet it is around 0.008 bits/dim.

Sample quality All FID values are computed on 50000 samples from p_{θ}^{ODE} , generated with numerical ODE solvers as in Chapter 7. We compute FIDs between samples and training/test data for CIFAR-10/ImageNet. Although likelihood weighting + importance sampling slightly increases FID scores, their samples have comparable visual quality, as demonstrated in Figs. B.55 and B.56.

B.7.2 Numerical Stability

In our previous theoretical discussion, we always assume that data are perturbed with an SDE starting from $t = 0$. However, in practical implementations, $t = 0$ often leads to numerical instability. As a pragmatic solution, we choose a small non-zero starting time



Figure B.56: Samples on ImageNet 32×32 . (a) Model with the best FID. (b) ScoreFlow trained with likelihood weighting + importance sampling + VP SDE. Samples of both models are generated with the same random seed.

$\epsilon > 0$, and consider the SDE in the time horizon $[\epsilon, T]$. Using the same proof techniques, we can easily see that when the time horizon is $[\epsilon, T]$ instead of $[0, T]$, the original bound in Theorem 9.1,

$$\begin{aligned} D_{\text{KL}}(p \parallel p_{\theta}^{\text{SDE}}) &\leq \mathcal{J}_{\text{SM}}(\boldsymbol{\theta}; g(\cdot)^2) + D_{\text{KL}}(p_T \parallel \pi) \\ &= \frac{1}{2} \int_0^T \mathbb{E}_{p_t(\mathbf{x})}[g(t)^2 \|\nabla_{\mathbf{x}} \log p_t(\mathbf{x}) - \mathbf{s}_{\theta}(\mathbf{x}, t)\|_2^2] dt + D_{\text{KL}}(p_T \parallel \pi) \end{aligned}$$

shall be replaced with

$$D_{\text{KL}}(\tilde{p} \parallel \tilde{p}_{\theta}^{\text{SDE}}) \leq \frac{1}{2} \int_{\epsilon}^T \mathbb{E}_{p_t(\mathbf{x})}[g(t)^2 \|\nabla_{\mathbf{x}} \log p_t(\mathbf{x}) - \mathbf{s}_{\theta}(\mathbf{x}, t)\|_2^2] dt + D_{\text{KL}}(p_T \parallel \pi) \quad (\text{B.38})$$

where $\tilde{p}(\mathbf{x}) := \int p(\tilde{\mathbf{x}}) p_{0\epsilon}(\mathbf{x} \mid \tilde{\mathbf{x}}) d\mathbf{x}$, and $\tilde{p}_{\theta}^{\text{SDE}}$ denotes the marginal distribution of $\hat{\mathbf{x}}_{\theta}(\epsilon)$. Here the stochastic process $\{\hat{\mathbf{x}}_{\theta}(t)\}_{t \in [0, T]}$ is defined according to Eq. (9.5). When ϵ is sufficiently small, we always have

$$D_{\text{KL}}(\tilde{p} \parallel \tilde{p}_{\theta}^{\text{SDE}}) \approx D_{\text{KL}}(p \parallel p_{\theta}^{\text{SDE}}),$$

so we train with Eq. (B.38) to approximately maximize the model likelihood for p_{θ}^{SDE} . However, at test time, we should report the likelihood bound for p_{θ}^{SDE} for mathematical rigor, not $\tilde{p}_{\theta}^{\text{SDE}}$. To this end, we first derive an analogous result to Theorem 9.3 with the time horizon $[\epsilon, T]$, given as below.

Theorem B.1. *Let $p_{0t}(\mathbf{x}' \mid \mathbf{x})$ denote the transition distribution from $p_0(\mathbf{x})$ to $p_t(\mathbf{x})$ for the SDE in Eq. (9.1). With the same notations and conditions in Theorem 9.3, as well as the definitions of \tilde{p} and $\tilde{p}_{\theta}^{\text{SDE}}$ given above, we have*

$$-\mathbb{E}_{p_{0\epsilon}(\mathbf{x}' \mid \mathbf{x})}[\log \tilde{p}_{\theta}^{\text{SDE}}(\mathbf{x}')] \leq \mathcal{L}_{\theta}^{\text{SM}}(\mathbf{x}, \epsilon) = \mathcal{L}_{\theta}^{\text{DSM}}(\mathbf{x}, \epsilon), \quad (\text{B.39})$$

where $\mathcal{L}_{\theta}^{\text{SM}}(\mathbf{x}, \epsilon)$ is defined as

$$\begin{aligned} -\mathbb{E}_{p_{0T}(\mathbf{x}' \mid \mathbf{x})}[\log \pi(\mathbf{x}')] + \frac{1}{2} \int_{\epsilon}^T \mathbb{E}_{p_{0t}(\mathbf{x}' \mid \mathbf{x})} &\left[2g(t)^2 \nabla_{\mathbf{x}'} \cdot \mathbf{s}_{\theta}(\mathbf{x}', t) \right. \\ &\left. + g(t)^2 \|\mathbf{s}_{\theta}(\mathbf{x}', t)\|_2^2 - 2\nabla_{\mathbf{x}'} \cdot \mathbf{f}(\mathbf{x}', t) \right] dt, \end{aligned}$$

and $\mathcal{L}_{\theta}^{DSM}(\mathbf{x}, \epsilon)$ is given by

$$\begin{aligned} & -\mathbb{E}_{p_{0T}(\mathbf{x}'|\mathbf{x})}[\log \pi(\mathbf{x}')]+\frac{1}{2} \int_{\epsilon}^T \mathbb{E}_{p_{0t}(\mathbf{x}'|\mathbf{x})}\left[g(t)^2\left\|\mathbf{s}_{\theta}(\mathbf{x}', t)-\nabla_{\mathbf{x}'} \log p_{0t}(\mathbf{x}'|\mathbf{x})\right\|_2^2\right] \mathrm{d} t \\ & \quad-\frac{1}{2} \int_{\epsilon}^T \mathbb{E}_{p_{0t}(\mathbf{x}'|\mathbf{x})}\left[g(t)^2\left\|\nabla_{\mathbf{x}'} \log p_{0t}(\mathbf{x}'|\mathbf{x})\right\|_2^2+2 \nabla_{\mathbf{x}'} \cdot \mathbf{f}(\mathbf{x}', t)\right] \mathrm{d} t . \end{aligned}$$

Proof. The proof closely parallels that of Theorem 9.3, by noting that $\tilde{p}(\mathbf{x})=\int p(\mathbf{x}') p_{0 \epsilon}(\mathbf{x}|\mathbf{x}') \mathrm{d} \mathbf{x}'$. \square

Although $\tilde{p}_{\theta}^{\text {SDE }}$ is a probabilistic model for \tilde{p} , we can transform it into a probabilistic model for p leveraging a denoising distribution $q_{\theta}(\mathbf{x}|\mathbf{x}')$ that approximately converts \tilde{p} to p . Suppose $p_{0 \epsilon}(\mathbf{x}'|\mathbf{x})=\mathcal{N}(\mathbf{x}'|\alpha \mathbf{x}, \beta^2 \mathbf{I})$. Inspired by Tweedie's formula, we choose

$$q_{\theta}(\mathbf{x}|\mathbf{x}'):=\mathcal{N}\left(\mathbf{x} \mid \frac{\mathbf{x}'}{\alpha}+\frac{\beta^2}{\alpha} \mathbf{s}_{\theta}(\mathbf{x}', \epsilon), \frac{\beta^2}{\alpha^2} \mathbf{I}\right),$$

and define $p_{\theta}(\mathbf{x}):=\int q_{\theta}(\mathbf{x}|\mathbf{x}') \tilde{p}_{\theta}^{\text {SDE }}(\mathbf{x}') \mathrm{d} \mathbf{x}'$, which is a probabilistic model for p . With slight abuse of notation, we identify p_{θ} with $p_{\theta}^{\text {SDE }}$ in Table 9.2. With Jensen's inequality, we have

$$-\log p_{\theta}(\mathbf{x}) \leqslant-\mathbb{E}_{p_{0 \epsilon}(\mathbf{x}'|\mathbf{x})}\left[\log \frac{q_{\theta}(\mathbf{x}|\mathbf{x}') \tilde{p}_{\theta}^{\text {SDE }}(\mathbf{x}')}{p_{0 \epsilon}(\mathbf{x}'|\mathbf{x})}\right].$$

Combined with Theorem B.1, we have

$$-\log p_{\theta}(\mathbf{x}) \leqslant-\mathbb{E}_{p_{0 \epsilon}(\mathbf{x}'|\mathbf{x})}[\log q_{\theta}(\mathbf{x}|\mathbf{x}')-\log p_{0 \epsilon}(\mathbf{x}'|\mathbf{x})]+\mathcal{L}_{\theta}^{\text {SM }}(\mathbf{x}, \epsilon) \quad(B.40)$$

$$=-\mathbb{E}_{p_{0 \epsilon}(\mathbf{x}'|\mathbf{x})}[\log q_{\theta}(\mathbf{x}|\mathbf{x}')-\log p_{0 \epsilon}(\mathbf{x}'|\mathbf{x})]+\mathcal{L}_{\theta}^{DSM}(\mathbf{x}, \epsilon) \quad(B.41)$$

The above bound Eq. (B.41) was applied to both computing the test-time likelihood bounds in Table 9.2, and training the flow model used in variational dequantization. Note that it was not used to train the time-dependent score-based model.

In practice, we choose $\epsilon=10^{-5}$ for VP SDEs and $\epsilon=10^{-2}$ for subVP SDEs, except that on ImageNet we use $\epsilon=5 \times 10^{-5}$ for VP SDE models trained with likelihood weighting and importance sampling. Note that Chapter 7 chooses $\epsilon=10^{-5}$ for all cases. We found that when using likelihood weighting and optionally importance sampling, $\epsilon=10^{-5}$ for subVP SDEs can cause stiffness for numerical ODE solvers. In contrast, using $\epsilon=10^{-2}$ for subVP SDEs sidesteps numerical issues without hurting the performance for score-based

models trained with original weightings in Chapter 7. For the bound values in Table 9.2, we draw 1000 time values uniformly in $[\epsilon, T]$ and use them to estimate $\mathcal{L}_{\theta}^{\text{DSM}}$ for each data point, with the same importance sampling technique in Eq. (9.12). We use the correction in Eq. (B.41) and report upper bounds for $-\log p_{\theta}(\mathbf{x})$. For computing the likelihood of p_{θ}^{ODE} , we use the Dormand-Prince RK45 ODE solver [DP80] with absolute and relevant tolerances set to 10^{-5} . We do not use the correction in Eq. (B.41) for p_{θ}^{ODE} , because it is still a valid likelihood for the data distribution even in the time horizon $[\epsilon, T]$.

Below is a related result to bound $\log \tilde{p}_{\theta}^{\text{SDE}}(\mathbf{x})$ directly. We include it here for completeness, though we do not use it for either training or inference in our experiments.

Theorem B.2. *Let $p_{0t}(\mathbf{x}' | \mathbf{x})$ denote the transition distribution from $p_0(\mathbf{x})$ to $p_t(\mathbf{x})$ for the SDE in Eq. (9.1). With the same notations and conditions in Theorem 9.3, as well as the definitions of \tilde{p} and $\tilde{p}_{\theta}^{\text{SDE}}$ in Theorem B.1, we have*

$$-\log \tilde{p}_{\theta}^{\text{SDE}}(\mathbf{x}) \leq \mathcal{L}_{\theta, \epsilon}^{\text{SM}}(\mathbf{x}) = \mathcal{L}_{\theta, \epsilon}^{\text{DSM}}(\mathbf{x}), \quad (\text{B.42})$$

where $\mathcal{L}_{\theta, \epsilon}^{\text{SM}}(\mathbf{x})$ is defined as

$$\begin{aligned} -\mathbb{E}_{p_{\epsilon T}(\mathbf{x}' | \mathbf{x})} [\log \pi(\mathbf{x}')] + \frac{1}{2} \int_{\epsilon}^T \mathbb{E}_{p_{\epsilon t}(\mathbf{x}' | \mathbf{x})} & \left[2g(t)^2 \nabla_{\mathbf{x}'} \cdot \mathbf{s}_{\theta}(\mathbf{x}', t) \right. \\ & \left. + g(t)^2 \|\mathbf{s}_{\theta}(\mathbf{x}', t)\|_2^2 - 2\nabla_{\mathbf{x}'} \cdot \mathbf{f}(\mathbf{x}', t) \right] dt, \end{aligned}$$

and $\mathcal{L}_{\theta, \epsilon}^{\text{DSM}}(\mathbf{x})$ is given by

$$\begin{aligned} -\mathbb{E}_{p_{\epsilon T}(\mathbf{x}' | \mathbf{x})} [\log \pi(\mathbf{x}')] + \frac{1}{2} \int_{\epsilon}^T \mathbb{E}_{p_{\epsilon t}(\mathbf{x}' | \mathbf{x})} & \left[g(t)^2 \|\mathbf{s}_{\theta}(\mathbf{x}', t) - \nabla_{\mathbf{x}'} \log p_{\epsilon t}(\mathbf{x}' | \mathbf{x})\|_2^2 \right] dt \\ & - \frac{1}{2} \int_{\epsilon}^T \mathbb{E}_{p_{\epsilon t}(\mathbf{x}' | \mathbf{x})} \left[g(t)^2 \|\nabla_{\mathbf{x}'} \log p_{\epsilon t}(\mathbf{x}' | \mathbf{x})\|_2^2 + 2\nabla_{\mathbf{x}'} \cdot \mathbf{f}(\mathbf{x}', t) \right] dt. \end{aligned}$$

Proof. Proof closely parallels those of Theorems 9.3 and B.1. \square

Appendix C

Code

Below we provide open-source implementations for methods presented in this dissertation, along with pre-trained checkpoints for many models.

- Chapter 3

https://github.com/ermongroup/sliced_score_matching

- Chapter 4

https://github.com/chenlin9/high_order_dsm

- Chapter 5

<https://github.com/ermongroup/nCSN>

- Chapter 6

<https://github.com/ermongroup/nCSNV2>

- Chapter 7

– JAX implementation: https://github.com/yang-song/score_sde

– PyTorch implementation: https://github.com/yang-song/score_sde_pytorch

- Chapter 8

https://github.com/yang-song/score_inverse_problems

- Chapter 9

https://github.com/yang-song/score_flow

Bibliography

- [AHS85] David H Ackley, Geoffrey E Hinton, and Terrence J Sejnowski. “A learning algorithm for Boltzmann machines”. In: *Cognitive science* 9.1 (1985), pp. 147–169.
- [Ala+16] Guillaume Alain, Yoshua Bengio, Li Yao, Jason Yosinski, Eric Thibodeau-Laufer, Saizheng Zhang, and Pascal Vincent. “GSNs: generative stochastic networks”. In: *Information and Inference* (2016). DOI: [10.1093/imaiai/iaw003](https://doi.org/10.1093/imaiai/iaw003). URL: <http://imaiai.oxfordjournals.org/content/early/2016/03/17/imaiai.iaw003.full.pdf>.
- [AS66] Syed Mumtaz Ali and Samuel D Silvey. “A general class of coefficients of divergence of one distribution from another”. In: *Journal of the Royal Statistical Society: Series B (Methodological)* 28.1 (1966), pp. 131–142.
- [AG03] Eugene L Allgower and Kurt Georg. *Introduction to numerical continuation methods*. SIAM, 2003.
- [AG12] Eugene L Allgower and Kurt Georg. *Numerical continuation methods: an introduction*. Vol. 13. Springer Science & Business Media, 2012.
- [And82] Brian DO Anderson. “Reverse-time diffusion equation models”. In: *Stochastic Processes and their Applications* 12.3 (1982), pp. 313–326.
- [Ant+20] Vegard Antun, Francesco Renna, Clarice Poon, Ben Adcock, and Anders C Hansen. “On instabilities of deep learning in image reconstruction and the potential costs of AI”. In: *Proceedings of the National Academy of Sciences* 117.48 (2020), pp. 30088–30095.

- [ACB17] Martin Arjovsky, Soumith Chintala, and Léon Bottou. “Wasserstein Generative Adversarial Networks”. In: *Proceedings of the 34th International Conference on Machine Learning*. Ed. by Doina Precup and Yee Whye Teh. Vol. 70. Proceedings of Machine Learning Research. International Convention Centre, Sydney, Australia: PMLR, June 2017, pp. 214–223. URL: <http://proceedings.mlr.press/v70/arjovsky17a.html>.
- [Arm+11] Samuel G Armato III, Geoffrey McLennan, Luc Bidaut, Michael F McNitt-Gray, Charles R Meyer, Anthony P Reeves, Binsheng Zhao, Denise R Aberle, Claudia I Henschke, Eric A Hoffman, et al. “The lung image database consortium (LIDC) and image database resource initiative (IDRI): a completed reference database of lung nodules on CT scans”. In: *Medical physics* 38.2 (2011), pp. 915–931.
- [BCB15] Dzmitry Bahdanau, Kyung Hyun Cho, and Yoshua Bengio. “Neural machine translation by jointly learning to align and translate”. In: *3rd International Conference on Learning Representations, ICLR 2015*. 2015.
- [Bak+17] Spyridon Bakas, Hamed Akbari, Aristeidis Sotiras, Michel Bilello, Martin Rozycki, Justin S Kirby, John B Freymann, Keyvan Farahani, and Christos Davatzikos. “Advancing the cancer genome atlas glioma MRI collections with expert segmentation labels and radiomic features”. In: *Scientific data* 4.1 (2017), pp. 1–13.
- [BS18] Shane Barratt and Rishi Sharma. “A note on the inception score”. In: *arXiv preprint arXiv:1801.01973* (2018).
- [Bar86] Andrew R. Barron. “Entropy and the Central Limit Theorem”. In: *Annals of Probability* 14.1 (1986), pp. 336–342.
- [BT09] Amir Beck and Marc Teboulle. “A fast iterative shrinkage-thresholding algorithm for linear inverse problems”. In: *SIAM journal on imaging sciences* 2.1 (2009), pp. 183–202.
- [BN03] Mikhail Belkin and Partha Niyogi. “Laplacian eigenmaps for dimensionality reduction and data representation”. In: *Neural computation* 15.6 (2003), pp. 1373–1396.

- [BB99] Yoshua Bengio and Samy Bengio. “Modeling high-dimensional discrete data with multi-layer neural networks”. In: *Advances in Neural Information Processing Systems* 12 (1999).
- [Ben+13] Yoshua Bengio, Li Yao, Guillaume Alain, and Pascal Vincent. “Generalized denoising auto-encoders as generative models”. In: *Advances in neural information processing systems*. 2013, pp. 899–907.
- [BSM17] David Berthelot, Thomas Schumm, and Luke Metz. “Began: Boundary equilibrium generative adversarial networks”. In: *arXiv preprint arXiv:1703.10717* (2017).
- [BHV17] Florian Bordes, Sina Honari, and Pascal Vincent. “Learning to generate samples from noise through infusion training”. In: *arXiv preprint arXiv:1703.06975* (2017).
- [BBV04] Stephen Boyd, Stephen P Boyd, and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.
- [Bra+18] James Bradbury et al. *JAX: composable transformations of Python+NumPy programs*. Version 0.3.13. 2018. URL: <http://github.com/google/jax>.
- [BDS19] Andrew Brock, Jeff Donahue, and Karen Simonyan. “Large Scale GAN Training for High Fidelity Natural Image Synthesis”. In: *International Conference on Learning Representations*. 2019. URL: <https://openreview.net/forum?id=B1xsqj09Fm>.
- [Bro+20] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. “Language models are few-shot learners”. In: *Advances in neural information processing systems* 33 (2020), pp. 1877–1901.
- [BP07] Giovanni Bussi and Michele Parrinello. “Accurate sampling using Langevin dynamics”. In: *Physical Review E* 75.5 (2007), p. 056707.
- [Buz11] Thorsten M Buzug. “Computed tomography”. In: *Springer handbook of medical technology*. Springer, 2011, pp. 311–342.

- [Cai+20a] Ruojin Cai, Guandao Yang, Hadar Averbuch-Elor, Zekun Hao, Serge Belongie, Noah Snavely, and Bharath Hariharan. “Learning Gradient Fields for Shape Generation”. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2020.
- [Cai+20b] Ruojin Cai, Guandao Yang, Hadar Averbuch-Elor, Zekun Hao, Serge Belongie, Noah Snavely, and Bharath Hariharan. “Learning Gradient Fields for Shape Generation”. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2020.
- [CS06] Stéphane Canu and Alex Smola. “Kernel methods and the exponential family”. In: *Neurocomputing* 69.7 (2006). New Issues in Neurocomputing: 13th European Symposium on Artificial Neural Networks, pp. 714–720. ISSN: 0925-2312.
- [CB21] George Casella and Roger L Berger. *Statistical inference*. Cengage Learning, 2021.
- [CS14] B Chandra and Rajesh Kumar Sharma. “Adaptive noise schedule for denoising autoencoder”. In: *International conference on neural information processing*. Springer. 2014, pp. 535–542.
- [Che+20] Jianfei Chen, Cheng Lu, Biqi Chenli, Jun Zhu, and Tian Tian. “Vflow: More expressive generative flows with variational data augmentation”. In: *International Conference on Machine Learning*. PMLR. 2020, pp. 1660–1669.
- [Che+17] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. “Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs”. In: *IEEE transactions on pattern analysis and machine intelligence* 40.4 (2017), pp. 834–848.
- [Che+21a] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. “Evaluating large language models trained on code”. In: *arXiv preprint arXiv:2107.03374* (2021).
- [Che+21b] Nanxin Chen, Yu Zhang, Heiga Zen, Ron J Weiss, Mohammad Norouzi, and William Chan. “WaveGrad: Estimating Gradients for Waveform Generation”. In: *International Conference on Learning Representations (ICLR)*. 2021.

- [Che+18a] Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. “Neural Ordinary Differential Equations”. In: *Advances in neural information processing systems*. 2018, pp. 6571–6583.
- [CFG14] Tianqi Chen, Emily Fox, and Carlos Guestrin. “Stochastic gradient hamiltonian monte carlo”. In: *International conference on machine learning*. 2014, pp. 1683–1691.
- [Che+18b] Xi Chen, Nikhil Mishra, Mostafa Rohaninejad, and Pieter Abbeel. “Pixelnail: An improved autoregressive generative model”. In: *International Conference on Machine Learning*. PMLR. 2018, pp. 864–872.
- [Chi21] Rewon Child. “Very Deep VAEs Generalize Autoregressive Models and Can Outperform Them on Images”. In: *International Conference on Learning Representations*. 2021. URL: <https://openreview.net/forum?id=RLRXCV6DbEJ>.
- [Chi+19] Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. “Generating long sequences with sparse transformers”. In: *arXiv preprint arXiv:1904.10509* (2019).
- [Cla+13] Kenneth Clark, Bruce Vendt, Kirk Smith, John Freymann, Justin Kirby, Paul Koppel, Stephen Moore, Stanley Phillips, David Maffitt, Michael Pringle, et al. “The Cancer Imaging Archive (TCIA): maintaining and operating a public information repository”. In: *Journal of digital imaging* 26.6 (2013), pp. 1045–1057.
- [CUH15] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. “Fast and accurate deep network learning by exponential linear units (elus)”. In: *arXiv preprint arXiv:1511.07289* (2015).
- [Csi64] Imre Csiszár. “Eine informationstheoretische ungleichung und ihre anwendung auf beweis der ergodizitaet von markoffschen ketten”. In: *Magyer Tud. Akad. Mat. Kutato Int. Koezl.* 8 (1964), pp. 85–108.
- [Dai+17] Zihang Dai, Zhilin Yang, Fan Yang, William W Cohen, and Ruslan Salakhutdinov. “Good semi-supervised learning that requires a bad GAN”. In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. 2017, pp. 6513–6523.

- [DK19] Arnak S Dalalyan and Avetik Karagulyan. “User-friendly guarantees for the Langevin Monte Carlo with inaccurate gradient”. In: *Stochastic Processes and their Applications* 129.12 (2019), pp. 5278–5311.
- [DF08] Sanjoy Dasgupta and Yoav Freund. “Random projection trees and low dimensional manifolds”. In: *Proceedings of the fortieth annual ACM symposium on Theory of computing*. 2008, pp. 537–546.
- [DN21] Prafulla Dhariwal and Alex Nichol. “Diffusion models beat gans on image synthesis”. In: *Advances in Neural Information Processing Systems (NeurIPS)* (2021).
- [DKB15] Laurent Dinh, David Krueger, and Yoshua Bengio. “NICE: Non-linear independent components estimation”. In: *International Conference in Learning Representations Workshop Track* (2015).
- [DSB17] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. “Density estimation using Real NVP”. In: *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL: <https://openreview.net/forum?id=HkpbnH9lx>.
- [DP80] John R Dormand and Peter J Prince. “A family of embedded Runge-Kutta formulae”. In: *Journal of computational and applied mathematics* 6.1 (1980), pp. 19–26.
- [DM19a] Yilun Du and Igor Mordatch. “Implicit Generation and Modeling with Energy Based Models”. In: *Advances in Neural Information Processing Systems*. Ed. by H. Wallach, H. Larochelle, A. Beygelzimer, F. d’Alché-Buc, E. Fox, and R. Garnett. Vol. 32. Curran Associates, Inc., 2019, pp. 3608–3618.
- [DG17] Dheeru Dua and Casey Graff. *UCI Machine Learning Repository*. 2017. URL: <http://archive.ics.uci.edu/ml>.
- [Dua+87] Simon Duane, Anthony D Kennedy, Brian J Pendleton, and Duncan Roweth. “Hybrid monte carlo”. In: *Physics letters B* 195.2 (1987), pp. 216–222.
- [Dud67] Richard M Dudley. “The sizes of compact subsets of Hilbert space and continuity of Gaussian processes”. In: *Journal of Functional Analysis* 1.3 (1967), pp. 290–330.

- [DSK17] Vincent Dumoulin, Jonathon Shlens, and Manjunath Kudlur. “A Learned Representation For Artistic Style”. In: *International Conference on Learning Representations 2017*. 2017. URL: <https://openreview.net/forum?id=BJ0-BuT1g>.
- [DM19b] Alain Durmus and Eric Moulines. “High-dimensional Bayesian inference via the unadjusted Langevin algorithm”. In: *Bernoulli* 25.4A (2019), pp. 2854–2882.
- [Efr11] Bradley Efron. “Tweedie’s formula and selection bias”. In: *Journal of the American Statistical Association* 106.496 (2011), pp. 1602–1614.
- [Erm+13] Stefano Ermon, Carla Gomes, Ashish Sabharwal, and Bart Selman. “Taming the curse of dimensionality: Discrete integration by hashing and optimization”. In: *International Conference on Machine Learning*. PMLR. 2013, pp. 334–342.
- [GM98] Andrew Gelman and Xiao-Li Meng. “Simulating normalizing constants: From importance sampling to bridge sampling to path sampling”. In: *Statistical science* (1998), pp. 163–185.
- [GS14] Krzysztof J Geras and Charles Sutton. “Scheduled denoising autoencoders”. In: *arXiv preprint arXiv:1406.3269* (2014).
- [Ger+15] Mathieu Germain, Karol Gregor, Iain Murray, and Hugo Larochelle. “Made: Masked autoencoder for distribution estimation”. In: *International Conference on Machine Learning*. 2015, pp. 881–889.
- [GK18] Muhammad Usman Ghani and W Clem Karl. “Deep learning-based sinogram completion for low-dose CT”. In: *2018 IEEE 13th Image, Video, and Multidimensional Signal Processing Workshop (IVMSP)*. IEEE. 2018, pp. 1–5.
- [GOW19] Davis Gilton, Greg Ongie, and Rebecca Willett. “Neumann networks for linear inverse problems in imaging”. In: *IEEE Transactions on Computational Imaging* 6 (2019), pp. 328–343.
- [GC11] Mark Girolami and Ben Calderhead. “Riemann manifold langevin and hamiltonian monte carlo methods”. In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 73.2 (2011), pp. 123–214.

- [Goo+14] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. “Generative adversarial nets”. In: *Advances in neural information processing systems*. 2014, pp. 2672–2680.
- [GM15] Jackson Gorham and Lester Mackey. “Measuring sample quality with Stein’s method”. In: *Advances in Neural Information Processing Systems 28* (2015).
- [GM17] Jackson Gorham and Lester Mackey. “Measuring sample quality with kernels”. In: *Proceedings of the 34th International Conference on Machine Learning*. 2017, pp. 1292–1301.
- [Goy+17] Anirudh Goyal Alias Parth Goyal, Nan Rosemary Ke, Surya Ganguli, and Yoshua Bengio. “Variational walkback: Learning a transition operator as a stochastic recurrent net”. In: *Advances in Neural Information Processing Systems*. 2017, pp. 4392–4402.
- [Gra06] Erik W Grafarend. *Linear and nonlinear models: fixed effects, random effects, and mixed models*. de Gruyter, 2006.
- [Gra+18] Will Grathwohl, Ricky TQ Chen, Jesse Bettencourt, Ilya Sutskever, and David Duvenaud. “FFJORD: Free-Form Continuous Dynamics for Scalable Reversible Generative Models”. In: *International Conference on Learning Representations*. 2018.
- [Gra+20] Will Grathwohl, Kuan-Chieh Wang, Joern-Henrik Jacobsen, David Duvenaud, Mohammad Norouzi, and Kevin Swersky. “Your classifier is secretly an energy based model and you should treat it like one”. In: *International Conference on Learning Representations*. 2020. URL: <https://openreview.net/forum?id=Hkxz0NtDB>.
- [Gra13] Alex Graves. “Generating sequences with recurrent neural networks”. In: *arXiv preprint arXiv:1308.0850* (2013).
- [GM94] Ulf Grenander and Michael I Miller. “Representations of knowledge in complex systems”. In: *Journal of the Royal Statistical Society: Series B (Methodological)* 56.4 (1994), pp. 549–581.

- [Gul+17] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. “Improved training of wasserstein gans”. In: *Advances in Neural Information Processing Systems*. 2017, pp. 5767–5777.
- [GH10] Michael Gutmann and Aapo Hyvärinen. “Noise-contrastive estimation: A new estimation principle for unnormalized statistical models”. In: *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. 2010, pp. 297–304.
- [GH11] Michael U Gutmann and Jun-ichiro Hirayama. “Bregman divergence as general framework to estimate unnormalized statistical models”. In: *Proceedings of the Twenty-Seventh Conference on Uncertainty in Artificial Intelligence*. AUAI Press. 2011, pp. 283–290.
- [Ham+] Kerstin Hammernik, Jo Schlemper, Chen Qin, Jinming Duan, Ronald M. Summers, and Daniel Rueckert. “Systematic evaluation of iterative deep neural networks for fast parallel MRI reconstruction with sensitivity-weighted coil combination”. In: *Magnetic Resonance in Medicine* n/a.n/a (). DOI: <https://doi.org/10.1002/mrm.28827>. eprint: <https://onlinelibrary.wiley.com/doi/pdf/10.1002/mrm.28827>. URL: <https://onlinelibrary.wiley.com/doi/abs/10.1002/mrm.28827>.
- [HP86] Ulrich G Haussmann and Etienne Pardoux. “Time reversal of diffusions”. In: *The Annals of Probability* (1986), pp. 1188–1205.
- [Heu+17] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. “GANs trained by a two time-scale update rule converge to a local Nash equilibrium”. In: *Advances in Neural Information Processing Systems*. 2017, pp. 6626–6637.
- [Hin02] Geoffrey E Hinton. “Training products of experts by minimizing contrastive divergence”. In: *Neural computation* 14.8 (2002), pp. 1771–1800.
- [Ho+19] Jonathan Ho, Xi Chen, Aravind Srinivas, Yan Duan, and Pieter Abbeel. “Flow++: Improving Flow-Based Generative Models with Variational Dequantization and Architecture Design”. In: *International Conference on Machine Learning*. 2019, pp. 2722–2730.

- [HE16] Jonathan Ho and Stefano Ermon. “Generative adversarial imitation learning”. In: *Advances in Neural Information Processing Systems*. 2016, pp. 4565–4573.
- [HJA20] Jonathan Ho, Ajay Jain, and Pieter Abbeel. “Denoising Diffusion Probabilistic Models”. In: *Advances in Neural Information Processing Systems 33* (2020).
- [HLA19] Jonathan Ho, Evan Lohn, and Pieter Abbeel. “Compression with Flows via Local Bits-Back Coding”. In: *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*. Ed. by Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d’Alché-Buc, Emily B. Fox, and Roman Garnett. 2019, pp. 3874–3883.
- [Hoo+19] Emiel Hoogeboom, Jorn W. T. Peters, Rianne van den Berg, and Max Welling. “Integer Discrete Flows and Lossless Compression”. In: *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*. Ed. by Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d’Alché-Buc, Emily B. Fox, and Roman Garnett. 2019, pp. 12134–12144. URL: <https://proceedings.neurips.cc/paper/2019/hash/9e9a30b74c49d07d8150c8c83b1ccf07-Abstract.html>.
- [HB17] Xun Huang and Serge Belongie. “Arbitrary style transfer in real-time with adaptive instance normalization”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 1501–1510.
- [Hus17] Ferenc Huszár. “Variational inference using implicit distributions”. In: *arXiv preprint arXiv:1702.08235* (2017).
- [Hut89] Michael F Hutchinson. “A stochastic estimator of the trace of the influence matrix for Laplacian smoothing splines”. In: *Communications in Statistics-Simulation and Computation* 18.3 (1989), pp. 1059–1076.
- [HD05] Aapo Hyvärinen and Peter Dayan. “Estimation of non-normalized statistical models by score matching.” In: *Journal of Machine Learning Research (JMLR)* 6.4 (2005).

- [Jal+21] Ajil Jalal, Marius Arvinte, Giannis Daras, Eric Price, Alexandros G Dimakis, and Jonathan I Tamir. “Robust Compressed Sensing MRI with Deep Generative Priors”. In: *arXiv preprint arXiv:2108.01368* (2021).
- [JB04] Oliver Johnson and Andrew Barron. “Fisher Information inequalities and the Central Limit Theorem”. In: *Probability Theory and Related Fields* 129.3 (2004), pp. 391–409.
- [Jol+21] Alexia Jolicoeur-Martineau, Rémi Piché-Taillefer, Ioannis Mitliagkas, and Remi Tachet des Combes. “Adversarial score matching and improved sampling for image generation”. In: *International Conference on Learning Representations*. 2021.
- [Jun+20] Heewoo Jun, Rewon Child, Mark Chen, John Schulman, Aditya Ramesh, Alec Radford, and Ilya Sutskever. “Distribution Augmentation for Generative Modeling”. In: *International Conference on Machine Learning*. PMLR. 2020, pp. 5006–5019.
- [KS20] Zahra Kadkhodaie and Eero P Simoncelli. “Solving Linear Inverse Problems Using the Prior Implicit in a Denoiser”. In: *arXiv preprint arXiv:2007.13640* (2020).
- [KHE87] Willi A Kalender, Robert Hebel, and Johannes Ebersberger. “Reduction of CT artifacts caused by metallic implants.” In: *Radiology* 164.2 (1987), pp. 576–577.
- [Kar+18] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. “Progressive Growing of GANs for Improved Quality, Stability, and Variation”. In: *International Conference on Learning Representations*. 2018. URL: <https://openreview.net/forum?id=Hk99zCeAb>.
- [Kar+20a] Tero Karras, Miika Aittala, Janne Hellsten, Samuli Laine, Jaakko Lehtinen, and Timo Aila. “Training generative adversarial networks with limited data”. In: *Advances in Neural Information Processing Systems* 33 (2020).
- [KLA19] Tero Karras, Samuli Laine, and Timo Aila. “A style-based generator architecture for generative adversarial networks”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 4401–4410.

- [Kar+20b] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. “Analyzing and improving the image quality of stylegan”. In: 2020.
- [KVE21] Bahjat Kawar, Gregory Vaksman, and Michael Elad. “SNIPS: Solving Noisy Inverse Problems Stochastically”. In: *arXiv preprint arXiv:2105.14951* (2021).
- [Kaz+19] Daniil Kazantsev, Edoardo Pasca, Martin J Turner, and Philip J Withers. “CCPi-Regularisation toolkit for computed tomographic image reconstruction with proximal splitting algorithms”. In: *SoftwareX* 9 (2019), pp. 317–323.
- [KW20] Daniil Kazantsev and Nicola Wadeson. “TOmographic MOdel-BAsed Reconstruction (ToMoBAR) software for high resolution synchrotron X-ray tomography”. In: *CT Meeting* (2020).
- [KB14] Diederik P Kingma and Jimmy Ba. “Adam: A method for stochastic optimization”. In: *arXiv preprint arXiv:1412.6980* (2014).
- [Kin+21] Diederik P Kingma, Tim Salimans, Ben Poole, and Jonathan Ho. “Variational Diffusion Models”. In: *arXiv preprint arXiv:2107.00630* (2021).
- [KW14] Diederik P. Kingma and Max Welling. “Auto-Encoding Variational Bayes”. In: *International Conference on Learning Representations*. 2014.
- [KD18] Durk P Kingma and Prafulla Dhariwal. “Glow: Generative Flow with Invertible 1x1 Convolutions”. In: *Advances in Neural Information Processing Systems 31*. Ed. by S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett. Curran Associates, Inc., 2018, pp. 10215–10224. URL: <http://papers.nips.cc/paper/8224-glow-generative-flow-with-invertible-1x1-convolutions.pdf>.
- [KL10] Durk P Kingma and Yann LeCun. “Regularized estimation of image statistics by score matching”. In: *Advances in neural information processing systems 23* (2010).
- [Kin+14] Durk P Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. “Semi-supervised learning with deep generative models”. In: *Advances in neural information processing systems*. 2014, pp. 3581–3589.

- [Kin+16] Durk P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. “Improved variational inference with inverse autoregressive flow”. In: *Advances in neural information processing systems* 29 (2016).
- [KGV83] Scott Kirkpatrick, C Daniel Gelatt Jr, and Mario P Vecchi. “Optimization by simulated annealing”. In: *science* 220.4598 (1983), pp. 671–680.
- [KP13] Peter E Kloeden and Eckhard Platen. *Numerical solution of stochastic differential equations*. Vol. 23. Springer Science & Business Media, 2013.
- [Kno+20] Florian Knoll, Jure Zbontar, Anuroop Sriram, Matthew J Muckley, Mary Bruno, Aaron Defazio, Marc Parente, Krzysztof J Geras, Joe Katsnelson, Hersh Chandarana, et al. “fastMRI: A publicly available raw k-space and DICOM dataset of knee images for accelerated MR image reconstruction using machine learning”. In: *Radiology: Artificial Intelligence* 2.1 (2020), e190007.
- [KF09] Daphne Koller and Nir Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.
- [Kon+20] Zhifeng Kong, Wei Ping, Jiaji Huang, Kexin Zhao, and Bryan Catanzaro. “DiffWave: A Versatile Diffusion Model for Audio Synthesis”. In: *arXiv preprint arXiv:2009.09761* (2020).
- [KH+09] Alex Krizhevsky, Geoffrey Hinton, et al. “Learning multiple layers of features from tiny images”. In: (2009).
- [KNH14] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. “The CIFAR-10 Dataset”. In: *online: http://www.cs.toronto.edu/kriz/cifar.html* 55 (2014).
- [KL51] Solomon Kullback and Richard A Leibler. “On information and sufficiency”. In: *The annals of mathematical statistics* 22.1 (1951), pp. 79–86.
- [LBH15] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. “Deep learning”. In: *nature* 521.7553 (2015), pp. 436–444.
- [LeC+06] Yann LeCun, Sumit Chopra, Raia Hadsell, M Ranzato, and F Huang. “A tutorial on energy-based learning”. In: *Predicting structured data* 1.0 (2006).
- [Léo14] Christian Léonard. “Some properties of path measures”. In: *Séminaire de Probabilités XLVI*. Springer, 2014, pp. 207–230.

- [LZT22] Ruilin Li, Hongyuan Zha, and Molei Tao. “Sqrt(d) Dimension Dependence of Langevin Monte Carlo”. In: *International Conference on Learning Representations*. 2022. URL: https://openreview.net/forum?id=5-2mX9_U5i.
- [Li+20] Xuechen Li, Ting-Kam Leonard Wong, Ricky T. Q. Chen, and David Duvenaud. “Scalable Gradients for Stochastic Differential Equations”. In: *Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics*. 2020.
- [LT18] Yingzhen Li and Richard E. Turner. “Gradient Estimators for Implicit Models”. In: *International Conference on Learning Representations*. 2018.
- [LCS19] Zengyi Li, Yubei Chen, and Friedrich T Sommer. “Learning energy-based models in high-dimensional spaces with multi-scale denoising score matching”. In: *arXiv* (2019), arXiv–1910.
- [Lin+17] Guosheng Lin, Anton Milan, Chunhua Shen, and Ian Reid. “Refinenet: Multi-path refinement networks for high-resolution semantic segmentation”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 1925–1934.
- [LLJ16] Qiang Liu, Jason Lee, and Michael Jordan. “A kernelized Stein discrepancy for goodness-of-fit tests”. In: *International conference on machine learning*. PMLR. 2016, pp. 276–284.
- [Liu+15] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. “Deep Learning Face Attributes in the Wild”. In: *Proceedings of International Conference on Computer Vision (ICCV)*. 2015.
- [Lyu09] Siwei Lyu. “Interpretation and Generalization of Score Matching”. In: *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence*. 2009, pp. 359–366.
- [MRO20] Dimitra Maoutsa, Sebastian Reich, and Manfred Opper. “Interacting particle solutions of Fokker-Planck equations through gradient-log-density estimation”. In: *arXiv preprint arXiv:2006.00702* (2020).
- [Mar+18] Morteza Mardani, Enhao Gong, Joseph Y Cheng, Shreyas S Vasanawala, Greg Zaharchuk, Lei Xing, and John M Pauly. “Deep generative adversarial neural networks for compressive sensing MRI”. In: *IEEE transactions on medical imaging* 38.1 (2018), pp. 167–179.

- [MG15] James Martens and Roger Grosse. “Optimizing neural networks with kronecker-factored approximate curvature”. In: *International conference on machine learning*. PMLR. 2015, pp. 2408–2417.
- [MSS12] James Martens, Ilya Sutskever, and Kevin Swersky. “Estimating the hessian by back-propagating curvature”. In: *Proceedings of the 29th International Conference on International Conference on Machine Learning*. 2012, pp. 963–970.
- [Mar+15] Martín Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: <https://www.tensorflow.org/>.
- [Men+21] Chenlin Meng, Yang Song, Wenzhe Li, and Stefano Ermon. “Estimating high order gradients of the data distribution by denoising”. In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 25359–25369.
- [Men+14] Bjoern H Menze, Andras Jakab, Stefan Bauer, Jayashree Kalpathy-Cramer, Keyvan Farahani, Justin Kirby, Yuliya Burren, Nicole Porz, Johannes Slotboom, Roland Wiest, et al. “The multimodal brain tumor image segmentation benchmark (BRATS)”. In: *IEEE transactions on medical imaging* 34.10 (2014), pp. 1993–2024.
- [Met+16] Luke Metz, Ben Poole, David Pfau, and Jascha Sohl-Dickstein. “Unrolled generative adversarial networks”. In: *arXiv preprint arXiv:1611.02163* (2016).
- [Miy61] KOICHI Miyasawa. “An empirical Bayes estimator of the mean of a normal population”. In: *Bull. Inst. Internat. Statist* 38.181-188 (1961), pp. 1–2.
- [Miy+18] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. “Spectral Normalization for Generative Adversarial Networks”. In: *International Conference on Learning Representations*. 2018. URL: <https://openreview.net/forum?id=B1QRgziT->.
- [MG14] Andriy Mnih and Karol Gregor. “Neural variational inference and learning in belief networks”. In: *International Conference on Machine Learning*. PMLR. 2014, pp. 1791–1799.

- [Moe+21] Taylor R Moen, Baiyu Chen, David R Holmes III, Xinhui Duan, Zhicong Yu, Lifeng Yu, Shuai Leng, Joel G Fletcher, and Cynthia H McCollough. “Low-dose CT image and projection dataset”. In: *Medical physics* 48.2 (2021), pp. 902–911.
- [Mou+19] Wenlong Mou, Yi-An Ma, Martin J Wainwright, Peter L Bartlett, and Michael I Jordan. “High-order Langevin diffusion yields an accelerated MCMC algorithm”. In: *arXiv preprint arXiv:1908.10859* (2019).
- [Mül97] Alfred Müller. “Integral probability metrics and their generating classes of functions”. In: *Advances in Applied Probability* 29.2 (1997), pp. 429–443.
- [NM10] Hariharan Narayanan and Sanjoy Mitter. “Sample complexity of testing the manifold hypothesis”. In: *Proceedings of the 23rd International Conference on Neural Information Processing Systems- Volume 2*. 2010, pp. 1786–1794.
- [Nea01] Radford M Neal. “Annealed importance sampling”. In: *Statistics and computing* 11.2 (2001), pp. 125–139.
- [Nea+11] Radford M Neal et al. “MCMC using Hamiltonian dynamics”. In: *Handbook of markov chain monte carlo* 2.11 (2011), p. 2.
- [Nes03] Yurii Nesterov. *Introductory lectures on convex optimization: A basic course*. Vol. 87. Springer Science & Business Media, 2003.
- [Ngi+11] Jiquan Ngiam, Zhenghao Chen, Pang W Koh, and Andrew Y Ng. “Learning deep energy models”. In: *Proceedings of the 28th international conference on machine learning (ICML-11)*. 2011, pp. 1105–1112.
- [ND21] Alex Nichol and Prafulla Dhariwal. “Improved denoising diffusion probabilistic models”. In: *arXiv preprint arXiv:2102.09672* (2021).
- [Nic+21] Alex Nichol, Prafulla Dhariwal, Aditya Ramesh, Pranav Shyam, Pamela Mishkin, Bob McGrew, Ilya Sutskever, and Mark Chen. “Glide: Towards photorealistic image generation and editing with text-guided diffusion models”. In: *arXiv preprint arXiv:2112.10741* (2021).
- [Nij+19] Erik Nijkamp, Mitch Hill, Tian Han, Song-Chun Zhu, and Ying Nian Wu. “On the Anatomy of MCMC-based Maximum Likelihood Learning of Energy-Based Models”. In: *arXiv preprint arXiv:1903.12370* (2019).

- [Nij+22] Erik Nijkamp, Bo Pang, Hiroaki Hayashi, Lifu Tu, Huan Wang, Yingbo Zhou, Silvio Savarese, and Caiming Xiong. “A conversational paradigm for program synthesis”. In: *arXiv preprint arXiv:2203.13474* (2022).
- [Niu+20] Chenhao Niu, Yang Song, Jiaming Song, Shengjia Zhao, Aditya Grover, and Stefano Ermon. “Permutation invariant graph generation via score-based generative modeling”. In: *International Conference on Artificial Intelligence and Statistics (AISTATS)*. 2020.
- [NCT16] Sebastian Nowozin, Botond Cseke, and Ryota Tomioka. “f-gan: Training generative neural samplers using variational divergence minimization”. In: *Advances in neural information processing systems*. 2016, pp. 271–279.
- [Oks13] Bernt Oksendal. *Stochastic differential equations: an introduction with applications*. Springer Science & Business Media, 2013.
- [Oor+16a] Aäron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alexander Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. “WaveNet: A Generative Model for Raw Audio”. In: *Arxiv*. 2016. URL: <https://arxiv.org/abs/1609.03499>.
- [Oor+16b] Aäron van den Oord, Nal Kalchbrenner, Oriol Vinyals, Lasse Espeholt, Alex Graves, and Koray Kavukcuoglu. “Conditional Image Generation with PixelCNN Decoders”. In: *Proceedings of the 30th International Conference on Neural Information Processing Systems*. NIPS’16. Barcelona, Spain: Curran Associates Inc., 2016, pp. 4797–4805. ISBN: 978-1-5108-3881-9. URL: <http://dl.acm.org/citation.cfm?id=3157382.3157633>.
- [Ost+17] Georg Ostrovski, Marc G Bellemare, Aäron van den Oord, and Rémi Munos. “Count-based exploration with neural density models”. In: *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org. 2017, pp. 2721–2730.
- [ODM18] Georg Ostrovski, Will Dabney, and Rémi Munos. “Autoregressive Quantile Networks for Generative Modeling”. In: *ICML*. Vol. 80. Proceedings of Machine Learning Research. PMLR, 2018, pp. 3933–3942.
- [Owe13] Art B. Owen. *Monte Carlo theory, methods and examples*. 2013.

- [Pan+20] Tianyu Pang, Taufik Xu, Chongxuan Li, Yang Song, Stefano Ermon, and Jun Zhu. “Efficient Learning of Generative Models via Finite-Difference Score Matching”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2020.
- [Pap+19] George Papamakarios, Eric Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. “Normalizing Flows for Probabilistic Modeling and Inference”. In: *arXiv preprint arXiv:1912.02762* (2019).
- [PPM17] George Papamakarios, Theo Pavlakou, and Iain Murray. “Masked autoregressive flow for density estimation”. In: *Advances in neural information processing systems* 30 (2017).
- [Par81] Giorgio Parisi. “Correlation functions and computer simulations”. In: *Nuclear Physics B* 180.3 (1981), pp. 378–384.
- [Par+18] Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Noam Shazeer, Alexander Ku, and Dustin Tran. “Image transformer”. In: *International Conference on Machine Learning*. PMLR. 2018, pp. 4055–4064.
- [Pas+17] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. “Automatic differentiation in pytorch”. In: (2017).
- [Pas+19] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. “PyTorch: An imperative style, high-performance deep learning library”. In: *Advances in Neural Information Processing Systems*. 2019, pp. 8024–8035.
- [Pop+21] Vadim Popov, Ivan Vovk, Vladimir Gogoryan, Tasnima Sadekova, and Mikhail Kudinov. “Grad-TTS: A Diffusion Probabilistic Model for Text-to-Speech”. In: *arXiv preprint arXiv:2105.06337* (2021).
- [Rab+11] Julien Rabin, Gabriel Peyré, Julie Delon, and Marc Bernot. “Wasserstein barycenter and its application to texture mixing”. In: *International Conference on Scale Space and Variational Methods in Computer Vision*. Springer. 2011, pp. 435–446.

- [Ram+22] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. “Hierarchical text-conditional image generation with clip latents”. In: *arXiv preprint arXiv:2204.06125* (2022).
- [Ram+21] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. “Zero-shot text-to-image generation”. In: *International Conference on Machine Learning*. PMLR. 2021, pp. 8821–8831.
- [RS07] Martin Raphan and Eero P Simoncelli. “Learning to Be Bayesian Without Supervision”. In: *Advances in Neural Information Processing Systems*. 2007, pp. 1145–1152.
- [RS11] Martin Raphan and Eero P Simoncelli. “Least Squares Estimation Without Priors or Supervision”. In: *Neural computation* 23.2 (2011), pp. 374–420.
- [Rav+18] Suman Ravuri, Shakir Mohamed, Mihaela Rosca, and Oriol Vinyals. “Learning Implicit Generative Models with the Method of Learned Moments”. In: *Proceedings of the 35th International Conference on Machine Learning*. Ed. by Jennifer Dy and Andreas Krause. Vol. 80. Proceedings of Machine Learning Research. Stockholmsmässan, Stockholm Sweden: PMLR, Oct. 2018, pp. 4314–4323. URL: <http://proceedings.mlr.press/v80/ravuri18a.html>.
- [ROV19] Ali Razavi, Aaron van den Oord, and Oriol Vinyals. “Generating diverse high-fidelity images with vq-vae-2”. In: *Advances in Neural Information Processing Systems*. 2019, pp. 14837–14847.
- [Raz+19] Ali Razavi, Aäron van den Oord, Ben Poole, and Oriol Vinyals. “Preventing Posterior Collapse with delta-VAEs”. In: *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL: <https://openreview.net/forum?id=BJe0Gn0cY7>.
- [RKK18] Sashank J. Reddi, Satyen Kale, and Sanjiv Kumar. “On the Convergence of Adam and Beyond”. In: *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. URL: <https://openreview.net/forum?id=ryQu7f-RZ>.

- [RMW14a] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. “Stochastic Backpropagation and Approximate Inference in Deep Generative Models”. In: *Proceedings of the 31st International Conference on Machine Learning*. 2014, pp. 1278–1286.
- [RMW14b] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. “Stochastic Backpropagation and Approximate Inference in Deep Generative Models”. In: *Proceedings of the 31st International Conference on Machine Learning*. Vol. 32. 2014.
- [Rob20] Herbert Robbins. *An empirical Bayes approach to statistics*. University of California Press, 2020.
- [RT+96] Gareth O Roberts, Richard L Tweedie, et al. “Exponential convergence of Langevin distributions and their discrete approximations”. In: *Bernoulli* 2.4 (1996), pp. 341–363.
- [RMK20] Geoffrey Roeder, Luke Metz, and Diederik P Kingma. “On Linear Identifiability of Learned Representations”. In: *arXiv preprint arXiv:2007.00810* (2020).
- [Ron20] Matteo Ronchetti. “TorchRadon: Fast Differentiable Routines for Computed Tomography”. In: *arXiv preprint arXiv:2009.14788* (2020). eprint: [arXiv : 2009.14788](https://arxiv.org/abs/2009.14788).
- [RPB15] O. Ronneberger, P. Fischer, and T. Brox. “U-Net: Convolutional Networks for Biomedical Image Segmentation”. In: *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*. Vol. 9351. LNCS. (available on arXiv:1505.04597 [cs.CV]). Springer, 2015, pp. 234–241. URL: <http://lmb.informatik.uni-freiburg.de/Publications/2015/RFB15a>.
- [Rot+17] Kevin Roth, Aurelien Lucchi, Sebastian Nowozin, and Thomas Hofmann. “Stabilizing training of generative adversarial networks through regularization”. In: *Advances in neural information processing systems* 30 (2017).
- [RS00] Sam T Roweis and Lawrence K Saul. “Nonlinear dimensionality reduction by locally linear embedding”. In: *science* 290.5500 (2000), pp. 2323–2326.
- [RHW86] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. “Learning representations by back-propagating errors”. In: *nature* 323.6088 (1986), pp. 533–536.

- [SZ+19] Sotirios Sabanis, Ying Zhang, et al. “Higher order langevin monte carlo algorithm”. In: *Electronic Journal of Statistics* 13.2 (2019), pp. 3805–3850.
- [Sah+22] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily Denton, Seyed Kamyar Seyed Ghasemipour, Burcu Karagol Ayan, S Sara Mardavi, Rapha Gontijo Lopes, et al. “Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding”. In: *arXiv preprint arXiv:2205.11487* (2022).
- [Sai96] Stephan R Sain. *The nature of statistical learning theory*. 1996.
- [Saj+18] Mehdi SM Sajjadi, Olivier Bachem, Mario Lucic, Olivier Bousquet, and Sylvain Gelly. “Assessing generative models via precision and recall”. In: *Advances in Neural Information Processing Systems*. 2018, pp. 5228–5237.
- [Sal+16] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. “Improved techniques for training gans”. In: *Advances in neural information processing systems*. 2016, pp. 2234–2242.
- [Sal+17] Tim Salimans, Andrej Karpathy, Xi Chen, and Diederik P. Kingma. “Pixel-CNN++: Improving the PixelCNN with Discretized Logistic Mixture Likelihood and Other Modifications”. In: *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL: <https://openreview.net/forum?id=BJrFC6ceg>.
- [SH19] Saeed Saremi and Aapo Hyvärinen. “Neural empirical bayes”. In: *Journal of Machine Learning Research* 20 (2019), pp. 1–23.
- [Sar+18] Saeed Saremi, Arash Mehrjou, Bernhard Schölkopf, and Aapo Hyvärinen. “Deep energy estimator networks”. In: *arXiv preprint arXiv:1805.08306* (2018).
- [SS19] Simo Särkkä and Arno Solin. *Applied stochastic differential equations*. Vol. 10. Cambridge University Press, 2019.
- [SHS14] Hiroaki Sasaki, Aapo Hyvärinen, and Masashi Sugiyama. “Clustering via mode seeking by direct estimation of the gradient of a log-density”. In: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer. 2014, pp. 19–34.

- [SR03] Lawrence K Saul and Sam T Roweis. “Think globally, fit locally: unsupervised learning of low dimensional manifolds”. In: *Departmental Papers (CIS)* (2003), p. 12.
- [See+99] I Seibold, B Lehmann, A Arribas, C Ruiz, TJ Shepherd, and KL Ashworth. “Development of tomographic systems for mining, mineral exploration and environmental purposes”. In: *Transactions of the Institution of Mining and Metallurgy* 108.B (1999), B105–B118.
- [SZX19] Liyue Shen, Wei Zhao, and Lei Xing. “Patient-specific reconstruction of volumetric computed tomography images from a single projection view via deep learning”. In: *Nature biomedical engineering* 3.11 (2019), pp. 880–888.
- [SSZ18] Jiaxin Shi, Shengyang Sun, and Jun Zhu. “A Spectral Approach to Gradient Estimation for Implicit Distributions”. In: *Proceedings of the 35th International Conference on Machine Learning*. 2018, pp. 4651–4660.
- [SD21] Samarth Sinha and Adji B Dieng. “Consistency Regularization for Variational Auto-Encoders”. In: *arXiv preprint arXiv:2105.14859* (2021).
- [Ski89] John Skilling. “The eigenvalues of mega-dimensional matrices”. In: *Maximum Entropy and Bayesian Methods*. Springer, 1989, pp. 455–466.
- [SBD11] Jascha Sohl-Dickstein, Peter Battaglino, and Michael R. DeWeese. “Minimum Probability Flow Learning”. In: *Proceedings of the 28th International Conference on International Conference on Machine Learning*. 2011, pp. 905–912.
- [Soh+15] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. “Deep Unsupervised Learning Using Nonequilibrium Thermodynamics”. In: *International Conference on Machine Learning*. 2015, pp. 2256–2265.
- [SE19a] Jiaming Song and Stefano Ermon. “Bridging the Gap Between f -GANs and Wasserstein GANs”. In: *arXiv preprint arXiv:1910.09779* (2019).
- [SZE17] Jiaming Song, Shengjia Zhao, and Stefano Ermon. “A-nice-mc: Adversarial training for mcmc”. In: *arXiv preprint arXiv:1706.07561* (2017).
- [Son+21a] Yang Song, Conor Durkan, Iain Murray, and Stefano Ermon. “Maximum Likelihood Training of Score-Based Diffusion Models”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2021.

- [SE19b] Yang Song and Stefano Ermon. “Generative Modeling by Estimating Gradients of the Data Distribution”. In: *Advances in Neural Information Processing Systems*. 2019, pp. 11918–11930.
- [SE20] Yang Song and Stefano Ermon. “Improved Techniques for Training Score-Based Generative Models”. In: *Advances in Neural Information Processing Systems* 33 (2020).
- [Son+19] Yang Song, Sahaj Garg, Jiaxin Shi, and Stefano Ermon. “Sliced Score Matching: A Scalable Approach to Density and Score Estimation”. In: *Proceedings of the Thirty-Fifth Conference on Uncertainty in Artificial Intelligence, UAI 2019, Tel Aviv, Israel, July 22-25, 2019*. 2019, p. 204.
- [Son+18a] Yang Song, Taesup Kim, Sebastian Nowozin, Stefano Ermon, and Nate Kushman. “PixelDefend: Leveraging Generative Models to Understand and Defend against Adversarial Examples”. In: *International Conference on Learning Representations*. 2018. URL: <https://openreview.net/forum?id=rJUYGxbCW>.
- [SK21] Yang Song and Diederik P Kingma. “How to Train Your Energy-Based Models”. In: *arXiv preprint arXiv:2101.03288* (2021).
- [Son+22] Yang Song, Liyue Shen, Lei Xing, and Stefano Ermon. “Solving Inverse Problems in Medical Imaging with Score-Based Generative Models”. In: *International Conference on Learning Representations*. 2022. URL: <https://openreview.net/forum?id=vaRCHVj0uGI>.
- [Son+18b] Yang Song, Rui Shu, Nate Kushman, and Stefano Ermon. “Constructing Unrestricted Adversarial Examples with Generative Models”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2018.
- [Son+21b] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. “Score-Based Generative Modeling through Stochastic Differential Equations”. In: *International Conference on Learning Representations*. 2021. URL: <https://openreview.net/forum?id=PxTIG12RRHS>.
- [Sri+17] Bharath Sriperumbudur, Kenji Fukumizu, Arthur Gretton, Aapo Hyvärinen, and Revant Kumar. “Density Estimation in Infinite Dimensional Exponential Families”. In: *Journal of Machine Learning Research* 18.57 (2017), pp. 1–59.

- [Sri+09] Bharath K Sriperumbudur, Kenji Fukumizu, Arthur Gretton, Bernhard Schölkopf, and Gert RG Lanckriet. “On integral probability metrics, ϕ -divergences and binary classification”. In: *arXiv preprint arXiv:0901.2698* (2009).
- [Sta59] A.J. Stam. “Some Inequalities Satisfied by the Quantities of Information of Fisher and Shannon”. en. In: *Information and Control* 2.2 (June 1959), pp. 101–112.
- [Ste72] Charles Stein. “A bound for the error in the normal approximation to the distribution of a sum of dependent random variables”. In: *Proceedings of the sixth Berkeley symposium on mathematical statistics and probability, volume 2: Probability theory*. Vol. 6. University of California Press. 1972, pp. 583–603.
- [Ste81] Charles M Stein. “Estimation of the mean of a multivariate normal distribution”. In: *The annals of Statistics* (1981), pp. 1135–1151.
- [ST99] O Stramer and RL Tweedie. “Langevin-type models I: Diffusions with given stationary distributions and their discretizations”. In: *Methodology and Computing in Applied Probability* 1.3 (1999), pp. 283–306.
- [Str+15] Heiko Strathmann, Dino Sejdinovic, Samuel Livingstone, Zoltan Szabo, and Arthur Gretton. “Gradient-free Hamiltonian Monte Carlo with efficient kernel exponential families”. In: *Advances in Neural Information Processing Systems*. 2015, pp. 955–963.
- [Sub+07] Sriram Subramaniam, Alberto Bartesaghi, Jun Liu, Adam E Bennett, and Rachid Sougrat. “Electron tomography of viruses”. In: *Current opinion in structural biology* 17.5 (2007), pp. 596–602.
- [Sun+19] Shengyang Sun, Guodong Zhang, Jiaxin Shi, and Roger Grosse. “Functional Variational Bayesian Neural Networks”. In: *International Conference on Learning Representations*. 2019.
- [Sut+18] Dougal Sutherland, Heiko Strathmann, Michael Arbel, and Arthur Gretton. “Efficient and principled score estimation with Nyström kernel exponential families”. In: *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*. 2018, pp. 652–660.

- [SVL14] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. “Sequence to sequence learning with neural networks”. In: *Advances in neural information processing systems* 27 (2014).
- [SW86] Robert H Swendsen and Jian-Sheng Wang. “Replica Monte Carlo simulation of spin-glasses”. In: *Physical review letters* 57.21 (1986), p. 2607.
- [TKK21] Jaesung Tae, Hyeongju Kim, and Taesu Kim. “EdiTTS: Score-based Editing for Controllable Text-to-Speech”. In: *arXiv preprint arXiv:2110.02584* (2021).
- [Tan+20] Matthew Tancik, Pratul P. Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T. Barron, and Ren Ng. “Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains”. In: *NeurIPS* (2020).
- [Tas+21] Yusuke Tashiro, Jiaming Song, Yang Song, and Stefano Ermon. “CSDI: Conditional Score-based Diffusion Models for Probabilistic Time Series Imputation”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2021.
- [TOB16a] L. Theis, A. van den Oord, and M. Bethge. “A note on the evaluation of generative models”. In: *International Conference on Learning Representations*. arXiv:1511.01844. 2016. URL: <http://arxiv.org/abs/1511.01844>.
- [TOB16b] Lucas Theis, Aäron van den Oord, and Matthias Bethge. “A note on the evaluation of generative models”. In: *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. 2016. URL: <http://arxiv.org/abs/1511.01844>.
- [Tia+20] Chunwei Tian, Lunke Fei, Wenxian Zheng, Yong Xu, Wangmeng Zuo, and Chia-Wen Lin. “Deep learning on image denoising: An overview”. In: *Neural Networks* (2020).
- [Tol+18] Ilya Tolstikhin, Olivier Bousquet, Sylvain Gelly, and Bernhard Schoelkopf. “Wasserstein Auto-Encoders”. In: *International Conference on Learning Representations*. 2018. URL: <https://openreview.net/forum?id=HkL7n1-0b>.
- [TBB19] James Townsend, Thomas Bird, and David Barber. “Practical lossless compression with latent variables using bits back coding”. In: *International Conference*

- on Learning Representations*. 2019. URL: <https://openreview.net/forum?id=ryE98iR5tm>.
- [TRB17] Dustin Tran, Rajesh Ranganath, and David Blei. “Hierarchical implicit models and likelihood-free variational inference”. In: *Advances in Neural Information Processing Systems*. 2017, pp. 5523–5533.
- [TR19] Belinda Tzen and Maxim Raginsky. “Neural Stochastic Differential Equations: Deep Latent Gaussian Models in the Diffusion Limit”. In: *arXiv:1905.09883* (2019).
- [UVL16] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. “Instance normalization: The missing ingredient for fast stylization”. In: *arXiv preprint arXiv:1607.08022* (2016).
- [Uri+16] Benigno Uria, Marc-Alexandre Côté, Karol Gregor, Iain Murray, and Hugo Larochelle. “Neural autoregressive distribution estimation”. In: *The Journal of Machine Learning Research* 17.1 (2016), pp. 7184–7220.
- [UML13] Benigno Uria, Iain Murray, and Hugo Larochelle. “RNADE: the real-valued neural autoregressive density-estimator”. In: *Proceedings of the 26th International Conference on Neural Information Processing Systems-Volume 2*. 2013, pp. 2175–2183.
- [Vaa98] A. W. van der Vaart. *Asymptotic Statistics*. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, 1998. doi: [10.1017/CBO9780511802256](https://doi.org/10.1017/CBO9780511802256).
- [VK20] Arash Vahdat and Jan Kautz. “NVAE: A Deep Hierarchical Variational Autoencoder”. In: *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*. Ed. by Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin. 2020. URL: <https://proceedings.neurips.cc/paper/2020/hash/e3b21256183cf7c2c7a66be163579d37-Abstract.html>.
- [VKK16] Aäron Van Den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. “Pixel Recurrent Neural Networks”. In: *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*. ICML’16. New

- York, NY, USA: JMLR.org, 2016, pp. 1747–1756. URL: <http://dl.acm.org/citation.cfm?id=3045390.3045575>.
- [Vas+17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. “Attention is all you need”. In: *Advances in neural information processing systems* 30 (2017).
- [Vin11] Pascal Vincent. “A Connection Between Score Matching and Denoising Autoencoders”. In: *Neural Computation* 23.7 (2011), pp. 1661–1674.
- [VB13] Marinus T Vlaardingerbroek and Jacques A Boer. *Magnetic resonance imaging: theory and practice*. Springer Science & Business Media, 2013.
- [Wan+18] Jianing Wang, Yiyuan Zhao, Jack H Noble, and Benoit M Dawant. “Conditional generative adversarial networks for metal artifact reduction in CT images of the ear”. In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer. 2018, pp. 3–11.
- [Wan+20] Ziyu Wang, Shuyu Cheng, Li Yueru, Jun Zhu, and Bo Zhang. “A wasserstein minimum velocity approach to learning unnormalized models”. In: *International Conference on Artificial Intelligence and Statistics*. PMLR. 2020, pp. 3728–3738.
- [Wei+20] Haoyu Wei, Florian Schiffers, Tobias Würfl, Daming Shen, Daniel Kim, Aggelos K Katsaggelos, and Oliver Cossairt. “2-Step Sparse-View CT Reconstruction with a Domain-Specific Perceptual Network”. In: *arXiv preprint arXiv:2012.04743* (2020).
- [WT11] Max Welling and Yee W Teh. “Bayesian learning via stochastic gradient Langevin dynamics”. In: *Proceedings of the 28th international conference on machine learning (ICML-11)*. Citeseer. 2011, pp. 681–688.
- [Wen+19] Li Wenliang, Dougal Sutherland, Heiko Strathmann, and Arthur Gretton. “Learning deep kernels for exponential family densities”. In: *Proceedings of the 36th International Conference on Machine Learning*. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. Long Beach, California, USA: PMLR, Sept. 2019, pp. 6737–6746. URL: <http://proceedings.mlr.press/v97/wenliang19a.html>.

- [Woo+14] David P Woodruff et al. “Sketching as a tool for numerical linear algebra”. In: *Foundations and Trends® in Theoretical Computer Science* 10.1–2 (2014), pp. 1–157.
- [Wür+18] Tobias Würfl, Mathis Hoffmann, Vincent Christlein, Katharina Breininger, Yixin Huang, Mathias Unberath, and Andreas K Maier. “Deep learning computed tomography: Learning projection-domain weights from image domain in limited angle problems”. In: *IEEE transactions on medical imaging* 37.6 (2018), pp. 1454–1463.
- [Xie+22] Tian Xie, Xiang Fu, Octavian-Eugen Ganea, Regina Barzilay, and Tommi S. Jaakkola. “Crystal Diffusion Variational Autoencoder for Periodic Material Generation”. In: *International Conference on Learning Representations*. 2022. URL: https://openreview.net/forum?id=03RLpj-tc_.
- [Xu+22] Minkai Xu, Lantao Yu, Yang Song, Chence Shi, Stefano Ermon, and Jian Tang. “GeoDiff: A Geometric Diffusion Model for Molecular Conformation Generation”. In: *International Conference on Learning Representations*. 2022. URL: <https://openreview.net/forum?id=PzcvxEMzvQC>.
- [Yan+21] Kaiyu Yang, Jacqueline Yau, Li Fei-Fei, Jia Deng, and Olga Russakovsky. “A Study of Face Obfuscation in ImageNet”. In: *arXiv preprint arXiv:2103.06191* (2021).
- [YK16] Fisher Yu and Vladlen Koltun. “Multi-scale context aggregation by dilated convolutions”. In: *International Conference on Learning Representations (ICLR)*. 2016.
- [YKF17] Fisher Yu, Vladlen Koltun, and Thomas Funkhouser. “Dilated Residual Networks”. In: *Computer Vision and Pattern Recognition (CVPR)*. 2017.
- [Yu+15] Fisher Yu, Ari Seff, Yinda Zhang, Shuran Song, Thomas Funkhouser, and Jianxiong Xiao. “Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop”. In: *arXiv preprint arXiv:1506.03365* (2015).
- [Yu+20] Lequan Yu, Zhicheng Zhang, Xiaomeng Li, and Lei Xing. “Deep sinogram completion with image prior for metal artifact reduction in CT images”. In: *IEEE Transactions on Medical Imaging* 40.1 (2020), pp. 228–238.

- [ZK16] Sergey Zagoruyko and Nikos Komodakis. “Wide residual networks”. In: *arXiv preprint arXiv:1605.07146* (2016).
- [Zbo+18] Jure Zbontar et al. “fastMRI: An Open Dataset and Benchmarks for Accelerated MRI”. In: 2018. arXiv: [1811.08839](https://arxiv.org/abs/1811.08839).
- [ZZ18] Qianjun Zhang and Lei Zhang. “Convolutional adaptive denoising autoencoders for hierarchical feature extraction”. In: *Frontiers of Computer Science* 12.6 (2018), pp. 1140–1148.
- [Zha19] Richard Zhang. “Making Convolutional Networks Shift-Invariant Again”. In: *ICML*. 2019.
- [ZFZ19] Hao Zheng, Faming Fang, and Guixu Zhang. “Cascaded Dilated Dense Network with Two-step Data Consistency for MRI Reconstruction”. In: *Advances in Neural Information Processing Systems*. Ed. by H. Wallach, H. Larochelle, A. Beygelzimer, F. d’Alché-Buc, E. Fox, and R. Garnett. Vol. 32. Curran Associates, Inc., 2019. URL: <https://proceedings.neurips.cc/paper/2019/file/1e48c4420b7073bc11916c6c1de226bb-Paper.pdf>.
- [ZZ20] Bo Zhou and S Kevin Zhou. “DuDoRNet: Learning a dual-domain recurrent network for fast MRI reconstruction with deep T1 prior”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 4273–4282.
- [Zho+19] Sharon Zhou, Mitchell Gordon, Ranjay Krishna, Austin Narcomey, Li F Fei-Fei, and Michael Bernstein. “HYPE: A Benchmark for Human eYe Perceptual Evaluation of Generative Models”. In: *Advances in Neural Information Processing Systems*. 2019, pp. 3444–3456.
- [ZSZ20] Yuhao Zhou, Jiaxin Shi, and Jun Zhu. “Nonparametric score estimators”. In: *International Conference on Machine Learning*. PMLR. 2020, pp. 11513–11522.
- [Zhu+18] Bo Zhu, Jeremiah Z Liu, Stephen F Cauley, Bruce R Rosen, and Matthew S Rosen. “Image reconstruction by domain-transform manifold learning”. In: *Nature* 555.7697 (2018), pp. 487–492.
- [Zim+21] Roland S Zimmermann, Lukas Schott, Yang Song, Benjamin A Dunn, and David A Klindt. “Score-Based Generative Classifiers”. In: *arXiv preprint arXiv:2110.00473* (2021).