RICE UNIVERSITY

From Guided Sampling to Learned Solvers: Diffusion Models for Inverse Problems

By

Nicolas Zilberstein

A THESIS SUBMITTED
IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE

Doctor of Philosophy

APPROVED, THESIS COMMITTEE

Santiago Segarra
Santiago Segarra (Sep 21, 2026 19:20:12 CDT)

Santiago Segarra (Chair)

W. M. Rice Trustee Associate Professor
of Electrical and Computer Engineering

Morteza Mardani
Morteza Mardani (Sep 21, 2026 17:05:36 PDT)

Morteza Mardani

Principal Research Scientist, NVIDIA

Cesar A Uribe
Cesar A Uribe (Sep 21, 2026 10:18:29 CDT)

Cesar A. Uribe

Louis Owen Assistant Professor of
Electrical and Computer Engineering

Αναστασιος Κυριλλιδης
Αναστασιος Κυριλλιδης (Sep 21, 2026 13:07:59 CDT)

Anastasios Kyrillidis

Noah Harding Associate Professor of
Computer Science

HOUSTON, TEXAS

December 2026

ABSTRACT

From Guided Sampling to Learned Solvers: Diffusion Models for Inverse Problems

by

Nicolas Zilberstein

During the last five years, Diffusion models (and their flow-matching counterparts) have emerged as expressive priors for *inverse problems*, where the goal is to recover an unknown signal from noisy, incomplete, or otherwise degraded measurements. Combining a pre-trained diffusion prior with a physical forward model gives, in principle, a plug-and-play recipe for posterior sampling or *maximum a posteriori* estimation. In practice, this strategy faces several challenges, broadly categorized in two groups: on the algorithmic side, naive posterior sampling is slow, prone to mode collapse when the diffusion is defined in a latent space, and unable to respect hard structural constraints; on the model side, sampling from the pre-trained prior requires integration over many small steps, only the score of the density is learned (not a normalized density), and the source distribution is task-agnostic.

This thesis develops different algorithms that address these challenges independently. In Part I (*inference-time algorithms on pre-trained priors*) we introduce (i) higher-order annealed Langevin diffusion for accelerated posterior sampling, applied to massive MIMO detection, channel estimation, and image restoration; (ii) repulsive latent score distillation, which injects a repulsive coupling across particles to provably prevent mode collapse in latent-space samplers; and (iii) shielded Langevin Monte Carlo, which augments the drift with a navigation potential to enforce hard

structural constraints. In Part II (*training priors for improved inference*) we introduce (iv) prior-informed flow matching, which replaces the Gaussian source with a task-informed one for graph inverse problems; (v) energy-based models with anisotropic noise, which learn a normalized density via a dual score-matching objective and unlock blind inverse problems; and (vi) flow-map denoisers, which train a flow map and use its two-time denoiser form to control the distortion–perception plane with a single model.

The unifying claim is that at every level of the diffusion-based sampling pipeline — sampling algorithm, source distribution, training objective (score alone vs. score-plus-density), and training target for velocity (instantaneous vs. average) — there is a design lever, and inverse problems benefit from pulling each in the direction dictated by the downstream task. We validate this across imaging, wireless communications, and graph inverse problems.

Acknowledgments

It's hard to summarize in just a couple of pages the impact the PhD has had on my life, both professionally and personally. I have to admit that I started this journey imagining certain things, but the path turned out to be completely different from what I had planned.

First, I want to thank Santiago: I will be forever grateful for the opportunity you gave me back in 2020, when we started working together. I remember quite clearly that as soon as I heard your talk at FIUBA, I was sure I wanted to do my PhD with you. What first convinced me was your intelligence and your conceptual clarity, but what ultimately won me over was your humanity. Six and a half years later, I can't imagine a better advisor. Thank you for your constant guidance and support. In particular, I want to thank you for the freedom and trust you always gave me to explore what I was passionate about. I admire you, and I hope that one day I can come even a little closer to being an advisor like you.

I want to thank my committee, Tasos Kyrillidis, Cesar Uribe, and Morteza Mardani, for their help and for taking the time to read my thesis. In particular, I want to thank Morteza for the opportunity to collaborate in 2024, which was fundamental to my transition into machine learning and to defining the direction of my thesis. Morteza, your contribution was key in helping me build the confidence to bet on diffusion.

I also want to thank the Rice ECE community: Ashutosh Sabharwal and Rahman Doost-Mohammady, with whom I collaborated on wireless papers in the early years of my PhD; and everyone in Santiago's group over the years: Mitch, Arindam, Boning, Madeline, Reza, Yuyujun, Rostyslav, and Ali, with whom I had the pleasure of collaborating on several papers. I also want to thank Harvey Chen for his dedication

to research and for trusting me. Outside of ECE at Rice, there are countless people I would like to thank, but doing so would make this thesis infinitely long. I would like to highlight a few people with whom I shared different moments at Rice: Gonchi, and the "casa" group that welcomed me; Tincho Sevilla, with whom I shared both an office and a house; and the "Erudovich" group (Prina, Guido, Juanse, and Facu). Last but not least, special thanks to Fernando Gama, who was key to my arrival in the PhD program and helped me adjust to life in the US: Fer, thank you so much for all your guidance and advice.

Beyond Rice, in the academic world, there are also countless people I'd like to thank. First, I want to thank all my internship hosts. Thank you to Akshay Malhotra and Shahab Hamidi-Rad at InterDigital for their trust. Thank you to Eero Simoncelli and Florentin Guth for an incredible summer at a place as special as the Flatiron Institute. Having the chance to work in an environment like Flatiron's was a true honor. Thank you to Mojtaba Sahraee-Ardakan, Kuldeep Purohit, Mauricio Delbracio, and the entire "Luma" team for hosting me at Google. Getting to see behind the scenes of the most advanced diffusion models was another great privilege I had during my PhD. Beyond internships, I want to thank Gonzalo Mateos, Antonio Marques, Victor Tenorio, and Luiz Chamon, with whom I had the privilege of collaborating: I learned a great deal from each of you. I also want to thank Samuel Rey, with whom I shared my first year at Rice during his visit, and Juan Cerviño, for our get-togethers in different places across the US and even in Uruguay.

Lastly, although they were not directly involved, I want to thank all the professors who taught me during my undergraduate studies at FIUBA. *Y quiero detenerme a agradecer a la Universidad de Buenos Aires y a la educación pública, gratuita y de calidad, que tan golpeada está siendo en este momento. Siento el deber de dejar un mensaje sobre lo importante que es la educación pública en nuestro país, y de que es algo que debemos cuidar.*

Moving beyond academia, I would now like to thank all the wonderful people in

Argentina. To my lifelong friends from my high school San Jose: Claus, Luca, Paco, Bauti, Luz, Ana, Juli and Vicky; to my friends from "Vamos los Jueves," for all the asados we had every time I came back to Buenos Aires, and especially Tomi Davila, for always keeping up with my life abroad and taking the time to catch up, and Juanma, for that wonderful visit in 2025. I also want to thank Pepa, Fran Banti and Tincho Davila, and "la mastur" (Pili, Maca and Facu Nola). And of course, to the friends FIUBA gave me: Diego (rotor), Klaus, and Alan.

Finally, I want to take a moment for my family, who have been extraordinarily generous to me. To my parents, Roslinda and Martin: my gratitude will be eternal for always pushing me to work harder, to believe it could be done, and for giving me the chance to choose my own path. To my brothers, Tomi and Luqui, for your joy and for always being willing to drop whatever you had going on to spend time with me every summer. Mamá, papá, Tomi, Luqui: I love you. To my grandparents who are no longer with me, and to Chana, with her now-traditional "sí se puede" ("yes, we can"). Chana, your love of life is contagious and an endless source of energy. To my aunts and uncles. And to my "new family," the Acquarones and the Etchevarrens.

And now, I want to close with the most important person I met during my PhD: Iris, my wife. Iris, I admire you deeply. From the first moment I saw you back in 2021, I knew there was something special between us. Being able to call you my wife is without a doubt the most beautiful thing the PhD gave me. Thank you for coming with me to every internship, trip, and move, and through everything we've done together. Thank you for all the days you put up with my (almost constant) overworking and my nerves during interview processes. Life with you is more beautiful, and I can't wait to see where life takes us next.

Contents

Abstract i

List of Illustrations xiv

List of Tables xxviii

1 Introduction 1

1.1 Generative models and inverse problems 1

1.2 Diffusion models as priors 2

1.3 Thesis contributions and organization 5

1.4 Outline of the thesis 7

1.5 List of publications 9

2 Background 12

2.1 Inverse problems and generative priors 13

2.1.1 Bayesian formulation 13

2.1.2 The distortion–perception trade-off 14

2.1.3 Priors: classical and learned 16

2.2 Diffusion models 17

2.2.1 Forward process 18

2.2.2 Reverse process 19

2.2.3 Probability-flow ODE 19

2.3 Latent diffusion models 20

2.4 Denoising score matching 21

2.5 Sampling algorithms 23

2.6 Solving inverse problems with diffusion priors 25

2.7 Design knobs in diffusion-based inverse problem solvers 27
2.8 Contributions of this thesis 29
2.9 Notation 30

I Inference-time algorithms on pre-trained priors 32

3 Higher-order annealed Langevin diffusion 34
3.1 Introduction 34
3.2 System model and problem formulation 36
3.3 Higher-order Langevin dynamics 37
3.3.1 Continuous-time underdamped Langevin dynamics 37
3.3.2 Generalized Langevin equation 39
3.4 Accelerating higher-order Langevin dynamics 42
3.4.1 Modified continuous-time higher-order Langevin dynamics . . 42
3.4.2 Numerical discretization 44
3.4.3 Sampling from the posterior via discrete annealed pre-conditioned higher-order Langevin 48
3.5 Numerical results 52
3.5.1 Massive MIMO detection 53
3.5.2 Channel estimation with learned score prior 59
3.5.3 Inverse problem in imaging with learned score prior 65
3.6 Conclusion 67

4 Repulsive latent score distillation 69
4.1 Introduction 69
4.2 Related Works 72
4.3 Background 73
4.4 Repulsive variational diffusion sampling 75

4.4.1 Tackling mode collapse: enhancing diversity via repulsion . . . 75
4.4.2 Tackling latent inversion: augmentation of the variational distribution . . . 78
4.4.3 Repulsive latent score distillation for solving inverse problems 79
4.5 Numerical results . . . 81
4.5.1 Inpainting . . . 83
4.5.2 Non-linear inverse problems . . . 84
4.6 Conclusion . . . 87

5 Sampling with shielded Langevin Monte Carlo using navigation potentials 89
5.1 Introduction . . . 89
5.2 Background . . . 91
5.2.1 Langevin diffusion . . . 92
5.2.2 Rimon-Koditschek (RK) navigation potential . . . 93
5.3 Shielded LMC sampler . . . 95
5.3.1 Building the sampler by using time-varying noise . . . 96
5.4 Numerical results . . . 99
5.4.1 2D Gaussian mixture model with obstacles . . . 99
5.4.2 MIMO detection: speeding-up the sampling . . . 100
5.5 Conclusion . . . 103

II Training priors for improved inference 104

6 Prior-informed flow matching for graph reconstruction 109
6.1 Introduction . . . 110
6.2 Related work . . . 112
6.3 Background . . . 114

6.3.1 Continuous flow matching for graph generation 115
6.3.2 Distortion–perception formulation for conditional graph reconstruction 116
6.4 Method 118
6.4.1 Approximating the posterior mean 119
6.4.2 Learning the flow model 120
6.5 Numerical results 127
6.5.1 Setup 127
6.5.2 Link prediction 128
6.5.3 Blind graph reconstruction 130
6.5.4 Distortion–perception trade-off 132
6.6 Conclusion 133

7 Learning normalized energy models 135

7.1 Introduction 136
7.2 Background 137
7.2.1 Diffusion models: from isotropic to anisotropic noise 137
7.2.2 Linear inverse problems 138
7.3 Learning EBMs through anisotropic denoising 139
7.3.1 Energy-based linear inverse problem solvers 140
7.3.2 Learning with anisotropic dual score matching 142
7.3.3 Architecture and covariance conditioning 143
7.3.4 Validation: comparison to baselines 146
7.4 Applications of anisotropic energy models 148
7.4.1 The probabilities of posterior samples 149
7.4.2 Energy-based schedules and corrector steps 151
7.4.3 Blind inverse problems 155
7.5 Discussion 156

8 Flow map denoisers: traversing the distortion–perception plane **159**

8.1 Introduction . . . 159
8.2 Background . . . 162
8.2.1 Flow matching and flow maps . . . 162
8.2.2 Inverse problems and Plug-and-Play methods . . . 163
8.2.3 Distortion–Perception Trade-off . . . 164
8.3 Average denoisers: traversing the DP plane . . . 164
8.3.1 Flow maps as denoisers . . . 164
8.3.2 Traversing the DP plane with flow maps . . . 165
8.3.3 Beyond the Gaussian case: image denoising experiments . . . 167
8.4 Plug-and-play with average denoisers . . . 169
8.5 Related work . . . 171
8.6 Numerical results . . . 172
8.6.1 Inverse problems . . . 174
8.6.2 Analysis of the DP trade-off . . . 175
8.7 Discussion . . . 176

9 Discussion and future directions **179**

9.1 Summary of contributions . . . 179
9.2 Open questions and future directions . . . 181
9.2.1 Methodological directions . . . 181
9.2.2 Scientific applications . . . 182

III Appendix **184**

A Supplement for Chapter 3 **185**

A.1 Proof of Proposition 1 . . . 185

A.2 Proof of Proposition 2 187

B Supplement for Chapter 4 190

B.1 Technical proofs 190

B.1.1 Data augmentation 190

B.1.2 Proof of proposition 1 191

B.1.3 Proof of proposition 2 192

B.2 Additional experiments 193

B.2.1 Implementation details of baselines 193

B.2.2 Super Resolution 195

B.2.3 Motion deblurring 197

B.2.4 Inpainting with masked box (half-face) 198

B.2.5 Inpainting with random mask 199

B.2.6 Inpainting free mask 199

B.2.7 Qualitative results 201

B.2.8 Ablation of RLSD 205

C Supplement for Chapter 6 217

Appendix 217

C.1 Proofs 217

C.1.1 Proof for Theorem 1 217

C.2 Experimental details 219

C.2.1 Details about the architecture 219

C.2.2 Details about the datasets 220

C.2.3 Hyperparameters 221

C.2.4 Metrics Calculation 224

C.2.5 Training cost 225

C.2.6 Inference cost 226

D Supplement for Chapter 7 **228**

D.1 Dual-score matching for anisotropic noise . . . 228

D.1.1 Algorithms . . . 231

D.2 Energy-guided sampling as a Bregman mirror descent . . . 233

D.3 Implementation . . . 233

D.3.1 Architecture and training hyperparameters . . . 233

D.3.2 Distribution covariances used for training . . . 236

D.3.3 Comparison with previous energy-based models . . . 237

D.3.4 Details of baselines . . . 238

D.4 Additional experiments . . . 240

D.4.1 Additional inverse problems . . . 240

D.4.2 Comparison between dual and single score matching . . . 244

D.4.3 Analysis of the posterior distribution . . . 246

D.4.4 Recovering the Gumbel distribution for the isotropic case . . . 249

D.4.5 Validation of the normalization constant in synthethic experiment . . . 251

D.4.6 Reconstructed images . . . 253

E Supplement for Chapter 8 **261**

E.1 Background . . . 261

E.1.1 Flow Maps . . . 261

E.2 Proofs and theoretical results . . . 263

E.2.1 Average denoiser. . . . 263

E.2.2 Full derivation for the Gaussian case (Theorem 2) . . . 264

E.2.3 Experiment on mixture of Gaussians . . . 268

E.2.4 Fixed point characterization . . . 272

E.2.5 Gaussian case: global contraction. . . . 276

E.3 Implementation . . . 281

E.3.1 Training details 281
E.3.2 Hyperparameters and baselines 283
E.3.3 Computational cost 284
E.3.4 Experiment of the teaser plot 285
E.4 Additional experiments 286
E.4.1 Results with error bars 286
E.4.2 Additional inverse problems 287
E.4.3 Visualization of the iterations for different lookaheads 291
E.4.4 Distortion–perception of CelebA 293
E.4.5 Performance as a function of the number of PnP iterations . . 294
E.4.6 Failure modes and limitations 297
E.4.7 Visual results 299

Bibliography **310**

Illustrations

3.1 Performance analysis of our proposed methods for MIMO symbol detection considering SER as a function of SNR. (a) Comparison between our annealed underdamped and third-order Langevin method and the overdamped variant for $L \in \{5, 10, 20\}$. (b) Comparison with classical and learning-based detection methods. (c) Comparison with different detection methods for an SNR = 16 dB and varying the running time in ms/symb. 59

3.2 Performance analysis of our proposed methods for channel estimation considering NMSE as a function of SNR in a CDL-C channel model. (a) Comparison of our proposed annealed underdamped and third-order Langevin methods using different discretization schemes. (b) Comparison of our proposed annealed underdamped and third-order Langevin method and the annealed overdamped Langevin. (c) Comparison with baseline methods. 64

4.1 Illustration of Repulsive Latent Score Distillation (RLSD): It propagates a set of particles by adding noise and applying two levels of regularization: (i) **Denoising**, via score-matching regularization, which directs particles toward modes of the distribution $p(\mathbf{x}_0|\mathbf{y})$ (blue arrows); and (ii) **Repulsion**, which pushes particles apart (red arrows) to explore other regions of the posterior density. During sampling, the repulsion gradient ensures particles remain separated, leading to different modes, as shown in the upper-right box. 71

4.2 Inpainting half a face (top to bottom): Ground truth, measurement, PSLD, NonRepuls-RLSD, and RLSD ($\gamma = 50$). Each method generates four samples from different initializations, with RLSD incorporating repulsion. Both RLSD and NonRepuls-RLSD outperform PSLD. RLSD produces diverse results (e.g., images 1 and 3 hide the left eye), whereas NonRepuls-RLSD samples appear similar. 84

4.3 Results for Phase Retrieval. Adding repulsion between particles allows to sample from different modes (top row). 87

5.1 Comparison of constraint types. Left: Our setting, where convex obstacles carve out forbidden regions, yielding a non-convex feasible space from which we must sample. Right: The classical case, where sampling is restricted to a convex feasible region within the ambient space. 92

5.2 Impact of parameter α on shielded LMC with obstacles. Blue dots are Langevin samples after 50k iterations, red dashed lines are obstacles, and black crosses mark mode means; contours show the target distribution.(a) $\alpha = 0.1$: repulsion is too strong, pushing samples away from true modes. (b) $\alpha = 1$: good balance between target sampling and obstacle avoidance. (c) $\alpha = 7$: repulsion is too weak, causing samples to collapse onto obstacle boundaries where gradients vanish. 97

5.3 Performance analysis of shielded LMC for MIMO symbol detection, considering SER as a function of SNR in a Rayleigh fading channel model. (a) QPSK constellation with obstacles. (b) Ablation of the obstacle-strength parameter α. (c) Comparison against the unconstrained Langevin sampler and the optimal ML detector. 102

6.1 Overview of the **Prior-Informed Flow Matching (PIFM)** graph reconstruction framework. Starting from a partially observed adjacency matrix $\mathbf{A}^{\mathcal{O}} = \xi \odot \mathbf{A}$, where ξ denotes a mask, we form an initialization $\mathbf{A}_0$ by combining the observed entries with prior predictions $f_{\text{prior}}(\mathbf{A}^{\mathcal{O}})$ obtained with an element-wise predictor. In dark red we denote the true edges that are masked, while in light red those masked positions where there is no edge between nodes. A rectified flow then interpolates linearly from $\mathbf{A}_0$ to the ground-truth graph $\mathbf{A}_1 = \mathbf{A}$, learning global structural information from a coupling of all the edges. The intermediate states $\mathbf{A}_t$ improve on the prior-informed initialization, enabling recovery of the missing edges. 112

6.2 Toy experiment showcasing the advantage of PIFM (in this case, for link prediction). (a) Graph $\mathcal{G}$ with four nodes, where the hidden edges are e_{02} and e_{13}. (b) Samples generated by node2vec vs. PIFM: PIFM learns a probabilistic coupling and generates only the two valid modes. (c) Proportions of samples generated with PIFM from each mode; the method also learns a good approximation of the probability of each mode. 126

6.3 Empirical DP behavior on the ENZYMES expansion experiment. (a) MMD measures distance to the reference graph distribution and is lower after additional integration in this setting. (b–d) At $K = 100$, degree, triangle, and clustering-coefficient statistics more closely match the ground truth than at $K = 1$. This dataset-specific comparison is not a claim that MMD decreases monotonically with K in every setting. Error bars show the standard deviation over 300 samples (10 per test graph). 133

7.1 Illustration of possible paths in image space (left) and covariance space (right). Isotropic models are limited to covariance schedules that are reparameterizations of the diagonal (orange), while anisotropic models can explore paths where different signal components are noise-corrupted at different rates (red, black, and purple). 140

7.2 Proposed architecture based on UNet. We incorporate the covariance information through the embedding network, with two dedicated branches for the two covariance domains (spatial and spectral). 145

7.3 Inpainting results on CelebA. DPS and our method generate sharper images while RED-Diff's solutions are smoother, which is consistent with our analysis of their posterior density values in Section 7.4.1. . . 149

7.4 **Left and middle:** Histograms of $\log p_\theta(\hat{\mathbf{x}})$ and $\log p_\theta(\hat{\mathbf{x}}|\mathbf{y})$ for inpainting solutions $\hat{\mathbf{x}}$ generated by DPS, RED-Diff, and our energy model from a given measurement $\mathbf{y}$, along with the ground truth $\mathbf{x}$. Our energy model is well-calibrated with respect to both prior and posterior probabilities. **Right:** Examples of generated images $\hat{\mathbf{x}}$ sorted from lowest to highest prior probability (in reading order). The colored border indicates the sampler used to produce the image (green for DPS, red for RED-Diff, blue for ours) and its prior probability (darker shades for lower probability images), matching the arrows in the left panel. Note that higher-probability images are smoother and less detailed. The ground truth and measurements are shown on the left with blue/black borders, respectively. 150

7.5 Three autoregressive generations for a model trained on MNIST, with identical initial and injected noise. The first two rows generate the four quadrants of the image respectively in reading and reversed reading order. The third row generates the image in 16 patches, ordered from the center to the outer border. 152

7.6 Classification error as a function of the number of measurements. The energy-guided scheduling (7.12, orange) consistently achieves a lower error rate than the fixed geometric scheduling $\boldsymbol{\delta\Sigma} \propto \boldsymbol{\Sigma}$ (blue) with few measurements, with both methods reaching the baseline classifier error at 300 measurements. 154

7.7 Blind reconstruction experiment. **Top:** two observations corrupted by noise of standard deviation σ_1 inside a central $s \times s$ square, with $(\sigma_1, s) \in \{(0.1, 14), (2, 20)\}$. **Middle and bottom:** Log probability $\log p_\theta(\mathbf{y}|\mathbf{\Sigma}_s)$ as a function of box size s and σ_1, respectively. The noise covariance parameters estimated by the energy model trained with dual score matching (red dashed vertical lines) accurately track their ground truth values (solid green vertical lines). 157

8.1 PnP with average denoisers on a 2D mixture-of-Gaussians inverse problem ($\mathbf{y} = \mathbf{H}\mathbf{x} + \sigma\boldsymbol{\epsilon}$, with $\mathbf{H}$ a rotation+scaling operator and $\sigma = 0.3$). $D_{s,t}$ denotes the average denoiser (defined formally in Section 8.3.1). The lookahead t controls the DP trade-off: at $t = s$ the denoiser is MMSE and PnP reconstructions concentrate at the posterior mean, which for this symmetric bimodal posterior lies between the two modes, yielding low distortion but poor perceptual quality; as $t \to 1$, the lookahead progressively restores the bimodal structure, recovering samples from both modes at the cost of higher distortion; details of this experiment are in Appendix E.3.4. 161

8.2 Analysis of the perception-distortion DP trade-off and variance restoration on CelebA 128×128, quantified as RMSE/Var ratio vs FID. **(a)** DP trade-off at different noise levels. Each curve is swept by varying the lookahead parameter t using a single trained model. **(b)** The near-collapse of curves across noise levels indicates a universal relationship between variance restoration and perceptual quality (FID) improvement. 167

8.3 Across five linear inverse problems on AFHQ (Gaussian/motion deblurring, box/random inpainting, SR×4), we sweep the lookahead t within a PnP solver and report RMSE (distortion) versus FID (perception). Each marker corresponds to a value of t, with small lookaheads favoring low distortion and large ones favoring low FID. A single trained model spans the full frontier across all tasks, without retraining or guidance tuning; reconstructions on the right illustrate the perceptual change as t increases (larger version in Fig. E.27). 175
8.4 DP behavior on JPEG compression. 177

B.1 Phase Retrieval. Adding repulsion allows to sample from different modes. 201
B.2 Inpainting half face using (from top to bottom): Ground truth and Measurement, PSLD, NonRepuls-RLSD, and RLSD ($\gamma = 50$). We generate four samples for each method from a different initialization; for RLSD, they interact through the repulsion term. First, NonRepuls-RLSD, and RLSD outperforms PSLD for all four images. Second, image 4 from RLSD (last row) looks different (has brown eye), while the four samples of NonRepuls-RLSD have blue eye; this illustrates that RLSD promotes diversity. 202
B.3 Random inpainting (80%): PSLD (top) and NonRepuls-RLSD (down), and four different samples for each method. Notice that NonRepuls-RLSD generates a better reconstruction of the background. 203

B.4 Qualitative examples of Inpainting with free mask on Imagenet. We consider PSLD (third column) and NonRepuls-RLSD (forth column). For rows 2, 4-7, RLSD outperforms PSLD by a large margin. Furthermore, notice that RLSD can reconstruct backgrounds that might be difficult, such as the one row 4. On the other hand, RLSD struggles to reconstruct fine-details, as it is shown in row 3. 204

B.5 Reconstruction for different number of steps. a) Ground-truth (left) and Measurement (right). b) The two on the left are the reconstruction using PSLD with 200 steps (running time = 1.9 min), while the two of the right are with RLSD ($\gamma = 0$) (running time = 1.75 min). c) The two on the left are the reconstruction using PSLD with 1000 steps (running time = 9.2 min), while the two of the right are with RLSD ($\gamma = 0$) (running time = 8.8 min). 206

B.6 Reconstruction as a function of L (number of steps in the optimization). Increasing L modifies the number of denoisers that are used (the limits of the interval are the same, and it changes the step-size). Also, we need to decrease l_{r_z} (from 1 to 0.8) as well the coupling term $\tilde{\rho}$, from 0.15 to 0.1. 207

B.7 Reconstruction as a function of γ (number of repulsion), for $L = 200$ and $N = 4$. Increasing γ enhances diversity: for $\gamma = 150$, 2 out of 4 samples show the mouth, while for $\gamma = 0$ all samples have the mouth hidden. 208

B.8 Reconstruction for $\gamma = 150$, for $L = 200$, $N = 4$ and repulsion between 0 and t_{repul}: when $t_{\text{repul}} = T$, we have repulsion in all the trajectory, while $t_{\text{repul}} = 0$ corresponds to NonRepuls-RLSD. 209

B.9 Reconstruction for $\gamma = 150$, for $L = 200$, $N = 4$ and repulsion for $t \in [t_{\text{repul}}, 200]$; notice that $t_{\text{repul}} = T$ corresponds to NonRepuls-RLSD. 210

B.10 Reconstruction when increasing the number of particles in the repulsion, for $N = 0, 2$ and 4. Notice that $N = 0$ corresponds to the NonRepuls-RLSD, $N = 2$ to Hybrid case, and $N = 4$ to RLSD. 211
B.11 Reconstruction when increasing the number of particles in the repulsion, for $N = 0, 2$ and 4. Notice that $N = 0$ corresponds to the NonRepuls-RLSD, $N = 2$ to Hybrid case, and $N = 4$ to RLSD. In this case, we need full repulsion to increase diversity w.r.t. the NonRepuls-RLSD case. 212
B.12 Comparison between RLSD (top row of Fig. b) and NonAug-RLSD (last row of Fig. b). RLSD achieves a sharper reconstructed image; in particular, the shirt for the case of NonAug-RLSD has not fine-detail, while RLSD has. 213
B.13 Results of the reconstruction of an image from the validation set of ImageNet using RLSD (b) and RED-diff with diffusion prior trained on FFHQ (c). This example showchases that using a large-pre trained model such as Stable Diffusion enables to use our method with images from very different classes (Imagenet and FFHQ). 214
B.14 Reconstruction as a function of ρ (coupling parameter). Increasing ρ enhances diversity but reduces quality: for $\rho = 0.12$, the 4 samples looks different but with a low quality, while with a lower ρ, the diversity is lower the performance is better. 216

D.1 Examples of particular $\mathbf{\Sigma}_t$ for different kernels and masks. 237
D.2 Examples of autoregressive patch-based covariances $\mathbf{\Sigma}_t$ used for training. We first sample the number of boxes, and then the variance of each box. 237
D.3 Noisy observation with a box covariance of size 30×30 and $\sigma_1 = 1$. . 244

D.4 Blind reconstruction experiment. Log probability $\log p_\theta(\mathbf{y}|\mathbf{\Sigma}_s)$ as a function of box size s and σ_1, respectively. The energy model trained with dual score matching correctly identifies the true box size (dashed red for estimated size and solid green vertical lines). 244

D.5 Log probability $\log p_\theta(\mathbf{y}|\mathbf{\Sigma}_s)$ as a function of box size s for $s = \{5, 14, 20\}$ with $\sigma_1 = \{2, 0.1, 2\}$ respectively. The energy model trained with dual score matching correctly identifies the true box size (dashed red for estimated size and solid green vertical lines), which is not the case for the model trained with single score matching (dashed blue vertical line). 245

D.6 MMSE for one-shot denoising in box inpainting (10×10) as a function of the noise in the box. 246

D.7 MMSE estimator computed by our model, which corresponds to $\hat{\mathbf{x}} = \mathbb{E}[\mathbf{x}|\mathbf{y}, \mathbf{\Sigma}]$. 246

D.8 Histograms of $\log p_\theta(\hat{\mathbf{x}})$ and $\log p_\theta(\hat{\mathbf{x}}|\mathbf{y})$ for inpainting solutions $\hat{\mathbf{x}}$ generated by DPS, RED-Diff, and our energy model for different measurements $\mathbf{y}$ associated to different $\mathbf{x}$, along with the ground truths $\mathbf{x}$. Our energy model is well-calibrated with respect to both prior and posterior probabilities. 247

D.9 Images sorted from high to low probability (from left-up corner to right-down corner). Images with highest prior probability contains less details than those associated with higer probability. 247

D.10 ImageNet posterior samples for a 21×21 center inpainting task, sorted by decreasing prior probability. Consistent with previous observations, samples with the highest probability feature uniform backgrounds and minimal detail, whereas lower-probability samples exhibit significantly higher structural complexity and dense detail. . . 248

D.11 ImageNet posterior samples (red border) generated by sampling from measurements $\mathbf{y}$ generated using a box inpainting mask with a box of size 21×21, and sorted from high to low posterior probability, next to their corresponding ground truth (blue border). 249
D.12 Histogram of log probabilities $\log p_\theta(\mathbf{x})$[dB/dim] for 50k in the ImageNet test set. The distribution is well-fit by a Gumbel distribution (red line), but there is a second mode (centered near 38 dB/dim) in contrast to the model in [1]. This additional mode is caused by grayscale images within ImageNet; these samples possess significantly higher probability under the model U_θ compared to standard RGB samples, creating an additional mode. 250
D.13 Images from the test set of ImageNet 64×64, sorted from low to high probability (left to right and top to down). 250
D.14 Comparison of single and dual score matching on a 1000-dimensional Gaussian scale mixture. Left: Dual score matching (red dashed) captures the true energy (blue solid), whereas single score matching (green) fails, even after global normalization. Right: Radial score components. Single score matching learns the score accurately within the data support (blue bar plot) but lacks accuracy outside of this region. 252
D.15 Examples of inpainting for CelebA. 253
D.16 Examples of inpainting for ImageNet. 254
D.17 Examples of deblurring for CelebA 255
D.18 Examples of deblurring for ImageNet 256
D.19 Examples of super-resolution for CelebA 257
D.20 Examples of inpainting for AFHQ-cats. 258
D.21 Examples of Motion Deblurring. 259

D.22 Comparison of the different corrector samplers a function of the number of steps. 260

E.1 Visual comparison for $\alpha = 0.3$. From left to right: clean image, noisy input, and reconstructions at increasing lookahead. The estimator smoothly transitions from a blurry conditional mean (MMSE) to a sharp sample, progressively restoring texture, contrast, and color saturation. 267

E.2 Distortion–perception comparison on a 2D mixture of Gaussians at three noise levels $s \in \{0.3, 0.5, 0.7\}$ (heavy, moderate, light noise). Blue: average denoiser $D_{s,t}$ swept over lookahead $t \in [s, 1]$. Red: [1] optimal DP estimator swept over perception level $\alpha \in [0, 1]$, computed via exact W_2-optimal transport between the MMSE pushforward and p_1. Both curves share the MMSE endpoint exactly (★); small residual gaps at the perfect-perception extreme (▲) reflect the velocity singularity near $\tau = 1$. The near-optimality of $D_{s,t}$ in the interior confirms that the flow-map trajectory closely approximates the W_2 geodesic even in a non-Gaussian setting, bridging Theorem 2 to the image experiments in Section 8.3.3. 272

E.3 The composite bound $s\, L_D(t)\, L_F$ exceeds 1 across all (s, t), so T_t is never a global contraction — empirical convergence relies on the local analysis of Theorem 3. $L_D(t)$ is estimated via power iteration on $\partial D_{s,t}/\partial \mathbf{x}$ at noisy samples (worst case over 20). The bound increases with t, with steepest growth at low noise (large s). 275

E.4 Two forward models are tested: **(a)** isotropic rotation $\mathbf{H}_1 = 0.8\,R(\pi/6)$ with $\lambda = 0.1$, and **(b)** anisotropic operator $\mathbf{H}_2 = R(\pi/6)\,\mathrm{diag}(1.0, 0.3)\,R(\pi/4)$ with $\lambda = 0.5$. Black: measurements $\mathbf{y}$. Gold crosses: ground truth $\mathbf{x}_{1,i}$. Yellow: Wiener/MAP estimator. Red dots: Tikhonov fixed point $\mathbf{x}^\star(t)$ predicted by Eq. (E.24). Red $\times$: PnP iterates after convergence (matching theory to machine precision, $\|\hat{\mathbf{x}} - \mathbf{x}^\star(t)\|_{\text{mean}} < 5 \cdot 10^{-16}$). Blue dots: samples from the stochastic PnP variant, illustrating the variance restoration discussed in Section 8.2.2. As t increases (left to right), the Tikhonov regularization weakens and the iterates spread further from the prior mean toward the Wiener solution. 279

E.5 Ablation of PnP-flow as a function of the number of steps with Poisson noise and Gaussian deblurring 291

E.6 Progression of random inpainting with lookahead $t = s$ 291

E.7 Progression of random inpainting with lookahead $t = 1$ 292

E.8 Progression of SR $\times 4$ with lookahead $t = s$ 292

E.9 Progression of SR $\times 4$ with lookahead $t = 1$ 293

E.10 **Distortion–perception trade-off traced by a single flow map on CelebA.** Same setup as Fig. 8.3, but on CelebA 128×128. For each of four linear inverse problems (Gaussian deblurring, motion deblurring, box inpainting, and super-resolution), we sweep the lookahead t of our average denoiser within a PnP solver and report RMSE (distortion, $\downarrow$) against FID (perception, $\downarrow$). Each marker corresponds to a different value of t; small lookaheads cluster near the low-distortion regime (top of each curve) while large lookaheads cluster near the low-FID regime (right). The same trained model spans the entire frontier across all four tasks. Reconstructions on the right illustrate the perceptual change as t increases. 294

E.11 AFHQ - Super-resolution ×4. Ablation of PnP-flow as a function of the number of steps. 295
E.12 AFHQ - Gaussian deblurring. Ablation of PnP-flow as a function of the number of steps. 296
E.13 AFHQ - Box inpainting. Ablation of PnP-flow as a function of the number of steps. 296
E.14 CelebA - Motion deblurring. Ablation of PnP-flow as a function of the number of steps. 296
E.15 Performance as a function of the number of steps 297
E.16 Reconstruction for phase retrieval 298
E.17 Gaussian deblurring with kernel size 61×61 and $\sigma_b = 3$, and $\sigma = 0.05$ 299
E.18 Motion deblurring with kernel size 61×61 and $\sigma_b = 1$, and $\sigma = 0.1$. 300
E.19 Super resolution ×4, and $\sigma = 0.05$ 301
E.20 Random inpainting with 90% drop rate, and $\sigma = 0.01$ 302
E.21 Box inpainting with mask 80×80, and $\sigma = 0.01$ 303
E.22 JPEG compresion 304
E.23 CelebA - Super resolution ×4, and $\sigma = 0.05$ 305
E.24 CelebA - Motion deblurring 306
E.25 Gaussian deblurring with Poisson noise 307
E.26 Colorization 308
E.27 Larger version of the images in Fig. 8.3. 309

Tables

2.1 Mapping of thesis contributions to identified failure modes and design knobs. The first three rows are *inference-time* algorithms on top of a pre-trained diffusion or flow prior (knob 4 of Section 2.7); the last three are *training-time* modifications in which the prior itself is modified along knobs 1–3. 31

3.1 Hyperparameters of the different algorithms based on Langevin dynamics. 57

3.2 Quantitative results on noiseless images ($\sigma_0 = 0.0002$) from FFHQ. We consider the average PSNR[dB], LPIPS and FID of Gaussian deblurring, inpainting and $4 \times$ SR. In bold is the best method for each metric and experiment. 67

3.3 Quantitative results on noisy ($\sigma = 0.05$) images from FFHQ. We consider the average PSNR[dB], LPIPS and FID of Gaussian deblurring, inpainting and $4 \times$ SR. In bold is the best method for each metric and experiment. 67

4.1 Comparison of our work with DPS, RED-Diff, PSLD, VSD, and CSD. Our method combines the strengths of variational samplers (no score Jacobian, computational efficiency) and posterior sampling algorithms (diversity at high dimensions). In addition, our formulation enables us to solve nonlinear inverse problems at 512×512. 72

4.2 Box inpainting (half face) with $\sigma_0 = 0.001$ - FFHQ 512. For evaluation, we downsample the estimated images of RLSD and PSLD to 256. The best method for each metric is bolded. 85

4.3 Phase-retrieval with $\sigma_0 = 0.001$ on FFHQ 512. The best method for each metric is bolded. 86

4.4 HDR with $\sigma_0 = 0.001$ on FFHQ 512. The best method for each metric is bolded. 87

6.1 Notation used in PIFM. 115

6.2 Graph reconstruction with **50% of unordered node pairs masked**. Values are five-seed means ± sample standard deviations in percent; RePaint values are single runs. Stage-1 rows are frozen edge predictors evaluated directly, and each PIFM row uses the named predictor unchanged as its source. **Best** and second-best are highlighted for the threshold-free metrics (AP, AUC); FNR and FPR are reported at the fixed 0.5 threshold, where they trade off against each other, and are therefore not ranked. 129

6.3 Cora results over five seeds. AUC measures held-out-edge ranking; the remaining columns are subgraph-pool MMD^2 values. **Bold blue** values indicate the best result(s) in each metric: higher is better for AUC, while lower is better for MMD^2. The prior is repeated because each configuration induces a different subgraph pool. 130

6.4 Performance for the **expansion** task with **50% of edges masked**. Values are five-seed means ± sample standard deviations in percent; † marks the RePaint baselines, which are single runs. Metrics are computed on the region the flow updates, $1 - \mathbf{A}^{\mathcal{O}}$. FNR and FPR use the fixed 0.5 threshold. 132

7.1 Quantitative results for inpainting and Gaussian deblurring on **CelebA** 64×64. Best results are in **bold**. 147
7.2 Quantitative results for inpainting and Gaussian deblurring on **ImageNet** 64×64. Best results are in **bold**. 147
7.3 LPIPS distance between ground truth $\mathbf{x}$ and reconstruction $\hat{\mathbf{x}}$ for different correction schemes on a CelebA inpainting task. 156

8.1 Quantitative results for random-mask inpainting (90%, $\sigma = 0.01$) and motion deblurring ($\sigma = 0.05$) on **CelebA** 128×128. Best results are in **bold**; second-best underlined. 173
8.2 Quantitative results for random-mask inpainting (90%, $\sigma = 0.01$) and motion deblurring ($\sigma = 0.05$) on **AFHQ** 256×256. Best results are in **bold**; second-best underlined. 173

B.1 Baselines used in the RLSD experiments. Pre-trained diffusion models: FFHQ from [2]; ImageNet from [3]. Latent RED-Diff uses the same variational sampler as RED-Diff but with a latent diffusion prior (Section 4.3). 194
B.2 SR $\times$ 8 with $\sigma_0 = 0.001$ - FFHQ 512. The best method for each metric and experiment is bolded. 196
B.3 SR $\times$ 8 with $\sigma_0 = 0.001$ - FFHQ 256. The best method for each metric and experiment is bolded. 196
B.4 SR $\times$ 8 with $\sigma_0 = 0.001$ - FFHQ 256. The best method for each metric and experiment is bolded. 197
B.5 Motion Blurring with $\sigma_0 = 0.001$ - FFHQ 512. The best method for each metric and experiment is bolded. 197
B.6 Motion Blurring with $\sigma_0 = 0.001$ - FFHQ 256. The best method for each metric and experiment is bolded. 198

B.7 Box inpainting (half face) with $\sigma_0 = 0.001$ - FFHQ 256. The best method for each metric and experiment is bolded. 198
B.8 Random inpainting with (80%) mask and with ($\sigma_0 = 0.001$) - FFHQ 512. In bold is the best method for each metric and experiment. . . . 199
B.9 Random inpainting with (80%) mask and with ($\sigma_0 = 0.001$) - FFHQ 256. In bold is the best method for each metric and experiment. . . . 199
B.10 Free mask [4] with ($\sigma_0 = 0.001$) - ImageNet 512. In bold is the best method for each metric and experiment. 200
B.11 Box inpainting (half face) with $\sigma_0 = 0.001$ - FFHQ 512. We consider the best particle across each batch of them. 201
B.12 Ablation γ for inpainting with free-mask. 208
B.13 Ablation when changing the number of particles in the repulsion term with $\sigma_0 = 0.001$ - FFHQ 512. 211
B.14 Ablation ρ for inpainting with free-mask - $\gamma = 0$ 215

C.1 Statistics of the datasets used for evaluation. 221
C.2 Training time of the flow model for 100 epochs on each dataset under 50% mask rate. 226
C.3 Per-graph inference time of PIFM as a function of the number of Euler steps. 227
C.4 Per-graph inference time comparison between standalone priors and PIFM. 227

D.1 SongUNet Architecture Summary 234
D.2 Training hyperparameters for CelebA and ImageNet64 236
D.3 Training hyperparameters for AFHQ 236
D.4 Inpainting - Hyperparameters for the Predictor-Corrector Sampler with fixed corrector. 239

D.5 Experiments on CelebA 64×64: Comparison on super-resolution $\times 4$ and half-mask inpainting. 241
D.6 Experiments on CelebA 64×64: Comparison on random inpainting (0.7%) and vertical half-mask inpainting. 242
D.7 Experiments on CelebA 64×64: Comparison on motion deblurring. . . . 242
D.8 Box 30×30 243
D.9 Experiments on CelebA 64×64: Comparison on box inpainting 30×30. . 243
D.10 Box 50×50 243
D.11 Experiments on CelebA 64×64: Comparison on box inpainting 50×50. . 243

E.1 Per-run training configuration. 283
E.2 Running time for each method. PnP with a flow map as a denoiser is the fastest method, while generating images that are closer to those from posterior sampling methods. 285
E.3 Quantitative results for random inpainting (90%) with $\sigma = 0.01$ on CelebA and AFHQ. Best results are in **bold**. 287
E.4 Quantitative results for motion deblurring with $\sigma = 0.05$ on CelebA and AFHQ. Best results are in **bold**. 287
E.5 Quantitative results for Gaussian deblurring w/$\sigma = 0.05$ across CelebA and AFHQ datasets. Best results are in **bold**. 288
E.6 Quantitative results for super-resolution $\times 4$ w/$\sigma = 0.05$ across CelebA and AFHQ datasets. Best results are in **bold**. 288
E.7 Quantitative results for box inpainting w/$\sigma = 0.01$ across CelebA and AFHQ datasets. Best results are in **bold**. 289
E.8 Quantitative results for JPEG w/$\sigma = 0.01$ on AFHQ datasets. Best results are in **bold**. 289
E.9 Quantitative results for Colorization w/$\sigma = 0.001$ on AFHQ datasets. Best results are in **bold**. 290

E.10 Quantitative results for Phase retrieval w/$\sigma = 0.01$ on AFHQ datasets. Best results are in **bold**. 297

Chapter 1

Introduction

1.1 Generative models and inverse problems

The last five years, which represent the time frame of this thesis, have witnessed a dramatic reshaping of the world where we live, with *generative models* as the centerpiece of this revolution. Large language models trained on internet-scale text, text-to-image systems trained on billions of images, and multi-modal systems have demonstrated that models trained to approximate the distribution of natural data can be used as remarkably general tools. Under the hood, all of these systems share a common recipe: given samples from an unknown data distribution, learn a parametric model that approximates it, and use the trained model to synthesize new samples that resemble the training data.

Synthesizing realistic content *per se* is impressive, but the scientific and engineering value of generative models comes from what one can do *with* them. In this thesis, we ask a specific instance of that question: can the strong priors learned by modern diffusion models be used to solve *inverse problems*, in which the goal is to recover an unknown signal from a noisy, incomplete, or otherwise degraded measurement?

Inverse problems are pervasive across science and engineering, which renders this question general enough to encompass a wide range of applications. For instance, in wireless communications, one wants to detect the transmitted symbols and estimate the channel of a massive multiple-input multiple-output (MIMO) receiver from noisy

pilot and data observations; in image restoration, one restores a clean image from a blurred, downsampled, inpainted, or otherwise degraded observation; in network science, one seeks to recover the topology of a graph from a partial set of edge observations. In every case the reasoning is Bayesian: a physical forward model links the unknown signal to the measurement, and a prior over the signal disambiguates the many candidate reconstructions consistent with the same observation.

Modern inverse problems raise two intertwined difficulties. First, the signals are *high-dimensional* — high-resolution images, multiple antennas in MIMO systems, graphs with thousands of nodes — so exact posterior computation is intractable and one must resort to iterative approximations or samplers. Second, they are *ill-posed*: many candidate signals are consistent with the same measurement, so recovery hinges on the prior. The choice and use of the prior is therefore the central design decision, and the more expressive the prior, the closer the reconstruction can come to the true signal.

1.2 Diffusion models as priors

Classical priors, such as sparsity in a chosen basis, total-variation regularization, or low-rank structure, are interpretable and cheap, but capture only a small fraction of the structure present in natural signals. Learned priors close this gap by learning from data. In a nutshell, these models are parametrized by a neural network whose parameters are learned from a large dataset of examples, and they can capture the complex, high-dimensional structure of natural signals. Among learned priors, *diffusion models* and their flow-matching counterparts have emerged as the most compelling choice: they provide a simple and scalable training loss, produce high-quality and diverse samples, and can be composed with a variety of downstream inference schemes [5–8].

Intuitively, a diffusion model learns a family of denoisers spanning a continuum of noise levels; running the reverse process transports samples from a simple noise distribution (e.g., a Gaussian distribution) into the data distribution. Chapter 2 introduces these ideas formally.

Combining this *generative* prior with the physical forward model of an inverse problem, one can in principle simulate a conditional reverse process that targets the posterior over the unknown signal given the observation. In practice, this idealized recipe breaks down in several ways. The goal of this thesis is to provide a set of algorithms and solutions to this problem, which split cleanly by *where* we choose to intervene.

Inference-time algorithms on pre-trained priors

The first part of the thesis focuses on the *sampling algorithm*, where the key assumption is that a pre-trained, off-the-shelf diffusion prior is known and given beforehand. Given this generative diffusion prior, we attempt to sample from the posterior for a given inverse problem. In this context, we propose three different samplers that address three distinct aspects of modern inverse problem solvers:

- **The sampler is slow.** Simulating the reverse process requires hundreds to thousands of evaluations of the score network. This is prohibitive for real-time systems (e.g. wireless receivers) and dominates the offline cost of inverse-problem solvers.

- **Variational samplers collapse with latent diffusion models.** When the diffusion is defined in a latent space, as in state-of-the-art text-to-image models, variational posterior-sampling schemes concentrate on a single mode of the

posterior, producing blurry or unrepresentative reconstructions.

- **Inability to enforce hard constraints.** Many applications carry hard structural requirements — forbidden regions in a MIMO constellation, privacy or copyright exclusions — that a naive posterior sampler is unable to enforce with probability one.

Training priors for improved inference

The second part of the thesis focuses on the *generative model* itself, where one has the freedom to design and train a new generative model from scratch. In this context, we identify three distinct ways in which the generative model can be modified to design better and tailored solvers for inverse problems:

- **The source distribution is task-agnostic.** Standard diffusion and flow matching are initialized with a Gaussian distribution and evolve into the data distribution. When strong task-specific side information is available (e.g., a classifier, a spectral embedding, a learned approximation of the posterior mean), replacing the Gaussian source with a task-informed one reduces the ill-posedness of the inverse problem.

- **Only the score is learned, not the density.** Standard denoising score matching estimates the gradient of the log-density, but not the log-density itself: the normalization constant is inaccessible. We show that energy models, combined with an extension of denoising score matching to matrix-valued *anisotropic noise*, unlock a companion covariance score which, together with the data score, makes the normalization tractable via a dual score-matching objective.

- **Only the instantaneous velocity is learned.** Standard flow matching learns the instantaneous velocity of a stochastic interpolant, which must be integrated over hundreds to thousands of small steps to transport samples from the source to the data distribution. *Flow maps* instead learn an *average velocity* between arbitrary time pairs, transporting samples in a single evaluation. We train a flow map and use its average/two-time *denoiser* form as a plug-and-play denoiser for inverse problems, showing that its lookahead gives a principled knob for traversing the distortion–perception plane.

The two lists differ in kind. The first is about the *sampling algorithm*: given a fixed model, how do we run it, and how can we improve the overall sampling process? The second is about the *model*: given an inverse problem in a particular domain, how should we design and train it? These two complementary approaches — inference-time algorithmic scaling on top of a classical diffusion or flow prior, versus training a modified diffusion or flow prior tailored to the task — organize this thesis.

1.3 Thesis contributions and organization

Part I — Inference-time algorithms on pre-trained priors

Part I takes a standard, off-the-shelf diffusion or flow-matching prior and modifies only the sampling algorithm. Each chapter addresses one failure mode of naive posterior sampling we described above:

1. *Higher-order annealed Langevin diffusion* [9] (Chapter 3) attacks slow mixing by augmenting the Langevin sampler with auxiliary variables and pre-conditioning. It is applied to massive MIMO detection, channel estimation, and image restoration.

2. *Repulsive latent score distillation* [10] (Chapter 4) attacks mode collapse in variational posterior sampling with latent diffusion priors. We recast posterior sampling as a variational optimization problem and introduce a repulsive coupling across particles that provably prevents concentration on a single mode.

3. *Shielded Langevin Monte Carlo with navigation potentials* [11] (Chapter 5) attacks the inability of standard samplers to respect hard constraints. We augment Langevin dynamics with a barrier-like potential inspired by robot navigation, enabling sampling that provably avoids user-specified forbidden regions.

The common thread is that all three methods leave the underlying generative model untouched. This is a form of *inference-time scaling*: given a fixed classical prior, better inverse-problem solvers can be built purely by investing more (or smarter) compute at sampling time.

Part II — Training priors for improved inference

Part II keeps the sampling recipe simple but *modifies the model itself*. We identify three orthogonal ways to modify a diffusion or flow-matching model, and develop one contribution along each:

1. *Prior-informed flow matching* [12] (Chapter 6) modifies the *source distribution*. A generic flow model transports a Gaussian source into the data distribution; we replace the Gaussian with a task-informed source that already carries structural information about the graph inverse problem at hand, and train the flow to complete the reconstruction.

2. *Energy-based models with anisotropic noise* [13] (Chapter 7) learn a *normalized density*, not only its gradient. We extend denoising score matching to matrix-

valued anisotropic noise via a dual score-matching objective; the companion covariance score it enables makes the density normalization tractable, and unlocks blind inverse problems, uncertainty quantification, and posterior-based comparison across solvers.

3. *Flow-map denoisers* [14] (Chapter 8) train a flow map and use its average/two-time *denoiser* form as a plug-and-play denoiser for image inverse problems. The lookahead of the trained flow map provides a principled, single-model traversal of the distortion–perception plane.

The common thread is that the modification is at training time. Rather than treat the source distribution, the training objective, and the training target for velocity as fixed choices, we turn each into a design variable and tune it toward the downstream inverse problem.

Design principle

Underlying both parts is a single design principle. A diffusion-based inverse problem solver has multiple degrees of freedom — the sampling algorithm, the source distribution, the training objective (score alone vs. score-plus-density), and the training target for velocity (instantaneous vs. average). Each of these can be treated as a fixed choice or as a design variable. This thesis argues that the most substantial improvements come from turning each one, in turn, into a lever and pulling it in the direction dictated by the downstream inverse problem.

1.4 Outline of the thesis

The remainder of the thesis is organized as follows.

Chapter 2 (Background) develops the mathematical machinery used throughout the thesis. It introduces the general formulation of inverse problems and the Bayesian view that puts a generative prior in the middle of the reconstruction problem; the continuous-time score-based formulation of diffusion generative models, including variance-exploding and variance-preserving stochastic differential equations and their flow-matching (ordinary differential equation) counterparts; latent diffusion models and their consequences for inverse problems; the denoising-score-matching training objective and Tweedie's formula; the machinery for using diffusion priors as posterior samplers for inverse problems; and the distortion–perception trade-off, which frames the reconstruction problem as a two-dimensional Pareto problem rather than a scalar minimization.

Part I (Inference-time algorithms on pre-trained priors) develops the three plug-and-play sampler modifications: higher-order annealed Langevin diffusion (Chapter 3) for accelerated posterior sampling, repulsive latent score distillation (Chapter 4) for mode-diverse posterior sampling with latent diffusion priors, and shielded Langevin Monte Carlo (Chapter 5) for constrained posterior sampling.

Part II (Training priors for improved inference) develops the three training-time modifications: prior-informed flow matching (Chapter 6) modifies the source distribution, energy-based models with anisotropic noise (Chapter 7) learn a normalized density via a dual-score-matching objective with matrix-valued noise, and flow-map denoisers (Chapter 8) train a flow map and use its average/two-time denoiser form as a plug-and-play denoiser for inverse problems. A short closing synthesis in Chapter 9 revisits the shared flow-matching machinery of Chapters 6 and 8.

Chapter 9 (Discussion and future work) synthesizes the contributions along the inference-time-vs-model axis, discusses the design principle just described, and outlines several open research directions: combining inference-time and training-time approaches, sharpening convergence guarantees under approximate scores, extending the framework to scientific inverse problems (MRI, cryo-EM, molecular design), and pushing the safety agenda beyond hard-constraint enforcement to privacy, copyright, and fairness constraints.

1.5 List of publications

The six research chapters of this thesis are adapted from the following papers, listed in the order in which they appear:

1. Nicolas Zilberstein, Ashutosh Sabharwal, and Santiago Segarra. "Solving Linear Inverse Problems using Higher-Order Annealed Langevin Diffusion." *IEEE Transactions on Signal Processing*, 2024 [9]. (Chapter 3)

2. Nicolas Zilberstein, Morteza Mardani, and Santiago Segarra. "Repulsive Latent Score Distillation for Solving Inverse Problems." *International Conference on Learning Representations (ICLR)*, 2025 [10]. (Chapter 4)

3. Nicolas Zilberstein, Santiago Segarra, and Luiz Chamon. "Sampling with Shielded Langevin Monte Carlo Using Navigation Potentials." *Asilomar Conference on Signals, Systems, and Computers*, 2025 [11]. (Chapter 5)

4. Harvey Chen, Nicolas Zilberstein, and Santiago Segarra. "Prior-Informed Flow Matching for Graph Inverse Problems." Accepted at Trans. Mach. Learn. Res. (also presented at SPIGM workshop at ICML 2026) [12]. (Chapter 6)

5. Nicolas Zilberstein, Santiago Segarra, Eero P. Simoncelli, and Florentin Guth. "Learning Normalized Energy Models for Linear Inverse Problems." ICML 2026 [13]. (Chapter 7)

6. Nicolas Zilberstein, Morteza Mardani, and Santiago Segarra. "Flow Map Denoisers: Traversing the Distortion–Perception Plane for Inverse Problems." Accepted at NeurIPS 2026 [14]. (Chapter 8)

Papers beyond the thesis. The six papers above form the core contributions of this thesis. During the PhD, the author also contributed to the following papers, on topics related to the material discussed here but not incorporated as chapters:

- Ali Azizpour, Nicolas Zilberstein, and Santiago Segarra. "Scalable Implicit Graphon Learning." *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2025 [15].

- Victor M. Tenorio, Nicolas Zilberstein, Santiago Segarra, and Antonio G. Marques. "Graph Guided Diffusion: Unified Guidance for Conditional Graph Generation." Under revision at IEEE Transactions on Signal and Information Processing over Networks [16].

- Fernando Gama, Nicolas Zilberstein, Martin Sevilla, Richard G. Baraniuk, and Santiago Segarra. "Unsupervised Learning of Sampling Distributions for Particle Filters." *IEEE Transactions on Signal Processing*, 2023 [17].

- Nicolas Zilberstein, Akshay Malhotra, Shahab Hamidi-Rad, and Yugeswar Deenoo. "Topology Preserving Regularization for Independent Training of Inter-operable Models." *UniReps: 2nd Edition of the Workshop on Unifying Representations in Neural Models*, 2024 [18].

- Martin Sevilla, Nicolas Zilberstein, Benjamin Cox, Sara Pérez-Vietes, Víctor Elvira, and Santiago Segarra. "State and Dynamics Estimation with the Kalman-Langevin Filter." *Asilomar Conference on Signals, Systems, and Computers*, 2023 [19].

- Fernando Gama, Nicolas Zilberstein, Richard G. Baraniuk, and Santiago Segarra. "Unrolling Particles: Unsupervised Learning of Sampling Distributions." *IEEE Intl. Conf. Acoust., Speech and Signal Process. (ICASSP)*, 2022 [20].

- Nicolas Zilberstein, Chris Dick, Rahman Doost-Mohammady, Ashutosh Sabharwal, and Santiago Segarra. "Accelerated massive MIMO detector based on annealed underdamped Langevin dynamics." *IEEE Intl. Conf. Acoust., Speech and Signal Process. (ICASSP)*, 2023 [21].

- Ali Azizpour, Nicolas Zilberstein, and Santiago Segarra. "Model-Driven Graph Contrastive Learning." *arXiv preprint arXiv:2506.06212*, 2025 [22].

Chapter 2

Background

This chapter introduces the mathematical tools used throughout the thesis. Section 2.1 formulates the inverse problem and the Bayesian view that places a generative model at the center of the reconstruction task through the prior distribution; it introduces the distortion–perception trade-off (Section 2.1.2) as the yardstick against which reconstructions are evaluated, and closes with a brief taxonomy of the generative-model families in which diffusion models sit. Section 2.2 develops diffusion models in their general continuous-time score-based formulation and Section 2.3 introduces latent diffusion models. Section 2.4 describes denoising score matching, the training objective that underlies almost every algorithm in this thesis, together with Tweedie's formula. Section 2.5 reviews the sampling algorithms used at inference time, focusing on DDPM, DDIM, and related methods. Section 2.6 shows how to use a pre-trained diffusion prior as a Bayesian posterior sampler for inverse problems, and reviews the main approximation strategies for the intractable likelihood term. Section 2.7 summarizes the preceding material into an explicit description of the four *design knobs* a diffusion-based inverse problem solver exposes; the entire thesis can be read as a study of how to exploit one knob at a time. Finally, Section 2.8 summarizes the contributions as a table, and Section 2.9 lists notation.

2.1 Inverse problems and generative priors

The goal in inverse problems is to recover an unknown signal $\mathbf{x}_0 \in \mathbb{R}^d$ from a measurement $\mathbf{y} \in \mathbb{R}^m$ produced by a physical *forward operator* $\mathcal{A} : \mathbb{R}^d \to \mathbb{R}^m$ corrupted by additive noise:

$$\mathbf{y} = \mathcal{A}(\mathbf{x}_0) + \mathbf{w}, \qquad \mathbf{w} \sim \mathcal{N}(\mathbf{0}, \sigma_0^2 \mathbf{I}). \tag{2.1}$$

In many settings of interest, such as massive multiple-input multiple-output (MIMO) receivers, computed tomography, magnetic resonance imaging (MRI), and pixel-space image restoration, the forward operator is (or is well-approximated by) a linear map, in which case $\mathcal{A}(\mathbf{x}_0) = \mathbf{H}\mathbf{x}_0$ for some matrix $\mathbf{H} \in \mathbb{R}^{m \times d}$. Other settings, such as non-linear phase retrieval, cryogenic electron microscopy, or optical imaging through turbulence, require the full non-linear $\mathcal{A}$. The number of measurements m is typically much smaller than the signal dimension d, or the operator $\mathcal{A}$ has a large null space; the problem is therefore *ill-posed* and admits infinitely many candidate reconstructions consistent with any given $\mathbf{y}$. Throughout this thesis we assume the forward operator $\mathcal{A}$ is known.

2.1.1 Bayesian formulation

From a Bayesian perspective, we can place a prior $p_{\mathbf{x}}(\mathbf{x}_0)$ on the unknown signal and let the likelihood $p_{\mathbf{w}}(\mathbf{y}|\mathbf{x}_0)$ be determined by (2.1). Bayes' rule then gives the posterior

$$p(\mathbf{x}_0|\mathbf{y}) \;\propto\; p_{\mathbf{w}}(\mathbf{y}|\mathbf{x}_0)\, p_{\mathbf{x}}(\mathbf{x}_0). \tag{2.2}$$

The posterior is the object of interest, and different reconstruction *estimators* correspond to different summaries of it. Two extremes stand out:

- *Maximum a posteriori (MAP)*: $\hat{\mathbf{x}}_0^{\mathrm{MAP}} = \operatorname{argmax}_{\mathbf{x}_0} p(\mathbf{x}_0|\mathbf{y})$. Minimizes zero–one distortion; often solved as an optimization problem with a data-fidelity term and a prior regularizer.

- *Posterior sampling*: $\hat{\mathbf{x}}_0 \sim p(\mathbf{x}_0|\mathbf{y})$. Samples reflect the ill-posedness of the problem, quantify uncertainty, and can be aggregated (e.g. via the posterior mean) into other estimators.

The posterior mean $\mathbf{x}^\star := \mathbb{E}[\mathbf{x}_0|\mathbf{y}]$ minimizes mean-squared error and is the minimum mean-squared-error (MMSE) estimator, but it typically produces overly smooth reconstructions that live off the data manifold. As formalized next, the choice between MAP-like, MMSE-like, and posterior-sample-like estimators is governed by a fundamental *distortion–perception trade-off*.

2.1.2 The distortion–perception trade-off

Historically, image inverse problems were evaluated purely on *distortion* metrics: given a reconstruction $\hat{\mathbf{x}}$ and the ground truth $\mathbf{x}$, the average distortion

$$D \;=\; \mathbb{E}[\Delta(\mathbf{x}, \hat{\mathbf{x}})] \tag{2.3}$$

measures a per-sample discrepancy Δ (mean-squared error, PSNR, SSIM, or a learned surrogate such as LPIPS [23]). The distortion-optimal estimator is the posterior mean $\mathbf{x}^\star = \mathbb{E}[\mathbf{x}_0|\mathbf{y}]$: it minimizes D under squared error, but as described above, averaging over the posterior smooths out the high-frequency content and produces reconstructions that appear visually blurry and off-manifold.

Perceptual metrics compare the *distribution* of reconstructions to the distribution of ground-truth signals rather than each reconstruction to a specific ground truth.

Given a divergence $d(\cdot, \cdot)$ between distributions (Wasserstein-2, total variation, or a surrogate such as FID [24] or KID [25]), the perceptual index

$$P \;=\; d(p_{\mathbf{x}}, p_{\hat{\mathbf{x}}}) \tag{2.4}$$

scores a reconstruction not by pixel proximity to the ground truth, but by how well its distribution matches the ground-truth distribution. Under P, the MMSE estimator scores poorly, whereas exact posterior samples — which live *on* the data manifold — yield an optimal perceptual score.

The Blau–Michaeli trade-off. Blau and Michaeli [26] formalized this tension as the *distortion–perception (DP) trade-off*. The DP function is the Pareto-optimal distortion achievable subject to a perceptual constraint:

$$D(P) \;=\; \min_{p_{\hat{\mathbf{x}}|\mathbf{y}}} \mathbb{E}[\Delta(\mathbf{x}, \hat{\mathbf{x}})] \quad \text{subject to} \quad d(p_{\mathbf{x}}, p_{\hat{\mathbf{x}}}) \;\leq\; P. \tag{2.5}$$

The function $D(P)$ is monotonically non-increasing and typically convex; the two objectives are fundamentally in tension. In particular, the MMSE estimator sits at the extreme low-distortion/high-perceptual-divergence end of the front ($D = D^\star$, $P = P^\star$), while an exact posterior sample sits at the low-divergence end ($D > D^\star$, $P = 0$).

For the MSE distortion $\Delta(\mathbf{x}, \hat{\mathbf{x}}) = \|\mathbf{x} - \hat{\mathbf{x}}\|_2^2$ and the 2-Wasserstein perceptual divergence $d(p_{\mathbf{x}}, p_{\hat{\mathbf{x}}}) = W_2(p_{\mathbf{x}}, p_{\hat{\mathbf{x}}})$, Freirich et al. [1] give an explicit characterization of the optimal DP curve:

$$D(P) \;=\; D^\star \;+\; \max\!\big\{(P^\star - P)^2,\, 0\big\}, \qquad P \in [0, P^\star], \tag{2.6}$$

where $D^\star = \mathbb{E}[\|\mathbf{x} - \mathbf{x}^\star\|_2^2]$ is the MMSE distortion and $P^\star = W_2(p_{\mathbf{x}}, p_{\mathbf{x}^\star})$ is the perceptual gap of the posterior mean. Moreover, the Pareto-optimal estimator at

each perception level P is the linear interpolation

$$\hat{\mathbf{x}}_P \;=\; \left(1 - \frac{P}{P^\star}\right)\hat{\mathbf{x}}_0 \;+\; \frac{P}{P^\star}\,\mathbf{x}^\star \tag{2.7}$$

between a perfectly-perceptual estimator $\hat{\mathbf{x}}_0$ (any $\hat{\mathbf{x}}$ with $d(p_{\mathbf{x}}, p_{\hat{\mathbf{x}}}) = 0$, e.g. a posterior sample) and the MMSE estimator $\mathbf{x}^\star$. The DP frontier is therefore *parameterized by a single scalar* $P/P^\star \in [0, 1]$ that dials the reconstruction from posterior-sample-like ($P/P^\star = 0$, recovering $\hat{\mathbf{x}}_0$) to posterior-mean-like ($P/P^\star = 1$, recovering $\mathbf{x}^\star$).

2.1.3 Priors: classical and learned

Classical priors encode prior knowledge as a hand-crafted regularizer: sparsity in a chosen basis (wavelets, Fourier domain), total-variation regularization (piecewise smoothness), or low-rank structure (in matrix-valued signals) [27, 28]; we refer the reader to [29] for a comprehensive overview. They are cheap to evaluate and principled, with theoretical guarantees, but their performance degrades in modern high-dimensional systems, where signals have complex structure — such as graph-supported signals or natural images. Learned priors close this gap by fitting a generative model p_θ to a dataset $\{\mathbf{x}_0^{(i)}\}_{i=1}^N \sim p_{\text{data}}$ and using the trained model as a plug-in for $p_{\mathbf{x}}(\mathbf{x}_0)$ in (2.2).

At a coarse level, modern generative models fall into a small number of categories. *Autoregressive models* (ARs) factorize the joint density in a fixed ordering and are the workhorse of language modeling. *Variational autoencoders* (VAEs) learn a probabilistic encoder–decoder pair and provide a tractable variational lower bound on the likelihood, at the cost of blurry samples in high dimensions. *Generative adversarial networks* (GANs) produce sharp samples but suffer from unstable training, mode collapse, and no explicit likelihood. *Normalizing flows* (NFs) provide exact likelihoods and invertible sampling, but their architectural constraints limit expressivity.

Energy-based models (EBMs) parameterize an unnormalized density directly and are trained by score matching or contrastive divergence, but the training is expensive.

Diffusion models, which we introduce in Section 2.2 (*flow matching*, the parameterization used by Chapters 6 and 8, can be viewed as the same model class under a different noise schedule [30]; the shared flow-matching background is collected in the Part II opener, Section II), have emerged as the strongest learned prior for inverse problems, achieving state-of-the-art performance. They enjoy a simple, scalable, and stable training loss; produce high-quality and diverse samples; compose easily with a measurement likelihood, favoring plug-and-play strategies; and admit tractable posterior-sampling algorithms. The remainder of this chapter develops the ingredients we need to use them as priors for the inverse problems studied throughout the thesis.

2.2 Diffusion models

Diffusion generative models specify a forward stochastic process that gradually corrupts data into noise, and learn to reverse this process to sample from the data distribution. The most useful formulation for our purposes is the continuous-time score-based formulation [7], which unifies the discrete-time constructions of Sohl-Dickstein et al. [5] and Ho et al. [6] and cleanly separates the choice of noising process from the choice of sampling algorithm.

2.2.1 Forward process

Let $\mathbf{x}_0 \sim p_{\text{data}}(\mathbf{x}_0)$ denote a data sample and consider the forward stochastic differential equation (SDE)

$$\mathrm{d}\mathbf{x}_t = \mathbf{f}(t)\mathbf{x}_t\,\mathrm{d}t + g(t)\,\mathrm{d}\mathbf{w}_t, \qquad t \in [0, T], \tag{2.8}$$

where $\mathbf{w}_t$ is a standard d-dimensional Brownian motion and the coefficients $\mathbf{f}(t) \in \mathbb{R}^{d\times d}$ and $g(t) \in \mathbb{R}$ define the noising *schedule*. The SDE maps $\mathbf{x}_0$ to a terminal distribution p_T that is chosen to be easy to sample from — typically $\mathcal{N}(\mathbf{0}, \mathbf{I})$. Two schedules dominate the literature:

- *Variance-exploding* (VE) SDE [7, 31]: $\mathbf{f}(t) = 0$ and $g(t) = \sqrt{\mathrm{d}\sigma_t^2/\mathrm{d}t}$ for a monotone-increasing σ_t. The marginals are $\mathbf{x}_t = \mathbf{x}_0 + \sigma_t\boldsymbol{\epsilon}$ with $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, so the perturbed variable is the clean signal plus independent Gaussian noise of standard deviation σ_t.

- *Variance-preserving* (VP) SDE [6, 7]: $\mathbf{f}(t) = -\frac{1}{2}\beta(t)\mathbf{I}$ and $g(t) = \sqrt{\beta(t)}$ for a noise-schedule $\beta(t) > 0$. The marginals are $\mathbf{x}_t = \alpha_t\mathbf{x}_0 + \sigma_t\boldsymbol{\epsilon}$ with (α_t, σ_t) satisfying $\alpha_t^2 + \sigma_t^2 = 1$, so the perturbed variable is a convex combination of the clean signal and Gaussian noise.

Both instances admit a Gaussian transition kernel $p_{t|0}(\mathbf{x}_t|\mathbf{x}_0)$ in closed form, which is critical for the denoising-score-matching training objective of Section 2.4. More general schedules, including the anisotropic matrix-valued extensions developed in Chapter 7, which replace the scalar coefficients with matrices, fit into the same framework.

2.2.2 Reverse process

A classical result of Anderson [32] states that the time reversal of (2.8) is itself an SDE:

$$\mathrm{d}\mathbf{x}_t = \left[\mathbf{f}(t)\mathbf{x}_t - g(t)^2 \nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t)\right] \mathrm{d}t + g(t)\, \mathrm{d}\overline{\mathbf{w}}_t, \tag{2.9}$$

where $\overline{\mathbf{w}}_t$ is a Brownian motion running backwards in time. The quantity

$$\mathbf{s}_t(\mathbf{x}_t) \;:=\; \nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t) \tag{2.10}$$

is the (marginal) *score function*. To generate samples, we simulate (2.9) from $t = T$ down to $t = 0$, starting from $\mathbf{x}_T \sim p_T$. All that is needed beyond the choice of forward SDE is an estimator of the score $\mathbf{s}_t$ at each time t, and this is exactly what a diffusion model learns.

2.2.3 Probability-flow ODE

The reverse SDE has a deterministic counterpart, the probability-flow ordinary differential equation (ODE)

$$\mathrm{d}\mathbf{x}_t = \left[\mathbf{f}(t)\mathbf{x}_t - \tfrac{1}{2} g(t)^2 \nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t)\right] \mathrm{d}t, \tag{2.11}$$

whose marginals at every time t coincide with those of (2.8) [7]. The probability-flow ODE plays two roles: it enables exact likelihood computation via the change-of-variables formula, and it opens the door to deterministic high-order ODE solvers (Section 2.5). It is also the point of contact with the family of *flow matching* [8,33,34] and *stochastic interpolant* [34] models, which parameterize a generative ODE directly via its velocity field rather than deriving it from a forward SDE. The flow-matching background used by Chapters 6 and 8 is collected in the Part II opener, Section II.

2.3 Latent diffusion models

Diffusion training and sampling scale cubically with the ambient dimension d: the score network processes tensors of that size at every step of the forward and reverse process, and needs to be evaluated hundreds to thousands of times per generation. For high-resolution images, this quickly becomes prohibitive. *Latent diffusion models* (LDMs) [35, 36] sidestep the issue by performing the diffusion in a compressed latent space produced by a pre-trained autoencoder (or learned jointly with the score as in [36]).

Concretely, an encoder $\mathcal{E}_\phi : \mathbb{R}^d \to \mathbb{R}^{d'}$ maps a pixel-space signal $\mathbf{x}_0$ to a latent code $\mathbf{z}_0 = \mathcal{E}_\phi(\mathbf{x}_0)$, with $d' \ll d$, and a decoder $\mathcal{D}_\phi : \mathbb{R}^{d'} \to \mathbb{R}^d$ approximately inverts this mapping via $\mathbf{x}_0 \approx \mathcal{D}_\phi(\mathbf{z}_0)$. The autoencoder is trained end-to-end with a reconstruction loss (typically a combination of pixel-wise, perceptual, and adversarial terms) on the target image distribution, and its parameters ϕ are frozen thereafter. A diffusion or flow-matching model is then trained on the latent distribution $p_{\mathbf{z}_0}$ induced by $\mathcal{E}_\phi$ applied to p_{data}. State-of-the-art text-to-image systems — Stable Diffusion 1.5 [35], SDXL [37], and Flux [38] — are all LDMs, with autoencoders whose compression factors range from $8\times$ to $16\times$ per spatial dimension. This drops the diffusion cost by two to three orders of magnitude while still generating high-quality images.

For inverse problems, LDMs introduce two additional challenges. First, the forward operator $\mathcal{A}$ is defined in pixel space, so evaluating the likelihood requires passing through the decoder, which is expensive and non-invertible; the measurement $\mathbf{y}$ itself lives in the ambient space and cannot be mapped into the latent space. Second, backpropagating through the decoder is memory-intensive and can bias reconstructions toward the blurred images that the (adversarially trained) autoencoder favors; the interaction with the diffusion prior further exacerbates mode collapse in latent-

space posterior samplers. Both issues motivate the *repulsive latent score distillation* approach developed in Chapter 4.

2.4 Denoising score matching

Both the reverse SDE (2.9) and the probability-flow ODE (2.11) require the marginal score $\mathbf{s}_t(\mathbf{x}_t) = \nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t)$ at every time t. Since p_t is not accessible in closed form, we approximate the score with a neural network $\mathbf{s}_\theta(\mathbf{x}_t, t)$ trained on samples from the joint $p(\mathbf{x}_t, \mathbf{x}_0)$. The natural objective is *explicit score matching* [39], which minimizes the squared error between the network and the true score under p_t; this is intractable because p_t and its score are both unknown.

Denoising score matching (DSM) [40] sidesteps this by matching the network to the score of the conditional $p_{t|0}(\mathbf{x}_t|\mathbf{x}_0)$, which is Gaussian and therefore known in closed form. The single-scale DSM loss is

$$\mathcal{L}_{\mathrm{DSM}}(\theta; t) \;=\; \mathbb{E}_{\mathbf{x}_0 \sim p_{\mathrm{data}},\, \mathbf{x}_t \sim p_{t|0}}\big[\|\mathbf{s}_\theta(\mathbf{x}_t, t) - \nabla_{\mathbf{x}_t} \log p_{t|0}(\mathbf{x}_t|\mathbf{x}_0)\|_2^2\big], \tag{2.12}$$

and Vincent [40] shows that its minimizer coincides with the marginal score $\nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t)$. Song and Ermon [31] extend this to a continuum of noise levels by integrating against a positive weighting schedule $\lambda(t)$:

$$\mathcal{L}_{\mathrm{DSM}}(\theta) \;=\; \int_0^T \lambda(t)\, \mathcal{L}_{\mathrm{DSM}}(\theta; t)\, \mathrm{d}t. \tag{2.13}$$

For the VE SDE, the conditional score is $\nabla_{\mathbf{x}_t} \log p_{t|0}(\mathbf{x}_t|\mathbf{x}_0) = -(\mathbf{x}_t - \mathbf{x}_0)/\sigma_t^2$, so (2.12) reduces to a residual-prediction loss — training the network to predict the noise added to $\mathbf{x}_0$. For the VP SDE, the same derivation yields a re-scaled version of the same loss. The weighting $\lambda(t)$ is a design choice that controls the effective allocation of network capacity across noise levels.

DSM as an upper bound on the KL. Beyond its role as a tractable surrogate for explicit score matching, the multi-scale DSM loss admits a direct maximum-likelihood interpretation. Song et al. [41] show that, with the *likelihood weighting* $\lambda(t) = g(t)^2$, the DSM objective upper-bounds the KL divergence between the data distribution and the model distribution p_θ induced by the learned score:

$$\mathrm{KL}\big(p_{\mathrm{data}} \,\|\, p_\theta\big) \;\le\; \mathcal{L}_{\mathrm{DSM}}(\theta) + C, \tag{2.14}$$

where C is a constant independent of θ. Equivalently, minimizing $\mathcal{L}_{\mathrm{DSM}}$ minimizes an upper bound on the negative log-likelihood of p_θ, so score-based training with this weighting is approximately maximum-likelihood training.

Tweedie's formula

The learned score has a tight connection to the posterior mean of the clean signal given the noisy iterate. For an isotropic Gaussian perturbation kernel with variance σ_t^2 (VE case), Tweedie's formula [42] gives

$$\mathbb{E}[\mathbf{x}_0|\mathbf{x}_t] \;=\; \mathbf{x}_t + \sigma_t^2 \,\nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t), \tag{2.15}$$

and the analogous VP identity is $\mathbb{E}[\mathbf{x}_0|\mathbf{x}_t] = (\mathbf{x}_t + \sigma_t^2 \mathbf{s}_t(\mathbf{x}_t))/\alpha_t$. In either case, evaluating the learned score $\mathbf{s}_\theta$ is equivalent to evaluating a *denoiser*

$$\mathbf{D}_\theta(\mathbf{x}_t, \sigma_t) \;:=\; \mathbb{E}[\mathbf{x}_0|\mathbf{x}_t] \tag{2.16}$$

that maps a noisy input at level σ_t to its posterior-mean estimate of the clean signal. The denoiser view is essential in three ways: it enables the "predict $\mathbf{x}_0$" parameterization used in modern diffusion samplers; it underlies the plug-and-play use of pre-trained denoisers as priors (e.g. Regularization by Denoising [43] and Plug-and-Play methods [44]); and it is central to the flow-map denoisers of Chapter 8, which learn a denoiser that jumps across arbitrary time pairs (s, t) in a single evaluation.

Anisotropic extension

The DSM loss in (2.13) implicitly assumes an isotropic noise perturbation. As we will see in Chapter 7, replacing the scalar σ_t with a full covariance $\mathbf{\Sigma}_t$ and correspondingly extending DSM to a matrix-valued *dual score-matching* objective produces energy models whose learned geometry is aligned with the posterior distribution of a target inverse problem. This is one of the three training-time modifications developed in Part II.

2.5 Sampling algorithms

Given a trained score network $\mathbf{s}_\theta$, drawing samples from the model requires simulating either the reverse SDE (2.9) or the probability-flow ODE (2.11) numerically. The choice of numerical scheme is a design axis in its own right — different schemes trade off between sample quality, number of function evaluations (NFE), and the amount of stochasticity injected at each step — and is one of the four knobs listed in Section 2.7. We summarize the samplers used most frequently in the thesis.

Ancestral sampling (DDPM). The original DDPM sampler [6] discretizes the reverse SDE at the training noise grid $\{t_k\}_{k=0}^{K}$ and takes one Gaussian denoising step per grid point:

$$\mathbf{x}_{k-1} \;=\; \frac{1}{\sqrt{1-\beta_k}}\big(\mathbf{x}_k + \beta_k\, \mathbf{s}_\theta(\mathbf{x}_k, t_k)\big) + \sqrt{\beta_k}\,\boldsymbol{\epsilon}_k, \qquad \boldsymbol{\epsilon}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{I}). \tag{2.17}$$

DDPM is faithful to the reverse SDE marginals but expensive: high-quality samples typically require K in the range 10^2–10^3.

DDIM. Song et al. [45] recast the diffusion generative process as a non-Markovian family indexed by a stochasticity parameter $\eta \in [0, 1]$. The DDIM update takes the

form

$$\mathbf{x}_{k-1} \;=\; \alpha_{k-1}\,\hat{\mathbf{x}}_0(\mathbf{x}_k) + \sqrt{1-\alpha_{k-1}^2-\eta_k^2\sigma_k^2}\,\boldsymbol{\epsilon}_\theta(\mathbf{x}_k, t_k) + \eta_k\sigma_k\,\boldsymbol{\epsilon}_k, \tag{2.18}$$

where $\hat{\mathbf{x}}_0(\mathbf{x}_k) = (\mathbf{x}_k - \sigma_k\boldsymbol{\epsilon}_\theta(\mathbf{x}_k, t_k))/\alpha_k$ is the Tweedie estimate of the clean signal from the network's noise prediction $\boldsymbol{\epsilon}_\theta$. Two limits are especially important:

- $\eta = 1$: DDIM reduces to ancestral DDPM.
- $\eta = 0$: the update becomes deterministic and coincides with a first-order discretization of the probability-flow ODE. In this regime, the sampler is invertible (any $\mathbf{x}_0$ can be re-encoded back to any $\mathbf{x}_T$ deterministically), and much larger step sizes are stable.

The deterministic DDIM regime typically produces high-quality samples with 10–50 steps, an order of magnitude fewer than DDPM. Intermediate η interpolates between the two, and controlled stochasticity via η turns out to be a useful design axis for inverse problem solvers.

Higher-order ODE solvers. Once the sampling problem is cast as an ODE, standard numerical-analysis machinery applies. Heun's method, Runge–Kutta schemes, and specialized diffusion solvers such as DPM-Solver [46] exploit the semi-linear structure of the probability-flow ODE to reach state-of-the-art sample quality in fewer than 20 NFE. Karras et al. [47] give a unified derivation and defend Heun's method with a truncation-error-controlled schedule as a strong default.

Langevin-based samplers. An orthogonal approach bypasses the reverse SDE altogether and runs a Markov chain — typically annealed overdamped Langevin dynamics with a decaying noise schedule — directly on the learned score field. This was the original recipe of Song and Ermon [31]. The *higher-order* annealed Langevin

diffusion of Chapter 3 augments this recipe with an auxiliary momentum variable and pre-conditioning, dramatically accelerating convergence; the shielded Langevin sampler of Chapter 5 augments it with a barrier potential to enforce hard constraints. We note that these samplers use an underlying reverse process that departs from the theoretical one in (2.9); empirically, they still produce high-quality samples. They can also be used inside a *predictor–corrector* framework [48, 49], which interleaves one step of a numerical SDE integrator (the predictor) with a few Langevin steps at each noise level (the correctors).

Variational samplers. A qualitatively different alternative is to sidestep sequential simulation entirely: parameterize a variational distribution over the clean signal and optimize its parameters via a score-matching objective, using the pre-trained diffusion model only for its score. RED-diff [50] and Score Distillation Sampling [51] are canonical examples. The *repulsive latent score distillation* sampler of Chapter 4 sits in this family and extends it to a particle variational family with a repulsive coupling that prevents mode collapse.

2.6 Solving inverse problems with diffusion priors

We now have the ingredients to use a diffusion model as a prior for the inverse problem (2.1). The posterior of interest (2.2) can be sampled by simulating a *conditional* reverse SDE (or the analogous ODE) that targets $p(\mathbf{x}_0|\mathbf{y})$ instead of $p(\mathbf{x}_0)$. The corresponding conditional score decomposes into a prior term and a likelihood term:

$$\nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t|\mathbf{y}) \;=\; \underbrace{\nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t)}_{\text{prior score (known)}} \;+\; \underbrace{\nabla_{\mathbf{x}_t} \log p_t(\mathbf{y}|\mathbf{x}_t)}_{\text{noisy likelihood score (intractable)}} \,. \tag{2.19}$$

The prior score is exactly what the pre-trained diffusion model provides, via $\mathbf{s}_\theta(\mathbf{x}_t, t)$. The noisy likelihood term, however, is *not* directly available, because it marginalizes

over the clean signal:

$$p_t(\mathbf{y}|\mathbf{x}_t) \;=\; \int p(\mathbf{y}|\mathbf{x}_0)\, p_{0|t}(\mathbf{x}_0|\mathbf{x}_t)\, \mathrm{d}\mathbf{x}_0. \tag{2.20}$$

The reverse process runs on the noisy iterate $\mathbf{x}_t$, but the physical forward model only relates the *clean* signal $\mathbf{x}_0$ to the observation $\mathbf{y}$. Approximating the integral in (2.20) is the central technical challenge of diffusion-based inverse problem solvers, and virtually every method in the literature can be classified by how it does so. Broadly:

- *Point-estimate approximations* replace the integrand with a delta at a plug-in estimate of $\mathbf{x}_0$. The most influential instance is *Diffusion Posterior Sampling* (DPS) [2], which uses the Tweedie estimate $\hat{\mathbf{x}}_0 := \mathbb{E}[\mathbf{x}_0|\mathbf{x}_t]$ from (2.15), giving $p_t(\mathbf{y}|\mathbf{x}_t) \approx \mathcal{N}(\mathbf{y}; \mathcal{A}(\hat{\mathbf{x}}_0), \sigma_0^2\mathbf{I})$. This is cheap but biased, and requires backpropagation through the score network at every step.

- *Projection- and SVD-based approximations* exploit the structure of a linear operator $\mathbf{H}$. Kawar et al. [52] use the singular-value decomposition of $\mathbf{H}$ to condition the diffusion in the spectral domain; ΠGDM [53] uses the pseudoinverse of $\mathbf{H}$. These methods are asymptotically exact for noiseless linear problems but degrade under noise and non-linearity.

- *Variational approximations* recast posterior sampling as an optimization problem, sidestepping the intractable integral by fitting a parametric variational family to the posterior. RED-diff [50] and its variants use a Gaussian family; the *repulsive latent score distillation* of Chapter 4 extends this to a particle family with a repulsive coupling that provably prevents mode collapse.

- *Asymptotically exact methods* target the true posterior directly and trade com-

pute for approximation error: as the number of network evaluations grows, they converge to samples from the exact posterior (up to score-network error). Sequential Monte Carlo methods with diffusion-based proposals [54, 55] are the canonical instance; they are principled but expensive, since the number of particles required to control variance grows quickly with the signal dimension.

All of these approximations either (i) require expensive backpropagation through the score network, (ii) require simulating the full reverse SDE, (iii) sacrifice sample diversity, or (iv) cannot enforce hard structural constraints. Reducing their compute cost, restoring diversity, and enforcing constraints are the three failure modes attacked by the inference-time algorithms of Part I.

2.7 Design knobs in diffusion-based inverse problem solvers

The preceding sections have set out the ingredients of a diffusion-based inverse problem solver. It is useful, before turning to the contributions of the thesis, to abstract those ingredients into an explicit catalogue of *design knobs*. Each knob is a component of the pipeline that can be modified independently, at either training or inference time; each of the six contributions of the thesis modifies exactly one knob. Table 2.1 in Section 2.8 maps knobs to contributions.

1. **Source distribution p_T.** Both the forward SDE (2.8) and the flow-matching interpolant require a terminal / base distribution to invert from. The default is an uninformative Gaussian $\mathcal{N}(\mathbf{0}, \mathbf{I})$, chosen for tractability. When additional information about the target signal is available (e.g., a learned posterior mean, paired clean-measurement data) replacing the Gaussian source with a task-informed one dramatically reduces the effective ill-posedness of the problem.

Chapter 6 realizes this knob for graph inverse problems by constructing a source distribution that encodes prior graph structure and training a flow model to close the gap.

2. **Training target: score vs. density.** Denoising score matching, in its standard form, learns only the *score* $\nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t)$ — the gradient of the log-density. The normalization constant of p_t itself remains inaccessible, which blocks any downstream task that needs a proper density: blind inverse problems in which the noise level must be recovered from the observation alone, principled uncertainty quantification, and posterior-based comparison across solvers. This knob is turned by extending DSM to matrix-valued *anisotropic noise*. Replacing scalar coefficients with matrix-valued ones $\mathbf{F}(t), \mathbf{G}(t)$ produces a companion covariance score $\nabla_{\mathbf{\Sigma}} \log p_t(\mathbf{x}_t, \mathbf{\Sigma}_t)$ that, together with the data score, makes the density normalization tractable. Chapter 7 realizes this knob via a dual score-matching objective, yielding energy-based models whose normalized density unlocks reliable energy-guided sampling, blind inverse problems, and uncertainty quantification.

3. **Training target: instantaneous vs. average velocity.** Standard diffusion and flow matching learn the *instantaneous* velocity of a stochastic interpolant, which via Tweedie's formula (2.15) corresponds to a single-noise-level denoiser $\mathbf{D}_\theta(\mathbf{x}_t, \sigma_t)$ (2.16) and forces many-step integration (10^2–10^3 solver evaluations) at inference. A model trained instead to output the *average velocity* between arbitrary time pairs (s, t) — a *flow map* [56–58] — transports samples in a single evaluation, and its Tweedie-analogue is a two-time denoiser $\mathbf{D}_\theta(\mathbf{x}_s, \sigma_s, \sigma_t)$ that predicts the clean signal from a noisy input at time s using lookahead

t. Chapter 8 trains such a flow map and uses this average/two-time denoiser as a plug-and-play primitive for image inverse problems; the lookahead of the trained flow map directly parameterizes the DP frontier (2.7), turning what was previously a solver choice into an intrinsic property of a single trained model.

4. **Sampling algorithm.** Given a fixed pre-trained model, the algorithm that generates samples is still a design choice: ancestral DDPM, deterministic DDIM at $\eta = 0$, higher-order ODE solvers, annealed Langevin chains, and variational samplers all target the same distribution but differ dramatically in cost, sample quality, and downstream flexibility (Section 2.5). For inverse problems in particular, the sampler must accommodate a likelihood term, and the choice of sampler determines how easily this can be done efficiently, diversely, and under constraints. Chapters 3, 4, and 5 realize this knob along three different axes: auxiliary variables in Langevin sampling and pre-conditioning for speed, particle repulsion for diversity, and navigation potentials for hard constraints.

These four knobs are not independent because pulling one often changes the analysis of the others. Nevertheless, they are logically separable, and separating them clarifies both the contributions of the thesis and the broader design space of diffusion-based inverse problem solvers. The remainder of the thesis works through one knob at a time.

2.8 Contributions of this thesis

The background above lets us make the thesis contributions precise. Table 2.1 maps each of the six failure modes identified in Section 1.2 to the specific knob it modifies (Section 2.7), the contribution that realizes the modification, and the chapter that

develops it. The unifying claim of the thesis is that solving inverse problems with diffusion priors benefits from modifications at every one of the four knobs, and that each knob can be pulled as an independent design lever.

2.9 Notation

Throughout the thesis we use bold lowercase letters for vectors ($\mathbf{x}$), bold uppercase for matrices ($\mathbf{H}$), and calligraphic for sets or distributions ($\mathcal{N}$, $\mathcal{X}$). Time indices for continuous-time processes are subscripts ($\mathbf{x}_t$), and iteration indices for discrete algorithms are subscripts too, with context distinguishing them. We reserve $\mathbf{x}_0$ for the clean signal, $\mathbf{y}$ for the measurement, $\mathbf{H}$ or $\mathcal{A}$ for the forward operator, $\mathbf{s}_\theta$ for a score network, and $\mathbf{v}_\theta$ for a velocity network. Additional notation is introduced in place.

Axis	Failure mode	Contribution	Chapter
Inference-time (Part I)	Slow sampling	Higher-order annealed Langevin diffusion [9]	Chapter 3
	Latent-space mode collapse	Repulsive latent score distillation [10]	Chapter 4
	Hard constraints not respected	Shielded Langevin Monte Carlo with navigation potentials [11]	Chapter 5
Model (Part II)	Task-agnostic source distribution	Prior-informed flow matching [12]	Chapter 6
	Only the score is learned, not the density	Energy-based models with anisotropic noise [13]	Chapter 7
	Only the instantaneous velocity is learned	Flow-map denoisers [14]	Chapter 8

Table 2.1 : Mapping of thesis contributions to identified failure modes and design knobs. The first three rows are *inference-time* algorithms on top of a pre-trained diffusion or flow prior (knob 4 of Section 2.7); the last three are *training-time* modifications in which the prior itself is modified along knobs 1–3.

Part I

Inference-time algorithms on pre-trained priors

This part of the thesis looks at a *classical* diffusion or flow-matching prior, i.e., a model pre-trained on a large dataset, with no knowledge of the target inverse problem, and modifies only the sampling algorithm at inference time. This is a form of *inference-time scaling*: given a fixed prior, better inverse-problem solvers are built by improving the sampling algorithm.

The three chapters of Part I address three different aspects based on failures identified in Section 1.2:

- Chapter 3 attacks *slow mixing* by augmenting overdamped Langevin dynamics with auxiliary variables and pre-conditioning, defining a higher-order annealed Langevin sampler that achieves substantial speed-ups over its overdamped counterpart while matching accuracy on massive MIMO detection, channel estimation, and image restoration.

- Chapter 4 attacks *mode collapse* in variational posterior sampling with latent diffusion priors. Posterior sampling is recast as a variational optimization problem in latent space and a repulsive coupling across particles is injected to provably prevent concentration on a single mode.

- Chapter 5 attacks the inability of standard samplers to *respect hard constraints*. Langevin dynamics is augmented with a barrier-like navigation potential inspired by robot navigation, yielding a sampler that provably avoids user-specified forbidden regions.

The unifying property of these three algorithms is that the underlying generative model is untouched. When the pre-trained prior is a large model trained on a diverse dataset, these models provide useful signal information.

Chapter 3

Higher-order annealed Langevin diffusion

Sampling from a diffusion posterior for a high-dimensional linear inverse problem is dominated by the cost of simulating a first-order Langevin recursion, whose mixing time scales poorly with the conditioning of the target. In this chapter we develop a family of *higher-order* annealed Langevin diffusions that augment the standard overdamped Langevin sampler with an auxiliary momentum variable and a preconditioner tied to the noise schedule. We derive the modified continuous-time process, prove that it retains the target as its invariant measure, and provide a numerical discretization amenable to plug-in with any pre-trained diffusion prior. Applied to massive multiple-input multiple-output (MIMO) detection, channel estimation, and image restoration with a learned score, the higher-order sampler matches the accuracy of the overdamped baseline while requiring an order of magnitude fewer iterations. The chapter closes with proofs of the two main propositions.

This chapter is adapted from [9].

3.1 Introduction

Inverse problems in massive MIMO detection [59], image restoration [60], and graph topology inference [61, 62] share a common structure: recover a high-dimensional signal from noisy, under-determined observations under a Bayesian prior. Diffusion-based samplers have recently emerged as the state-of-the-art approach for such prob-

lems [2, 7, 52]. As set out in Section 2.6, a natural approach is to sample from the posterior directly using annealed Langevin dynamics on top of a learned diffusion prior [31, 63–65]. This chapter develops a family of *higher-order* annealed Langevin samplers that accelerate this approach without touching the underlying prior.

The bottleneck of standard annealed Langevin sampling is the mixing time of the underlying Markov chain. On high-dimensional, ill-conditioned posteriors, overdamped Langevin dynamics require thousands of iterations to reach the target, and each iteration invokes the score network. Two orthogonal accelerations from the sampling literature promise to reduce this cost: adding auxiliary momentum variables — the *underdamped* and *third-order* Langevin dynamics enjoy strictly better non-asymptotic convergence rates than their overdamped counterpart [66, 67], and have already been exploited in a score-based generative model built on underdamped diffusion [68] — and pre-conditioning the dynamics to match the geometry of the target [63, 69–71]. Neither has been combined into a single sampler for inverse problems with learned diffusion priors, and combining them is non-trivial: the invariant distribution must be preserved (a question best analyzed in the general SDE framework of Ma et al. [72]), and the choice of numerical discretization has a first-order effect on sample accuracy.

In this chapter we develop such a combined framework. We derive pre-conditioned continuous-time formulations of the underdamped and third-order Langevin dynamics, prove that both preserve the target posterior as their unique invariant distribution, and give a splitting-based numerical discretization that is materially more accurate than a naive Euler–Maruyama scheme. The resulting sampler is drop-in for any pre-trained diffusion prior and reduces the iteration budget by an order of magnitude on massive MIMO detection, channel estimation, and image restoration,

without sacrificing accuracy.

3.2 System model and problem formulation

We instantiate the general inverse-problem setup of Section 2.1 in the concrete form used throughout this chapter. The linear forward model

$$\mathbf{y} = \mathbf{H}\mathbf{x} + \mathbf{z}, \tag{3.1}$$

now has $\mathbf{H} \in \mathbb{R}^{N_r \times N_u}$ (the channel matrix in a MIMO system, the degradation operator in an imaging problem, etc.), $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \sigma_0^2 \mathbf{I}_{N_r})$ Gaussian noise with known variance σ_0^2, and $\mathbf{x} \in \mathcal{X}^{N_u}$ the signal of interest with domain $\mathcal{X}$ that may be finite or infinite. We assume perfect knowledge of σ_0^2 and either $\mathbf{H}$ or $\mathbf{x}$; the case in which both are unknown is treated in Section 3.5. Under this configuration the estimation problem specializes to:

Problem 1 Given perfect knowledge of $\mathbf{H}$ (or $\mathbf{x}$) and an observed $\mathbf{y}$ following (3.1), find an estimate of $\mathbf{x}$ (or $\mathbf{H}$).

To facilitate the presentation throughout the chapter, we derive everything for the case when $\mathbf{x}$ is unknown; the derivation when $\mathbf{H}$ is unknown is equivalent. Bayes' optimal decision rule (cf. (2.2)) gives

$$\hat{\mathbf{x}}_{\text{MAP}} = \underset{\mathbf{x} \in \mathcal{X}^{N_u}}{\operatorname{argmax}} \; p(\mathbf{x}|\mathbf{y}, \mathbf{H}) = \underset{\mathbf{x} \in \mathcal{X}^{N_u}}{\operatorname{argmax}} \; p_{\mathbf{z}}(\mathbf{y} - \mathbf{H}\mathbf{x}) p(\mathbf{x}). \tag{3.2}$$

Several schemes have been proposed in the last decades to provide efficient approximate solutions to Problem 1 (see Section 3.1). Recently, samplers based on Langevin diffusion were proposed for different applications [63–65, 73, 74], achieving high performance. However, these samplers are demanding in terms of computation and running time.

Hence, we are motivated by the following question: *Can we derive an efficient algorithm to solve inverse problems by sampling from the posterior? More specifically, compared to the performance of the overdamped Langevin dynamic, can we reduce its running time but still achieve state-of-the-art performance?* We achieve this objective by leveraging an annealed version of two different higher-order Langevin dynamics. In particular, we propose to solve Problem 1 by (approximately) sampling from the posterior distribution in (3.2) using *pre-conditioned* annealed underdamped (second-order) and third-order Langevin dynamics.

3.3 Higher-order Langevin dynamics

In Section 3.3.1, we introduce the continuous-time ULD, also known as the second-order Langevin dynamic. In Section 3.3.2, we describe the GLE, which serves as a framework to derive higher-order dynamics.

3.3.1 Continuous-time underdamped Langevin dynamics

The continuous-time ULD is the Markov process on variables $\mathbf{x}_t \in \mathbb{R}^d$ and $\mathbf{v}_t \in \mathbb{R}^d$ that solves the stochastic differential equations (SDEs)

$$\begin{aligned} \mathrm{d}\mathbf{x}_t &= \mathbf{M}^{-1}\mathbf{v}_t \mathrm{d}t, \\ \mathrm{d}\mathbf{v}_t &= -\nabla_{\mathbf{x}_t} U(\mathbf{x}_t)\mathrm{d}t - \gamma \mathbf{v}_t \mathrm{d}t + \sqrt{2\gamma\tau}\mathbf{M}^{\frac{1}{2}}\mathrm{d}\mathbf{W}_t, \end{aligned} \tag{3.3}$$

where $\mathbf{W}$ is a standard d-dimensional Brownian motion, $U \in \mathcal{C}^2(\mathbb{R}^d)$ is called the potential, $\gamma \in \mathbb{R}^+$ is a friction parameter, $\mathbf{M}$ is a mass matrix that controls the coupling between $\mathbf{x}_t$ and $\mathbf{v}_t$, and τ is a temperature parameter. Under mild conditions, it can be shown [75] that the invariant distribution of the continuous-time process (3.3)

lies in the kernel of the Fokker-Planck equation, and it is given by

$$\pi(\mathbf{X}) \propto \exp\left[-\tau^{-1} H\left(\mathbf{X}\right)\right], \tag{3.4}$$

where $\mathbf{X} = [\mathbf{x}, \mathbf{v}]$ and $H(\mathbf{X}) = U(\mathbf{x}) + \frac{\mathbf{v}^\top \mathbf{M}^{-1}\mathbf{v}}{2}$ is the Hamiltonian of the system. Therefore, the state distribution of the process in (3.3) converges to $\pi(\mathbf{X})$ when $t \to \infty$. Given a target distribution $p(\mathbf{x})$ from which we want to generate samples $\mathbf{x} \in \mathbb{R}^d$, if we define $U(\mathbf{x}) = -\log p(\mathbf{x})$, then (3.3) defines an MCMC sampler as $\pi(\mathbf{x}) \propto p(\mathbf{x})^{1/\tau}$. In particular, if $\tau = 1$, then $\pi(\mathbf{x}) \propto p(\mathbf{x})$. In essence, the dynamic in (3.3) explores the target distribution by moving in the direction of the gradient of the logarithm of the target density $\nabla_{\mathbf{x}} \log p(\mathbf{x})$, known as *score function*.

Accelerated mixing time and overdamped (first-order) Langevin as limit dynamic. The *overdamped* Langevin dynamic can be obtained as a particular regime of the dynamic in (3.3), when the friction parameter $\gamma \to \infty$ [75, Section 6.5]. Therefore, the main difference between the overdamped and the underdamped case is the momentum variable $\mathbf{v}_t$. Through the lens of sampling, the momentum term entails, under some assumptions on the log-probability density $\log p(\mathbf{x})$, an accelerated version of the sampler compared to the overdamped Langevin dynamic. In particular, [66] establishes a non-asymptotic convergence rate for a specific discretization scheme of (3.3) when $\log p(\mathbf{x})$ is strongly concave and has Lipschitz continuous gradient (this result was later improved in [76]). In [77], another convergence rate is given by considering a splitting strategy as the discretization scheme; details about discretization schemes are postponed to Section 3.4.2. Overall, these results demonstrate that there is a significant improvement when considering the underdamped case.

3.3.2 Generalized Langevin equation

The GLE represents a more general stochastic dynamical system represented by the following SDEs

$$\begin{aligned} \mathrm{d}\mathbf{x}_t &= \mathbf{M}^{-1}\mathbf{v}_t \mathrm{d}t, \\ \mathrm{d}\mathbf{v}_t &= -\nabla_{\mathbf{x}_t} U(\mathbf{x}_t) - \left[\int_0^t \mathbf{K}(t-s)\mathbf{M}^{-1}\mathbf{v}_s \mathrm{d}s\right]\mathrm{d}t + \boldsymbol{\eta}_t \mathrm{d}t, \end{aligned} \tag{3.5}$$

where $\mathbf{K}(t)$ is a memory kernel defining a drag force (or linear dissipation term) in the form of a convolution with the momentum. Therefore, the dissipation term depends on the history of the position. The stochastic force $\boldsymbol{\eta}_t$ is related to the dissipation term through the fluctuation-dissipation theorem [75]. In particular, $\boldsymbol{\eta}_t$ is defined as a stationary Gaussian process with zero mean and auto-covariance function given by

$$\mathbb{E}\left[\boldsymbol{\eta}(s+t)\boldsymbol{\eta}^\top(s)\right] = \tau\mathbf{K}(t), \tag{3.6}$$

where τ is the temperature of the system. The GLE is a non-Markovian dynamical model, in the sense that the evolution of the system depends on the history of the trajectory through the convolution integral in the momentum equation. Therefore, from a numerical point of view, its implementation is computationally demanding as it requires the storage of the momentum history – whose numerical evaluation at each time step can be non-trivial – as well as the generation of a stochastic force with non-trivial correlations. Under certain conditions on the memory kernel $\mathbf{K}(t)$, the GLE is ergodic with unique invariant distribution given by the Gibbs distribution (3.4). Thus, we can also generate samples from a target distribution $p(\mathbf{x})$ by defining $U(\mathbf{x}) = -\log p(\mathbf{x})$.

Although the inclusion of the memory kernel enhances the sampling properties [78], it makes this formulation analytically intractable. Therefore, we can circumvent this issue by working with a particular choice of the memory kernel, which

allows us to reformulate the dynamic in a Markovian representation by adding a set of auxiliary variables.

Extended-variable Markovian representation. To rewrite the SDEs in (3.5) in a Markovian representation, we define the memory kernel as $\mathbf{K}(t) = K(t)\mathbf{I}$, where $K(t)$ is a scalar-valued function. Under this representation of the kernel, we define the function $K(t)$ as a Prony series

$$K(t) = \sum_{j=1}^{j_{\max}} \lambda_j^2 \exp\left(-\alpha_j t\right), \tag{3.7}$$

where there are $j_{\max}$ modes, and each λ_j and α_j are constant parameters. With this choice of $K(t)$, we can represent the GLE (3.5) as a Markovian system by introducing $j_{\max}$ additional variables. From now on, we consider only one Prony mode, i.e., $j_{\max} = 1$, so that we drop the subindex j; the dynamic that we derive can be extended easily to the case $j_{\max} > 1$ [79].

Formally, we define the extended variable $\boldsymbol{\xi}$ associated with the Prony mode as the dissipation term given by the convolution operation in (3.5) divided by λ

$$\boldsymbol{\xi}_t = -\frac{1}{\lambda}\int_0^t \lambda^2 \exp\left(-\alpha(t-s)\right)\mathbf{M}^{-1}\mathbf{v}_s \mathrm{d}s. \tag{3.8}$$

Then, we can rewrite (3.5) as

$$\begin{aligned} \mathrm{d}\mathbf{x}_t &= \mathbf{M}^{-1}\mathbf{v}_t \mathrm{d}t \\ \mathrm{d}\mathbf{v}_t &= -\nabla_{\mathbf{x}_t} U(\mathbf{x}_t)\mathrm{d}t + \lambda\boldsymbol{\xi}_t \mathrm{d}t + \boldsymbol{\eta}_t \mathrm{d}t, \end{aligned} \tag{3.9}$$

Now, we redefine the drag force and the stochastic force to obtain a Markovian system.

First, using the Laplace transform, it can be seen that the variable $\boldsymbol{\xi}_t$ defined in (3.8) is the solution of the ordinary differential equation given by

$$\mathrm{d}\boldsymbol{\xi}_t = -\alpha\boldsymbol{\xi}_t \mathrm{d}t - \lambda\mathbf{M}^{1/2}\mathbf{v}_t \mathrm{d}t. \tag{3.10}$$

Second, we construct a random force for the Prony mode by noticing that the random force has to satisfy the fluctuation-dissipation theorem [80], so it can be viewed as the solution of a linear SDE

$$\mathrm{d}\tilde{\boldsymbol{\eta}}_t = -\alpha\tilde{\boldsymbol{\eta}}_t\mathrm{d}t + \sqrt{2\tau\alpha}\mathrm{d}\mathbf{W}_t. \tag{3.11}$$

Thus, the random force is defined as the solution of an Ornstein-Uhlenbeck process induced by the Prony series. Moreover, we need to multiply the solution by λ to satisfy the fluctuation-dissipation theorem, i.e., $\boldsymbol{\eta}_t = \lambda\tilde{\boldsymbol{\eta}}_t$. Hence, the process $\boldsymbol{\eta}_t$ has zero mean and time correlation function given by $\mathbb{E}\left[\boldsymbol{\eta}(s+t)\boldsymbol{\eta}(s)^\top\right] = \tau\lambda^2\exp\left(-\alpha t\right)\mathbf{I}$.

We can combine both results (3.10) and (3.11) and define a new variable $\mathbf{z}_t = \boldsymbol{\xi}_t + \tilde{\boldsymbol{\eta}}_t$, to reformulate the GLE as a Markovian system

$$\begin{aligned}
\mathrm{d}\mathbf{x}_t &= \mathbf{M}^{-1}\mathbf{v}_t\mathrm{d}t, \\
\mathrm{d}\mathbf{v}_t &= -\nabla_{\mathbf{x}_t}U(\mathbf{x}_t)\mathrm{d}t + \lambda\mathbf{z}_t\mathrm{d}t, \\
\mathrm{d}\mathbf{z}_t &= -\lambda\mathbf{M}^{-1}\mathbf{v}_t\mathrm{d}t - \alpha\mathbf{z}_t\mathrm{d}t + \sqrt{2\tau\alpha}\,\mathrm{d}\mathbf{W}_t.
\end{aligned} \tag{3.12}$$

Under this representation of the GLE, we can define a Fokker-Planck operator associated with the dynamic and, consequently, an invariant distribution of the dynamic. In comparing the GLE with the (second-order) ULD, we first notice that the latter can be seen as a particular case of the former when $K(t) = \gamma\delta(t)$ [81]. Moreover, in [67], it was shown that the third-order case generates smoother trajectories than the corresponding ones under the ULD, which enhances the sampling properties and exhibits an improvement in the convergence rate in comparison to the overdamped and underdamped cases. Also, [82] introduced an annealing scheme that shows better performance and higher exploration of the state space compared to the underdamped and overdamped cases.

3.4 Accelerating higher-order Langevin dynamics

In Section 3.4.1, we define two modified continuous-time Markovian Langevin dynamics, one associated with the second-order and the other one with the third-order dynamic. We show that the two processes have a unique invariant distribution given by (3.4). Then, in Section 3.4.2, we explain the numerical implementation of these dynamics via discretization schemes based on the splitting method. Lastly, in Section 3.4.3, we derive our posterior sampling algorithms, where we present different possible choices of the score function of the posterior distribution and the pre-conditioning matrix involved in the algorithms.

3.4.1 Modified continuous-time higher-order Langevin dynamics

This section aims to construct continuous-time diffusion processes based on the dynamics introduced in Section 3.3 that leverage the geometry of the space to accelerate the convergence. We achieve this acceleration by incorporating carefully crafted pre-conditioning denoted by $\mathbf{C}_t \in \mathbb{R}^{d\times d}$. We assume the following for $\mathbf{C}_t$

AS1 *The matrix* $\mathbf{C}_t \in \mathbb{R}^{d\times d}$ *is symmetric positive definite and has uniformly bounded matrix norm, i.e.,* $||\mathbf{C}_t|| < M$ *for all* t*.*

Before introducing the dynamics, we assume also the following for the potential function.

AS2 *The gradient of the potential is Lipschitz continuous, i.e.,* $||\nabla_{\mathbf{x}} U(\mathbf{x}) - \nabla_{\mathbf{x}} U(\mathbf{x}')|| \leq L||\mathbf{x} - \mathbf{x}'||, \quad \forall\, \mathbf{x}, \mathbf{x}' \in \mathbb{R}^d$*.*

This assumption is a necessary condition to guarantee the uniqueness of the invariant distribution of our proposed pre-conditioned dynamics.

Pre-conditioned ULD. As defined in Section 3.3.1, this dynamic has two variables: position and momentum. Thus, we have $\mathbf{X}_t = [\mathbf{x}_t, \mathbf{v}_t]$ and

$$H(\mathbf{X}_t) = U(\mathbf{x}_t) + \frac{1}{2}\mathbf{v}_t^\top \mathbf{M}^{-1}\mathbf{v}_t, \tag{3.13}$$

and the gradient $\nabla H(\mathbf{X}_t) = [\nabla_{\mathbf{x}_t} U(\mathbf{x}_t), \mathbf{M}^{-1}\mathbf{v}_t]$. Moreover, we consider a symmetric positive definite matrix $\mathbf{C}_t \in \mathbb{R}^{d\times d}$. Then, the pre-conditioned second-order dynamic is defined as follows

$$\begin{aligned} \mathrm{d}\mathbf{x}_t &= \mathbf{C}_t\mathbf{M}^{-1}\mathbf{v}_t \mathrm{d}t, \\ \mathrm{d}\mathbf{v}_t &= -\mathbf{C}_t\nabla_{\mathbf{x}_t} U(\mathbf{x}_t) - \gamma\mathbf{v}_t\mathrm{d}t + \sqrt{2\tau\gamma}\mathbf{M}^{\frac{1}{2}}\mathrm{d}\mathbf{W}_t, \end{aligned} \tag{3.14}$$

The following proposition guarantees that the invariant distribution is still (3.4).

Proposition 1 Given Assumptions 1 and 2 and the Hamiltonian defined in (3.13)*, the distribution* (3.4) *is the unique invariant distribution of the dynamics* (3.14)*.*

The proof is deferred to Appendix A.1. This proposition generalizes existing results [83] and [71] by accounting for the effect of a mass matrix $\mathbf{M}$.

Pre-conditioned third-order LD. Following a similar recipe to the ULD case, we can derive a pre-conditioned version of the dynamical system defined in (3.12). First, the variables involved in this dynamic are $\mathbf{X}_t = [\mathbf{x}_t, \mathbf{v}_t, \mathbf{z}_t]$, so that

$$H(\mathbf{X}_t) = U(\mathbf{x}_t) + \frac{1}{2}\mathbf{v}_t^\top \mathbf{M}^{-1}\mathbf{v}_t + \frac{1}{2}\mathbf{z}_t^\top \mathbf{M}^{-1}\mathbf{z}_t, \tag{3.15}$$

and the gradient is $\nabla H(\mathbf{X}_t) = [\nabla_{\mathbf{x}_t} U(\mathbf{x}_t), \mathbf{M}^{-1}\mathbf{v}_t, \mathbf{M}^{-1}\mathbf{z}_t]$. Then, the pre-conditioned

third-order dynamic is defined as follows*

$$\begin{aligned} \mathrm{d}\mathbf{x}_t &= \mathbf{C}_t\mathbf{M}^{-1}\mathbf{v}_t\mathrm{d}t, \\ \mathrm{d}\mathbf{v}_t &= -\mathbf{C}_t\nabla_{\mathbf{x}_t}U(\mathbf{x}_t)\mathrm{d}t + \lambda\mathbf{z}_t\mathrm{d}t, \\ \mathrm{d}\mathbf{z}_t &= -\lambda\mathbf{v}_t\mathrm{d}t - \alpha\mathbf{z}_t\mathrm{d}t + \sqrt{2\tau\alpha}\mathbf{M}^{1/2}\mathrm{d}\mathbf{W}_t, \end{aligned} \tag{3.16}$$

Again, the following proposition guarantees that the invariant distribution is (3.4).

Proposition 2 Given Assumptions 1 and 2 and the Hamiltonian defined in (3.15)*, the distribution* (3.4) *is the unique invariant distribution of the dynamics* (3.16)*.*

The proof is deferred to Appendix A.2.

Considering a pre-conditioning matrix is key in real-time applications, as it accelerates the convergence of the dynamic while yielding good performance. Recently, pre-conditioned *overdamped* Langevin dynamics have been proposed for denoising tasks in image processing [63] and MIMO detection in communications [65,73], where the negative inverse of the Hessian matrix of the score-posterior is used as a pre-conditioning matrix. Here we have extended this analysis, enabling the use of pre-conditioned higher-order Langevin dynamics for the solution of inverse problems.

3.4.2 Numerical discretization

In order to define a sampling algorithm based on the continuous-time dynamics introduced in Section 3.4.1, we need to discretize the equation. In this work, we rely

*The square root of $\mathbf{M}$ is multiplying the stochastic term, while in (3.3) the inverse of $\mathbf{M}$ is multiplying the velocity variable. Both representations are equivalent, where the difference comes from a re-scaling of the kernel in (3.7) with $\mathbf{M}$. Then, the fluctuation-dissipation theorem (3.6) has to hold, so the stochastic term is multiplied by $\mathbf{M}^{1/2}$.

on splitting methods for the discretization of (3.14) and (3.16). In a nutshell, given the operator that describes the time evolution of the state $\mathbf{x}_t$, the idea is to split the operator into tractable sub-operators and then compose them to approximate the full operator. Formally, given an initial state $\mathbf{X}_0$, the solution of the SDE (3.3) can be constructed as $\mathbf{X}_t = \exp(t\mathcal{L}_{\text{ULD}})\mathbf{X}_0$, where $\exp(t\mathcal{L}_{\text{ULD}})$ is the operator that defines the propagation of the state and $\mathcal{L}_{\text{ULD}}$ is the (infinitesimal) generator of the Markov process [75]. Then, we split the operator $\mathcal{L}_{\text{ULD}}$ into sub-operators [79], and rewrite the dynamic as the composition of each sub-operation. Since we focus on sampling throughout this work, we work directly with $U(\mathbf{x}_t) = -\log p(\mathbf{x}_t)$ in deriving the numerical discretization schemes.

Numerical discretization of ULD. For the ULD (3.14), we rewrite the dynamic as

$$\mathrm{d}\begin{bmatrix}\mathbf{x}_t\\ \mathbf{v}_t\end{bmatrix} = \underbrace{\begin{bmatrix}\mathbf{C}_t\mathbf{M}^{-1}\mathbf{v}_t\\ \mathbf{0}\end{bmatrix}\mathrm{d}t}_{\text{A}} + \underbrace{\begin{bmatrix}\mathbf{0}\\ \mathbf{C}_t\nabla_{\mathbf{x}_t}\log p(\mathbf{x}_t)\end{bmatrix}\mathrm{d}t}_{\text{B}} + \underbrace{\begin{bmatrix}\mathbf{0}\\ -\gamma\mathbf{v}_t\mathrm{d}t + \sqrt{2\gamma\tau}\mathbf{M}^{\frac{1}{2}}\mathrm{d}\mathbf{W}\end{bmatrix}}_{\text{O}}. \tag{3.17}$$

The terms labeled A and B refer to the Hamiltonian components, which can be solved using any deterministic numerical integrator, and the term labeled O refers to the Ornstein-Uhlenbeck process, which has a closed-form expression when integrating it in an interval $[k\epsilon, (k+1)\epsilon)$. Given this splitting strategy, we can derive a family of schemes. The simpler one, denoted by ABO, has the following state propagation in

an interval of length ϵ

$$\exp(\epsilon\mathcal{L}_{\text{ULD}}) = \exp(\epsilon\mathcal{L}_O)\exp(\epsilon\mathcal{L}_B)\exp(\epsilon\mathcal{L}_A). \tag{3.18}$$

Therefore, given a step size ϵ, the discretization scheme is given by

$$\begin{aligned}
\mathbf{x}_{k+1} &= \mathbf{x}_k + \epsilon\mathbf{C}_\epsilon\mathbf{M}^{-1}\mathbf{v}_k, \\
\mathbf{v}_{k+1/2} &= \mathbf{v}_k + \epsilon\mathbf{C}_\epsilon\nabla_{\mathbf{x}_{k+1}}\log p(\mathbf{x}_{k+1}), \\
\mathbf{v}_{k+1} &= \exp(-\gamma\epsilon)\mathbf{v}_{k+1/2} + \sqrt{\tau(1-\exp(-2\gamma\epsilon))}\mathbf{M}^{\frac{1}{2}}\mathbf{w}_k,
\end{aligned} \tag{3.19}$$

where $\mathbf{w}_k \sim \mathcal{N}(0, \mathbf{I})$. This particular scheme corresponds to the adjoint symplectic Euler scheme to solve the Newtonian part of the Langevin dynamics SDE, followed by an exact Ornstein-Uhlenbeck solution. In general, we can construct different methods by interleaving and propagating each sub-operator k times, i.e., we propagate each sub-operator with a step-size $\frac{\epsilon}{k}$; propagating for a total time of ϵ is a requirement to have a consistent method [79]. As a result, the convention is that if each letter associated with each sub-operator appears k times, then it is updated with step-size $\frac{\epsilon}{k}$. For example, the symmetric Langevin velocity-Verlet, denoted by BAOAB, is given by the following state propagation

$$\begin{aligned}
\exp(\epsilon\mathcal{L}_{\text{ULD}}) = \exp\left(\frac{\epsilon}{2}\mathcal{L}_B\right)\exp\left(\frac{\epsilon}{2}\mathcal{L}_A\right)\exp(\epsilon\mathcal{L}_O) \\
\exp\left(\frac{\epsilon}{2}\mathcal{L}_A\right)\exp\left(\frac{\epsilon}{2}\mathcal{L}_B\right),
\end{aligned} \tag{3.20}$$

where $k = 2$ for the components A and B. We do not write the discretization obtained from this scheme as the procedure to obtain the equations is equivalent to the ABO.

The numerical integration scheme used to discretize (3.3) largely determines the performance of the algorithm [84], as we will show in Section 3.5.2. Although there are simpler schemes like the *Euler-Maruyama discretization*, which is a first-order

integrator, we use the splitting method in (3.19) as it shows better empirical performance.

Numerical discretization for third-order LD. Following a similar strategy to the ULD, we rewrite the dynamic in (3.16) as

$$\mathrm{d}\begin{bmatrix}\mathbf{x}_t\\ \mathbf{v}_t\\ \mathbf{z}_t\end{bmatrix} = \underbrace{\begin{bmatrix}\mathbf{C}_t\mathbf{M}^{-1}\mathbf{v}_t\\ \mathbf{0}\\ \mathbf{0}\end{bmatrix}\mathrm{d}t}_{\mathrm{A}} + \underbrace{\begin{bmatrix}\mathbf{0}\\ \mathbf{C}_t\nabla_{x_t}\log p(\mathbf{x}_t)\\ \mathbf{0}\end{bmatrix}\mathrm{d}t}_{\mathrm{B}} + \underbrace{\begin{bmatrix}\mathbf{0}\\ \lambda\mathbf{z}_t\\ \mathbf{0}\end{bmatrix}\mathrm{d}t}_{\mathrm{C}} + \underbrace{\begin{bmatrix}\mathbf{0}\\ \mathbf{0}\\ -\lambda\mathbf{v}_t\mathrm{d}t - \alpha\mathbf{z}_t\mathrm{d}t + \sqrt{2\tau\alpha}\mathbf{M}^{1/2}\mathrm{d}\mathbf{W}_t\end{bmatrix}}_{\mathrm{O}}. \tag{3.21}$$

In this case, we consider also two possible discretization methods. For the first one, we combine the sub-operators B and C and discretize this combination with a velocity Verlet numerical integrator [85]. We denote this method by (BC)OA(BC) and is given by

$$\mathbf{v}_{k+1/2} = \mathbf{v}_k + \frac{\epsilon}{2}\mathbf{C}_\epsilon\nabla_{x_k}\log p(\mathbf{x}_k) + \frac{\epsilon}{2}\lambda\mathbf{z}_k, \tag{3.22}$$

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \epsilon\mathbf{C}_\epsilon\mathbf{M}^{-1}\mathbf{v}_{k+1/2}, \tag{3.23}$$

$$\mathbf{z}_{k+1} = \theta\mathbf{z}_k - (1-\theta)\frac{\lambda}{\alpha}\mathbf{v}_{k+1/2} + \kappa\sqrt{\tau}\mathbf{M}^{1/2}\mathbf{w}_k, \tag{3.24}$$

$$\mathbf{v}_{k+1} = \mathbf{v}_{k+1/2} + \frac{\epsilon}{2}\mathbf{C}_\epsilon\nabla_{x_{k+1}}\log p(\mathbf{x}_{k+1}) + \frac{\epsilon}{2}\lambda\mathbf{z}_{k+1}, \tag{3.25}$$

where $\theta = \exp(-\epsilon\alpha)$ and $\kappa = \sqrt{1-\theta^2}$ [85]. Notice that we need to propagate each sub-operator for a total time of ϵ. Hence, each application of the $\mathcal{L}_{BC}$ operator – (3.22) and (3.25) – has a step size of $\epsilon/2$. The other method that we use in Section 3.5.2 is denoted by BACOCAB, following the convention that we introduced previously.

3.4.3 Sampling from the posterior via discrete annealed pre-conditioned higher-order Langevin

Recall that our goal is to solve Problem 1 by sampling (approximately) from the posterior defined in (3.2) using the discrete dynamics introduced in Section 3.4.2. As reviewed in Section 2.6, sampling from the posterior with an unconditional prior amounts to running the Langevin dynamics on the conditional score, which decomposes (Bayes' rule) as

$$\nabla_{\mathbf{x}} \log p(\mathbf{x}|\mathbf{y}, \mathbf{H}) = \nabla_{\mathbf{x}} \log p(\mathbf{y}|\mathbf{x}, \mathbf{H}) + \nabla_{\mathbf{x}} \log p(\mathbf{x}), \tag{3.26}$$

where the first term is the guidance (or likelihood) score and the second is the prior score. In addition, in this work, we leverage an annealing process, which accelerates the convergence of the dynamic [31]. Therefore, in this section, we present our algorithms based on the annealed version of both the pre-conditioned underdamped and third-order Langevin dynamics. Below, we give details on different strategies for defining both the guidance score function and the score of the prior for an annealed version of (3.26).

Annealing process. We consider an annealed version of our dynamic, where random noise of decreasing variance is added to our state $\mathbf{x}$. This procedure helps accelerate convergence and allows us to accommodate discrete states $\mathbf{x}$ [21]. First, we define a sequence of noise levels $\{\sigma_l\}_{l=1}^{L+1}$ such that $\sigma_1 > \sigma_2 > \cdots > \sigma_L > \sigma_{L+1} = 0$. Then, at each level, we define a perturbed version of the true signal $\mathbf{x}$ as

$$\tilde{\mathbf{x}}_l = \mathbf{x} + \mathbf{n}_l, \tag{3.27}$$

where $\mathbf{n}_l \sim \mathcal{N}(0, \sigma_l^2 \mathbf{I})$. Given the perturbed signal in (3.27), the forward model

in (3.1) can be rewritten as

$$\mathbf{y} = \mathbf{H}\tilde{\mathbf{x}}_l + (\mathbf{z} - \mathbf{H}\mathbf{n}_l). \tag{3.28}$$

Therefore, the annealing process entails a posterior distribution for each noise level l given by [cf. (3.26)]

$$\nabla_{\tilde{\mathbf{x}}_l} \log p(\tilde{\mathbf{x}}_l|\mathbf{y}, \mathbf{H}) = \nabla_{\tilde{\mathbf{x}}_l} \log p(\mathbf{y}|\tilde{\mathbf{x}}_l, \mathbf{H}) + \nabla_{\tilde{\mathbf{x}}_l} \log p(\tilde{\mathbf{x}}_l). \tag{3.29}$$

Thus, we have to specify both constituent terms in this score function. We now provide details of different strategies for computing each term.

i) Score of the guidance term (or likelihood): In the new forward model in (3.28), the likelihood is given by $p(\mathbf{y}|\tilde{\mathbf{x}}_l, \mathbf{H}) = p(\mathbf{z} - \mathbf{H}\mathbf{n}_l|\tilde{\mathbf{x}}_l)$, which is not Gaussian: although $p(\mathbf{n}_l)$ is a Gaussian distribution, after conditioning on $\tilde{\mathbf{x}}_l$ the conditional distribution $p(\mathbf{n}_l|\tilde{\mathbf{x}}_l)$ is no longer Gaussian. Therefore, to circumvent this, we consider *two different approximations*, one for each system model considered in Section 3.5. For the MIMO channel estimation problem, we rely on the approximation introduced in [64, 86]: $\nabla_{\tilde{\mathbf{x}}_l} \log p(\mathbf{y}|\tilde{\mathbf{x}}_l, \mathbf{H}) \approx \nabla_{\mathbf{x}} \log p(\mathbf{y}|\mathbf{x}, \mathbf{H}) = \frac{\mathbf{H}^{\mathrm{H}}(\mathbf{y}-\mathbf{H}\mathbf{x})}{\sigma_0^2}$, with $\mathbf{H}^{\mathrm{H}}$ denoting the conjugate transpose of $\mathbf{H}$. Although this is only correct when $\sigma_l = 0$, they circumvent this error by adding heuristically a sequence of annealing terms $\{\gamma_l\}_{l=0}^{L}$ such that the final score is $\nabla_{\tilde{\mathbf{x}}_l} \log p(\mathbf{y}|\tilde{\mathbf{x}}_l, \mathbf{H}) \approx \frac{\mathbf{H}^{\mathrm{H}}(\mathbf{y}-\mathbf{H}\mathbf{x})}{\sigma_0^2+\gamma_l^2}$. The second approximation, which we use for the MIMO symbol detection problem, defines the annealing process in the singular value decomposition (SVD) domain. This was considered in [63, 65, 73], and it entails a closed-form expression.

ii) Score of the prior: By Tweedie's formula (Section 2.4, (2.15)), the prior score at annealed noise level σ_l satisfies

$$\nabla_{\tilde{\mathbf{x}}_l} \log p(\tilde{\mathbf{x}}_l) = \frac{\mathbb{E}_{\sigma_l}[\mathbf{x}|\tilde{\mathbf{x}}_l] - \tilde{\mathbf{x}}_l}{\sigma_l^2}. \tag{3.30}$$

When the conditional expectation is available in closed form (as for a discrete prior over a MIMO constellation [73]) we use it directly; otherwise we approximate it by a score network trained with denoising score matching (Section 2.4).

Algorithm 1 Pre-conditioned annealed ULD with ABO discretization

Require: $T, L, \{\sigma_l, \mathbf{M}_l, \mathbf{C}_l, \epsilon_l\}_{l=1}^{L}, \sigma_0, \mathbf{H}, \mathbf{y}, \tau$

Initialize $\tilde{\mathbf{x}}_{t=0,l=1}, \mathbf{v}_{t=0,l=1}$ randomly

for $l = 1$ to L **do**

for $k = 0$ to $T - 1$ **do**

Draw $\mathbf{w}_k \sim \mathcal{N}(0, \mathbf{I})$

Compute $\nabla_{\tilde{\mathbf{x}}_{k,l}} \log p(\tilde{\mathbf{x}}_{k,l}|\mathbf{y}, \mathbf{H})$

$\tilde{\mathbf{x}}_{k+1,l} = \tilde{\mathbf{x}}_{k,l} + \epsilon_l \mathbf{C}_l \mathbf{M}_l^{-1} \mathbf{v}_{k,l}$

$\mathbf{v}_{k+1/2,l} = \mathbf{v}_{k,l} + \epsilon_l \mathbf{C}_l \nabla_{\tilde{\mathbf{x}}_{k+1,l}} \log p(\tilde{\mathbf{x}}_{k+1,l}|\mathbf{y}, \mathbf{H})$

$\mathbf{v}_{k+1,l} = e^{-\gamma\epsilon_l} \mathbf{v}_{k+1/2,l} + \sqrt{\tau(1 - e^{-2\gamma\epsilon_l})} \mathbf{M}_l^{\frac{1}{2}} \mathbf{w}_k$

end for

$\tilde{\mathbf{x}}_{0,l+1} = \tilde{\mathbf{x}}_{T,l}$, $\mathbf{v}_{0,l+1} = \mathbf{v}_{T,l}$

end for

return $\tilde{\mathbf{x}}_{T,L}$

Algorithm 2 Pre-conditioned annealed third-order LD with (BC)OA(BC) discretization

Require: $T, L, \{\sigma_l, \mathbf{M}_l, \mathbf{C}_l, \epsilon_l\}_{l=1}^{L}, \sigma_0, \mathbf{H}, \mathbf{y}, \tau, \alpha, \lambda$

Initialize $\tilde{\mathbf{x}}_{t=0,l=1}, \tilde{\mathbf{v}}_{t=0,l=1}, \tilde{\mathbf{z}}_{t=0,l=1}$ randomly

for $l = 1$ to L **do**

$\theta_l = \exp(-\epsilon_l \alpha)$

$\kappa_l = \sqrt{1-\theta_l^2}$

for $k = 0$ to $T-1$ **do**

Draw $\mathbf{w}_k \sim \mathcal{N}(0, \mathbf{I})$

Compute $\nabla_{\tilde{\mathbf{x}}_{k,l}} \log p(\tilde{\mathbf{x}}_{k,l}|\mathbf{y}, \mathbf{H})$

$\mathbf{v}_{k+1/2,l} = \mathbf{v}_{k,l} + \frac{\epsilon_l}{2}\mathbf{C}_l \nabla_{\tilde{\mathbf{x}}_{k,l}} \log p(\tilde{\mathbf{x}}_{k,l}|\mathbf{y}, \mathbf{H})$

$+\frac{\epsilon_l}{2}\lambda \mathbf{z}_{k,l}$

$\tilde{\mathbf{x}}_{k+1,l} = \tilde{\mathbf{x}}_{k,l} + \epsilon_l \mathbf{C}_l \mathbf{M}_l^{-1} \mathbf{v}_{k+1/2,l}$

$\mathbf{z}_{k+1,l} = \theta_l \mathbf{z}_{k,l} - (1-\theta_l)\frac{\lambda}{\alpha}\mathbf{v}_{k+1/2,l} + \kappa_l \sqrt{\tau}\mathbf{M}_l^{1/2}\mathbf{w}_k$

$\mathbf{v}_{k+1,l} = \mathbf{v}_{k+1/2,l} + \frac{\epsilon_l}{2}\mathbf{C}_l \nabla_{\tilde{\mathbf{x}}_{k,l}} \log p(\tilde{\mathbf{x}}_{k,l}|\mathbf{y}, \mathbf{H})$

$+\frac{\epsilon_l}{2}\lambda \mathbf{z}_{k+1,l}$

end for

$\tilde{\mathbf{x}}_{0,l+1} = \tilde{\mathbf{x}}_{T,l}$, $\mathbf{v}_{0,l+1} = \mathbf{v}_{T,l}$, $\mathbf{z}_{0,l+1} = \mathbf{z}_{T,l}$

end for

return $\tilde{\mathbf{x}}_{T,L}$

Algorithm. The algorithms to generate samples $\hat{\mathbf{x}}$ from the (approximate) posterior $p(\mathbf{x}|\mathbf{y}, \mathbf{H})$ using ULD and the third-order LD are shown in Algorithm 1 and Algorithm 2, respectively. Notice that our methods allow the inclusion of a preconditioning matrix $\mathbf{C}_l$ for each level of noise l, by considering a piecewise constant

discretization of $\mathbf{C}_t$ in (3.14) and (3.16).

Intuitively, the algorithm works as follows. First, we initialize $\mathbf{x}_0, \mathbf{v}_0$ – and $\mathbf{z}_0$ for the third-order LD – randomly. Then, at each level l of the L noise levels we run T iterations of the ULD or third-order LD, following the direction of the score function of the log-posterior density of the perturbed signal $\nabla_{\tilde{\mathbf{x}}_l} \log p(\tilde{\mathbf{x}}_l|\mathbf{y}, \mathbf{H})$. We start at a high noise level, so the contribution of the score of the prior is negligible and can be ignored, meaning that the Langevin dynamic considers the observation as a denoised estimation. Then, we gradually decrease its value until $\tilde{\mathbf{x}} \approx \mathbf{x}$; in the low noise levels, the idea is to refine the estimation by considering the prior information combined with the guidance term. Apart from enabling the approximation of the score function of the prior distribution, the annealing process also improves the mixing time of the Langevin dynamic [31]. Notice that the auxiliary variables are defined at each noise level. Finally, similar algorithms can be considered for other discretization schemes, like the ones we used in Section 3.5.2. We omit them here to avoid redundancy.

Regarding the computational complexity, it strongly depends on the expression of (3.29). Thus, we define the computational complexity for each system in Section 3.5.

3.5 Numerical results

We start by applying our framework to the symbol detection problem in a massive MIMO system. In particular, we compare the symbol error rate (SER) performance of both algorithms introduced in Section 3.4.3 in comparison with the annealed overdamped case. Then, we compare our methods with both classical and learning-based baseline detectors and, finally, we analyze the complexity in terms of running time. Second, in Section 3.5.2, we consider the problem of channel estimation. We first

study the normalized mean squared error (NMSE) performance of our methods considering different discretization schemes, and then their performance compared to the overdamped case. Finally, we compare with baseline methods and show that the third-order dynamic outperforms all the other methods.

3.5.1 Massive MIMO detection

In this section, we analyze the proposed methods for the problem of massive MIMO symbol detection. This problem can be cast as (3.1), where N_u are the number of single-antenna transmitters or users and N_r the antennas at the base station. In this context, $\bar{\mathbf{H}} \in \mathbb{C}^{N_r \times N_u}$ is the channel matrix, $\bar{\mathbf{z}} \sim \mathcal{CN}(\mathbf{0}, \sigma_0^2 \mathbf{I}_{N_r})$ is a vector of complex circular Gaussian noise, $\bar{\mathbf{y}} \in \mathbb{C}^{N_r}$ is the received vector, $\bar{\mathbf{x}} \in \mathcal{X}^{N_u}$ is the vector of transmitted symbols, which *is the unknown variable*, and $\mathcal{X}$ is a finite set of constellation points. Although the variables involved are complex, we can rewrite the model in its equivalent real-valued representation obtained by considering the real $\Re(.)$ and imaginary $\Im(.)$ parts separately. Define $\mathbf{x} = [\Re(\bar{\mathbf{x}})^\top, \Im(\bar{\mathbf{x}})^\top]^\top$, $\mathbf{y} = [\Re(\bar{\mathbf{y}})^\top, \Im(\bar{\mathbf{y}})^\top]^\top$, $\mathbf{z} = [\Re(\bar{\mathbf{z}})^\top, \Im(\bar{\mathbf{z}})^\top]^\top$ and

$$\mathbf{H} = \begin{bmatrix} \Re(\bar{\mathbf{H}}) & -\Im(\bar{\mathbf{H}}) \\ \Im(\bar{\mathbf{H}}) & \Re(\bar{\mathbf{H}}) \end{bmatrix}. \tag{3.31}$$

Thus, the system can be rewritten in the equivalent real-valued representation as in (3.1). As we assume that the symbols' prior distribution is uniform among the constellation elements and the measurement noise $\mathbf{z}$ is Gaussian, the maximum a posteriori (MAP) detector boils down to a maximum likelihood (ML) detector

$$\hat{\mathbf{x}}_{\text{ML}} = \underset{\mathbf{x} \in \mathcal{X}^{N_u}}{\operatorname{argmin}} \ ||\mathbf{y} - \mathbf{H}\mathbf{x}||_2^2, \tag{3.32}$$

The exact solution to this problem is NP-hard due to the discrete nature of the constellation $\mathcal{X}$. Therefore, approximate solutions have been proposed. We consider

the following existing solutions as baseline methods.

- **MMSE** [59]: Linear detector, which formally solves first a relaxation of (1), by considering $\mathbf{x} \in \mathbb{C}^{N_u}$ instead of $\mathbf{x} \in \mathcal{X}^{N_u}$, and then projects the solution back onto one of the constellation elements.

- **V-BLAST** [87]: Multi-stage interference cancellation BLAST algorithm using zero-forcing as the detection stage.

- **RE-MIMO** [88]: Recurrent permutation equivariant neural detector based on an encoder-predictor architecture. The encoder and the predictor are parameterized by a transformer and a multi-layer perceptron, respectively. We follow the training procedure proposed in the paper.

- **OAMPNet** [89]: Unfolding of the orthogonal approximate message passing (OAMP) algorithm. We use 10 layers as it is proposed in the paper. At each layer, a matrix pseudo inverse is required and has 2 learnable parameters.

- **Overdamped Langevin-based detector** [**73**]: Detector based on an annealed version of overdamped Langevin dynamics, where the symbol's estimation is obtained by sampling from the (approximate) posterior distribution. We specify the hyperparameters in Table 3.1.

- **ML**: The optimal solver for (3.32) using Gurobi [90], a highly-optimized mixed integer programming package.

The signal-to-noise ratio (SNR) is given by

$$\text{SNR} = \frac{\mathbb{E}[||\mathbf{Hx}||^2]}{\mathbb{E}[||\mathbf{z}||^2]}. \tag{3.33}$$

For all the experiments in the following subsections, we consider a 16-QAM modulation. The simulation environment includes a base station with $N_r = 64$ receiver antennas and $N_u = 32$ single-antenna users. We consider a Kronecker correlated model given by

$$\mathbf{H} = \mathbf{R}_r^{1/2}\mathbf{H}_e\mathbf{R}_u^{1/2}, \tag{3.34}$$

where $\mathbf{H}_e$ is a Rayleigh fading channel matrix and $\mathbf{R}_r$ and $\mathbf{R}_u$ are the spatial correlation matrices at the receiver and transmitters, respectively. These correlation matrices are generated according to the exponential correlation matrix model with a coefficient $\rho = 0.6$; see [91] for details. Before going to the experiments, we give the closed-form expression of the constituent terms in (3.29), based on the SVD of the channel matrix given by $\mathbf{H} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$; for details see [73].

Algorithm and computational complexity. We define the spectral representation of $\tilde{\mathbf{x}}_l$ and $\mathbf{y}$ as $\tilde{\boldsymbol{\chi}}_l = \mathbf{V}^\top\tilde{\mathbf{x}}_l$ and $\boldsymbol{\eta} = \mathbf{U}^\top\mathbf{y}$, and s_j the j-th singular value. Then, the score of the likelihood is:

$$\nabla_{\tilde{\boldsymbol{\chi}}_l}\log p(\boldsymbol{\eta}|\tilde{\boldsymbol{\chi}}_l, \mathbf{H}) = \mathbf{\Sigma}^\top \ |\sigma_0^2\mathbf{I} - \sigma_l^2\mathbf{\Sigma}\mathbf{\Sigma}^\top|^\dagger \ (\boldsymbol{\eta} - \mathbf{\Sigma}\tilde{\boldsymbol{\chi}}_l). \tag{3.35}$$

For the score of the prior [cf. (3.30)], we need the conditional expectation, which can be calculated elementwise as

$$\mathbb{E}_{\sigma_l}[x_j|[\tilde{\mathbf{x}}_l]_j] = \frac{1}{Z}\sum_{x_k\in\mathcal{X}} x_k \exp\left(\frac{-||[\tilde{\mathbf{x}}_l]_j - x_k||^2}{2\sigma_l^2}\right), \tag{3.36}$$

where $Z = \sum_{x_k\in\mathcal{X}} \exp\left(\frac{-||[\tilde{\mathbf{x}}_l]_j - x_k||^2}{2\sigma_l^2}\right)$ and $j = 1, \cdots, N_u$. Then, given the orthogonality of $\mathbf{V}$, we have $\nabla_{\tilde{\boldsymbol{\chi}}_l}\log p(\tilde{\boldsymbol{\chi}}_l) = \mathbf{V}^\top\nabla_{\tilde{\mathbf{x}}_l}\log p(\tilde{\mathbf{x}}_l)$.

Additionally, given that the sample after the annealing process will be very close to the constellation but not exactly, we take $\bar{\mathbf{x}} = \text{argmin}_{\mathbf{x}\in\mathcal{X}^{N_u}} ||\mathbf{x} - \mathbf{V}\tilde{\boldsymbol{\chi}}_{T,L}||_2^2$. In our implementation, we generate U different Langevin samples $\{\bar{\mathbf{x}}_u\}_{u=1}^U$ for each pair

$\{\mathbf{y}, \mathbf{H}\}$ by running Algorithms 1 and 2 multiple times and keep the sample that minimizes

$$\hat{\mathbf{x}} = \underset{\mathbf{x}\in\{\bar{\mathbf{x}}_u\}_{u=1}^{U}}{\operatorname{argmin}} ||\mathbf{y} - \mathbf{H}\mathbf{x}||_2^2. \tag{3.37}$$

Notice that these U Langevin trajectories can be run in parallel, as they are independent of each other. We fix $U = 20$ as explained in [73].

We set $\gamma = 1$, and for the third-order, $\lambda = 1$ and $\alpha = 1.2$. Moreover, we consider the same ϵ for all the noise levels, given by $\epsilon = \frac{\epsilon_0}{\sigma_L^2}$. Lastly, the pre-condition matrix is a diagonal matrix that depends on the noise level l, and is given by

$$[\mathbf{C}_l]_{jj} = \begin{cases} \sigma_l^2 \left(1 - \frac{\sigma_l^2}{\sigma_0^2} s_j^2\right) & \text{if } \sigma_l s_j \leq \sigma_0 \\ \sigma_l^2 - \frac{\sigma_0^2}{s_j^2} & \text{if } \sigma_l s_j > \sigma_0 \end{cases} \tag{3.38}$$

and the mass as $\mathbf{M}_l = \frac{\gamma^2}{4}\mathbf{C}_l^{-1}$. Now, we present the results of the experiments. In the three experiments in this section, we consider the ABO numerical method for the underdamped and the (BC)OA(BC) for the third-order algorithm.

Regarding the computational complexity, the first step in this case is to compute the SVD of the channel $\mathbf{H}$, whose complexity is $\mathcal{O}(N_u N_r \min\{N_u, N_r\})$, and is done only once per channel. Then, the discretization schemes entail three steps. The first one as well as the third are just vector summations, since we consider the mass parameter $\boldsymbol{\Xi} = \xi\mathbf{I}$ to be just a scalar in this work. The second step requires the computation of (3.35) and (3.36), which entails a complexity of $\mathcal{O}(N_u^2 + KN_u)$ per iteration. Therefore, the overall complexity, including the SVD computation and all the iterations, is $\mathcal{O}(N_u N_r \min\{N_u, N_r\} + LT(N_u^2 + KN_u))$. Compared to the overdamped case (see [73]), we see that we are not adding computational burden to the detector. Regarding the M trajectories, observe that these are independent of each other, so they can be computed in parallel.

Table 3.1 : Hyperparameters of the different algorithms based on Langevin dynamics.

Parameter	Overdamped			Underdamped			Third-order		
L	5	10	20	5	10	20	5	10	20
σ_1	0.4	1	1	0.4	1	1	0.4	1	1
σ_L	0.02	0.01	0.01	0.02	0.01	0.01	0.02	0.01	0.01
ϵ_0	6×10^{-4}	3×10^{-5}	3×10^{-5}	6×10^{-4}	3×10^{-5}	3×10^{-5}	2.2×10^{-4}	5×10^{-5}	5×10^{-5}
T	30	70	70	30	70	70	30	70	70
τ	0.01	0.5	0.5	0.01	0.5	0.5	0.023	0.084	0.084

Comparison with the overdamped Langevin detector. In this experiment, we compare the advantage of using the pre-conditioned annealed underdamped and third-order Langevin detector w.r.t. the overdamped case. We consider three cases where $L \in \{5, 10, 20\}$. All hyperparameters are summarized in Table 3.1 and the results are shown in Fig. 3.1a. First, we observe that when $L = 20$, all methods have comparable performance. On the other hand, for both $L = 10$ and $L = 5$, our proposed methods outperform the overdamped Langevin detector, with the third-order based detector outperforming the underdamped case. Notice that the number of iterations ($L \times T$) was reduced by a factor of 10 (150 vs 1400) and 2 for $L = 5$ and $L = 10$, respectively. Therefore, this experiment illustrates that there is a *trade-off between performance and running time* and that *adding auxiliary variables entails a better performance with a reduced running time.*

Comparison with other methods. Based on our previous experiments, we consider the case of $L = 5$ for all the methods as it is the case with the best trade-off between running time and performance. The comparison with baseline detectors is shown in Fig. 3.1b. The figure reveals that our proposed method markedly outper-

forms the other detectors. Notice that our method can handle a varying number of users without the need for any retraining as required in, e.g., OAMPNet [89]. This is key in MIMO communications, as the number of users connected to the network might be constantly changing.

Running time comparison. In this third experiment, we compute the SER of our method and the other baselines w.r.t. running time in ms/symb. We assume a coherence time such that each block contains 1000 samples. The comparison is shown in Fig. 3.1c for an SNR $= 16\,\text{dB}$, $U = 1$, and 5000 symbols. Given the coherence time, we have to compute 5 SVDs that correspond to each $\{\mathbf{H}_i\}_{i=1}^{5}$. First, notice that both underdamped and overdamped methods have the same performance when considering $L = 20$, something expected given the result in the first experiment. Moreover, the third-order slightly outperforms both. However, when $L = 5$, both the underdamped and the third-order cases successfully reduce the running time while achieving a better performance compared to the overdamped case. Furthermore, the third-order outperforms the underdamped without adding computational burden.

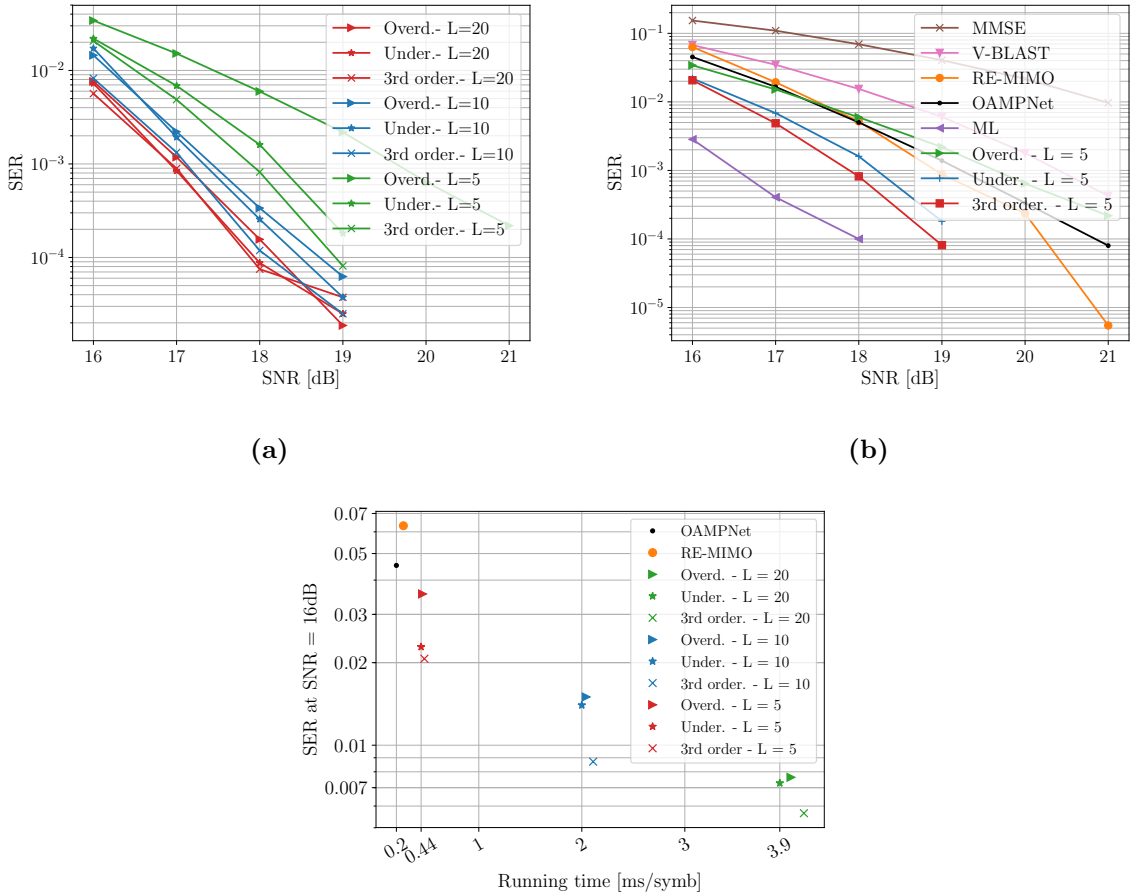


Figure 3.1 : Performance analysis of our proposed methods for MIMO symbol detection considering SER as a function of SNR. (a) Comparison between our annealed underdamped and third-order Langevin method and the overdamped variant for $L \in \{5, 10, 20\}$. (b) Comparison with classical and learning-based detection methods. (c) Comparison with different detection methods for an SNR $= 16\,\text{dB}$ and varying the running time in ms/symb.

3.5.2 Channel estimation with learned score prior

We analyze the proposed methods for the problem of MIMO channel estimation. Again, we cast this problem as in (3.1), where N_u are the number of single-antenna transmitters or users and N_r the antennas at the base station. The channel matrix

$\mathbf{H} \in \mathbb{C}^{N_r \times N_u}$ is *the unknown variable*, $\mathbf{Z}$ is a random matrix where each column $\mathbf{z} \sim \mathcal{CN}(\mathbf{0}, \sigma_0^2 \mathbf{I}_{N_r})$ is a vector of complex circular Gaussian noise, and $\mathbf{Y} \in \mathbb{C}^{N_r \times N_p}$ is the received vector. In this problem, the variable $\mathbf{x}$ is a matrix denoted as $\mathbf{P} \in \mathcal{X}^{N_u \times p}$, which are known as pilots, and typically they are pre-specified. Thus, the forward model is $\mathbf{Y} = \mathbf{HP} + \mathbf{Z}$, and the optimal solution in 3.2 is

$$\hat{\mathbf{H}}_{\text{MAP}} = \underset{\mathbf{H} \in \mathbb{C}^{N_r \times N_u}}{\operatorname{argmax}} \; p_{\mathbf{z}}(\mathbf{Y} - \mathbf{HP}) p(\mathbf{H}). \tag{3.39}$$

Similar to the case of symbol detection, the problem of channel estimation can also be rewritten to treat the real and imaginary parts separately [64], boiling down to the expression in (3.1). The dimension $N_p = \alpha_p N_u$ denotes the number of pilots, and the value of α_p determines if the problem is ill-conditioned or not; if $\alpha < 1$, then the problem is underdetermined. We consider $\alpha_p = 0.6$. Thus, it becomes necessary to incorporate prior knowledge about the channel distribution to obtain an accurate estimation. We describe the methods that we use as baselines.

- **Lasso** [92]: This is a compressive-sensing-based approach that uses ℓ_1-norm element-wise regularization in the two-dimensional Fourier (beamspace) domain.

- **fsAD** [93]: This is another classical compressive-sensing-based approach for recovering channel matrices assumed sparse in the continuous spatial frequency domain, similar to Newtonized OMP (orthogonal matching pursuit) [94].

- **L-DAMP** [95,96]: This method is a data-driven algorithm that uses algorithm unrolling [97] and end-to-end learning. A different model is trained for each SNR.

- **Overdamped Langevin-based detector** [64]: This is the annealed first-order Langevin dynamic. The score of the prior is learned from data using denoising score-matching.

In all the experiments, we consider the SNR as in (3.33) and the estimation quality as the normalized mean squared error (NMSE)

$$\text{NMSE[dB]} = 10\log_{10}\frac{||\hat{\mathbf{H}}-\mathbf{H}||_F^2}{||\mathbf{H}||_F^2}. \tag{3.40}$$

For the channel model, we use the standardized clustered delay line (CDL)-C model, which is adopted in 5G specifications [98]. We consider $N_u = 64$ and $N_r = 16$.

Algorithm and computational complexity. In this case, the score of the likelihood is:

$$\nabla_{\tilde{\mathbf{H}}_l}\log p(\mathbf{Y}|\tilde{\mathbf{H}}_l,\mathbf{P}) \approx \frac{(\mathbf{Y}-\tilde{\mathbf{H}}_l\mathbf{P})\mathbf{P}^{\mathrm{H}}}{\sigma_0^2+\sigma_l^2}. \tag{3.41}$$

For the score of the prior [cf. (3.30)], we parameterize it with a neural network, known as score *network*, and we train using denoising score-matching [31]. We use the architecture proposed in [99]. We trained the model as it is explained in [64], and we use the same score network for our higher-order methods.

For the step-size, we consider $\epsilon_l = \epsilon_0\frac{\sigma_l^2}{\sigma_L^2}$. For the underdamped case we consider $\gamma = 1$, while for the third-order case we set $\lambda = 1$ and $\alpha = 2$. Moreover, we consider $\epsilon_0 = 2.2\times 10^{-10}$, $L = 58$, and $T = 3$ for both methods, and an early stopping at the iteration 60. The mass matrix depends on the SNR level: we consider $\mathbf{M} = 3\mathbf{I}$ for the first two levels of SNR and $\mathbf{M} = 2\mathbf{I}$ for the remaining levels. Lastly, the pre-condition matrix is not used, i.e., $\mathbf{C}_l = \mathbf{I}$.

The computational complexity depends on the neural network in this case, which is explained in [64]. However, similar to the experiment in Section 3.5.1, both algorithms 1 and 2 do not add computational burden.

Comparison between discretization schemes. In this first experiment, we compare the performance of the annealed underdamped and third-order algorithms when considering different discretization schemes. In particular, we consider the methods introduced in Section 3.4.2, i.e., ABO and BAOAB for ULD, and (BC)OA(BC) and BACOCAB for the third-order dynamic. The results are shown in Fig. 3.2a. First, notice that the annealed underdamped algorithm is unstable when considering the ABO discretization, while it becomes stable when considering the BAOAB method. In the case of the third-order dynamic, the results reveal that it is more robust to the choice of numerical discretization. However, we observe that the BACOCAB outperforms the (BC)OA(BC), in particular for SNRs higher than 0 dB. Overall, we see that in both cases, propagating more times each sub-operator within one discretization interval entails a better performance.

Comparison with the overdamped Langevin detector. In this second experiment, given the discretizations BAOAB and BACOCAB for the underdamped and third-order respectively, we compare their performance with the overdamped case proposed in [64], which uses $L = 2311$. Also, we add the case of $L = 58$, with the same hyperparameters as the underdamped and third-order case. The results are shown in Fig. 3.2b. Although the overdamped case with $L = 2311$ outperforms our proposed methods, the difference compared to the third-order case is small for SNRs higher than 0 dB. For lower SNRs, all the methods have a similar performance. There is a trade-off between running time and performance, and which to prioritize depends on the accuracy of the estimation that is needed.

Comparison with other methods. Based on our previous experiments, we consider the case of $L = 58$ for all the methods as it is the case of less complexity within

all the combinations that we proposed and compare with the baseline detectors. The comparison is shown in Fig. 3.2c. We observe that our proposed methods outperform the other baselines for SNRs greater than 0 dB. Thus, despite the trade-off that was shown in the previous experiment where our proposed annealed third-order algorithm performs slightly worse than the overdamped case, it still outperforms the other methods.

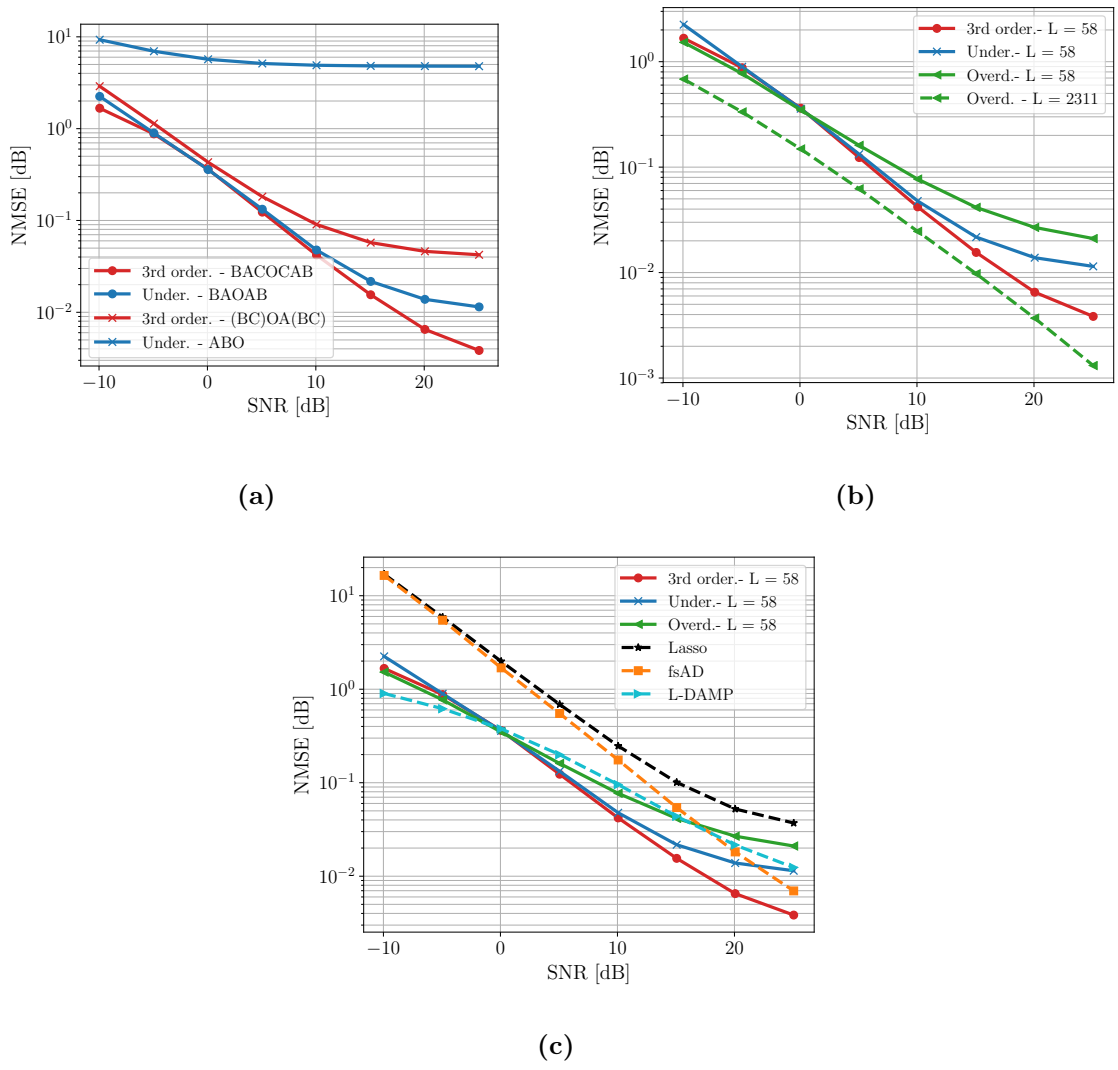


Figure 3.2 : Performance analysis of our proposed methods for channel estimation considering NMSE as a function of SNR in a CDL-C channel model. (a) Comparison of our proposed annealed underdamped and third-order Langevin methods using different discretization schemes. (b) Comparison of our proposed annealed underdamped and third-order Langevin method and the annealed overdamped Langevin. (c) Comparison with baseline methods.

3.5.3 Inverse problem in imaging with learned score prior

Finally, we analyze the proposed methods for three inverse problems in imaging, namely image deblurring, inpainting, and super-resolution. In this context, the degradation operator $\mathbf{H}$ is known, and we want to estimate the signal given a noisy observation. Thus, the problem is equivalent to (3.2). Similar to the experiment in Section 3.5.2, the problem is ill-conditioned, and therefore, we need to regularize the problem by incorporating prior information. We do this by using the score network proposed in [99] as well as the pre-trained model on the FFHQ dataset [100]. We consider 100 validation samples at random, and the following methods as baselines

- **DDRM** [52]: Denoising diffusion restoration model, which is based on the denoising diffusion probabilistic model [6]. We use the pre-trained model from [101]. We consider the same setting proposed by the authors.

- **SNIPS** [63]: This is the annealed first-order Langevin dynamic. For the score network, we use the pre-trained model from [99]. We consider 20 levels of noise, with 3 levels per sample, $\sigma_1 = 348$ and $\sigma_L = 0.01$.

- **DPS** [2]: Solver of general noisy (non)linear inverse problems based on diffusion and an approximation of the posterior sampling. To be fair in the comparison, we consider 60 noise levels, which corresponds to the same number of evaluations of the NN as Langevin with 20 levels of noise and 3 samples per level. We use the pre-trained model given by the authors.

We generate only one sample with SNIPS and our two proposed methods, similar to diffusion-based models. As evaluation metrics, we consider PSNR, LPIPS [23], and FID [24]. See the appendix for the images obtained for each task and the hyper-

parameters of our methods. The results for noiseless experiments – $\sigma_0 = 0.0002$ – are in Table 3.2, while for noisy case – $\sigma_0 = 0.05$ – in Table 3.3.

Image deblurring with anisotropic Gaussian filter. In this first experiment, we consider the anisotropic Gaussian blur kernel with $\sigma = 20$ horizontally and $\sigma = 1$ vertically, which was used in [52]. We observe that for the noiseless case, the algorithm based on 3rd-order Langevin outperforms the other baselines. Furthermore, we observe a similar behavior to the experiments in Section 3.5.2: when the SNR between the signal and the noise is high – noiseless case –, then the performance when considering higher-order methods increases as well. However, when the SNR is low – noisy case – the three dynamics achieve similar performance. In this case, the 3rd order Langevin sampler outperforms all the other methods in LPIPS and FID, while for the PSNR DDRM achieves the best performance.

Image inpainting. For inpainting, we randomly drop 50% of the pixels from the original image. Again, the algorithm based on 3rd order Langevin outperforms the other baselines. We can see also that for the noisy case the three Langevin dynamics perform similarly, while for the noiseless case, the gap in performance increases. Notice also that our proposed method outperforms DDRM in the noisy setting.

Super-resolution. Lastly, we consider super-resolution, using a block averaging filter of 4 on each axis. In this case, DDRM achieves the best performance in terms of PSNR, although the 3rd-order Langevin sampler achieves a slightly better LPIPS in the noiseless case.

Table 3.2 : Quantitative results on noiseless images ($\sigma_0 = 0.0002$) from FFHQ. We consider the average PSNR[dB], LPIPS and FID of Gaussian deblurring, inpainting and $4\times$ SR. In bold is the best method for each metric and experiment.

	Deblurr (Gauss)			Inpainting			SRx4		
Method	PSNR [dB] ↑	LPIPS ↓	FID ↓	PSNR [dB] ↑	LPIPS ↓	FID ↓	PSNR [dB] ↑	LPIPS ↓	FID ↓
DDRM	35.82	0.017	13.81	30.77	0.04	38.74	**28.53**	0.13	**58.06**
DPS	18.64	0.36	110.65	19.06	0.35	117.27	15.64	0.45	120.38
SNIPS	34.76	0.009	22.51	30.68	0.035	35.2	28.18	0.15	68.05
Underd. Lang	35.55	0.008	16.34	30.98	0.026	29.17	28.22	0.13	61.41
3rd. order Lang	**36.78**	**0.004**	**11.22**	**31.46**	**0.02**	**21.65**	28.41	**0.12**	58.81

Table 3.3 : Quantitative results on noisy ($\sigma = 0.05$) images from FFHQ. We consider the average PSNR[dB], LPIPS and FID of Gaussian deblurring, inpainting and $4\times$ SR. In bold is the best method for each metric and experiment.

	Deblurr (Gauss)			Inpainting			SRx4		
Method	PSNR [dB] ↑	LPIPS ↓	FID ↓	PSNR [dB] ↑	LPIPS ↓	FID ↓	PSNR [dB] ↑	LPIPS ↓	FID ↓
DDRM	**28.25**	0.175	84.09	29.2	0.09	52.8	**27.02**	**0.187**	**90.54**
DPS	18.14	0.38	114.32	18.64	0.37	115.7	15.53	0.46	117.67
SNIPS	27.2	0.21	92.5	29.54	0.077	57.61	26.33	0.22	104.17
Underd. Lang	27.06	0.18	89.14	29.4	0.06	52.49	26.16	0.21	100.61
3rd. order Lang	27.29	**0.173**	**82.83**	**29.73**	**0.054**	**44.13**	26.36	0.191	94.17

3.6 Conclusion

We proposed a pre-conditioned annealed higher-order Langevin sampler for linear inverse problems, and validated it on massive MIMO detection, channel estimation, and image restoration. Two chapter-specific limitations are worth noting. First, the sampler introduces additional hyperparameters (momentum, preconditioner, discretization) whose automatic selection — for example via an unrolling framework [97,

102, 103] — is a natural next step. Second, joint estimation of the linear operator $\mathbf{H}$ and the hidden signal $\mathbf{x}$ [104] is left open. Chapter 9 discusses broader directions, including non-asymptotic convergence guarantees under approximate scores and combinations with the training-time modifications of Part II.

Chapter 4

Repulsive latent score distillation

Latent diffusion models such as Stable Diffusion provide expressive priors for image inverse problems, but posterior sampling in the compressed latent space is notoriously prone to *mode collapse*: variational schemes concentrate mass on a single mode of the posterior and produce blurred or unrepresentative reconstructions. In this chapter we recast posterior sampling as a variational optimization problem in latent space and introduce a repulsive coupling across an ensemble of particles that provably prevents collapse. The resulting method, *repulsive latent score distillation* (RLSD), inherits the low per-step cost of variational latent samplers while restoring the diversity of the underlying posterior. Applied to linear and non-linear image inverse problems — inpainting, deblurring, non-linear phase retrieval — RLSD delivers competitive fidelity and substantially improved perceptual diversity relative to prior latent-space posterior samplers.

This chapter is adapted from [10].

4.1 Introduction

Latent diffusion models such as Stable Diffusion [105] provide expressive priors for image restoration, and Section 2.6 reviewed how these priors can be combined with a measurement likelihood to solve inverse problems. Two families dominate: *diffusion posterior samplers* [2, 53, 106] run the reverse SDE with a plug-in approximation of

the noisy likelihood, and *variational samplers* [50, 51] recast posterior sampling as a stochastic optimization problem. This chapter targets the variational family and attacks its two well-known pathologies when combined with latent diffusion: *mode collapse* and *latent-space inversion*.

The first challenge ($c1$) mode collapse is inherent to variational samplers with a unimodal q: minimizing the reverse KL against a multimodal posterior concentrates all mass on a single mode. Prior attempts to inject diversity through independent particles [107] or Stein Variational Gradient Descent [108, 109] either collapse to the same local minimum or fail in high dimensions [110, 111]. The second challenge ($c2$) latent-space inversion arises because the adversarially trained autoencoder of Stable Diffusion makes it hard to recover latents whose decoder image matches the measurement; latent-space samplers [112–114] require expensive correction steps to stay on the image manifold, echoing the classical GAN-inversion problem [115, 116]. To our knowledge, no prior work handles both pathologies simultaneously.

In this chapter we introduce *Repulsive Latent Score Distillation* (RLSD), a variational sampler that addresses both. Against ($c1$) we replace the unimodal q with an ensemble of interactive particles subject to a *repulsive* coupling inspired by kernel-density estimation [117], defined through pairwise similarities in DINO feature space [118]. Against ($c2$) we augment the variational distribution to jointly optimize latent and pixel variables, in the spirit of half-quadratic splitting [119]. KL minimization of the resulting augmented particle approximation yields a solver with two interpretable gradient terms: a *denoising* regularizer along the diffused trajectory and a *repulsion* regularizer that promotes diversity in the latent trajectory (see Fig. 4.1). RLSD combines the compute efficiency of variational samplers with the diversity of posterior samplers, controlled by a single scalar weight, and delivers state-of-the-art perfor-

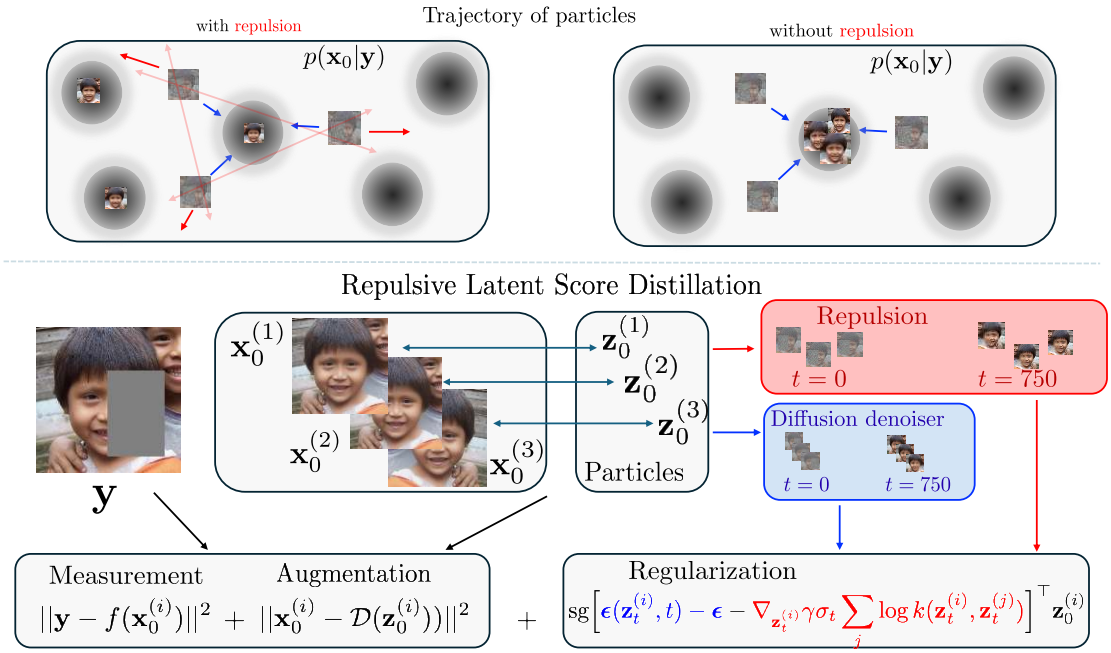


Figure 4.1 : Illustration of Repulsive Latent Score Distillation (RLSD): It propagates a set of particles by adding noise and applying two levels of regularization: (*i*) **Denoising**, via score-matching regularization, which directs particles toward modes of the distribution $p(\mathbf{x}_0|\mathbf{y})$ (blue arrows); and (*ii*) **Repulsion**, which pushes particles apart (red arrows) to explore other regions of the posterior density. During sampling, the repulsion gradient ensures particles remain separated, leading to different modes, as shown in the upper-right box.

mance on linear and non-linear inverse problems at 512×512 resolution, including inpainting and phase retrieval.

A detailed comparison of RLSD's properties against representative posterior samplers and variational samplers is summarized in Table 4.1.

	Posterior samplers		Variational samplers			
	DPS	PSLD	RED-Diff	VSD	CSD	**RLSD (Ours)**
Diversity (Low-dim)	✓	✓	✗	✓	✓	✓
Diversity (High-dim)	✓	✓	✗	✗	✗	✓
Latent Diffusion	✗	✓	✗	✓	✓	✓
Linear Inv. Problems	✓	✓	✓	-	-	✓
Nonlinear Inv. Problems	✓	-	✓	-	-	✓
No Score Jacobian	✗	✗	✓	✓	✓	✓

Table 4.1 : Comparison of our work with DPS, RED-Diff, PSLD, VSD, and CSD. Our method combines the strengths of variational samplers (no score Jacobian, computational efficiency) and posterior sampling algorithms (diversity at high dimensions). In addition, our formulation enables us to solve nonlinear inverse problems at 512×512.

4.2 Related Works

This chapter is primarily related to diffusion models at its core, and two related lines of work: inverse problems and score distillation sampling.

Diffusion models for inverse problems.

Several works have used diffusion models as priors to solve inverse problems in various domains [120, 121]. A recent approach termed RED-Diff [50] uses variational inference for solving inverse problems with diffusion priors, similar to plug-and-play methods [44]; see also [122, 123]. This method employs the diffusion model as a denoiser at different scales, akin to the RED framework [43]. Despite successfully balancing quality and runtime, it suffers from mode collapse due to the unimodal approximation. Furthermore, optimizing directly in the pixel domain restricts the

ability to leverage latent diffusion models like Stable Diffusion [105] for solving inverse problems at high-resolution. Recent works incorporate latent diffusion models as prior [112–114]. While they partially alleviate the computational demands of pixel-domain solvers, they introduce additional steps to correct the deviations from the image manifold, which arise from the adversarial training of the autoencoder. Thus, it is still an open problem to develop methods that are fast, promote diversity, and optimize in the latent space of the diffusion model.

Score distillation: diversity and mode collapse.

Recently, SDS enabled the use of pre-trained diffusion models for text-to-3D generation [51]. Although SDS provides an efficient mechanism for the aforementioned task, it often suffers from mode collapse and saturated images. ProlificDreamer [107] aims to fix the mode collapse using a data-driven dispersion fine-tuned at each iteration via LoRA [124]. It is, however, costly, and the independence of particles hinders diversity. Recently, the authors in [108] propose to use the well-known Stein variational gradient descent (SVGD) as an update direction, which yields an interactive particle system. However, it is known that SVGD suffers from the curse of dimensionality [110]. Other related works are [125], where the authors leverage the negative prompt to eliminate undesired perspectives, and [126], where an entropic regularization is proposed.

4.3 Background

The general background on diffusion models, latent diffusion models, denoising score matching, and posterior sampling with diffusion priors is developed in Sections 2.2–2.6. Here we recap only what is needed to state the RLSD contribution: notation for latent diffusion models specialized to inverse problems, and the RED-diff variational

reformulation of posterior sampling on which RLSD builds.

Latent diffusion notation. Following the general setup of Section 2.3, we work with a pre-trained latent diffusion model consisting of an encoder $\mathcal{E}_\phi(\mathbf{x}_0) = \mathbf{z}_0$, a decoder $\mathcal{D}_\phi(\mathbf{z}_0) \approx \mathbf{x}_0$, and a score network trained on the latent distribution. We use the variance-preserving parameterization so that diffused latents satisfy $\mathbf{z}_t = \alpha_t \mathbf{z}_0 + \sigma_t \boldsymbol{\epsilon}$ with $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and $\alpha_t^2 + \sigma_t^2 = 1$, and the score network is parameterized as a noise predictor $\boldsymbol{\epsilon}_\theta(\mathbf{z}_t, t) \approx -\sigma_t \nabla_{\mathbf{z}_t} \log p(\mathbf{z}_t)$. Stable Diffusion [105] is the canonical instance and the one we use in experiments.

Inverse problem. We consider the general (possibly non-linear) forward model

$$\mathbf{y} = \mathcal{A}(\mathbf{x}_0) + \mathbf{w}, \qquad \mathbf{w} \sim \mathcal{N}(\mathbf{0}, \sigma_0^2 \mathbf{I}), \tag{4.1}$$

in pixel space. Sampling from the posterior $p(\mathbf{x}_0|\mathbf{y}) \propto p(\mathbf{y}|\mathbf{x}_0)p(\mathbf{x}_0)$ with a latent diffusion prior is the object of interest. As discussed in Section 2.6, the two dominant families of solvers are (i) *diffusion posterior sampling* approaches [2,53,106,127], which run the reverse process using a Gaussian approximation of the noisy likelihood based on Tweedie's formula, at the cost of expensive score-Jacobian evaluations [112]; and (ii) *variational* approaches, described next, which sidestep the reverse process and the score Jacobian entirely.

RED-diff variational sampling. RED-diff [50] formulates posterior sampling as a variational optimization*

$$q^\star(\mathbf{z}_0|\mathbf{y}) \;=\; \operatorname*{argmin}_{q(\mathbf{z}_0|\mathbf{y})} \; \mathrm{KL}\big(q(\mathbf{z}_0|\mathbf{y}) \,\|\, p(\mathbf{z}_0|\mathbf{y})\big), \tag{4.2}$$

*RED-diff was originally proposed in the pixel domain; we express it in the latent domain for consistency with the setup above.

with $\mathbf{x}_0 = \mathcal{D}_\phi(\mathbf{z}_0)$. When $q(\mathbf{z}_0|\mathbf{y}) \sim \mathcal{N}(\boldsymbol{\mu}_z, \sigma_z^2\mathbf{I})$, (4.2) reduces to a tractable MAP-style optimization in which the diffusion model's trajectory acts as a regularizer. This is cheap and avoids the score Jacobian, but the unimodal Gaussian q suffers from two structural pathologies: it collapses to a single mode of the posterior (bad for highly ill-posed problems such as inpainting), and, in latent space, its interaction with the adversarially trained decoder biases reconstructions toward blurred images (Appendix B.2.8). RLSD addresses both.

4.4 Repulsive variational diffusion sampling

In Section 4.4.1, we address ($c1$) by introducing a repulsion mechanism, promoting diversity through a multimodal variational approximation using interactive particles. Then, in Section 4.4.2, we tackle ($c2$) by proposing an augmented variational formulation. Finally, in Section 4.4.3 we combine both techniques to derive **Repulsive Latent Score Distillation** (RLSD), our proposed solver for inverse problems using latent diffusion models.

4.4.1 Tackling mode collapse: enhancing diversity via repulsion

We aim to solve inverse problems characterized by the forward model in (4.1) by minimizing the reverse KL divergence (4.2). However, as explained in Section 4.3, minimizing (4.2) with a Gaussian variational distribution leads to a unimodal approximation of a multimodal posterior, which is problematic for highly ill-posed problems like inpainting. To circumvent this, we propose a *particle approximation* for defining a *multimodal* variational distribution. More precisely, we incorporate a repulsion force within the particles to encourage the exploration of multiple modes.

Particle interpretation of SDS.

To facilitate the presentation, throughout this section we consider the unconditional case of (4.2), i.e., without measurements:

$$q(\mathbf{z}_0) = \underset{q(\mathbf{z}_0)}{\operatorname{argmin}} \, \mathrm{KL}(q(\mathbf{z}_0)||p(\mathbf{z}_0)) \tag{4.3}$$

Following [41], we rewrite (4.3) in terms of the diffused trajectory as

$$q(\mathbf{z}_0) = \underset{q(\mathbf{z}_0)}{\operatorname{argmin}} \, \mathbb{E}_{t\sim\mathcal{U}[0,T]} \left[\omega(t)\mathrm{KL}(q(\mathbf{z}_t) \,\|\, p(\mathbf{z}_t))\right], \tag{4.4}$$

where $\omega(t)$ is a weighting function and $q(\mathbf{z}_t) = \int q(\mathbf{z}_t|\mathbf{z}_0)q(\mathbf{z}_0)\mathrm{d}\mathbf{z}_0$ depends on the diffused trajectory $q(\mathbf{z}_t|\mathbf{z}_0) \sim \mathcal{N}(\alpha_t\mathbf{z}_0, \sigma_t^2\mathbf{I})$ and the *variational approximation* $q(\mathbf{z}_0)$. The optimization in (4.4) corresponds to score distillation, which can be formulated as a Wasserstein gradient flow. At the particle level, the WGF is described by the following ODE

$$\mathrm{d}\mathbf{z}_{0,\tau}^{(i)} = \mathbb{E}_t \left[\omega(t) \left(\nabla_{\mathbf{z}_{t,\tau}^{(i)}} \log p\left(\mathbf{z}_{t,\tau}^{(i)}\right) - \nabla_{\mathbf{z}_{t,\tau}^{(i)}} \log q_\tau\left(\mathbf{z}_{t,\tau}^{(i)}\right)\right)\right] \mathrm{d}\tau, \tag{4.5}$$

where $p\left(\mathbf{z}_{t,\tau}^{(i)}\right)$ is the target distribution, $q_\tau\left(\mathbf{z}_{t,\tau}^{(i)}\right)$ is the marginal distribution of a generic particle i at time-step τ, and t is the noise level of the diffusion model. In a nutshell, the WGF of $\mathbf{z}_0$ in (4.5) is computed as an expectation over its diffused trajectory (noise levels t), involving the gradients of $\mathbf{z}_t$. This formulation sheds light on how the particles are propagated when optimizing (4.4). More precisely, it becomes evident that for an *initial Gaussian variational approximation at* $\tau = 0$*, the marginal for all* τ *is also Gaussian.* The dynamic in (4.5) yields a deterministic trajectory where *the mode of the Gaussian variational approximation will match one of the modes of* $p(\mathbf{z}_0)$*.* Consequently, assuming the same initial position, *all particles will converge to the same mode.* Although this can be mitigated ad hoc by changing the particles' initial positions, we seek a more principled method.

In this context, we can enhance diversity by considering 1) a multimodal (but most likely intractable) variational distribution or 2) interactive particle systems; in this work, we focus on the second one due to its tractability. When considering an interactive set of particles, the key design factor is the coupling term, which prevents the ensemble from collapsing to the same mode. In particular, we propose using a repulsion term.

Repulsive variational distribution.

Inspired by [117], we consider an ensemble of interacting particles coupled via a *repulsive force* that pushes particles away from collapsing to the same solution. In a nutshell, we introduce a repulsive force that yields the following modification of the gradient flow in (4.5)

$$\mathrm{d}\mathbf{z}_{0,\tau}^{(i)} = \mathbb{E}_t \left[\omega(t) \left(\nabla_{\mathbf{z}_{t,\tau}^{(i)}} \log p \left(\mathbf{z}_{t,\tau}^{(i)} \right) - \nabla_{\mathbf{z}_{t,\tau}^{(i)}} \log q_\tau \left(\mathbf{z}_{t,\tau}^{(i)} \right) - \nabla_{\mathbf{z}_{t,\tau}^{(i)}} \mathcal{R}(\mathbf{z}_{t,\tau}^{(1)}, \cdots, \mathbf{z}_{t,\tau}^{(N)}) \right) \right] \mathrm{d}\tau, \tag{4.6}$$

where N is the number of particles and $\mathcal{R}(\mathbf{z}_{t,\tau}^{(1)}, \cdots, \mathbf{z}_{t,\tau}^{(N)})$ is the coupling between particles such that its gradient is the repulsive force[†]. Notice that the marginal distribution in (4.6) at each time-step τ is given by (where Z is a normalizing constant)

$$q_\tau \left(\mathbf{z}_{t,\tau}^{(1)}, \cdots, \mathbf{z}_{t,\tau}^{(N)} \right) = \frac{1}{Z} \mathcal{R} \left(\mathbf{z}_{t,\tau}^{(1)}, \cdots, \mathbf{z}_{t,\tau}^{(N)} \right) \prod_{i=1}^{N} q_\tau \left(\mathbf{z}_{t,\tau}^{(i)} \right). \tag{4.7}$$

Throughout this work, we consider a pairwise kernel function k such that the repulsive force adopts the form $\nabla_{\mathbf{z}_{t,\tau}^{(i)}} \mathcal{R} \left(\mathbf{z}_{t,\tau}^{(1)}, \cdots, \mathbf{z}_{t,\tau}^{(N)} \right) = \nabla_{\mathbf{z}_{t,\tau}^{(i)}} \sum_{j=1}^{N} \log \left[k \left(\mathbf{z}_{t,\tau}^{(i)}, \mathbf{z}_{t,\tau}^{(j)} \right) \right]^\gamma$; see the numerical experiments for the particular instances of the kernel k. The repulsive force allows us to consider simple and flexible variational distributions that

[†]We consider here a repulsive force because we seek diversity. However, an attractive force can be considered within this same framework.

can discover multiple modes. Finally, notice that when $\gamma = 0$, we recover the i.i.d. (non-repulsive) case.

4.4.2 Tackling latent inversion: augmentation of the variational distribution

As discussed in Section 4.3, solving (4.2) directly in the latent space yields blurry solutions. To tackle this, we propose to solve an augmented version of this problem, allowing us to decouple the data and the latent space of the diffusion model. Formally, we introduce an auxiliary variable $\mathbf{x}_0$ defined in the data (pixel) space, which yields an augmented variational distribution $q(\mathbf{z}_0, \mathbf{x}_0|\mathbf{y})$ and an augmented posterior as

$$p(\mathbf{z}_0, \mathbf{x}_0|\mathbf{y}) \propto \exp\left(-\frac{1}{2\sigma_0^2}||\mathbf{y} - \mathcal{A}(\mathbf{x}_0)||^2 - \lambda g(\mathbf{z}_0) - \frac{1}{2\rho^2}||\mathbf{x}_0 - \mathcal{D}_\phi(\mathbf{z}_0)||^2\right), \tag{4.8}$$

where ρ controls the correlation between the variables $\mathbf{x}_0$ and $\mathbf{z}_0$, and $\exp(-\lambda g(\mathbf{z}_0))$ represents the prior distribution parameterized by the latent diffusion model. Notice that the definition in (4.8) implies that $\mathbf{x}_0|\mathbf{z}_0 \sim \mathcal{N}(\mathcal{D}_\phi(\mathbf{z}_0), \rho^2\mathbf{I})$, i.e., the conditional distribution of the data point ($\mathbf{x}_0$) is centered at the value of the decoder applied to the latent point ($\mathbf{z}_0$). However, this is not a delta but has some variance given by ρ^2. It can be shown that $p(\mathbf{x}_0|\mathbf{y}, \lambda, \rho^2)$ converges in total variation to the true posterior $p(\mathbf{x}_0|\mathbf{y}, \lambda)$ when $\rho \to 0$ [128, 129] (details can be found in Appendix B.1.1). We can reformulate the optimization problem in (4.2) as

$$q(\mathbf{z}_0, \mathbf{x}_0|\mathbf{y}) = \underset{q(\mathbf{z}_0, \mathbf{x}_0|\mathbf{y})}{\operatorname{argmin}} \operatorname{KL}(q(\mathbf{z}_0, \mathbf{x}_0|\mathbf{y}) \| p(\mathbf{z}_0, \mathbf{x}_0|\mathbf{y})). \tag{4.9}$$

When considering a diffusion model as data prior, our problem boils down to minimizing the variational lower bound, formalized in Proposition 3.

Proposition 3 Assuming we have access to a diffusion model $\nabla_{\mathbf{z}_t} \log p(\mathbf{z}_t)$ *for the prior on* $\mathbf{z}_0$*, then the KL minimization w.r.t* q *in* (4.9) *is equivalent to minimizing the variational bound, which can be done by solving the following optimization problem*

$$\min_{q(\mathbf{x}_0,\mathbf{z}_0|\mathbf{y})} \mathbb{E}_{q(\mathbf{z}_0|\mathbf{y})}[H(q(\mathbf{x}_0|\mathbf{z}_0,\mathbf{y}))] + \mathbb{E}_{q(\mathbf{x}_0,\mathbf{z}_0|\mathbf{y})}\left[\frac{1}{2\sigma_0^2}\|\mathbf{y}-\mathcal{A}(\mathbf{x}_0)\|^2\right] + \tag{4.10}$$
$$\mathbb{E}_{q(\mathbf{x}_0,\mathbf{z}_0|\mathbf{y})}\left[\frac{1}{2\rho^2}\|\mathbf{x}_0-\mathcal{D}_\phi(\mathbf{z}_0)\|^2\right] + \int_0^T \tilde{\omega}(t)\mathbb{E}_{q(\mathbf{z}_t|\mathbf{y})}\left[\|\nabla_{\mathbf{z}_t}\log q(\mathbf{z}_t \mid \mathbf{y}) - \nabla_{\mathbf{z}_t}\log p(\mathbf{z}_t)\|_2^2\right] dt.$$

The proof is in Appendix B.1.2. When $\rho \to 0$, then $\mathbf{x}_0 = \mathcal{D}_\phi(\mathbf{z}_0)$ and the augmented KL optimization boils down to the objective (4.2). To solve the problem in Proposition 3, we need to specify the variational distribution $q(\mathbf{x}_0, \mathbf{z}_0|\mathbf{y})$; we now incorporate our result from Section 4.4.1.

4.4.3 Repulsive latent score distillation for solving inverse problems

We now derive RLSD, which integrates the techniques from Sections 4.4.1 and 4.4.2. Specifically, we apply the repulsive variational distribution introduced in Section 4.4.1 to instantiate the augmented variational formulation detailed in Proposition 3. By employing the particle approximation defined in (4.7), we define a multimodal distribution that enables a better exploration of the posterior's search space, facilitating the discovery of multiple modes and addressing ($c1$). Notably, this variational approximation yields a tractable gradient, formalized in Proposition 4.

Proposition 4 When considering the variational distribution defined in (4.7)*, the KL minimization w.r.t* $q(\mathbf{x}_0, \mathbf{z}_0|\mathbf{y})$ *defined in Proposition 3 can be approximated with an ensemble of* N *particles and admits the following gradient*

$$\frac{1}{N}\sum_{i=1}^{N} \nabla_{\mathbf{u}^{(i)}}\left[\frac{1}{2\sigma_0^2}\|\mathbf{y}-\mathcal{A}(\mathbf{x}_0^{(i)})\|^2 + \frac{1}{2\rho^2}\|\mathbf{x}_0^{(i)}-\mathcal{D}_\phi(\mathbf{z}_0^{(i)})\|^2\right] + \nabla_{\mathbf{z}_{0,\tau}^{(i)}}\mathrm{reg}(\mathbf{z}_{t,\tau}^{(1)},\cdots,\mathbf{z}_{t,\tau}^{(N)}) \tag{4.11}$$

for $i = 1, \cdots, N$ *and where* $\mathbf{u}^{(i)} = [\mathbf{x}_0^{(i)}, \mathbf{z}_0^{(i)}]$. *The regularization term is given by*

$$\nabla_{\mathbf{z}_{0,\tau}^{(i)}} \mathrm{reg}(\mathbf{z}_{t,\tau}^{(1)}, \cdots, \mathbf{z}_{t,\tau}^{(N)}) = \mathbb{E}_{\boldsymbol{\epsilon},t}\left[\lambda_t \left(\boldsymbol{\epsilon}_\theta(\mathbf{z}_{t,\tau}^{(i)}, t) - \boldsymbol{\epsilon} - \nabla_{\mathbf{z}_{t,\tau}^{(i)}} \gamma \sigma_t \log \sum_{j=1}^{N} k\left(\mathbf{z}_{t,\tau}^{(i)}, \mathbf{z}_{t,\tau}^{(j)}\right)\right)\right]. \tag{4.12}$$

where $\lambda_t := \frac{T\alpha_t}{\sigma_t}\frac{d\omega(t)}{dt}$ *and* $\gamma \geq 0$.

The proof can be found in Appendix B.1.3. The gradient defined in Proposition 4 comprises three terms: a *measurement matching term*, an *error term* measuring the discrepancy between the variable in the pixel space and the decoded latent, and a *regularization term* that combines a *score-matching regularizer with a diversity-promoting component*. This repulsion term acts as a second regularizer, enhancing diversity during the sampling process. Our approach is not limited to any specific latent diffusion model.

Practical algorithm for sampling using RLSD.

The underlying optimization of the gradient update in Proposition 4, a particular case of Propositon 3, is highly non-convex (diffusion denoiser and the repulsion term) and challenging to solve. To alleviate this, we adopt a half quadratic splitting technique [119]. The algorithm is shown in Algorithm 3; we define sg[·] as the stopped-gradient operator to emphasize that the term inside it is not differentiated during the optimization step. We denote $\tilde{\rho} = \frac{\sigma_0^2}{\rho^2}$. For the weighting function λ_t ($\omega(t)$ is embedded) and timesteps, we follow the strategy introduced in [50], where $\lambda_t = \lambda(\sigma_t/\alpha_t)$, and the timesteps follow a decreasing order (from $t_{\max}$ to $t_{\min}$); we fix $t_{\max} = T$ and $t_{\min} = 0$. Regarding *computational burden*, our final algorithm only performs one backpropagation through the decoder in the $\mathbf{z}$-step and N backpropagations to compute the repulsive kernel (lines 7 and 9 in Alg. 3 are with respect to all particles).

The complexity of the repulsive force depends on the number of particles as well as the domain of the kernel. Notice that the amount of particles is a hyperparameter, allowing us to control the trade-off between diversity and speed. Importantly, in contrast to previous works, we do not backpropagate through the score network.

Algorithm 3 RLSD for solving inverse problems

Require: $\mathbf{y}, \mathcal{A}(\cdot), L, \boldsymbol{\epsilon}_\theta(\mathbf{z}_t, t), \mathcal{D}_\phi(\cdot), \{\lambda, \gamma, \tilde{\rho}, l_{r_x}, l_{r_z}\}$

Initialize $\{\mathbf{x}_{i,0}^0\}_{i=1}^N, \{\mathbf{z}_{i,0}^0\}_{i=1}^N$

for $\ell = 1$ to L **do**

  $t = T - \frac{\ell}{L}T$ and $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I})$

  $\lambda_t = \lambda(\sigma_t/\alpha_t)$

  $\mathbf{z}_{i,t}^\ell = \alpha_t \mathbf{z}_{i,0}^\ell + \sigma_t \boldsymbol{\epsilon}$

  $\mathcal{L}_z = \sum_{i=1}^N \|\mathbf{x}_i^\ell - \mathcal{D}_\phi(\mathbf{z}_{i,0}^\ell)\|^2 + \lambda_t \left(\text{sg}\left[\boldsymbol{\epsilon}_\theta(\mathbf{z}_{i,t}^\ell, t) - \boldsymbol{\epsilon} - \gamma \nabla_{\mathbf{z}_t^{(i)}} \sigma_t \log \sum_{j=1}^N k\left(\mathbf{z}_t^{(i)}, \mathbf{z}_t^{(j)}\right)\right]\right)^\top \mathbf{z}_{i,0}^\ell$

  $\mathbf{z}_0^\ell = \text{OptimizerStep}_{\mathbf{z}_0^\ell}(\mathcal{L}_z, l_{r_z})$

  $\mathcal{L}_x = \sum_{i=1}^N \|\mathbf{y} - \mathcal{A}(\mathbf{x}_i^l)\|^2 + \tilde{\rho}\|\mathbf{x}_i^l - \mathcal{D}_\phi(\mathbf{z}_{i,0}^l)\|^2$

  $\mathbf{x}_0^\ell = \text{OptimizerStep}_{\mathbf{x}_0^\ell}(\mathcal{L}_x, l_{r_x})$

end for

return $\{\mathbf{x}_{i,0}^L\}_{i=1}^N$

4.5 Numerical results

In this section, we compare RLSD against state-of-the-art (SoTA) methods for solving inverse problems using latent diffusion models. We consider 100 samples from the validation set of FFHQ [100] used in [2]. We compute PSNR [dB], LPIPS, and FID as metrics. Throughout the experiments, we seek to show the following:

- Our method generates more diverse solutions, in particular for tasks like inpainting and phase retrieval,
- When diversity is not relevant, our augmented formulation generates high-quality samples and outperforms baseline methods.

Sampling setup.

Unless we state otherwise, we consider 1000 steps (the full denoising trajectory) for all the cases. We denote by NonAug-RLSD our repulsive method without augmentation (see Alg. 7 in Appendix D.3), and by NonRepuls-RLSD our method with augmentation but without repulsion. We consider Adam [130] in the optimization steps (lines 7 and 9 in Alg. 3) and set the momentum pair $(0.9, 0.99)$. We randomly initialize variables $\mathbf{x}$ and $\mathbf{z}$ and generate a batch of $N = 4$ particles per noisy measurement. Regarding the pre-trained model, we consider Stable Diffusion v2.1, although other latent diffusion models can be used. As a diversity metric, we evaluate the pairwise diversity as the $1 -$ cosine similarity between the N particles. Lastly, for the kernel function, we consider a RBF: $k(\mathbf{z}_i, \mathbf{z}_j) = \exp(-\frac{||g_{\text{DINO}}(\mathbf{z}_i) - g_{\text{DINO}}(\mathbf{z}_j)||^2}{h_t})$, where $h_t = m_t^2 / \log N$, m_t is the median particle distance [109] and g_{DINO} is a pre-trained neural network [118]. Details about implementation are in Appendix D.3, and ablation analysis in Appendix B.2.8.

Baselines methods.

As we focus on methods that leverage large pre-trained models such as Stable diffusion, we compare with the recent PSLD [112] and Latent RED-Diff. For completeness, we also include a comparison with SoTA methods in the pixel-domain, namely DPS [2] and RED-diff [50]. Details about the implementation of each method are in Appendix D.3. Given that pixel-based diffusion solvers generate images at 256×256, we follow the strategy from [112] and downsample the results generated by our sampler, which have a 512×512 resolution, to do a fair comparison.

4.5.1 Inpainting

Inpainting, with its inherent ambiguity, provides a suitable benchmark to showcase two key aspects of RLSD: 1) high-quality reconstruction and 2) enhanced diversity achieved through the repulsion term. Additional linear inverse problems such as super resolution and deblurring are detailed in Appendix B.2. Specifically, we consider a box hiding half of the faces (see Fig. 4.2 and Appendix B.2.7). For the hyperparameters, we set $\lambda = 0.14$, $\tilde{\rho} = 0.075$, $l_{r_x} = 0.4$ and $l_{r_z} = 0.8$. Results in Table 4.2 show that *RLSD outperforms their baselines* in image quality (PSNR and FID); in particular, it outperforms PSLD, the other sampler at a resolution of 512×512. Moreover, it demonstrates that our method can trade-off diversity for quality by modifying the weight γ: while RLSD ($\gamma = 50$) achieves higher diversity than NonRepuls-RLSD, the latter achieves better performance. In addition, in Table B.11 we demonstrate that RLSD achieves the best performance when selecting the sample with best performance.

This highlights RLSD's ability to combine superior reconstruction with higher diversity, suggesting a *mixed strategy where some particles interact and others do not*. Indeed, this strategy (Hybrid-RLSD), where three particles interact and one particle is propagated independently of the ensemble, combines the best of both worlds, achieving the best balance between quality and diversity.

Diversity-quality trade off.

Fig. 4.2 showcases the diversity-quality trade-off. While PSLD generates four diverse samples of lower quality, Non-Repuls RLSD tends to fill the four images with very similar solutions. On the other hand, when considering RLSD ($\gamma = 50$), the results are more diverse while maintaining high quality: in images 1 and 3, the woman has

the left eye hidden, while none of the generated images with NonRepuls-RLSD show this. We defer to Appendix B.2.8 a more exhaustive analysis of diversity-quality trade off.

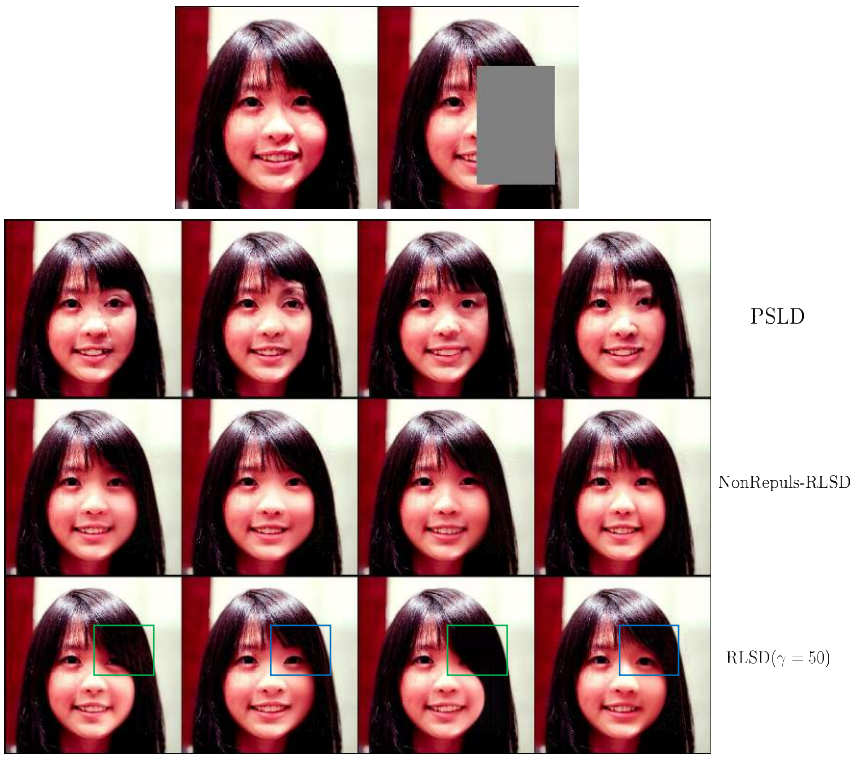


Figure 4.2 : Inpainting half a face (top to bottom): Ground truth, measurement, PSLD, NonRepuls-RLSD, and RLSD ($\gamma = 50$). Each method generates four samples from different initializations, with RLSD incorporating repulsion. Both RLSD and NonRepuls-RLSD outperform PSLD. RLSD produces diverse results (e.g., images 1 and 3 hide the left eye), whereas NonRepuls-RLSD samples appear similar.

4.5.2 Non-linear inverse problems

We consider in this case nonlinear inverse problems. Given that PSLD only works on linear inverse problems, we compare against latent DPS and latent RED-Diff.

Table 4.2 : Box inpainting (half face) with $\sigma_0 = 0.001$ - FFHQ 512. For evaluation, we downsample the estimated images of RLSD and PSLD to 256. The best method for each metric is bolded.

Sampler	PSNR [dB] ↑	LPIPS ↓	FID ↓	Diversity
PSLD	22.72	0.082	57.7	0.03
Latent RED-diff	23.5	0.15	92.59	0.009
ReSample	19.44	0.2	146.68	-
RED-Diff (Pixel)	23.1	**0.067**	<u>29.79</u>	0.004
DPS (Pixel)	23.4	0.14	78.88	**0.04**
NonAug-RLSD ($\gamma = 50$)	23.34	0.164	98.65	<u>0.035</u>
NonRepuls-RLSD	**24.98**	<u>0.079</u>	**29.18**	0.004
RLSD ($\gamma = 50$)	24.69	0.111	31.41	0.015
Hybrid-RLSD	<u>24.72</u>	0.096	30.48	0.018

Phase retrieval. We first consider phase retrieval, which deals with reconstructing the phase from only magnitude observations in the Fourier domain. Phase retrieval is known as a highly ill-posed problem, given that it is invariant to 180° rotation, which yields two equally probable modes. Thus, its posterior has multiple modes, which are discrete and isolated. We follow the strategy from DPS [2], where an oversampling of rate 2 is used. We consider 6 particles for the particle variational approximation, and 6 independent particles for the non-repulsive case. Furthermore, we set $\gamma = 30$, and we only consider repulsion between $t \in [0.4T, T]$. Results are shown in Table 4.3. First, we observe that for this experiment, the augmented variational approximation

(NonRepuls-RLSD and RLSD) entails a more unstable algorithm and thus does not converge to good modes.

Second, the results show that our proposed method NonAug-RLSD is more stable, and that the particle approximation effectively captures more modes. In particular, in Fig. 4.3 we show an example where the latent RED-diff generates 5 images that look similar, while NonAug-RLSD generates 6 samples that corresponds to different modes. This showcases that our methods indeed promote diversity, even for nonlinear inverse problems and at a resolution of 512×512.

Table 4.3 : Phase-retrieval with $\sigma_0 = 0.001$ on FFHQ 512. The best method for each metric is bolded.

Sampler	**PSNR [dB]** $\uparrow$	**LPIPS** $\downarrow$	**FID** $\downarrow$
Latent-DPS	14.98	0.618	291.68
Latent-RED-diff ($\gamma = 0$)	19.65	0.458	173.18
NonAug-RLSD ($\gamma = 30$)	**24.21**	**0.359**	**130.09**
NonRepuls-RLSD	18.33	0.495	223.14
RLSD ($\gamma = 30$)	20.43	0.449	207

High dynamic range (HDR). We try HDR, which performs the clipping function $\mathcal{A}(\mathbf{x}) = \text{clip}(2\mathbf{x}, -1, 1)$ on the normalized RGB pixels. HDR is known to be simpler than phase retrieval, and where diversity is not fundamental. Again, we consider Latent DPS as PSLD does not work for nonlinear inverse problems. The results are in Table 4.4, where RLSD outperforms all other baselines (when choosing the best sample from the group). This showcases that repulsion promotes better exploration and reaches better modes.



(a) NonAug-RLSD ($\gamma = 30$).



(b) Latent RED-Diff.

Figure 4.3 : Results for Phase Retrieval. Adding repulsion between particles allows to sample from different modes (top row).

Table 4.4 : HDR with $\sigma_0 = 0.001$ on FFHQ 512. The best method for each metric is bolded.

Sampler	**PSNR [dB]** $\uparrow$	**LPIPS** $\downarrow$	**FID** $\downarrow$
Latent-DPS	15.77	0.449	181.19
Latent-RED-diff	25.68	0.200	93.71
NonAug-RLSD ($\gamma = 30$)	24.84	0.210	94.15
NonRepuls-RLSD	27.72	0.087	37.57
RLSD ($\gamma = 50$)	**28.13**	**0.089**	**37.32**

4.6 Conclusion

We introduced **Repulsive Latent Score Distillation (RLSD)**, a variational posterior sampler for latent-diffusion priors that combines a particle-based multimodal variational family (attacking mode collapse via a repulsive kernel) with a distribution-

augmented objective (decoupling latent and pixel spaces to handle adversarial-decoder inversion). Chapter-specific limitations concentrate on the repulsion kernel: the DINO-based radial-basis choice is not obviously optimal at high noise levels, calling for *adaptive kernel learning*, and its interaction with the diffusion noise schedule — and with the forward operator — introduces hyperparameters that would benefit from a principled coupling scheme. Chapter 9 discusses broader directions on combining RLSD with other Part II priors.

Chapter 5

Sampling with shielded Langevin Monte Carlo using navigation potentials

Classical Langevin Monte Carlo samples from a target distribution using only the score of the potential, but it offers no mechanism to enforce hard constraints on the generated samples — an essential requirement in applications ranging from robotics and control to constrained image generation. In this chapter we introduce *shielded Langevin Monte Carlo*, a sampler for target distributions supported on *non-convex punctured domains*. Drawing on the Rimon–Koditschek navigation potential from robot motion planning, we augment the Langevin drift with a repulsive force around each forbidden region and modulate the diffusion coefficient by a spatially varying temperature that decays near obstacle boundaries. The resulting sampler provably keeps trajectories in the free space while retaining the target as its restricted stationary distribution. We illustrate the method on a two-dimensional Gaussian mixture with obstacles and on constrained massive MIMO symbol detection.

This chapter is adapted from [11].

5.1 Introduction

Sampling algorithms play a central role in modern generative modeling [7] and probabilistic inference [131]. In real-world deployments, however, the target distribution is often subject to constraints that arise from physical feasibility, system dynamics,

or user-specified safety requirements [132, 133] — forbidden regions in motion planning [134], chemical validity in molecular design [135], and content restrictions in image generation [136]. This chapter develops a Langevin-based sampler that provably respects such constraints, targeting a class the literature has not addressed: *distributions with non-convex support*, in which convex forbidden regions carve holes out of an otherwise convex ambient space (see Fig. 5.1).

Langevin Monte Carlo (LMC) [137, 138] — one of the workhorses of MCMC [139] and the primary sampler behind the other Part I contributions — does not natively enforce constraints on its output. A rich literature has extended it via projection [140], mirror and proximal maps [141–143], barriers [144, 145], penalties [146], and primal–dual methods [147]. Each of these is elegant on convex feasible sets, and each breaks down on non-convex ones: projection is generally intractable, mirror maps require a strictly convex Legendre dual on the feasible set, and barrier or proximal evaluations lose closed forms once the set is punctured. The gap is structural, not merely computational.

In this chapter we propose *shielded Langevin Monte Carlo* (shielded LMC), a sampler designed specifically for non-convex punctured domains. Drawing on control theory [148], we adapt the Rimon–Koditschek artificial potential [149] — originally designed for robot navigation around convex obstacles — into a modification of the Langevin diffusion. The potential contributes two ingredients: a spatially varying temperature that reduces stochasticity near obstacle boundaries, and an additional drift that produces a repulsive force around forbidden regions. Together these guarantee that trajectories stay in the feasible set while preserving the target as the restricted stationary distribution. We demonstrate the sampler on a two-dimensional Gaussian mixture with convex obstacles and on constrained massive-MIMO symbol

detection.

5.2 Background

We consider the problem of sampling from a punctured domain. Specifically, let $p(\mathbf{x})$ be a target (unconstrained) distribution supported on a space $\mathcal{X}$. Typically, $p(\mathbf{x})$ is known only up to a normalization factor, i.e., we know only $U(\mathbf{x})$ s.t. $p(\mathbf{x}) \propto e^{-U(\mathbf{x})}$. Our goal is to obtain samples from the *constrained* distribution

$$p_{\mathcal{F}}(\mathbf{x}) \propto p(\mathbf{x})\, \mathbb{I}\{\mathbf{x} \notin \mathcal{C}\}, \tag{5.1}$$

where $\mathcal{C} = \bigcup_{i=1}^{n} \mathcal{O}_i$ denotes a union of non-overlapping *obstacles* $\mathcal{O}_i$ we seek to avoid. Each obstacle $\mathcal{O}_i$ is assumed to be *closed* and *convex* and we associate with it a function $\beta_i(\mathbf{x})$ such that

$$\mathcal{O}_i = \{\mathbf{x} \in \mathcal{X} : \beta_i(\mathbf{x}) < 0\}. \tag{5.2}$$

In addition, we define the aggregated function $\beta(\mathbf{x}) = \prod_{i=1}^{n} \beta_i(\mathbf{x})$ that characterizes the overall constraint region $\mathcal{C}$. Hence, our goal is to sample from $p_{\mathcal{F}}$, whose support corresponds to the *punctured space*, or *free space* $\mathcal{F} = \mathcal{X} \setminus \mathcal{C}$. In Fig. 5.1, we provide an illustrative $\mathcal{F}$ and the corresponding obstacles $\mathcal{O}_i$. Note that the support $\mathcal{F}$ of $p_{\mathcal{F}}$ is non-convex despite the convexity of $\mathcal{C}$. Indeed, contrary to the traditional constrained sampling problem, we aim to sample from $p_{\mathcal{F}}$ in (5.1) (Fig. 5.1, left) as opposed to $p(\mathbf{x})\, \mathbb{I}\{\mathbf{x} \in \mathcal{C}\}$ (Fig. 5.1, right). In what follows, we introduce the two main ingredients that form the basis of our proposed constrained sampling method.

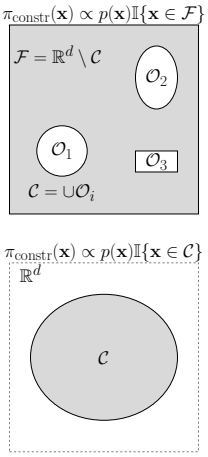


Figure 5.1 : Comparison of constraint types. Left: Our setting, where convex obstacles carve out forbidden regions, yielding a non-convex feasible space from which we must sample. Right: The classical case, where sampling is restricted to a convex feasible region within the ambient space.

5.2.1 Langevin diffusion

The Langevin diffusion is the stochastic process on $\mathbb{R}^d$ that satisfies the stochastic differential equation

$$\mathrm{d}\mathbf{x}_t = -\nabla U(\mathbf{x}_t) + \sqrt{2\tau}\mathrm{d}\mathbf{W}_t, \tag{5.3}$$

where $\mathbf{W}_t$ is a standard d-dimensional Brownian motion, $U \in \mathcal{C}^2(\mathbb{R}^d)$ is called the *potential*, and τ is a temperature parameter. The dynamic in (5.3) describes the path of a particle moving according to Newton's second law under a force $\nabla U(\mathbf{x}_t)$ while in contact with a thermal reservoir, i.e., collisions with other particles, represented by the random term (Brownian motion). Under mild conditions, it can be shown that the invariant distribution $\pi(\mathbf{x})$ of this continuous-time process is proportional to $\exp(-\tau^{-1}U(\mathbf{x}))$ [75]. In particular, given a target distribution $p(\mathbf{x})$ and fixing $\tau = 1$, we can obtain an MCMC sampler by letting $U(\mathbf{x}) = -\log p(\mathbf{x})$. The celebrated

MCMC algorithm [137, 139] known as unadjusted Langevin algorithm, is obtained as the Euler-Maruyama discretization of (5.3) and is described by the discrete-time equation

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \epsilon_k \nabla_{\mathbf{x}_k} \log p(\mathbf{x}_k) + \sqrt{2\tau\epsilon_k}\, \mathbf{w}_k, \tag{5.4}$$

where k is an iteration index, $\epsilon_k > 0$ is the discretization step size (which can be time-varying), and $\mathbf{w}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_d)$. Under some regularity conditions, the distribution of $\mathbf{x}_k$ converges to $\pi(\mathbf{x}) \propto p(\mathbf{x})^{1/\tau}$ as $\epsilon \to 0$ and $k \to \infty$. Although this result is asymptotic, non-asymptotic convergence results have been obtained under additional conditions on the target distribution [138].

5.2.2 Rimon-Koditschek (RK) navigation potential

The problem of navigation in the presence of obstacles is fundamental in robotics and control theory. The goal is to design trajectories that guide an agent (e.g., a robot or drone) from an initial configuration in state-space to a desired goal configuration, while avoiding forbidden regions of the configuration space (i.e., obstacles). In the absence of obstacles, this problem can be described in terms of a (convex) potential $U(\mathbf{x})$ whose minimum $U(\mathbf{x}^*) = 0$ describes the goal configuration. Then, its solution can be obtained by implementing the gradient flow $\dot{\mathbf{x}}_t = -\nabla_{\mathbf{x}_t} U(\mathbf{x}_t)$, where the minimum $\mathbf{x}^*$ corresponds to the goal configuration.

In the presence of obstacles $\mathcal{O}_i$, which is the setting of this chapter, the challenge is to design a potential that both attracts the agent to the goal and repels it from obstacles. A typical approach is to construct a navigation function which entails an *artificial potential* that achieves both aforementioned desiderata. It can be shown that implementing gradient flow along the navigation function indeed achieves the goal while avoiding obstacles as long as it satisfies the following properties [149].

Definition 5.1 (Navigation Function) A function $\phi : \mathcal{X} \to [0, 1]$ is a navigation function on the configuration space $\mathcal{X}$ if it satisfies:

1. *ϕ is smooth (at least C^2) on $\mathcal{X}$*
2. *ϕ has a unique minimum at the goal configuration*
3. *ϕ increases to 1 on the boundary of obstacles*
4. *The only critical point in the interior of the free space is $\mathbf{x}^*$.*

For instance, the RK artificial potential from [149] is a proper navigation function. Explicitly, it is given by

$$\phi_\alpha(\mathbf{x}) = \frac{U(\mathbf{x})}{\left(U(\mathbf{x})^\alpha + \beta(\mathbf{x})\right)^{1/\alpha}}, \tag{5.5}$$

where $\alpha > 0$ is a tuning parameter and β is defined as in (5.2). The RK potential combines a given potential $U(\mathbf{x})$, which encodes the configuration goal, with a repulsive potential $\beta(\mathbf{x})$, encoding the obstacle information. Therefore, the goal $\mathbf{x}^*$ can be reached without hitting obstacles $\mathcal{O}_i$ by implementing $\frac{\mathrm{d}\mathbf{x}_t}{\mathrm{d}t} = -\nabla_{\mathbf{x}_t}\phi_\alpha(\mathbf{x}_t)$, where the gradient is given by

$$\nabla_{\mathbf{x}}\phi_\alpha(\mathbf{x}) = \frac{\beta(\mathbf{x})\nabla_{\mathbf{x}}U(\mathbf{x}) - \frac{U(\mathbf{x})}{\alpha}\nabla_{\mathbf{x}}\beta(\mathbf{x})}{\left(U(\mathbf{x})^\alpha + \beta(\mathbf{x})\right)^{1+1/\alpha}}. \tag{5.6}$$

This gradient has an intuitive interpretation:

- Near the goal x^* (where $U(\mathbf{x}) \approx 0$), the original gradient $\nabla_{\mathbf{x}}U(\mathbf{x})$ dominates pulling the agent toward the goal;
- Near an obstacle (where $\beta(\mathbf{x}) \approx 0$), the repulsive term $-\nabla_{\mathbf{x}}\beta(\mathbf{x})$ dominates pushing the agent away from the obstacle;

- The tuning parameter α controls the balance between attraction and repulsion.

We will leverage this approach from navigation to build shielded LMC, a constrained sampler for spaces with convex obstacles.

5.3 Shielded LMC sampler

Recall that our goal is to sample from $p_{\mathcal{F}}(\mathbf{x})$ in (5.1) supported on the free space $\mathcal{F}$. To do this, we will use the Langevin dynamics introduced in Section 5.2.1. However, we need to incorporate the constraint $\mathbb{I}\{\mathbf{x} \notin \mathcal{C}\}$, which is non-differentiable and, therefore, cannot be directly plugged into the dynamic. To overcome this challenge, we leverage the artificial potential framework from Section 5.2.2 to construct a constrained variant of the Langevin dynamics. Hence, from now on, we consider $U(\mathbf{x}) = -\log p(\mathbf{x}) + C$ in (5.5), where we assume that $p(\mathbf{x})$ is log-concave (although in the numerical experiments we will consider cases where this does not hold) and C is such that $U(\mathbf{x}) > 0 \; \forall \; \mathbf{x}$.

A natural first attempt would be to replace the unconstrained distribution in (5.4) with the RK potential ϕ_α from (5.5). This yields the modified dynamics

$$\mathrm{d}\mathbf{x}_t = -\nabla\phi_\alpha(\mathbf{x}_t)\mathrm{d}t + \sqrt{2\tau}\mathrm{d}\mathbf{W}_t. \tag{5.7}$$

While this modification may appear reasonable, it is in fact incorrect due to the isotropic nature of the noise. Specifically, as the trajectory approaches an obstacle, the noise component remains isotropic and of constant variance, which can still push the particle into the forbidden region. Hence, a more careful treatment of the stochastic term is required to ensure the dynamic generates trajectories that lie in the free space $\mathcal{F}$. We tackle this next.

5.3.1 Building the sampler by using time-varying noise

In order to control the diffusion term, we modify not only the potential but also the variance of the noise. In particular, we propose the dynamics

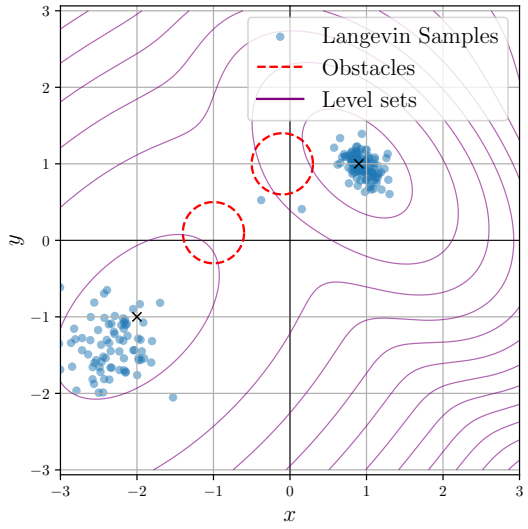
$$\mathrm{d}\mathbf{x}_t = -\nabla\phi_\alpha(\mathbf{x}_t)\mathrm{d}t + \sqrt{2\tau\beta^2(\mathbf{x}_t)}\mathrm{d}\mathbf{W}_t, \tag{5.8}$$

where $\beta(\mathbf{x}_t)$ is the aggregated function encoding the obstacles (see Section 5.2). We can interpret this new variance as a *spatially-aware* temperature function that modulates the diffusion term based on the proximity to obstacles. After discretization, we obtain the update rule

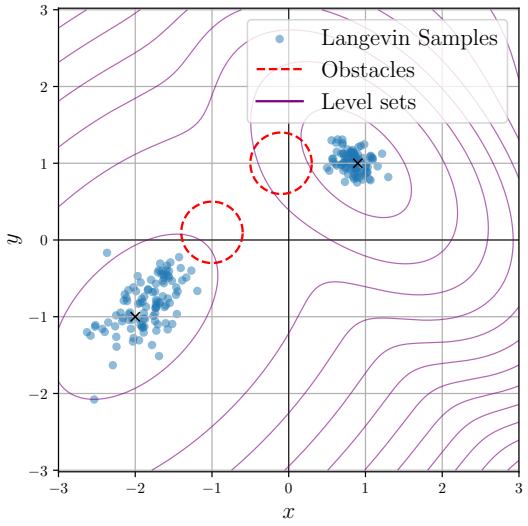
$$\begin{aligned}\mathbf{x}_{k+1} = \mathbf{x}_k + \eta_k \left[\beta(\mathbf{x}_k)\nabla_{\mathbf{x}_k}\log p(\mathbf{x}_k) - \frac{\log p(\mathbf{x}_k)\nabla_{\mathbf{x}_k}\beta(\mathbf{x}_k)}{\alpha}\right] \\ + \sqrt{2\eta_k\tau\beta^2(\mathbf{x}_k)}\mathbf{w}_k,\end{aligned} \tag{5.9}$$

where $\mathbf{w}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. Note that we incorporated the strictly positive denominator in (5.6) into the step size η_k. We refer to this modified dynamic as the *shielded LMC sampler*. The value of the parameter $\alpha > 0$ controls the behavior close to the optimum (i.e., the bias of the algorithm), trading off between the pure Langevin dynamics (5.4) and the repulsive strength of the obstacle. In particular, if α is too small, the repulsion acts over a large neighborhood around the obstacle and can slow down the convergence to the stationary distribution. On the other hand, if α is too large, the repulsion is very sharp (and only affects a narrow region near the obstacle), which can cause samples to get stuck at the boundaries of the obstacle. We illustrate this compromise in the numerical experiments (Section 5.4).

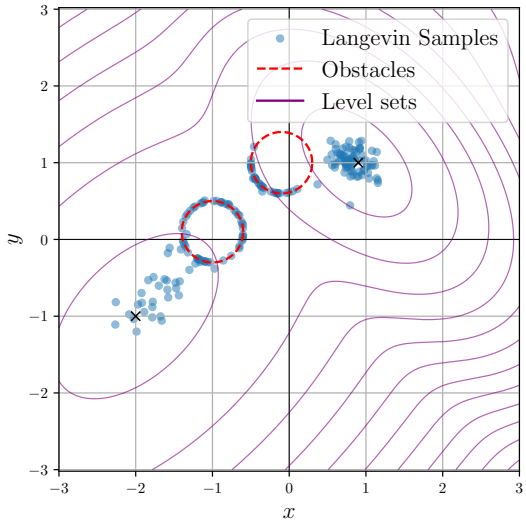
Once α is fixed, we can analyze the behavior of the shielded LMC sampler in the free space $\mathcal{F}$. To do this, we look at how the potential and spatially-aware temperature interact across different regions:



(a) $\alpha = 0.1$



(b) $\alpha = 1$



(c) $\alpha = 7$

Figure 5.2 : Impact of parameter α on shielded LMC with obstacles. Blue dots are Langevin samples after 50k iterations, red dashed lines are obstacles, and black crosses mark mode means; contours show the target distribution.(a) $\alpha = 0.1$: repulsion is too strong, pushing samples away from true modes. (b) $\alpha = 1$: good balance between target sampling and obstacle avoidance. (c) $\alpha = 7$: repulsion is too weak, causing samples to collapse onto obstacle boundaries where gradients vanish.

Algorithm 4 Shielded LMC

Require: $\nabla \log p(x), \beta(x), \tau$; Optional if available $\log p(x)$

1: Initialize $\mathbf{x}_0$ at random

2: **for** $k \leftarrow 1$ to K **do**

3: Compute $\beta(\mathbf{x}_k), \nabla_{\mathbf{x}_k}\beta(\mathbf{x}_k)$

4: Compute $\nabla_{\mathbf{x}_k} \log p(\mathbf{x}_k)$ (If available, compute $\log p(\mathbf{x}_k)$)

5: Draw $\mathbf{w}_k \sim \mathcal{N}(0, \mathbf{I})$

6: Compute

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \eta_k[\beta(\mathbf{x}_k)\nabla_{\mathbf{x}_k} \log p(\mathbf{x}_k) - \frac{\log p(\mathbf{x}_k)\nabla_{\mathbf{x}_k}\beta(\mathbf{x}_k)}{\alpha}] + \sqrt{2\eta_k\tau\beta^2(\mathbf{x}_k)}\mathbf{w}_k$$

7: **end for**

8: **return** $\mathbf{x}_K$

1. In the free space far from both the goal and the obstacle, the term $\beta(\mathbf{x})\nabla \log p(\mathbf{x})$ dominates. Since $\beta(\mathbf{x}) > 0$, the particle moves according to (5.4) with step size $\eta\beta(\mathbf{x})$. As a result, the sampler distributes the particle according to $p(\mathbf{x})$ while remaining largely unaffected by obstacles.

2. Near obstacle boundaries, where $\beta(\mathbf{x}) \to 0^+$, the attractive term $\beta(\mathbf{x})\nabla \log p(\mathbf{x})$ disappears and the gradient becomes dominated by the repulsive component $-\log p(\mathbf{x})\nabla\beta(\mathbf{x})$. This induces a strong repulsion that prevents samples from crossing into forbidden regions.

Overall, these two regimes allow the sampler to adapt to different regions of the space, balancing broad exploration when far from obstacles with stronger constraint avoidance as it approaches them. The final algorithm is shown in Alg. 4.

Dependency on $\log p(\mathbf{x})$. An alternative interpretation of the update in (5.9) is as a classical Langevin dynamic like (5.4) with an effective step size $\eta_k\beta(\mathbf{x}_k)$, augmented by an additional drift term $-\frac{\log p(\mathbf{x}_k)\,\nabla_{\mathbf{x}_k}\beta(\mathbf{x}_k)}{\alpha}$. This extra term explicitly depends

on the target density $p(\mathbf{x})$, which is generally intractable and, even when known, is available only up to a normalization constant. In such cases, a practical workaround is to approximate $\frac{\log p(\mathbf{x})}{\alpha}$ by a global constant $\bar{\alpha}$ that is independent of $\mathbf{x}$, thereby avoiding the need to evaluate $\log p(\mathbf{x})$ during sampling.

5.4 Numerical results

We describe two experiments that demonstrate the advantages of our proposed sampler and how it compares with the unconstrained case. First, in Section 5.4.1 we analyze its behavior when sampling from 2D Gaussian Mixture Model with circular obstacles. Then, in Section 5.4.2, we show how adding obstacles and thereby constraining the exploration space, can accelerate sampling in a MIMO application.

5.4.1 2D Gaussian mixture model with obstacles

Setup. We consider first a sampling task in $\mathcal{X} = \mathbb{R}^2$ where the target distribution is a mixture of two Gaussian components $p(\mathbf{x}) = \frac{1}{2}\mathcal{N}(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1) + \frac{1}{2}\mathcal{N}(\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2)$ with parameters

$$\boldsymbol{\mu}_1 = \begin{bmatrix} -2 \\ -1 \end{bmatrix}, \quad \boldsymbol{\mu}_2 = \begin{bmatrix} 0.9 \\ 1 \end{bmatrix},$$

$$\boldsymbol{\Sigma}_1 = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}, \quad \boldsymbol{\Sigma}_2 = \begin{bmatrix} 0.5 & -0.25 \\ -0.25 & 0.5 \end{bmatrix}.$$

In addition, we incorporate two circular obstacles $\mathcal{O}_i = \{\mathbf{x} \in \mathbb{R}^2 : (\mathbf{x} - \mathbf{c}_i)^2 - 0.4^2\}$ with $\mathbf{c}_1 = [-1, 1]^\top$ and $\mathbf{c}_2 = [-1, 0.1]^\top$; see Fig. 5.2 for a representation of this setting.

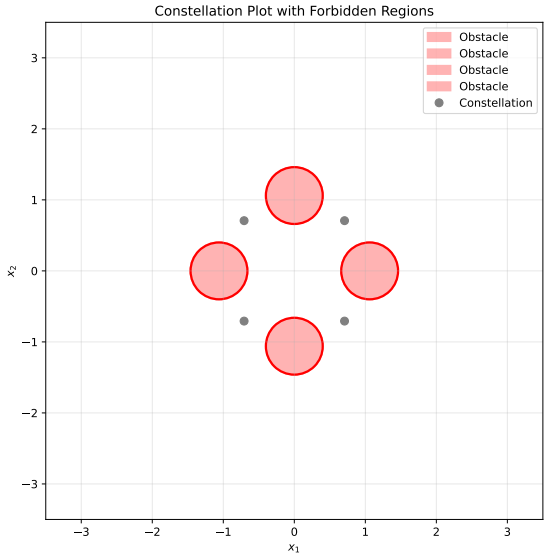
This setup provides a simple yet illustrative benchmark to analyze how the sampler handles multi-modal sampling and obstacle avoidance.

Analysis. Starting from random positions $\mathbf{x}_0 \sim \mathcal{N}(0, \mathbf{I})$, we run Algorithm 4 for 50×10^4 iterations with $\alpha = \{0.1, 1, 7\}$ and $\tau = 0.2$. The results are shown in Fig. 5.2. As discussed in Section 5.3.1, the proposed sampler has different behaviors for different values of α. In particular, $\alpha = 1$ corresponds to the best value, achieving a good trade-off between sampling from the target distribution while avoiding the obstacles. For $\alpha = 0.1$, it is clear that the repulsive field is too strong, pushing the particles away even from the modes of the target distribution. On the other hand, for $\alpha = 7$ the particles collapse to the boundary of the obstacles since the second term in (5.9) is negligible. In contrast, the first term becomes zero at the boundaries of the obstacles.

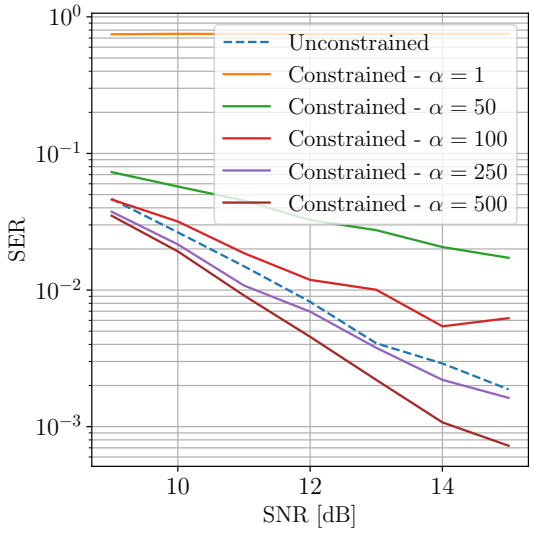
5.4.2 MIMO detection: speeding-up the sampling

Chapter 3 accelerated the annealed-Langevin MIMO symbol detector of [73] by augmenting the overdamped recursion with momentum and a preconditioner. Shielded LMC offers a qualitatively different route to the same speed-up: rather than modifying the sampler's inertia, it places obstacles around the QPSK constellation points and lets the shielded recursion (5.9) push the chain away from low-probability regions of the discrete posterior, encouraging faster traversal. We reuse the full MIMO setup of Section 3.5.1 — linear model (3.1), QPSK constellation $\mathcal{S} = \{\pm 0.74 \pm j0.84\}$, $N_u = N_r = 8$ users/antennas, Rayleigh fading channel $[\mathbf{H}]_{ij} \sim \mathcal{CN}(0, 1/N_r)$, and SER-vs-SNR evaluation — and only vary the sampler. As in Chapter 3, we cut the annealing budget from $L = 20$ [73] to $L = 5$ and retain the best of 10 posterior candidates per SER estimate.

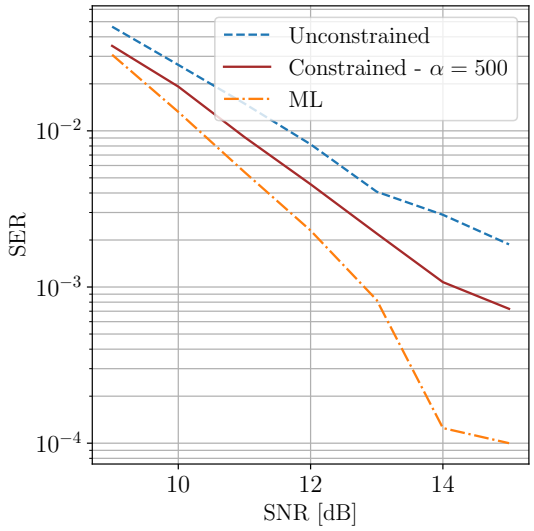
Fig. 5.3a illustrates the obstacle-augmented constellation. We first ablate the obstacle-strength parameter α of (5.9) and confirm that obstacles improve exploration; see Fig. 5.3b. In particular, $\alpha > 250$ yields a clear improvement over the unconstrained case. Fixing $\alpha = 500$, we compare against the optimal ML detector implemented via Gurobi [90] in Fig. 5.3c: a gap to optimal ML remains at high SNR, but the constrained sampler significantly outperforms the unconstrained Langevin baseline at the same reduced iteration budget.



(a)



(b)



(c)

Figure 5.3 : Performance analysis of shielded LMC for MIMO symbol detection, considering SER as a function of SNR in a Rayleigh fading channel model. (a) QPSK constellation with obstacles. (b) Ablation of the obstacle-strength parameter α. (c) Comparison against the unconstrained Langevin sampler and the optimal ML detector.

5.5 Conclusion

We introduced *shielded LMC*, a Langevin sampler for non-convex punctured domains that borrows the Rimon–Koditschek navigation potential to combine a spatially varying temperature with a repulsive drift, provably keeping trajectories in the feasible set while preserving the target as the restricted stationary distribution. Two chapter-specific limitations remain. First, the current construction assumes *convex* obstacles; extending the navigation potential to arbitrarily shaped forbidden regions is an obvious next step. Second, our experiments are restricted to a 2D toy mixture and a constrained MIMO detector: integrating the shielded drift into a full diffusion-based image sampler is the natural bridge to the real-world safety-constrained generation applications discussed in Chapter 9, which also outlines the open question of non-asymptotic convergence guarantees for the shielded dynamics.

Part II

Training priors for improved inference

This part of the thesis keeps the sampling recipe simple but *modifies the generative model itself* so that the resulting prior is better matched to the target inverse problem. Section 2.7 identified four design knobs of a diffusion-based solver; the sampler-side knob was pulled in Part I, and the three remaining knobs are exactly the training-time levers pulled here, one per chapter:

- Chapter 6 modifies the *source distribution*. A generic flow model transports a Gaussian source into the data distribution; we replace the Gaussian with a task-informed source that already carries structural information about the graph inverse problem at hand, and train the flow to complete the reconstruction.

- Chapter 7 modifies the *training target: from score alone to score-plus-density*. Standard denoising score matching estimates the gradient of the log-density but leaves the normalization constant inaccessible. We extend it to matrix-valued anisotropic noise via a dual score-matching objective; the companion covariance score it enables makes the density normalization tractable and unlocks blind inverse problems, uncertainty quantification, and posterior-based comparison across solvers.

- Chapter 8 modifies the *training target for velocity: from instantaneous to average*. Standard flow matching learns the instantaneous velocity of a stochastic interpolant, which must be integrated over hundreds to thousands of small steps. We train a flow map that outputs the *average velocity* across arbitrary time pairs and use its average/two-time *denoiser* form as a plug-and-play denoiser for inverse problems; the lookahead of the trained flow map is a principled, single-model knob for traversing the distortion–perception plane.

The unifying property of these three approaches is that they act at training time,

not at inference time. Two of the three chapters — Chapter 6 and Chapter 8 — share the underlying flow-matching machinery but pull two orthogonal levers within it; before diving into either, we collect the shared flow-matching background below. The cross-cutting theme is revisited in the closing discussion of Chapter 9.

Background: flow matching

The chapters in this part build on the diffusion machinery of Chapter 2 but state their contributions in the *flow-matching* formalism [8, 33, 34]. Flow matching and continuous-time diffusion are two faces of the same object — flow matching corresponds to the probability-flow ODE of Section 2.2 under a different noise schedule [30] — but the flow-matching parameterization is more natural whenever the source distribution or the transport structure is itself the design knob. Chapters 6 and 8 both fall in that regime. We collect here the notation and objects they share.

Stochastic interpolant. A flow-matching model specifies two distributions — a *source* p_0 (typically $\mathcal{N}(\mathbf{0}, \mathbf{I})$, but not required, see Chapter 6) and a *target* p_1 (the data distribution) — and interpolates between them along the process

$$\mathbf{x}_t \;=\; \alpha_t\, \mathbf{x}_0 \;+\; \beta_t\, \mathbf{x}_1, \qquad t \in [0, 1], \tag{5.10}$$

where $\mathbf{x}_0 \sim p_0$, $\mathbf{x}_1 \sim p_1$, and (α_t, β_t) are continuously differentiable coefficients satisfying $\alpha_0 = \beta_1 = 1$ and $\alpha_1 = \beta_0 = 0$. The pair $(\mathbf{x}_0, \mathbf{x}_1)$ need not be independent: a joint coupling $\pi(\mathbf{x}_0, \mathbf{x}_1)$ is admissible, and this generality is exactly what Chapter 6 exploits to encode a task-informed source. The *rectified-flow* special case [33] takes the linear schedule $\alpha_t = 1 - t$, $\beta_t = t$ and is the choice used throughout Chapters 6 and 8.

Marginal velocity field. The marginal velocity of the interpolant is the conditional expectation

$$\mathbf{v}(\mathbf{x}, t) \;:=\; \mathbb{E}\big[\dot{\alpha}_t\, \mathbf{x}_0 + \dot{\beta}_t\, \mathbf{x}_1 \;\big|\; \mathbf{x}_t = \mathbf{x}\big], \tag{5.11}$$

and the deterministic ODE

$$\dot{\mathbf{x}}_t \;=\; \mathbf{v}(\mathbf{x}_t, t), \qquad \mathbf{x}_0 \sim p_0, \tag{5.12}$$

transports samples from p_0 at $t = 0$ to samples from p_1 at $t = 1$. When p_0 is Gaussian and (α_t, β_t) trace out the VP or VE noising schedules of Section 2.2, (5.12) coincides with the probability-flow ODE (2.11) of the corresponding diffusion, and $\mathbf{v}$ is a linear function of the marginal score [30].

Training objective. The velocity field is learned by fitting a neural network $\mathbf{v}_\theta(\mathbf{x}, t)$ to the pathwise regression target $\dot{\alpha}_t \mathbf{x}_0 + \dot{\beta}_t \mathbf{x}_1$:

$$\mathcal{L}_{\mathrm{FM}}(\theta) \;=\; \mathbb{E}_{t\sim\mathcal{U}[0,1],\,(\mathbf{x}_0,\mathbf{x}_1)\sim\pi}\Big[\big\|\mathbf{v}_\theta(\mathbf{x}_t, t) - \big(\dot{\alpha}_t \mathbf{x}_0 + \dot{\beta}_t \mathbf{x}_1\big)\big\|_2^2\Big]. \tag{5.13}$$

The minimizer of (5.13) coincides with the marginal velocity (5.11) [34]. This is the flow-matching analogue of denoising score matching (Section 2.4): the pathwise target is tractable even when the marginal $\mathbf{v}$ is not, and DSM (2.13) can be recovered from (5.13) by a change of variables when $p_0 = \mathcal{N}(\mathbf{0}, \mathbf{I})$.

Instantaneous denoiser. Tweedie's formula (2.15) translates directly across: for the linear (rectified-flow) interpolant, the MMSE denoiser at time t is

$$D_t(\mathbf{x}) \;:=\; \mathbb{E}[\mathbf{x}_1 \mid \mathbf{x}_t = \mathbf{x}] \;=\; \mathbf{x} + (1 - t)\, \mathbf{v}(\mathbf{x}, t), \tag{5.14}$$

so a trained velocity network is equivalently a trained denoiser at every noise level $t \in [0, 1]$. This is the object plug-and-play methods (Section 2.6) invoke as their proximal step, and it is the starting point of the *average denoiser* developed in Chapter 8.

With these ingredients in place, Chapter 6 pulls the source-distribution knob by replacing p_0 with a task-informed distribution built from a coarse graph predictor, and Chapter 8 pulls the training-target-for-velocity knob by replacing the instantaneous velocity (5.11) with an average velocity across time pairs. Chapter 7 sits on the diffusion side of the equivalence, and reuses the DSM machinery of Section 2.4 directly.

Chapter 6

Prior-informed flow matching for graph reconstruction

Standard flow-matching generative models transport a generic Gaussian source distribution into the data distribution and know nothing about the downstream task. For structured inverse problems on graphs — link prediction and blind graph reconstruction from partial observations — this task-agnostic source wastes capacity on modeling structure that a coarse task-specific predictor can already provide. In this chapter we introduce *prior-informed flow matching* (PIFM), a training-time modification of continuous flow matching in which the Gaussian source is replaced by a task-informed source distribution that carries structural priors about the target graph (node embeddings, GNN-based link predictors, variational graph autoencoders, or graph transformers). The flow then only has to complete the reconstruction from the informed source, dramatically reducing the effective ill-posedness of the problem. Motivated by a permutation-equivariant version of the distortion–perception trade-off, PIFM admits two distortion instantiations — squared error on the residual velocity, and cross-entropy on the endpoint — and a closure guarantee: the reconstruction map is permutation-equivariant whenever the prior and the velocity field are. On link prediction and two blind reconstruction settings across several benchmarks, PIFM outperforms both task-agnostic diffusion priors and hand-designed graph baselines.

This chapter is adapted from [12].

6.1 Introduction

Graph generative models have made rapid progress in recent years, with diffusion-based [150–152] and flow-based [153, 154] approaches now setting the state of the art for unconditional and property-controlled generation in domains such as drug design [155] and social-network modeling [156]. Their use for *inverse problems on graphs* — reconstructing a graph from partially observed edges — remains largely unexplored, and is the focus of this chapter. As set out in Section 2.7, the design lever we pull is the *source distribution* of a flow-matching model: replacing the generic Gaussian source with one that already carries structural information about the target graph.

Conditional graph reconstruction sits at the intersection of two literatures whose strengths do not currently combine. *Link-prediction* methods — classical embedding techniques such as node2vec [157], inductive GNNs such as GraphSAGE [158], variational graph autoencoders (VGAE [159]), neural common-neighbour predictors (NCNC [160]), and graph transformers (LPFormer [161]) — produce expressive edge-wise scores but generate each edge prediction essentially independently, with no mechanism to couple predictions across the graph. *Diffusion or flow-based* graph generators [16, 152, 162, 163] produce globally consistent samples but are primarily designed for constrained generation, not fidelity-driven reconstruction from a specific observation. The gap is structural: no existing method combines an informative local prior with a global coupling over the full edge set trained for reconstruction.

We introduce *Prior-Informed Flow Matching* (PIFM), a flow-based estimator for high-fidelity graph reconstruction (see Fig. 6.1). We frame the problem through a permutation-equivariant version of the distortion–perception trade-off of Section 2.1.2 [1, 26, 164], which decomposes an optimal estimator into two stages: a marginal edge

prediction (approximating the posterior mean), followed by an optimal-transport-like refinement from the marginal estimate to the ground-truth distribution. PIFM realizes both stages. For the marginal stage, we use any of the link predictors listed above to produce per-edge probabilities. For the transport stage, we train a rectified flow [33, 34] using permutation-equivariant architectures that guarantee a permutation-equivariant reconstruction map (Theorem 1). Two distortion instantiations are supported: a squared-error velocity loss that recovers standard rectified flow matching, and a class-weighted cross-entropy endpoint loss more suited to sparse graphs. On benchmarks spanning dense and sparse graphs, PIFM outperforms both local link-prediction methods and task-agnostic generative baselines on link prediction and two blind variants (expansion and denoising).

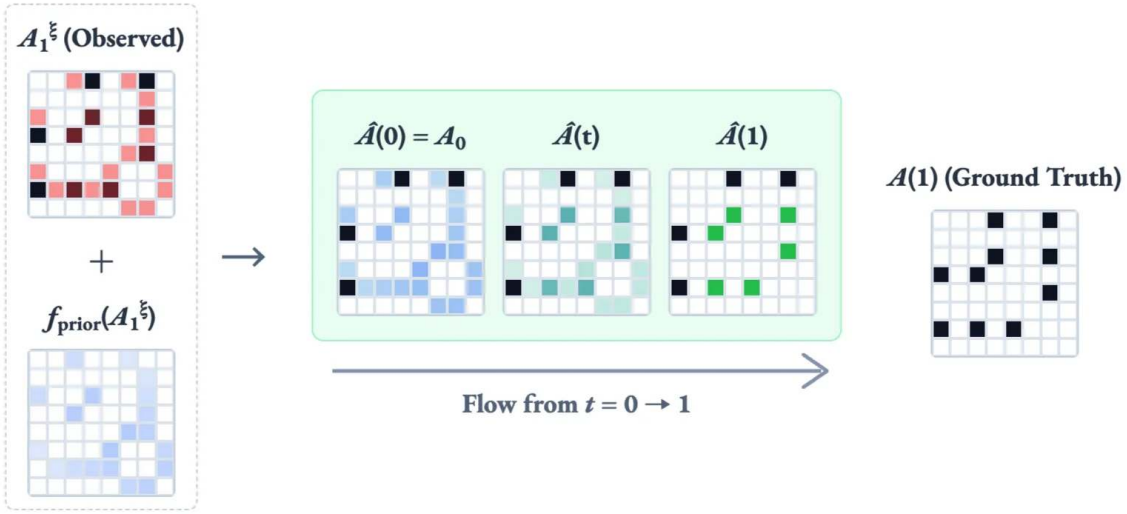


Figure 6.1 : Overview of the **Prior-Informed Flow Matching (PIFM)** graph reconstruction framework. Starting from a partially observed adjacency matrix $\mathbf{A}^{\mathcal{O}} = \xi \odot \mathbf{A}$, where ξ denotes a mask, we form an initialization $\mathbf{A}_0$ by combining the observed entries with prior predictions $f_{\text{prior}}(\mathbf{A}^{\mathcal{O}})$ obtained with an element-wise predictor. In dark red we denote the true edges that are masked, while in light red those masked positions where there is no edge between nodes. A rectified flow then interpolates linearly from $\mathbf{A}_0$ to the ground-truth graph $\mathbf{A}_1 = \mathbf{A}$, learning global structural information from a coupling of all the edges. The intermediate states $\mathbf{A}_t$ improve on the prior-informed initialization, enabling recovery of the missing edges.

6.2 Related work

Flow/diffusion models on graphs. Diffusion and flow-based graph generative models have shown impressive performance in recent years. Early models — namely EDP-GNN [150] and GDSS [151] — employ score-based *continuous* diffusion over a relaxation of the graph structure. However, given that graphs are inherently discrete, subsequent work has explored discrete diffusion processes [165]. Models like

DiGress [152] demonstrated the effectiveness of this approach, which has been further advanced by discrete flow-based models like DeFoG [153] and variational approaches like CatFlow [154]. A common point of these models is their reliance on a simple source distribution, such as Gaussian (continuous) or uniform (discrete) noise. While effective for unconstrained generation, recent work on image-based inverse problems demonstrates the advantages of learning a data-dependent flow using a prior-informed source distribution [34, 164, 166]. Concurrently to our work, Flowette [167] uses domain-specific graphette priors and flow matching for graph generation. In contrast, PIFM is conditioned on a partially observed graph and mask, pairing each prior-informed source with the corresponding clean graph for reconstruction. While both methods exploit informative structural priors, they address different tasks and differ in conditioning, coupling, architecture, and evaluation, making them complementary rather than competing approaches.

Graph topology inference via flow/diffusion-based solvers. Graph topology inference — the task of recovering hidden edges from a partially observed graph — is a long-standing inverse problem [61, 62]. Several methods adapt diffusion for constrained graph generation, which is related to but distinct from topology inference. DiGress [152] introduced an inpainting mechanism, inspired by Repaint [168], to generate graph structures consistent with a partial observation. Similarly, [162] use a related mechanism for completing partially observed graphs. PRODIGY [163] enforces hard constraints by projecting the graph estimate onto a feasible set at each sampling step. More recently, GGDiff [16] incorporates a guidance mechanism as a flexible alternative to inpainting. However, all these methods are designed for *constrained generation* (e.g., molecule generation with a given scaffold) rather than

recovering masked edges from a partially observed graph. Hence, to the best of our knowledge, designing a diffusion-based model explicitly for graph topology inference remains an open problem.

Link prediction and graph autoencoders. Graph reconstruction from partial observations is closely related to link prediction, which aims to infer whether unobserved edges should exist in a partially observed graph [169–171]. Classical approaches rely on topology-based heuristics, while embedding-based methods such as DeepWalk [172] and node2vec [157] learn node representations from random walks and score candidate edges from pairwise embedding features. Graph neural network methods further improve link prediction by learning node representations through message passing: GraphSAGE learns inductive neighbourhood aggregation functions that can generalize across nodes or graphs [158], while variational graph autoencoders (VGAE) encode the observed graph into latent node embeddings and decode pairwise edge probabilities [159]. Other methods model pairwise or path-based structure more directly, such as NBFNet, which formulates link prediction as neural path aggregation through a Bellman–Ford-style recurrence [173]. LPFormer [161] combines structural encodings with attention for link prediction. All of these methods return edge-wise scores; the flow refinement developed here adds a global coupling on top of any of them.

6.3 Background

We represent an undirected graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, where $\mathcal{V}$ denotes the nodes and $\mathcal{E}$ the edges, by its binary symmetric adjacency matrix $\mathbf{A} \in \mathbb{R}^{N \times N}$. Table 6.1 summarizes the chapter-specific notation.

Table 6.1 : Notation used in PIFM.

Symbol	Meaning
$\mathbf{A}_1$	Clean ground-truth adjacency matrix
$\mathbf{A}^{\mathcal{O}}$	Observed partial adjacency matrix
ξ	Binary observation mask
$\hat{\mathbf{A}}^*$	Prior estimate or approximate MMSE estimate
$\mathbf{A}_0$	Prior-informed source sample for flow matching
$\mathbf{A}_t$	Intermediate state along the flow path
$\widehat{\mathbf{A}}$	Final reconstructed adjacency matrix

6.3.1 Continuous flow matching for graph generation

The general flow-matching machinery, including source p_0, target p_1, stochastic interpolant (5.10), marginal velocity field (5.11), generative ODE (5.12), and the pathwise regression loss (5.13), was introduced in the Part II and Section II. We use it here specialized to graph adjacency matrices: source and target both live in $\mathbb{R}^{N\times N}$, and $\mathbf{A}_0 \sim p_0$, $\mathbf{A}_1 \sim p_1$ play the roles of $\mathbf{x}_0, \mathbf{x}_1$ in Section II. Two specialization choices matter for the rest of this chapter.

Rectified-flow schedule. We adopt the linear interpolant [33] $\alpha_t = 1 - t$, $\beta_t = t$ throughout, i.e., $\mathbf{A}_t = (1-t)\mathbf{A}_0 + t\mathbf{A}_1$, so the network target $\dot{\alpha}_t\mathbf{A}_0 + \dot{\beta}_t\mathbf{A}_1 = \mathbf{A}_1 - \mathbf{A}_0$ is the straight-line displacement between source and target. As shown in [174], the associated velocity field approximates the optimal-transport vector field when $\pi(\mathbf{A}_0, \mathbf{A}_1)$ closely resembles the optimal coupling of its marginals. The specialized loss is

$$\mathcal{L}_{\mathrm{FM}} = \mathbb{E}_{t,\mathbf{A}_0,\mathbf{A}_1}\left[\left\|v_\theta(\mathbf{A}_t, t) - (\mathbf{A}_1 - \mathbf{A}_0)\right\|_F^2\right]. \tag{6.1}$$

Non-independent coupling. The training objective does not require $\mathbf{A}_0$ and $\mathbf{A}_1$

to be independent: any joint distribution $\pi(\mathbf{A}_0, \mathbf{A}_1)$ is admissible and yields a valid flow between its marginals. This has been exploited to solve inverse problems on images [164,166,175], and is exactly the lever we pull below — our task-informed source p_0 is constructed from a coarse graph predictor applied to the partial observation, inducing a data-dependent coupling with $\mathbf{A}_1$.

6.3.2 Distortion–perception formulation for conditional graph reconstruction

We specialize the distortion–perception (DP) trade-off of Section 2.1.2 to graph reconstruction. Given a partially observed graph $\mathbf{A}^{\mathcal{O}}$, we seek an estimator $\hat{\mathbf{A}}$ of the ground-truth adjacency matrix $\mathbf{A}$; assessment happens along two complementary axes: (i) the average closeness of the reconstruction to the ground truth (a distortion metric), and (ii) the similarity in distribution between reconstructed and real graphs (a perceptual index). Following (2.5), the DP function is

$$D(P) = \min_{p(\hat{\mathbf{A}}|\mathbf{A}^{\mathcal{O}})} \Big\{ \mathbb{E}_{p(\mathbf{A},\hat{\mathbf{A}})} \Big[\Delta(\mathbf{A}, \hat{\mathbf{A}}) \Big] \ : \ d(p_{\mathbf{A}}, p_{\hat{\mathbf{A}}}) \leq P \Big\}, \tag{6.2}$$

where $\Delta(\cdot,\cdot)$ is a distortion metric and $d(\cdot,\cdot)$ a divergence between the distribution of clean graphs $p_{\mathbf{A}}$ and the distribution of estimates $p_{\hat{\mathbf{A}}}$. Two features distinguish the graph setting from the image setting reviewed in Section 2.1.2: the estimator must respect permutation invariance (enforced by permutation-equivariant architectures below), and the extremes of the frontier admit graph-domain interpretations that differ from images. We consider two distortion instantiations.

MSE distortion. For $\Delta(\mathbf{A}, \hat{\mathbf{A}}) = \|\mathbf{A} - \hat{\mathbf{A}}\|_F^2$, the two extremes of the frontier are well-understood [1]. When $P = \infty$, the optimal estimator is the posterior mean $\hat{\mathbf{A}}^\star = \mathbb{E}[\mathbf{A} \mid \mathbf{A}^{\mathcal{O}}]$; it minimizes distortion but produces *fractional* adjacency matrices that

may violate discreteness and structural constraints. When $P = 0$, the reconstructed distribution matches the ground-truth distribution ($p_{\hat{\mathbf{A}}} = p_{\mathbf{A}}$), and the corresponding estimator can be characterized through the optimal-transport problem

$$p^{\star}_{\hat{\mathbf{A}},\hat{\mathbf{A}}^{\star}} = \underset{p \in \Pi(p_{\mathbf{A}}, p_{\hat{\mathbf{A}}^{\star}})}{\operatorname{argmin}} \mathbb{E}\left[\|\hat{\mathbf{A}} - \hat{\mathbf{A}}^{\star}\|_F^2\right], \tag{6.3}$$

where $\Pi(p_{\mathbf{A}}, p_{\hat{\mathbf{A}}^{\star}})$ denotes the set of couplings with marginals $p_{\mathbf{A}}$ and $p_{\hat{\mathbf{A}}^{\star}}$. Combining (2.7) with $\Delta =$ Frobenius yields the linear interpolation $\hat{\mathbf{A}}_P = (1 - P/P^{\star})\hat{\mathbf{A}}_0 + (P/P^{\star})\hat{\mathbf{A}}^{\star}$ between a perfectly perceptual estimator $\hat{\mathbf{A}}_0$ and the MMSE estimator, so obtaining any point on the DP curve amounts to solving an optimal-transport problem between $p_{\mathbf{A}}$ and $p_{\hat{\mathbf{A}}^{\star}}$. We approximate a $P = 0$ estimator by (i) using an element-wise structural predictor $\hat{\mathbf{A}}^{\star}$ as an approximation for the posterior mean $\mathbb{E}[\mathbf{A} \mid \mathbf{A}^{\mathcal{O}}]$, and (ii) learning a transport from $p_{\hat{\mathbf{A}}^{\star}}$ to $p_{\mathbf{A}}$ with the rectified flow described below. We do not establish that these predictors recover $\mathbb{E}[\mathbf{A} \mid \mathbf{A}^{\mathcal{O}}]$, so the posterior-mean view motivates the choice of source rather than characterizing it. The learned map is trained to push $p_{\hat{\mathbf{A}}^{\star}}$ toward $p_{\mathbf{A}}$ — approximately, both because of finite model capacity and because a continuous flow cannot exactly match a discrete distribution of binary matrices — and we likewise do not claim that it recovers the exact optimal-transport coupling of (6.3). Intuitively, the procedure refines the coarse estimate $\hat{\mathbf{A}}^{\star}$ into a graph consistent with the underlying data distribution. We focus on the $P = 0$ case: while $P = \infty$ is MSE-optimal, it may produce graphs that violate structural constraints (e.g., chemical validity in molecule generation), whereas $P = 0$ guarantees distributional consistency.

Cross-entropy distortion. Since adjacency matrices are discrete-valued, cross-entropy is a natural alternative to squared error. Let $p^{d}_{\theta}(k \mid \mathbf{A}_t, t)$ denote the cate-

gorical distribution induced by the flow model for edge variable d, and set

$$\Delta(\mathbf{A}, \hat{\mathbf{A}}) = -\sum_{d=1}^{D}\sum_{k=1}^{K^d} \mathbb{I}[A^d = k] \log p_\theta^d(k \mid \mathbf{A}_t, t),$$

where $D = N(N-1)/2$ is the number of edge variables. In the binary instantiation we actually optimize (6.6) below, the sum is restricted to supervised node pairs rather than taken over all D of them, and a positive-class weight is applied to counteract edge sparsity. Unlike the MSE case, this formulation does not admit the same closed-form characterization of the DP trade-off, but it exhibits qualitatively similar behavior empirically.

6.4 Method

PIFM can be viewed as a two-stage refinement framework for conditional graph reconstruction, motivated by the DP formulation of Section 6.3.2. First, a structural prior estimates the conditional mean $\mathbb{E}[\mathbf{A} \mid \mathbf{A}^{\mathcal{O}}]$ from the observed subgraph $\mathbf{A}^{\mathcal{O}}$ by predicting marginal probabilities for the masked node pairs. While this estimate minimizes distortion locally, it does not necessarily produce a globally consistent graph. To address this, PIFM uses the resulting adjacency estimate as the source state of a rectified flow that learns to refine it toward the true graph distribution, recovering higher-order structural dependencies not captured by edge-wise predictions alone. The flow is trained on paired samples $(\mathbf{A}_0, \mathbf{A}_1)$ from the same graph, where $\mathbf{A}_0$ combines observed adjacency entries with prior predictions on the masked region. At inference, only $\mathbf{A}^{\mathcal{O}}$ and ξ are available: the prior produces the initial estimate, and the flow integrates the learned velocity field to generate the final reconstruction.

6.4.1 Approximating the posterior mean

Our goal is to approximate the conditional mean $\mathbb{E}[\mathbf{A} \mid \mathbf{A}^{\mathcal{O}}]$ with a *permutation-equivariant* estimator. We adopt the following latent-variable model.

AS3 *Each edge in* $\mathbf{A} \in \{0,1\}^{N\times N}$ *follows a Bernoulli distribution whose probabilities depend on latent node variables* $\mathbf{z}_1, \ldots, \mathbf{z}_N \in \mathcal{Z}$*:*

$$A_{ij} \sim \text{Bernoulli}\big(f(\mathbf{z}_i, \mathbf{z}_j)\big), \qquad 1 \le i < j \le N, \tag{6.4}$$

where f *maps pairs of latent variables to edge probabilities, i.e.,* $f(\mathbf{z}_i, \mathbf{z}_j) = P(A_{ij} \mid \mathbf{z}_i, \mathbf{z}_j) = p_{ij}$.

AS4 *The edges are conditionally independent given the latent structure:* $P(\mathbf{A} \mid Z) = \prod_{1\le i<j\le N} P(A_{ij} \mid \mathbf{z}_i, \mathbf{z}_j)$*, where* $Z = \{\mathbf{z}_1, \ldots, \mathbf{z}_N\}$.

Under these two assumptions, and assuming access to the mapping $\mathbf{z}^{-1} : \mathbf{A} \to \mathcal{Z}$ that recovers the latent coordinates $Z = \mathbf{z}^{-1}(\mathbf{A}^{\mathcal{O}})$, the posterior mean can be computed element-wise: on observed entries it equals the known value $A^{\mathcal{O}}_{ij}$, while on masked entries $[\mathbb{E}[\mathbf{A} \mid \mathbf{A}^{\mathcal{O}}]]_{ij} = P(A_{ij} = 1 \mid \mathbf{z}_i, \mathbf{z}_j) = f(\mathbf{z}_i, \mathbf{z}_j)$. We adopt two families of priors: *inductive* methods (trained on the whole dataset) and *transductive* methods (fit per graph).

Inductive prior (dataset-informed). We use *GraphSAGE* [158], trained on the partially observed graphs in the dataset to produce node embeddings $\{\mathbf{z}_i\}_{i=1}^N$. From these embeddings, we train a single logistic predictor on Hadamard edge features $(\mathbf{z}_i \odot \mathbf{z}_j)$ to estimate edge probabilities, giving $[\mathbb{E}[\mathbf{A} \mid \mathbf{A}^{\mathcal{O}}]]_{ij} \approx f_\phi(\mathbf{z}_i \odot \mathbf{z}_j)$.

Transductive priors (instance-specific). We use four per-graph priors that fit an embedding and predictor to each partially observed graph $\mathbf{A}^{\mathcal{O}}$.

- *node2vec* [157]: learns embeddings $\{\mathbf{z}_i\}_{i=1}^N$ from biased random walks; a per-graph logistic predictor on Hadamard edge features gives $[\mathbb{E}[\mathbf{A} \mid \mathbf{A}^{\mathcal{O}}]]_{ij} \approx f_\phi(\mathbf{z}_i \odot \mathbf{z}_j)$.
- *VGAE* [159]: uses a GCN encoder producing a Gaussian posterior over per-node latents $q(\mathbf{z}_i \mid \mathbf{A}^{\mathcal{O}}) = \mathcal{N}(\boldsymbol{\mu}_i, \mathrm{diag}(\boldsymbol{\sigma}_i^2))$, and an inner-product decoder yielding $[\mathbb{E}[\mathbf{A} \mid \mathbf{A}^{\mathcal{O}}]]_{ij} \approx \sigma(\boldsymbol{\mu}_i^\top \boldsymbol{\mu}_j)$, trained by ELBO maximization.
- *NCNC* [160]: a neural common-neighbour predictor combining a GCN encoder with a pooled sum over softly-completed common-neighbour representations, giving $[\mathbb{E}[\mathbf{A} \mid \mathbf{A}^{\mathcal{O}}]]_{ij} \approx \sigma(\mathrm{NCNC}(i, j; \mathbf{A}^{\mathcal{O}}, \mathbf{X}))$.
- *LPFormer* [161]: combines a GCN node encoder with pair features, PPR-conditioned attention, and structural counts. We train a fresh model per graph on observed edges and observed non-edges, then score every masked pair.

The remaining machinery is agnostic to which prior is used, so we treat f_{prior} as a generic edge-probability predictor throughout.

6.4.2 Learning the flow model

We now learn a flow model that approximates the joint distribution (coupling) $p(\mathbf{A}, \hat{\mathbf{A}}^\star)$ of the clean graph and its MMSE estimate. Following the flow-matching setup of Section II specialized to graphs (Section 6.3.1), we need to specify the forward path $\mathbf{A}_t$ and the velocity field v. Inspired by [164], we incorporate prior information as the initialization of the forward path; with slight abuse of notation, we denote by f_{prior} the prediction of the full graph (i.e., $f_{\text{prior}}(\mathbf{A}^{\mathcal{O}}) \approx \mathbb{E}[\mathbf{A} \mid \mathbf{A}^{\mathcal{O}}]$). The source sample is

$$\mathbf{A}_0 = \mathbf{A}^{\mathcal{O}} + (1 - \xi) \odot \big(f_{\text{prior}}(\mathbf{A}^{\mathcal{O}}) + \boldsymbol{\epsilon}_s\big), \tag{6.5}$$

where $\xi \in \{0, 1\}^{N \times N}$ is the mask (with $\xi_{ij} = 1$ if pair (i, j) is observed), $\mathbf{A}^{\mathcal{O}} = \xi \odot \mathbf{A}$ is the observed graph, $f_{\text{prior}}(\mathbf{A}^{\mathcal{O}})$ is the approximate MMSE estimator on the masked

node pairs, and $\boldsymbol{\epsilon}_s \sim \mathcal{N}(\mathbf{0}, \sigma_s^2 \mathbf{I})$ is a small additive noise following [175]. We define $\mathbf{A}_1 = \mathbf{A}$ for the target distribution.

Distortion instantiations. Given the source distribution, we learn the flow by minimizing a distortion $\Delta(\hat{y}_\theta, y)$ between a network prediction $\hat{y}_\theta$ and a target y. We consider two instances following Section 6.3.2, which differ in what the network predicts and in the target it is regressed against.

(i) MSE. The network is the velocity field: $\hat{y}_\theta = v_\theta(\mathbf{A}_t, t)$, $y = \mathbf{A}_1 - \mathbf{A}_0$, and $\Delta(\hat{y}_\theta, y) = \|\hat{y}_\theta - y\|_F^2$, recovering the rectified flow-matching loss (6.1).

(ii) CE. The network is instead an *endpoint predictor* $g_\theta(\mathbf{A}_t, t) \in \mathbb{R}^{N \times N}$ emitting per-pair logits for the clean adjacency, so that $\hat{y}_\theta = \sigma(g_\theta(\mathbf{A}_t, t))$ with σ the logistic sigmoid, and the target is the endpoint $y = \mathbf{A}_1$ rather than the residual [154,176–178]. The distortion is a class-weighted binary cross-entropy over the supervised pairs $\mathcal{M} = \{(i,j) : i < j,\ \xi_{ij} = 0\}$:

$$\Delta(\hat{y}_\theta, \mathbf{A}_1) = -\frac{1}{|\mathcal{M}|} \sum_{(i,j)\in\mathcal{M}} \Big[w\, A_{1,ij} \log \sigma(g_{\theta,ij}) + (1 - A_{1,ij}) \log\big(1 - \sigma(g_{\theta,ij})\big) \Big], \quad (6.6)$$
$$w = \min\left(\frac{n_-}{n_+},\, 50\right),$$

averaged per graph and then over the batch, where n_+ and n_- count positives and negatives among the supervised pairs. The weight w prevents the minimizer on sparse graphs from collapsing to predicting no edge. For the CE experiments we set $\sigma_s = 0$.

Training. For the velocity field v, we use the architecture from [151], a GNN-based network that yields a permutation-equivariant parameterization (see Appendix C.2 for details). For each training graph $\mathbf{A}_1$, we sample a mask ξ and time $t \sim U[0,1]$, construct $\mathbf{A}_0$ via (6.5), and minimize the flow-matching objective with the chosen distortion Δ over the resulting paired samples $(\mathbf{A}_0, \mathbf{A}_1)$. Both instantiations additionally

place explicit training weight at $t = 0$ — the only point the $K = 1$ sampler evaluates — via an endpoint-anchor term (MSE) or a boundary-aware time-sampling law (CE); both act only during training and leave the sampler of Algorithm 5 unchanged.

Inference. Given an observed graph $\mathbf{A}^{\mathcal{O}}$ and mask ξ, we initialize $\hat{\mathbf{A}}$ from the prior prediction and integrate the learned velocity field using K Euler steps, as shown in the sampling procedure of Algorithm 5. For a fixed learned field and terminal time, K determines the discretization of the Euler integration; different finite-step choices therefore produce different estimators. Although K is not theoretically a DP control parameter, it correlates empirically with the trade-off [164, 166]: smaller values of K produce estimates closer to the posterior mean, whereas larger values behave more like posterior sampling. Under the MSE objective, the network directly predicts the velocity field. Under the CE objective, it predicts edge-value distributions, from which the velocity field is computed before each update:

$$v_\theta(\mathbf{A}_t, t) = \frac{\sigma\big(g_\theta(\mathbf{A}_t, t)\big) - \mathbf{A}_t}{1 - t + \varepsilon}, \qquad \varepsilon = 10^{-5}, \tag{6.7}$$

which carries the current state $\mathbf{A}_t$ to the predicted endpoint over the remaining time $1 - t$; the constant ε keeps the denominator bounded as $t \to 1$.

Permutation invariance. Combined with the posterior-mean parameterization of Section 6.4.1, the permutation-equivariant velocity field endows PIFM with two related symmetry guarantees. First, the reconstruction map is permutation-equivariant: relabeling the observation and mask $(\mathbf{A}^{\mathcal{O}}, \xi)$ produces a correspondingly relabeled reconstruction $\tilde{\mathbf{A}}_1$. This holds for the full sampler of Algorithm 5 (both the MSE and CE branches, for any number of steps K) and needs only equivariance and well-posedness of the dynamics; it is formalized in Theorem 1. Second, in the idealized

continuous-time flow the conditional density of the reconstruction is itself invariant under simultaneous relabeling of $(\tilde{\mathbf{A}}_1, \mathbf{A}^{\mathcal{O}}, \xi)$; this stronger statement is formalized in Proposition 5. Since graphs are exchangeable, the model's output should not depend on node ordering, making these symmetries a desirable inductive bias.

Theorem 1 (Permutation equivariance of the reconstruction) *Assume:*

(a) the prior $f_{prior} : \mathbb{R}^{N\times N} \to \mathbb{R}^{N\times N}$ *and the velocity field* $v_\theta : \mathbb{R}^{N\times N} \times [0,1] \to \mathbb{R}^{N\times N}$ *are permutation-equivariant;*

(b) v_θ *is locally Lipschitz in its first argument (e.g., a smooth neural network on a bounded domain), so the sampling dynamics are well-posed.*

For an observed graph $\mathbf{A}^{\mathcal{O}}$ *supported on the observed entries (i.e.,* $(1-\xi)\odot\mathbf{A}^{\mathcal{O}} = 0$*) and mask* ξ*, let* $\mathrm{PIFM}(\mathbf{A}^{\mathcal{O}}, \xi)$ *be the reconstruction returned by Algorithm 5 under either the MSE or the CE branch, for any number of Euler steps* $K \geq 1$*. Then, for every permutation matrix* $\mathbf{P}$*,*

$$\mathrm{PIFM}(\mathbf{P}^\top\mathbf{A}^{\mathcal{O}}\mathbf{P},\, \mathbf{P}^\top\xi\mathbf{P}) \;\overset{d}{=}\; \mathbf{P}^\top\,\mathrm{PIFM}(\mathbf{A}^{\mathcal{O}}, \xi)\,\mathbf{P}, \tag{6.8}$$

with equality holding pointwise either under the coupled noise draw $\mathbf{P}^\top\boldsymbol{\epsilon}_s\mathbf{P}$ *or when* $\sigma_s = 0$ *(as in our CE experiments).*

Proposition 5 (Density invariance in the idealized flow) In the idealized continuous, unclipped MSE flow with a C^1 *symmetry-preserving velocity field and* $\sigma_s > 0$*, the conditional density of* $\tilde{\mathbf{A}}_1$ *given* $(\mathbf{A}^{\mathcal{O}}, \xi)$ *is invariant under simultaneous relabeling of* $(\tilde{\mathbf{A}}_1, \mathbf{A}^{\mathcal{O}}, \xi)$ *by any permutation matrix* $\mathbf{P}$*.*

The proofs are in Appendix C.1.1. Theorem 1 provides a closure guarantee for the overall reconstruction map, assuming an equivariant prior and velocity field; the guarantee is exact for GraphSAGE, VGAE, and NCNC (up to coupled randomness

in stochastic inference). For node2vec, the guarantee is only approximate, since the learned embeddings depend on stochastic random walks and optimization and therefore do not define a deterministic equivariant map (see Appendix C.1.1).

Final algorithm. Algorithm 5 describes our training and sampling procedures. In essence, PIFM is a general framework that learns a global graph structure to enhance simple, conditionally independent edge-wise priors.

To illustrate what we mean by *learning a global and dependent predictor*, we now describe a toy experiment. Consider a four-node graph $\mathcal{G}$ (Fig. 6.2(a)) where the goal is to predict the diagonal edges under a specific constraint: the only valid outcomes are either both edges are present or both are absent, i.e., $\mathcal{E} \in \{[e_{02} = 1, e_{13} = 1], [e_{02} = 0, e_{13} = 0]\}$. The probability of the first case is 0.6, and of the second 0.4. We first train an edge-wise prior using node2vec, which yields a probability of 0.6 for each diagonal edge. Crucially, because node2vec models each edge prediction independently, this prior is misspecified: a standard predictor based on it would always predict $[1, 1]$ if used as conditional mean or, if sampled, could generate invalid predictions such as $[1, 0]$. We then train a flow model using this node2vec prior to construct the initial state $\mathbf{A}_0$ as in (6.5). After training (Appendix C.2), we generate 200 samples, illustrated in Fig. 6.2(b); the proportion of each mode is shown in Fig. 6.2(c). The flow model (i) successfully leverages global information, learning a *probabilistic coupling* between the edges to generate samples only from the two valid states, and (ii) learns the probability of each mode.

Algorithm 5 PIFM: training and sampling

Training

Sample $\mathbf{A}_1 \sim p(\mathbf{A})$, a mask ξ, and time $t \sim U[0,1]$ (the CE instantiation instead draws t from $p\,\delta_0 + (1-p)\,U[0,1]$ with $p = 0.25$, weighting the $t = 0$ endpoint).

Train approximate MMSE estimator: $f_{\text{prior}}(\mathbf{A}^{\mathcal{O}})$

Compute $\mathbf{A}_0 = \mathbf{A}^{\mathcal{O}} + (1-\xi) \odot (f_{\text{prior}}(\mathbf{A}^{\mathcal{O}}) + \boldsymbol{\epsilon}_s)$, $\boldsymbol{\epsilon}_s \sim \mathcal{N}(\mathbf{0}, \sigma_s^2 \mathbf{I})$

Compute $\mathbf{A}_t = (1-t)\mathbf{A}_0 + t\mathbf{A}_1$

Train flow model: $\theta^\star = \text{argmin}_\theta\, \mathbb{E}_{\mathbf{A}_1, \mathbf{A}_0, \xi, t}\big[\Delta\big(\hat{y}_\theta(\mathbf{A}_t, t),\, y(\mathbf{A}_0, \mathbf{A}_1)\big)\big]$ (the MSE instantiation adds an endpoint-anchor term at $t = 0$ with weight $\lambda_0 = 5$)

Sampling (Reconstruction)

Initialize $\hat{\mathbf{A}} \leftarrow \xi \odot \mathbf{A}^{\mathcal{O}} + (1-\xi) \odot f_{\text{prior}}(\mathbf{A}^{\mathcal{O}}) + (1-\xi) \odot \boldsymbol{\epsilon}_s$, $\boldsymbol{\epsilon}_s \sim \mathcal{N}(\mathbf{0}, \sigma_{\text{samp}}^2 \mathbf{I})$

for $i \leftarrow 0, \ldots, K-1$ **do**

    if MSE distortion **then**

        $\hat{\mathbf{A}} \leftarrow \hat{\mathbf{A}} + \frac{1}{K}\, v_{\theta^\star}\big(\hat{\mathbf{A}}, \frac{i}{K}\big)$

        $\hat{\mathbf{A}} \leftarrow \xi \odot \mathbf{A}^{\mathcal{O}} + (1-\xi) \odot \hat{\mathbf{A}}$

    else

        // CE distortion

        $\mu \leftarrow \sigma\big(g_{\theta^\star}(\hat{\mathbf{A}}, \frac{i}{K})\big)$

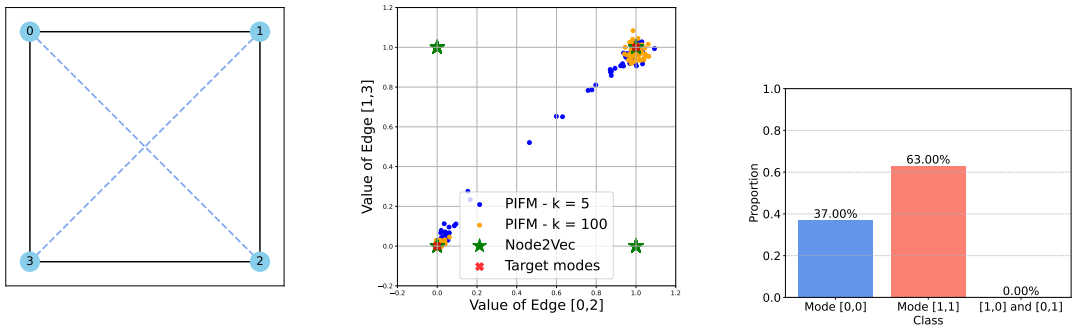
        $\hat{\mathbf{A}} \leftarrow \text{clip}\Big(\hat{\mathbf{A}} + \frac{1}{K} \cdot \frac{\mu - \hat{\mathbf{A}}}{1 - i/K + \varepsilon},\, 0,\, 1\Big)$

        $\hat{\mathbf{A}} \leftarrow \xi \odot \mathbf{A}^{\mathcal{O}} + (1-\xi) \odot \hat{\mathbf{A}}$

    end if

end for

Return $\hat{\mathbf{A}}$



(a) **(b)** **(c)**

Figure 6.2 : Toy experiment showcasing the advantage of PIFM (in this case, for link prediction). (a) Graph $\mathcal{G}$ with four nodes, where the hidden edges are e_{02} and e_{13}. (b) Samples generated by node2vec vs. PIFM: PIFM learns a probabilistic coupling and generates only the two valid modes. (c) Proportions of samples generated with PIFM from each mode; the method also learns a good approximation of the probability of each mode.

Computational cost. The flow-model training time of PIFM is the **same** as flow-model training with a Gaussian prior; the only added expense is training the prior when one is not already available. Inference is roughly 0.04–0.05 s per graph at $K = 1$ on an NVIDIA A100 (vs. 0.0002–0.002 s for the priors alone) and scales linearly in K (e.g., $\sim$2.8 s at $K = 100$); full timings are in Appendices C.2.5 and C.2.6.

Source distribution and prior model. PIFM is agnostic to the prior architecture: any model that produces edge-wise marginal probabilities on the masked region can serve as the source distribution. In practice, we recommend validating candidate priors independently before flow training and selecting the strongest available edge predictor for the target domain.

6.5 Numerical results

6.5.1 Setup

We evaluate PIFM on IMDB-B, PROTEINS, and ENZYMES [179], three inductive graph datasets (training and testing on disjoint graphs), and on Cora [180, 181], which is transductive and large-scale, adding additional challenges for training the flow-matching module. We focus on families of graphs that are diverse to show that our model learns a general predictor. For the inductive datasets, each one is split into 85% train, 10% validation, and 5% test graphs. We evaluate reconstruction quality under two masking levels: each unordered node pair in the upper triangle is independently masked with probability 10% or 50%, regardless of whether the pair is an edge or a non-edge. Implementation details are in Appendix C.2.

Evaluation metrics. Performance is evaluated exclusively on masked unordered node pairs. We report both threshold-dependent and threshold-independent metrics. Threshold-dependent results use a fixed threshold of 0.5 in the main benchmark tables and include the false positive rate (FPR) and false negative rate (FNR). Threshold-independent performance is measured using ROC-AUC and average precision (AP). To assess distributional similarity between reconstructed and ground-truth graphs, we additionally report maximum mean discrepancy (MMD) [182], which serves as a proxy for perceptual quality. Metric definitions are in Appendix C.2.4.

Baselines. We compare PIFM against several baselines, including diffusion-based ones. Recall that PIFM is composed of a one-shot prediction (prior) followed by a flow model. We compare PIFM to the accuracy of the one-shot prediction (without the flow) and with a flow with a random starting point:

- **node2vec [157] / GraphSAGE [158] / NCNC [160] priors**: one-shot predictions using the structural prior directly.
- **LPFormer prior** [161]: a task-specific graph transformer.
- **Flow with Gaussian prior**: flow model initialized from Gaussian $\mathcal{N}(0.5, 1)$ noise on masked entries.
- **Direct GNN refiner**: a non-flow control matched to PIFM in inputs and parameter count but predicting the corrected adjacency in one forward pass at $t = 0$, with no interpolation and no ODE integration.
- **DiGress + RePaint [152]**: unconditional DiGress combined with RePaint-style resampling [168].
- **GDSS + RePaint [151]**: unconditional GDSS combined with RePaint-style resampling.
- **VGAE** [159]: a graph autoencoder that learns latent node embeddings from the observed graph and reconstructs edges using a probabilistic decoder.

6.5.2 Link prediction

Table 6.2 reports results for 50% masking, averaged over five random seeds. Across all three datasets, PIFM trained with the MSE objective improves the AUC of every corresponding Stage-1 prior. In particular, LPFormer is the strongest Stage-1 prior considered in this study, yet PIFM further increases its AUC by 7.54, 6.45, and 5.20 percentage points on ENZYMES, PROTEINS, and IMDB-B, respectively. These consistent gains suggest that the flow-based refinement captures global and joint information that the prior missed.

We notice that both distortions improve ranking over their Stage-1 sources in most metrics. At the fixed 0.5 cutoff, CE generally lowers FNR on sparse graphs

Table 6.2 : Graph reconstruction with **50% of unordered node pairs masked**. Values are five-seed means ± sample standard deviations in percent; RePaint values are single runs. Stage-1 rows are frozen edge predictors evaluated directly, and each PIFM row uses the named predictor unchanged as its source. **Best** and second-best are highlighted for the threshold-free metrics (AP, AUC); FNR and FPR are reported at the fixed 0.5 threshold, where they trade off against each other, and are therefore not ranked.

	ENZYMES				PROTEINS				IMDB-B			
Method	AP↑	AUC↑	FNR↓	FPR↓	AP↑	AUC↑	FNR↓	FPR↓	AP↑	AUC↑	FNR↓	FPR↓
Stage-1 priors (no flow)												
node2vec	$18.39_{\pm 1.01}$	$53.47_{\pm 0.32}$	$49.65_{\pm 2.18}$	$44.57_{\pm 2.34}$	$26.62_{\pm 2.64}$	$54.94_{\pm 1.19}$	$50.28_{\pm 2.10}$	$41.96_{\pm 1.84}$	$59.55_{\pm 2.84}$	$54.04_{\pm 1.61}$	$45.21_{\pm 2.50}$	$46.69_{\pm 2.69}$
GraphSAGE	$24.90_{\pm 2.93}$	$57.55_{\pm 1.54}$	$37.59_{\pm 0.98}$	$56.13_{\pm 2.47}$	$31.95_{\pm 2.49}$	$57.23_{\pm 1.56}$	$36.12_{\pm 2.02}$	$59.75_{\pm 2.92}$	$75.88_{\pm 2.19}$	$73.51_{\pm 2.67}$	$23.96_{\pm 3.23}$	$43.63_{\pm 2.71}$
VGAE	$27.04_{\pm 2.29}$	$64.57_{\pm 2.00}$	$36.29_{\pm 3.50}$	$42.25_{\pm 0.34}$	$34.14_{\pm 2.99}$	$64.36_{\pm 1.64}$	$39.73_{\pm 2.69}$	$40.80_{\pm 0.77}$	$71.76_{\pm 2.96}$	$69.75_{\pm 2.17}$	$40.80_{\pm 1.88}$	$31.85_{\pm 3.04}$
NCNC	$33.17_{\pm 2.36}$	$64.46_{\pm 1.10}$	$92.32_{\pm 1.15}$	$1.20_{\pm 0.52}$	$39.93_{\pm 2.45}$	$64.97_{\pm 1.23}$	$90.67_{\pm 1.51}$	$1.46_{\pm 0.52}$	$84.19_{\pm 0.90}$	$78.55_{\pm 2.38}$	$71.50_{\pm 3.93}$	$0.20_{\pm 0.15}$
LPFormer	$32.32_{\pm 2.91}$	$65.62_{\pm 1.20}$	$65.79_{\pm 3.84}$	$12.60_{\pm 1.38}$	$41.15_{\pm 3.94}$	$67.67_{\pm 1.67}$	$59.82_{\pm 3.66}$	$14.40_{\pm 1.84}$	$91.25_{\pm 1.41}$	$89.79_{\pm 0.52}$	$22.03_{\pm 2.27}$	$7.48_{\pm 0.84}$
Generative/flow baselines												
DiGress + RePaint	17.34	55.22	77.95	11.62	23.65	55.45	71.46	17.65	56.47	58.89	73.00	10.27
GDSS + RePaint	16.43	49.65	69.45	30.46	22.33	51.42	66.44	32.23	53.39	51.20	69.35	29.22
Flow w/ Gaussian prior	$19.81_{\pm 2.89}$	$54.41_{\pm 1.41}$	$96.68_{\pm 2.67}$	$2.15_{\pm 1.42}$	$27.10_{\pm 3.26}$	$55.36_{\pm 1.39}$	$89.41_{\pm 3.74}$	$6.37_{\pm 3.60}$	$80.66_{\pm 1.72}$	$80.37_{\pm 1.58}$	$28.26_{\pm 5.63}$	$22.96_{\pm 4.61}$
PIFM (MSE loss)												
PIFM (node2vec)	$25.29_{\pm 2.65}$	$62.48_{\pm 1.25}$	$96.82_{\pm 2.07}$	$1.50_{\pm 1.12}$	$34.88_{\pm 2.79}$	$64.96_{\pm 1.33}$	$86.16_{\pm 2.49}$	$5.90_{\pm 2.42}$	$85.77_{\pm 1.48}$	$84.75_{\pm 1.20}$	$24.27_{\pm 2.67}$	$18.18_{\pm 2.61}$
PIFM (GraphSAGE)	$27.14_{\pm 2.61}$	$62.55_{\pm 0.88}$	$96.80_{\pm 1.80}$	$1.16_{\pm 0.91}$	$35.61_{\pm 3.30}$	$64.91_{\pm 2.14}$	$85.15_{\pm 5.13}$	$5.11_{\pm 2.49}$	$92.95_{\pm 1.15}$	$93.46_{\pm 1.11}$	$15.81_{\pm 0.71}$	$9.02_{\pm 2.67}$
PIFM (VGAE)	$30.90_{\pm 3.15}$	$68.57_{\pm 2.26}$	$94.78_{\pm 1.55}$	$1.91_{\pm 0.93}$	$38.09_{\pm 2.65}$	$69.22_{\pm 1.13}$	$83.50_{\pm 4.62}$	$7.65_{\pm 1.77}$	$92.52_{\pm 2.02}$	$92.46_{\pm 1.37}$	$15.32_{\pm 2.17}$	$11.21_{\pm 5.97}$
PIFM (NCNC)	$34.38_{\pm 4.43}$	$67.07_{\pm 3.01}$	$87.87_{\pm 6.11}$	$2.63_{\pm 1.71}$	$41.52_{\pm 4.41}$	$68.00_{\pm 3.51}$	$63.64_{\pm 5.26}$	$14.60_{\pm 5.99}$	$92.92_{\pm 0.52}$	$91.07_{\pm 2.00}$	$18.53_{\pm 3.59}$	$10.10_{\pm 6.17}$
PIFM (LPFormer)	$35.99_{\pm 3.01}$	$73.16_{\pm 1.39}$	$92.35_{\pm 2.07}$	$1.79_{\pm 0.59}$	$43.40_{\pm 3.05}$	$74.12_{\pm 0.98}$	$80.20_{\pm 6.00}$	$6.74_{\pm 1.86}$	**$95.28_{\pm 1.11}$**	**$94.99_{\pm 0.48}$**	$9.79_{\pm 1.67}$	$7.59_{\pm 2.01}$
PIFM (CE loss)												
PIFM (node2vec)	$25.13_{\pm 2.65}$	$62.17_{\pm 1.24}$	$28.48_{\pm 10.30}$	$60.52_{\pm 10.36}$	$33.17_{\pm 3.33}$	$63.79_{\pm 1.44}$	$9.03_{\pm 5.23}$	$86.45_{\pm 6.76}$	$85.74_{\pm 1.31}$	$84.59_{\pm 1.46}$	$19.16_{\pm 5.91}$	$25.81_{\pm 6.06}$
PIFM (GraphSAGE)	$27.35_{\pm 3.57}$	$61.97_{\pm 2.14}$	$33.00_{\pm 7.47}$	$58.51_{\pm 7.87}$	$34.24_{\pm 3.04}$	$63.40_{\pm 1.15}$	$9.45_{\pm 4.63}$	$88.24_{\pm 5.23}$	$91.96_{\pm 1.28}$	$92.14_{\pm 1.11}$	$14.92_{\pm 3.11}$	$13.91_{\pm 3.05}$
PIFM (VGAE)	$30.19_{\pm 2.36}$	$68.21_{\pm 1.55}$	$24.71_{\pm 8.44}$	$55.02_{\pm 7.94}$	$37.75_{\pm 2.87}$	$68.54_{\pm 1.18}$	$10.57_{\pm 3.84}$	$81.21_{\pm 6.14}$	$91.92_{\pm 1.43}$	$91.41_{\pm 0.70}$	$16.01_{\pm 1.61}$	$13.33_{\pm 3.50}$
PIFM (NCNC)	**$38.45_{\pm 3.97}$**	**$75.35_{\pm 1.77}$**	$26.42_{\pm 4.29}$	$39.40_{\pm 5.75}$	**$45.05_{\pm 1.99}$**	**$75.01_{\pm 1.71}$**	$14.03_{\pm 5.30}$	$62.70_{\pm 8.45}$	$94.14_{\pm 0.83}$	$93.07_{\pm 1.79}$	$12.13_{\pm 2.02}$	$13.59_{\pm 4.93}$
PIFM (LPFormer)	$34.21_{\pm 3.17}$	$72.06_{\pm 2.02}$	$23.91_{\pm 6.46}$	$49.68_{\pm 8.25}$	$42.03_{\pm 3.66}$	$72.95_{\pm 2.59}$	$12.97_{\pm 2.88}$	$69.07_{\pm 6.98}$	$95.08_{\pm 1.42}$	$94.75_{\pm 0.71}$	$9.28_{\pm 2.51}$	$10.37_{\pm 3.09}$

while increasing FPR; MSE emphasizes the continuous-score distortion used by the flow objective. We therefore report both ranking and thresholded behavior without treating the losses as interchangeable. The prior supplies task-specific edge scores, while the flow learns a correction over the reconstructed adjacency matrix, and their benefits remain complementary across the tested prior families.

Large-scale graph reconstruction. We evaluate PIFM on Cora over five seeds as a large-scale feasibility check, not as a leaderboard comparison. The MSE model uses one-hop, single-edge subgraphs, whereas the CE model uses two-hop, multi-edge subgraphs.

Table 6.3 : Cora results over five seeds. AUC measures held-out-edge ranking; the remaining columns are subgraph-pool MMD^2 values. **Bold blue** values indicate the best result(s) in each metric: higher is better for AUC, while lower is better for MMD^2. The prior is repeated because each configuration induces a different subgraph pool.

Configuration	Method	AUC ↑	Clustering MMD ↓	Degree MMD ↓	NSPDK MMD ↓
MSE, single-edge, 1-hop	NCNC prior	**93.49 ± 0.55**	.122 ± .062	.070 ± .030	.061 ± .026
	PIFM	93.36 ± 0.66	.100 ± .080	.072 ± .063	.060 ± .044
CE, multi-edge, 2-hop	NCNC prior	**93.49 ± 0.55**	.172 ± .086	.085 ± .041	.030 ± .009
	PIFM	87.17 ± 11.11	**.053 ± .044**	**.040 ± .025**	**.020 ± .007**

For MSE, PIFM is on par with the NCNC prior on AUC (the small paired difference does not reach significance at five seeds) and slightly improves the clustering and NSPDK means, while degree MMD is nearly unchanged. The CE multi-edge configuration shows higher AUC variability but improves all three distributional metrics on every seed: even where PIFM does not improve ranking, it brings the reconstructed graphs closer to the clean distribution.

6.5.3 Blind graph reconstruction

We focus on two blind versions of link prediction, namely expansion and denoising. In the expansion case, we only get to observe a subset of the edges (but no non-edges), and we need to determine which other entries correspond to existing edges.

Conversely, for denoising, we observe a set of edges that includes some spurious ones (but no confirmed non-edges), and we need to determine which of the observed edges are spurious and should be removed. These cases are more challenging than link prediction since transductive priors like node2vec cannot be trained on the masked graphs (since we do not have positive and negative edges). We report expansion here.

Expansion. The goal in expansion is to predict a set of hidden edges $\mathcal{E}_M$ given $\mathbf{A}^{\mathcal{O}}$, such that the edge set of the ground truth is $\mathcal{E} = \mathcal{E}_M \cup \mathcal{E}_{\mathcal{O}}$. Therefore, defining $\mathbf{A}_1 = \mathbf{A}$, the initialization becomes $\mathbf{A}_0 = \mathbf{A}^{\mathcal{O}} + (1 - \mathbf{A}^{\mathcal{O}}) \odot (f_{\text{prior}}(\mathbf{A}^{\mathcal{O}}) + \boldsymbol{\epsilon}_s)$. Throughout the blind-reconstruction experiments, we use PIFM with the MSE loss. The results for a drop rate of 50% are shown in Table 6.4.

Across five seeds, PIFM (GraphSAGE) attains the top AP and AUC on all three datasets, improving over the diffusion baselines. Furthermore, it raises its own Stage-1 prior AUC by 10.3, 10.2, and 13.6 percentage points on ENZYMES, PROTEINS, and IMDB-B, respectively, with the gaps exceeding the seed-to-seed standard deviations. The uninformed flow with a Gaussian source stays near random guess with $p = 0.5$ on the sparse datasets (50.60 on ENZYMES and 51.73 on PROTEINS), so the gain for the flow module requires an informative prior estimate. On the denser IMDB-B, the uninformed flow reaches 63.72 AUC, illustrating that denser graphs have a richer structure that can be exploited even without a prior. As in link prediction, FNR at the fixed 0.5 threshold remains high on the sparse datasets under the MSE loss. Lastly, we remark that varying the Euler step count yields empirical operating points whose distortion and distributional behavior are inspected in the following subsection; the step count is not treated as a calibrated trade-off parameter.

Table 6.4 : Performance for the **expansion** task with **50% of edges masked**. Values are five-seed means ± sample standard deviations in percent; † marks the RePaint baselines, which are single runs. Metrics are computed on the region the flow updates, $1 - \mathbf{A}^{\mathcal{O}}$. FNR and FPR use the fixed 0.5 threshold.

Mask Rate: 50% (0.5 Drop)												
	ENZYMES				PROTEINS				IMDB-B			
Method	AP↑	AUC↑	FNR↓	FPR↓	AP↑	AUC↑	FNR↓	FPR↓	AP↑	AUC↑	FNR↓	FPR↓
Baselines												
GraphSAGE	16.02±2.22	57.54±1.69	**37.79**±1.16	56.02±2.52	22.73±2.82	57.01±1.48	**36.22**±2.19	60.83±2.85	67.30±2.72	73.26±2.91	**24.09**±3.36	43.81±3.02
DiGress + RePaint†	8.09	50.69	87.21	11.40	13.50	52.46	80.02	15.07	39.70	53.07	81.10	13.00
GDSS + RePaint†	9.21	49.63	69.45	30.80	14.66	51.03	66.44	32.06	39.43	50.68	69.35	30.00
Flow w/ Gaussian prior	9.94±1.07	50.60±1.48	100.00±0.00	**0.00**±0.00	16.81±2.67	51.73±0.50	94.71±2.83	**4.25**±2.62	56.61±3.40	63.72±2.47	64.63±7.49	12.26±5.64
Ours												
PIFM (GraphSAGE)	**19.59**±2.00	**67.86**±1.17	99.05±0.88	0.27±0.20	**26.34**±2.05	**67.22**±1.94	92.44±3.25	4.43±1.79	**82.52**±0.90	**86.89**±2.21	28.55±3.77	**10.47**±4.29

6.5.4 Distortion–perception trade-off

We further analyze the DP trade-off using MMD as a proxy for perceptual quality, following GraphRNN [183], GDSS [151], and DeFoG [153].

Fig. 6.3 illustrates this trade-off on ENZYMES. Increasing the number of Euler steps decreases MMD^2 and moves degree, triangle, and clustering statistics closer to the reference distribution, whereas fewer steps produce reconstructions that favor edge-wise distortion metrics. Similar behavior appears in the five-seed LPFormer study with MSE loss and 50% masking: increasing the number of steps from $K = 1$ to $K = 100$ reduces NSPDK MMD^2 by approximately 3.1× on ENZYMES and 3.5× on PROTEINS, although at the cost of lower AUC. On IMDB-B, however, increasing K slightly increases MMD, indicating that the effect is dataset-dependent.

Similar empirical trends with increasing integration steps have been reported for PMRF [164] and InDI [166]. These observations should not be interpreted as estab-

lishing a theoretical or solver-independent DP control. Understanding estimator-level interpolation, fixed-accuracy solver comparisons, and explicit terminal-time control remains future work.

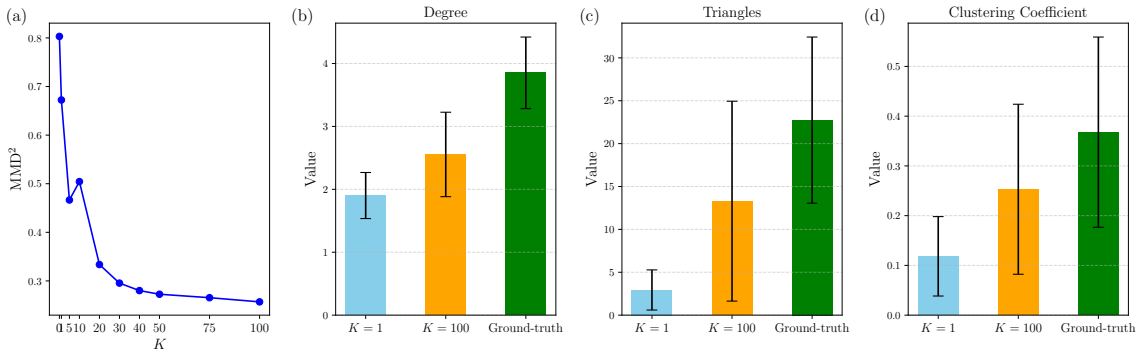


Figure 6.3 : Empirical DP behavior on the ENZYMES expansion experiment. (a) MMD measures distance to the reference graph distribution and is lower after additional integration in this setting. (b–d) At $K = 100$, degree, triangle, and clustering-coefficient statistics more closely match the ground truth than at $K = 1$. This dataset-specific comparison is not a claim that MMD decreases monotonically with K in every setting. Error bars show the standard deviation over 300 samples (10 per test graph).

6.6 Conclusion

We introduced **Prior-Informed Flow Matching (PIFM)**, a flow-based reconstructor that refines a local edge-prediction prior (GraphSAGE, node2vec, VGAE, NCNC, or LPFormer) toward the clean-graph distribution under either an MSE or a class-weighted cross-entropy distortion, and consistently outperforms both classical embedding methods and recent diffusion/flow-based graph baselines. Three chapter-specific limitations point to natural directions: (i) the Stage-1 prior is trained separately and cannot adapt to the flow — end-to-end training would require handling the result-

ing non-stationary source distribution; (ii) the continuous relaxation of the binary adjacency could be replaced by categorical-simplex [154] or discrete Markov [153] dynamics for heterogeneous graphs; and (iii) scaling beyond the dense $B \times N \times N$ edge state will require sparse or block-wise subgraph training. Chapter 9 discusses broader synthesis and cross-part combinations.

Chapter 7

Learning normalized energy models

Standard denoising score matching learns only the *gradient* of the log-density of a diffusion model, not the log-density itself. The normalization constant remains inaccessible, which blocks any downstream task that requires a proper density: blind inverse problems in which the noise level must be recovered from the observation alone, principled uncertainty quantification, and posterior-based comparison across solvers. In this chapter we develop *energy-based models with anisotropic noise*, a training-time modification that learns a normalized density directly. The key mechanism is to extend the noising process from scalar-valued to matrix-valued covariances $\mathbf{\Sigma}_t$; this unlocks a companion *covariance score* $\nabla_{\mathbf{\Sigma}} \log p(\mathbf{y}, \mathbf{\Sigma})$ that, together with the data score, is regressed via a *dual score-matching* objective and normalizes the learned density across noise levels. We prove that the resulting energy is aligned with the target posterior geometry, connect anisotropic energy-guided sampling to a Bregman mirror-descent scheme, and validate the approach on linear and blind image inverse problems, where it enables the density-dependent tasks listed above and improves reliability over standard isotropic energy-based baselines.

This chapter is adapted from [13].

7.1 Introduction

Diffusion models [5–7, 31] provide expressive priors for image inverse problems [120]. As reviewed in Section 2.6, the two dominant families of solvers built on these priors are *plug-and-play* methods [2, 52, 106], which combine an unconditional prior with the measurement likelihood via Bayes' rule at test time, and *regression* methods [4, 166, 184–186], which learn conditional models tied to a specific degradation. This chapter argues that neither family gives us what we most need for inverse problems — *access to a normalized density* — and develops a training-time modification that provides it.

The plug-and-play family is flexible but biased: the noisy likelihood is intractable, and every practical approximation introduces error that accumulates over the reverse process [2, 187]. The regression family avoids the approximation but locks the prior to a single degradation. Both families are, moreover, *score-based*: they learn $\nabla_{\mathbf{x}_t} \log p_t$ rather than $\log p_t$ itself, so posterior-density evaluation is computationally expensive when it is possible at all. As Section 2.7 makes precise, this rules out a range of downstream tasks that require a normalized density — blind inverse problems in which the noise level is unknown, principled uncertainty quantification, entropy and mutual-information estimation, and posterior-based comparison across solvers. Recent work has combined energy-based models with unconditional diffusion for compositional generation [48, 188] and introduced regularizers that enforce proper normalization [189–191], but these methods are limited to isotropic noise and therefore cannot handle the correlated (colored) noise structure that arises naturally in linear inverse problems.

In this chapter we introduce *Anisotropic Covariance Score Matching* (A-CSM), an extension of the dual-score-matching framework of [189] to anisotropic (colored)

noise. The starting observation is a change of variables: a linear inverse problem $\mathbf{y} = \mathbf{H}\mathbf{x} + \sigma\mathbf{v}$ with invertible $\mathbf{H}$ is equivalent to observing $\tilde{\mathbf{y}} = \mathbf{x} + \tilde{\mathbf{v}}$ with $\tilde{\mathbf{v}} \sim \mathcal{N}(0, \sigma^2\mathbf{H}^{-1}\mathbf{H}^{-\top})$, so solving inverse problems amounts to *removing anisotropic noise* indexed by a full covariance $\mathbf{\Sigma}$. We accordingly learn a model of $\log p(\mathbf{y}, \mathbf{\Sigma})$ across a family of covariances, which produces a companion *covariance score* $\nabla_{\mathbf{\Sigma}} \log p(\mathbf{y}, \mathbf{\Sigma})$ that ties inverse problems at different noise levels together and normalizes the density across the family. The A-CSM objective is derived from a Fokker–Planck consistency condition and is complemented by a novel noise-covariance embedding in the network architecture. A single trained model handles a range of degradations consistently and unlocks capabilities the score-only literature could not deliver: energy-guided adaptive sampling, unbiased MCMC correction, and blind inverse-problem solutions. We validate on ImageNet, CelebA, and AFHQ across inpainting and deblurring, matching or improving on established diffusion-based solvers.

7.2 Background

7.2.1 Diffusion models: from isotropic to anisotropic noise

We build on the continuous-time score-based diffusion framework introduced in Section 2.2. Throughout this chapter we work with the variance-exploding (VE) parameterization $\mathbf{x}_t = \mathbf{x} + \sigma_t\mathbf{v},\ \mathbf{v} \sim \mathcal{N}(0, \mathbf{I})$ and its discrete reverse process

$$\mathbf{x}_{t-1} = \mathbf{x}_t + \delta\sigma_t^2 \nabla \log p(\mathbf{x}_t) + \delta\sigma_t\, \mathbf{v}_t, \qquad \delta\sigma_t^2 = \sigma_t^2 - \sigma_{t-1}^2, \tag{7.1}$$

where the score $\nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t)$ is approximated by a neural network trained via denoising score matching (DSM; see Section 2.4). We use throughout the fact that the trained model provides an *implicit* representation of $p(\mathbf{x})$ via its family of scores at all noise levels [106].

Beyond isotropic noise.

While the majority of diffusion models assume an isotropic forward process, several recent studies have explored more general processes, including anisotropic processes with vector-valued parameterizations [192–197] and learnable [198, 199] and higher-order processes [68, 200]. Here, we make use of anisotropic forward diffusion process with covariance schedule $(\mathbf{\Sigma}_t)_{t=0}^T$:

$$\mathbf{x}_t = \mathbf{x} + \mathbf{\Sigma}_t^{\frac{1}{2}} \mathbf{v}, \quad \mathbf{v} \sim \mathcal{N}(0, \mathbf{I}), \tag{7.2}$$

with the corresponding reverse process

$$\mathbf{x}_{t-1} = \mathbf{x}_t + \boldsymbol{\delta} \mathbf{\Sigma}_t \nabla \log p(\mathbf{x}_t) + \boldsymbol{\delta} \mathbf{\Sigma}_t^{\frac{1}{2}} \mathbf{v}_t. \tag{7.3}$$

7.2.2 Linear inverse problems

We instantiate the general inverse-problem setup of Section 2.1 in the linear form used throughout this chapter,

$$\mathbf{y} = \mathbf{H}\mathbf{x} + \sigma \mathbf{v}, \quad \mathbf{v} \sim \mathcal{N}(0, \mathbf{I}), \tag{7.4}$$

with $\mathbf{H}$ a known linear degradation operator. Section 2.6 classifies the existing diffusion-based solvers into four families; here we focus on the two that are most relevant for the anisotropic energy models developed in the remainder of the chapter.

Plug-and-play approaches.

Plug-and-play methods condition the reverse process (7.1) on the observation $\mathbf{y}$ using the Bayes-rule decomposition of the conditional score (Section 2.6, (2.19)). The prior score is provided by a pre-trained *isotropic* diffusion model; the guidance term $\nabla_{\mathbf{x}_t} \log p(\mathbf{y}|\mathbf{x}_t)$ is intractable because it marginalizes over the clean signal, and existing

methods [2,52,53,106] rely on Gaussian approximations whose evaluation typically requires backpropagation through the learned score — an expensive Jacobian computation. Beyond these guidance approaches, optimization-based methods [10,50,122,201] recast posterior sampling as stochastic optimization, and sequential-Monte-Carlo methods [54] leverage particle systems.

Regression-based approaches.

A second family learns the conditional score of (2.19) directly, treating the inverse problem as a conditional generation task with paired data $(\mathbf{x}, \mathbf{y})$ [166, 184, 185, 202]. Recent work [186, 203] has proposed degradation-aware parameterizations of the conditional denoiser $\mathbb{E}[\mathbf{x} \mid \mathbf{y}, \mathbf{H}]$; these introduce their own computational overhead through repeated ambient-to-feature-space mappings within the architecture.

7.3 Learning EBMs through anisotropic denoising

We aim to learn a *single* model $p_\theta(\mathbf{x}|\mathbf{y})$ that explicitly approximates the posterior *density* for measurements $\mathbf{y}$ arising from a variety of different degradation operators, in contrast to previous score-based approaches. Learning the posterior density gives access not only to the conditional score through differentiation, but also unlocks several new applications that we describe in Section 7.4. We first describe in Section 7.3.1 the connection between linear inverse problems and anisotropic denoising, and how this can be leveraged to learn an energy model. We introduce our generalized dual score matching loss in Section 7.3.2 and our anisotropic energy architecture in Section 7.3.3. Finally, we show in Section 7.3.4 how our energy model can be used both as a denoiser and as a posterior sampler for linear inverse problems.

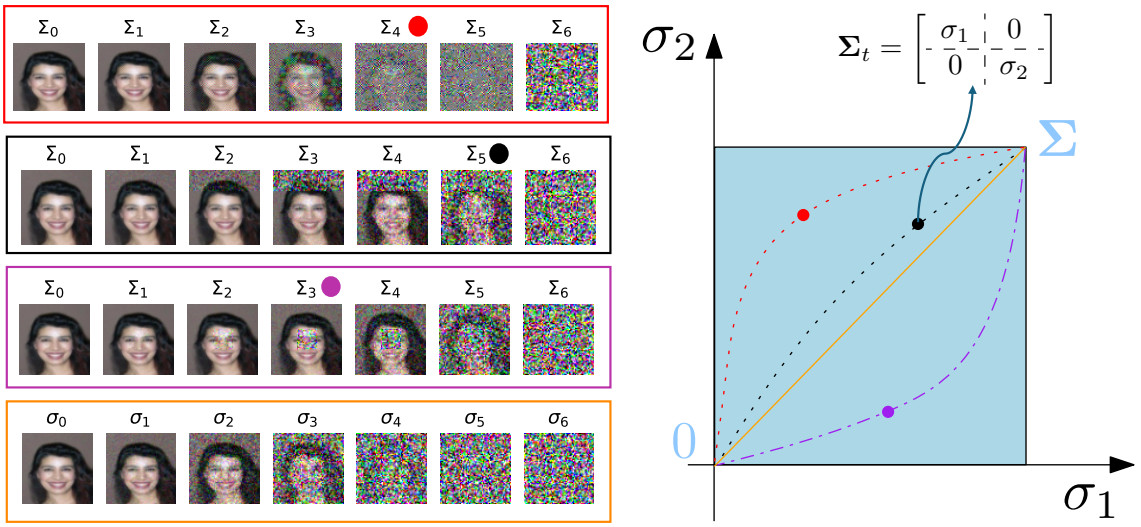


Figure 7.1 : Illustration of possible paths in image space (left) and covariance space (right). Isotropic models are limited to covariance schedules that are reparameterizations of the diagonal (orange), while anisotropic models can explore paths where different signal components are noise-corrupted at different rates (red, black, and purple).

7.3.1 Energy-based linear inverse problem solvers

We leverage the equivalence between linear degradations and colored noise to show how an energy-based model conditioned on different noise covariances enables access to normalized posterior densities, posterior means, and posterior samples.

From linear inverse problems to anisotropic denoising.

Consider a linear inverse problem of the form $\mathbf{y} = \mathbf{H}\mathbf{x} + \sigma\mathbf{v}$. The posterior is unchanged by redefining $\mathbf{y}$ as $\mathbf{H}^{-1}\mathbf{y}$, which corresponds to observing

$$\mathbf{y} = \mathbf{x} + \boldsymbol{\Sigma}^{\frac{1}{2}}\mathbf{v}', \quad \mathbf{v}' \sim \mathcal{N}(0, \mathbf{I}), \tag{7.5}$$

where $\boldsymbol{\Sigma} = \sigma^2\mathbf{H}^{-1}(\mathbf{H}^{-1})^\top$. Linear inverse problems are thus equivalent to denoising problems for images contaminated with correlated additive noise, with the noise covariances $\boldsymbol{\Sigma}$ depending on the linear measurement $\mathbf{H}$. If $\mathbf{H}$ is not invertible, it can

be stabilized by addition of a small multiple of the identity, which corresponds to adding large noise in the nullspace of $\mathbf{H}$. In the following, $\mathbf{y}$ refers to the observation model (7.5).

Anisotropic EBMs.

Using Bayes rule, the posterior distribution can be expressed as $p(\mathbf{x}|\mathbf{y}) = p(\mathbf{x})p(\mathbf{y}|\mathbf{x})/p(\mathbf{y})$. Its normalization requires having a model of $p(\mathbf{y})$, which depends on $\mathbf{\Sigma}$. To this end, we propose to learn an energy model $U_\theta(\mathbf{y}, \mathbf{\Sigma})$ that approximates $U(\mathbf{y}, \mathbf{\Sigma}) = -\log p(\mathbf{y}|\mathbf{\Sigma})$. Notice that we can recover $p(\mathbf{x})$ by setting $\mathbf{\Sigma} = 0$.

The posterior mean can be computed using an anisotropic generalization of Tweedie's identity: [204–206]:

$$\mathbb{E}[\mathbf{x} \,|\, \mathbf{y}, \mathbf{\Sigma}] = \mathbf{y} - \mathbf{\Sigma}\nabla_{\mathbf{y}} U(\mathbf{y}, \mathbf{\Sigma}), \tag{7.6}$$

where the score can be recovered from the energy model as $\nabla_{\mathbf{y}} U(\mathbf{y}, \mathbf{\Sigma}) \approx \nabla_{\mathbf{y}} U_\theta(\mathbf{y}, \mathbf{\Sigma})$. Therefore, by *learning an anisotropic denoiser*, we gain access to the posterior mean for different linear inverse problems.

Learning a score model over a family of covariances $\mathbf{\Sigma}$ enables posterior sampling. This is achieved by defining a noise schedule that bridges between measurements and the corresponding clean data. In particular, we need to specify a sequence of covariance matrices $(\mathbf{\Sigma}_t)_{t=0}^T$ that interpolate between the measurement covariance $\mathbf{\Sigma}$ at $t = T$ and the zero covariance at $t = 0$, as in (7.3). Several potential paths are illustrated in Fig. 7.1. Note that unlike the isotropic setting, these schedules do not simply correspond to different discretizations of a corresponding continuous-time process.

7.3.2 Learning with anisotropic dual score matching

We learn a model of the energy $U(\mathbf{y}, \mathbf{\Sigma}) = -\log p(\mathbf{y}|\mathbf{\Sigma})$ by developing an anisotropic generalization of the (isotropic) dual score matching approach of [189]. In a nutshell, dual score matching consists of matching *both* derivatives of an energy model $U_\theta(\mathbf{y}, \mathbf{\Sigma})$ to the data, i.e., the gradients of the energy with respect to *(i)* the noisy input $\mathbf{y}$ (the "data score" used in diffusion models) and *(ii)* the covariance $\mathbf{\Sigma}$ (which we dub the "covariance score"). Each of these derivatives can be trained by minimizing a corresponding loss function. As we explain below, the dual score matching formulation enables learning of *normalized* densities.

Data score.

The data score can be learned using the anisotropic generalization of Tweedie's formula (7.6), which leads to the anisotropic denoising score matching objective:

$$\ell_{\mathrm{A-DSM}}(\theta) = \mathbb{E}\left[\|\mathbf{\Sigma}^{\frac{1}{2}}\nabla_{\mathbf{y}} U_\theta(\mathbf{y}, \mathbf{\Sigma}) - \mathbf{\Sigma}^{-\frac{1}{2}}(\mathbf{y} - \mathbf{x})\|^2\right]. \tag{7.7}$$

The reweighting by $\mathbf{\Sigma}^{\frac{1}{2}}$ leads to a scale-invariant loss, a generalization of the maximum-likelihood weighting used in [41, 189].

Covariance score.

The covariance score can be shown to satisfy an analogue of the Tweedie formula (see Appendix D.1):

$$\nabla_{\mathbf{\Sigma}} U(\mathbf{y}, \mathbf{\Sigma}) = \mathbb{E}\left[\frac{1}{2}\mathbf{\Sigma}^{-1} - \frac{1}{2}\mathbf{\Sigma}^{-1}(\mathbf{y} - \mathbf{x})(\mathbf{y} - \mathbf{x})^\top \mathbf{\Sigma}^{-1}\right].$$

This expression generalizes the time score identity of [189–191], and leads to a covariance score matching objective:

$$\ell_{\text{A}-\text{CSM}}(\theta) = \mathbb{E}_{\mathbf{x},\mathbf{y},\boldsymbol{\Sigma}}\Big[\Big\|\boldsymbol{\Sigma}^{\frac{1}{2}}\nabla_{\boldsymbol{\Sigma}}U_\theta(\mathbf{y},\boldsymbol{\Sigma})\boldsymbol{\Sigma}^{\frac{1}{2}} - \frac{1}{2}\mathbf{I} + \frac{1}{2}\boldsymbol{\Sigma}^{-\frac{1}{2}}(\mathbf{y}-\mathbf{x})(\mathbf{y}-\mathbf{x})^\top\boldsymbol{\Sigma}^{-\frac{1}{2}}\Big\|_2^2\Big], \tag{7.8}$$

that has been weighted by $\boldsymbol{\Sigma}^{\frac{1}{2}}$ on both sides to be scale-invariant, and where $\|\cdot\|_F$ is the Frobenius norm. The A-CSM objective (7.8) acts as a regularizer that ties all the different marginals $p(\mathbf{y}|\boldsymbol{\Sigma})$ together in a meaningful way. It can be thought of as a proxy to enforce the continuity equation given by the Fokker-Planck equation [75]. This allows learning of a normalized density (as we show below), but also improves the denoising performance of the model, as illustrated in Appendix D.4.2.

Normalization of the learned density model.

We define an overall objective as a weighted sum of the two objectives defined above: $\frac{1}{d}\ell_{\text{A}-\text{DSM}} + \frac{1}{d^2}\ell_{\text{A}-\text{CSM}}$ (see Appendix D.1). After training, the learned energy model $U_\theta(\mathbf{y},\boldsymbol{\Sigma})$ approximates the true energy up to a constant: $U_\theta(\mathbf{y},\boldsymbol{\Sigma}) \approx U(\mathbf{y},\boldsymbol{\Sigma}) + \text{cst}$. Importantly, the A-CSM objective ensures mass conservation through the space of covariances, and thus this constant is independent of $\boldsymbol{\Sigma}$. For large $\boldsymbol{\Sigma}$, $\mathbf{y}$ is approximately distributed as $\mathcal{N}(0,\boldsymbol{\Sigma})$, allowing to normalize the trained energy model as follows:

$$U_\theta(\mathbf{y},\boldsymbol{\Sigma}) \quad \rightarrow \quad U_\theta(\mathbf{y},\boldsymbol{\Sigma}) \; - \; \mathbb{E}_{\mathbf{y}}[U_\theta(\mathbf{y},\boldsymbol{\Sigma}) \quad | \quad \boldsymbol{\Sigma}] \; + \; \frac{1}{2}\log\det(2\pi\mathrm{e}\boldsymbol{\Sigma}). \tag{7.9}$$

7.3.3 Architecture and covariance conditioning

We compute $U_\theta(\mathbf{y},\boldsymbol{\Sigma})$ with a neural network, with a novel architecture that is suitable for the problem. Since our proposed energy model is trained as an anisotropic

denoiser, the first requirement is that the score $\nabla_{\mathbf{y}} U_\theta(\mathbf{y}, \mathbf{\Sigma})$ derived from the energy preserves the inductive biases of score architectures used in previous literature to achieve high-quality denoising (and thus, score estimation). Following [43, 188, 189], we define

$$U_\theta(\mathbf{y}, \mathbf{\Sigma}) = \frac{1}{2}\langle \mathbf{y}, \mathbf{s}_\theta(\mathbf{y}, \mathbf{\Sigma})\rangle, \tag{7.10}$$

where $\mathbf{s}_\theta$ is an existing score network. Specifically, we implement $\mathbf{s}_\theta$ using the UNet architecture of [7, 47].

The second requirement is that the architecture can operate under conditioning with a wide range of covariances $\mathbf{\Sigma}$ arising from different linear inverse problems. However, an arbitrary covariance has $d(d-1)/2$ degrees of freedom, which is problematic in terms of both memory and computational cost. To alleviate this, we limit ourselves to covariances that are diagonal in either the spatial or spatial frequency domain. This reduces the parameterization to d coefficients, while still covering many standard inverse problems: inpainting (block-diagonal covariances in pixel space), deblurring (diagonal in frequency domain), and super-resolution (approximately diagonal in frequency domain).

Finally, we must define a mechanism by which the conditioning covariance matrix $\mathbf{\Sigma}$ is incorporated into the score network. Our design builds on the multiplicative conditioning mechanism used in isotropic score architectures via gain control [207]. At each layer ℓ, consisting of c_ℓ feature channels, an embedding vector $\mathbf{e}_\ell \in \mathbb{R}^{c_\ell}$ of the input noise variance σ^2 is computed and used to modulate (multiply) the channels. We represent spatial covariance matrices as spatially varying noise maps in $\mathbb{R}^d$, yielding *spatially varying embeddings* $\mathbf{e}_\ell \in \mathbb{R}^{c_\ell \times d_\ell}$, where d_ℓ denotes the spatial resolution at layer ℓ. Spectral covariance matrices are incorporated through an analogous mechanism in the frequency domain, producing embeddings that modulate the correspond-

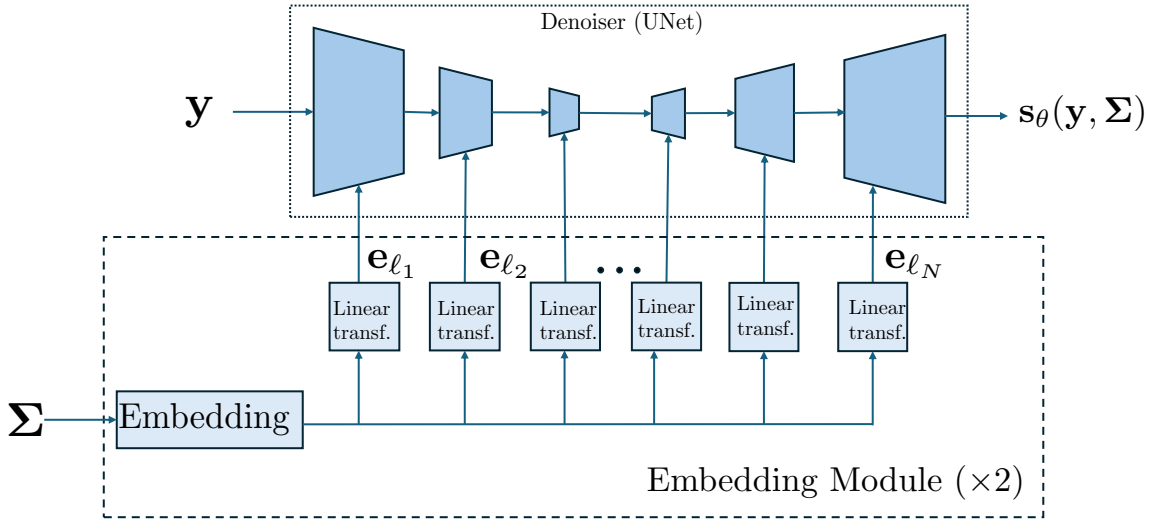


Figure 7.2 : Proposed architecture based on UNet. We incorporate the covariance information through the embedding network, with two dedicated branches for the two covariance domains (spatial and spectral).

ing feature channels (but not the spatial dimensions). At each layer, conditioning is applied through a gain modulation of the form $\mathbf{x}_\ell \leftarrow \text{SiLU}\left(\mathbf{x}_\ell \odot (1 + \mathbf{e}_\ell)\right)$, where SiLU(.) is the swish function [208]. The resulting architecture is schematically illustrated in Fig. 7.2. Additional implementation details are provided in Appendix D.3.1. We emphasize that this new embedding module does not introduce significant additional computational overhead relative to those used for isotropic noise [47].

Training and sampling.

We now describe our implementation and training pipeline. We assume a covariance distribution $p(\mathbf{\Sigma})$ for training in which the two classes (spatial and frequency covariances) are selected with probability 0.5. In the spatial domain, we consider center-box and horizontal-box masks with sizes from 1 to $d^2 = 64^2$. In the spectral domain, we consider Gaussian deblurring and $\times 4$ super-resolution. Additional details regarding hyperparameters and examples of the different covariances are pro-

vided in Appendix D.3, and the full training and sampling algorithms are provided in Appendix D.1.1.

7.3.4 Validation: comparison to baselines

Experimental setting.

We compare our energy-based approach with several well-established methods [120]. Among plug-and-play baselines, we include the guidance methods DPS [2] and DAPS [209] and the variational inference-based RED-Diff [50]. We also include Palette [4] as a conditional model trained to solve a particular inverse problem. Details of each method are provided in Appendix D.3.4. We implement all methods, including our own, using the base architecture illustrated in Fig. 7.2. The differences lie in $p(\boldsymbol{\Sigma})$—plug-and-play models have $\boldsymbol{\Sigma} = \sigma^2\mathbf{I}$—and in the input provided to the model—conditional models stack the measurements with the input noisy image. Models were trained on CelebA (64×64) [210], ImageNet64 (64×64) [211], AFHQ-Cat (192×192) [212], and MNIST (28×28) [213].

We assess the reconstruction quality of the different samplers with mean squared error (in terms of PSNR $= -10 \log_{10}$ MSE) and two perceptual metrics, LPIPS [23] and DISTS [214]. Each metric is averaged over 400 examples with ground truth images from the test set. In addition, we compute the FID [24] between ensembles of test set images and posterior samples. All methods use up to $1,000$ neural function evaluations ($1,200$ for inpainting on CelebA). For each method, the number of steps is tuned via ablation and then fixed for all experiments; no ground-truth information is used at test time. This is especially relevant for RED-Diff, which performs better with fewer noise levels, consistent with [50].

Table 7.1 : Quantitative results for inpainting and Gaussian deblurring on **CelebA** 64×64. Best results are in **bold**.

Sampler	Inpainting				Deblurring			
	PSNR↑	**LPIPS↓**	**FID↓**	**DISTS↓**	**PSNR↑**	**LPIPS↓**	**FID↓**	**DISTS↓**
DPS [2]	16.10	0.110	36.76	0.19	32.98	0.004	43.01	0.08
RED-Diff [50]	**17.96**	0.100	47.82	0.17	**34.53**	0.006	44.33	0.08
DAPS [209]	17.25	0.098	45.76	0.16	30.86	0.005	65.41	0.10
Palette [4]	15.23	0.154	75.22	0.20	17.28	0.348	148.04	0.44
Ours	17.70	**0.093**	**34.57**	**0.14**	32.27	**0.002**	**41.87**	**0.04**

Table 7.2 : Quantitative results for inpainting and Gaussian deblurring on **ImageNet** 64×64. Best results are in **bold**.

Sampler	Inpainting				Deblurring			
	PSNR↑	**LPIPS↓**	**FID↓**	**DISTS↓**	**PSNR↑**	**LPIPS↓**	**FID↓**	**DISTS↓**
DPS [2]	23.01	0.051	55.61	0.114	29.10	0.007	59.09	0.10
RED-Diff [50]	**24.22**	0.065	58.50	0.135	**29.93**	0.008	63.10	0.11
DAPS [209]	23.87	**0.048**	54.07	0.113	27.54	0.013	79.43	0.15
Ours	22.90	0.052	**47.54**	**0.108**	29.10	**0.004**	**44.82**	**0.07**

Experimental results.

We consider the two linear inverse problems that were used to jointly train the model. To test the spatial covariance configuration, we perform inpainting on a square of size 45×45 in the center of the image, with $\sigma = 10^{-4}$. For the spectral configuration, we perform deblurring with a Gaussian kernel of size 8×8 and a standard deviation of 0.8, with $\sigma = 10^{-2}$. Results are shown in Tables 7.1 and 7.2 for CelebA and ImageNet respectively, and in Fig. 7.3 we illustrate a few inpainting examples on CelebA. We observe that our proposed sampler is competitive with, and in most cases outperforms, the baseline samplers. We include additional comparisons on AFHQ-Cat (192×192) [212] in Appendix D.4.1. Overall, these experiments demonstrate that energy models are fully competitive with or even improve on score models for solving linear inverse problems, and that a single trained model can be used to tackle a variety of inverse problems.

7.4 Applications of anisotropic energy models

Our energy model offers benefits beyond its use in solving linear inverse problems. We demonstrate in Section 7.4.1 that having access to the learned energy enables more refined comparisons between posterior sampling algorithms. In Sections 7.4.2 and 7.4.2, we develop samplers that leverage access to the energy to automatically define adaptive anisotropic schedules and use *unbiased* corrector steps via Metropolis-Hastings proposals. Finally, we show in Section 7.4.3 that our energy model can estimate unknown degradation operators and noise levels, enabling blind inverse problem solvers.

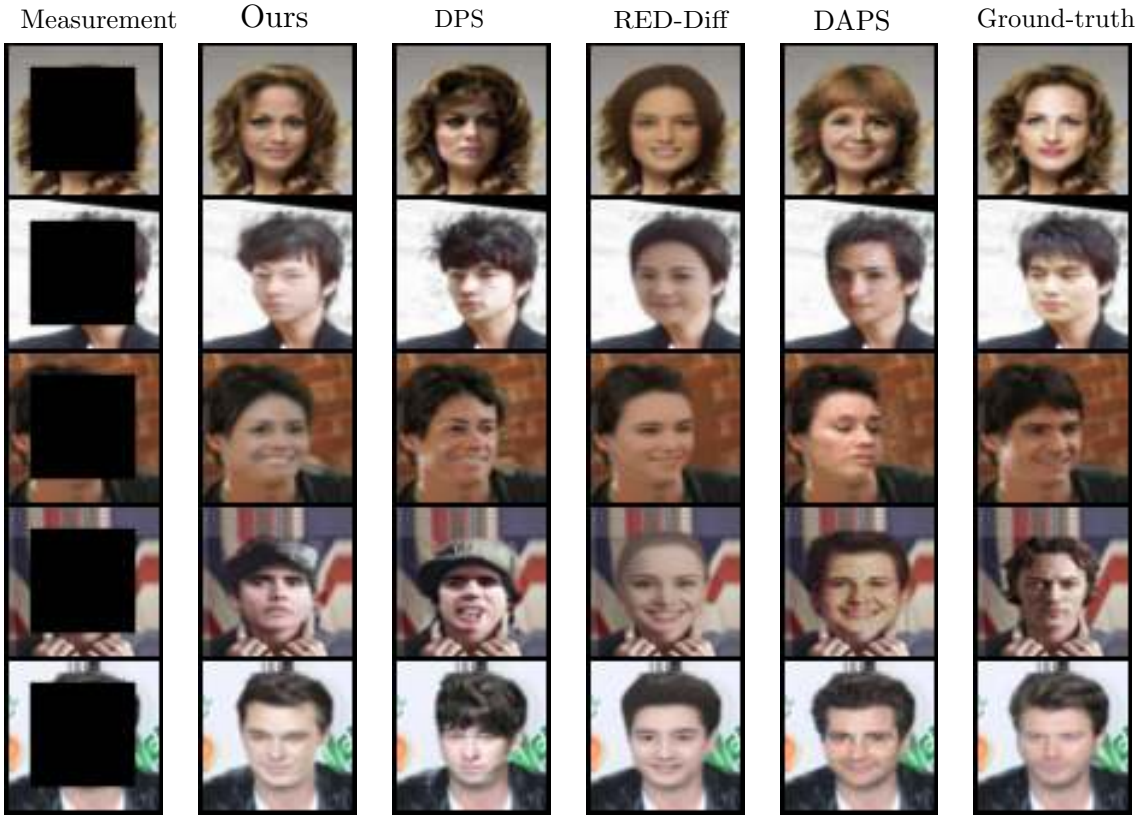


Figure 7.3 : Inpainting results on CelebA. DPS and our method generate sharper images while RED-Diff's solutions are smoother, which is consistent with our analysis of their posterior density values in Section 7.4.1.

7.4.1 The probabilities of posterior samples

In this section, we compare the prior and posterior probabilities of samples generated by three solvers, which include a plug-and-play model that relies on likelihood approximation (DPS), a variational inference procedure that approximates maximum-a-posteriori (MAP) optimization (RED-Diff), and a learned posterior sampler (ours). We fix one particular observation $\mathbf{y}$ and generate several potential solutions $\{\hat{\mathbf{x}}_i\}_{i=1}^N$ with each sampler. We then compute the probabilities of these samples under the prior $p(\mathbf{x})$ and the posterior $p(\mathbf{x}|\mathbf{y})$ (via Bayes rule) using our learned energy model. The resulting distributions of log probabilities are shown in Fig. 7.4. Examples of solu-

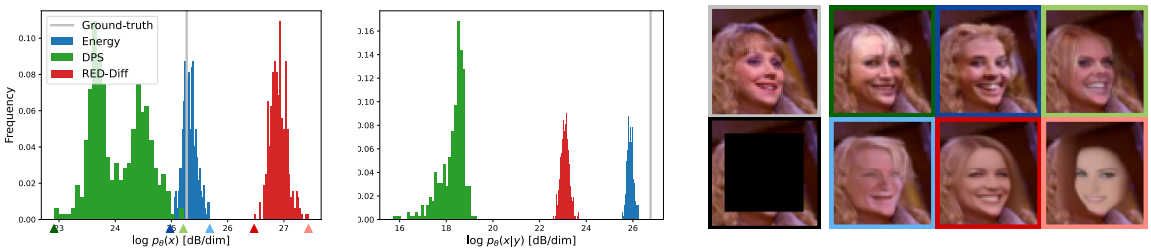


Figure 7.4 : Left and middle: Histograms of $\log p_\theta(\hat{\mathbf{x}})$ and $\log p_\theta(\hat{\mathbf{x}}|\mathbf{y})$ for inpainting solutions $\hat{\mathbf{x}}$ generated by DPS, RED-Diff, and our energy model from a given measurement $\mathbf{y}$, along with the ground truth $\mathbf{x}$. Our energy model is well-calibrated with respect to both prior and posterior probabilities. **Right:** Examples of generated images $\hat{\mathbf{x}}$ sorted from lowest to highest prior probability (in reading order). The colored border indicates the sampler used to produce the image (green for DPS, red for RED-Diff, blue for ours) and its prior probability (darker shades for lower probability images), matching the arrows in the left panel. Note that higher-probability images are smoother and less detailed. The ground truth and measurements are shown on the left with blue/black borders, respectively.

tions at different prior probabilities are shown in the right panel: generally, smoother images tend to have higher probability, while more detailed, textured images have lower probability. In particular, we see that DPS generates samples that are lower-probability under the prior due to their likelihood approximations, whereas RED-Diff generates higher-probability samples due to its maximum-a-posteriori behavior. Both methods produce samples of low posterior probability. Our energy-based sampler achieves a better balance: notably, the prior probabilities of solutions lie near the ground truth, and their posterior probability is only slightly lower than that of the ground truth. These results demonstrate that our anisotropic energy-based approach leads to accurate posterior samples that are consistent with both the prior and the measurements. A similar analysis across multiple observations $\mathbf{y}$ is provided in Appendix D.4.3.

7.4.2 Energy-based schedules and corrector steps

Energy-guided generation

An important feature of anisotropic diffusion is the freedom it allows in designing sampling paths [185, 195], as illustrated in Fig. 7.1. For any sequence of covariance steps $\boldsymbol{\delta\Sigma}_t = \boldsymbol{\Sigma}_t - \boldsymbol{\Sigma}_{t-1}$, a sample can be generated with the iterations

$$\begin{cases} \mathbf{x}_{t-1} = \mathbf{x}_t - \boldsymbol{\delta\Sigma}_t \nabla_{\mathbf{y}} U_\theta(\mathbf{x}_t, \boldsymbol{\Sigma}_t) + (\boldsymbol{\delta\Sigma}_t)^{\frac{1}{2}} \mathbf{v}_t \\ \boldsymbol{\Sigma}_{t-1} = \boldsymbol{\Sigma}_t - \boldsymbol{\delta\Sigma}_t \end{cases} \tag{7.11}$$

Designing samplers therefore reduces to selecting the *covariance step* $\boldsymbol{\delta\Sigma}_t$ at each iteration. After demonstrating this flexibility in the setting of any-order generation, we introduce an energy-guided sampler that selects $\delta\boldsymbol{\Sigma}_t$ automatically and adaptively by exploiting the variations of the energy with the noise covariance $\boldsymbol{\Sigma}$, leading to superior sampling quality.

Any-order generation.

We consider a family of block-diagonal covariances in pixel space. Each covariance from the family has a fixed variance on each $b \times b$ image patch, where b can take any value in $\{1, 2, 4, 7, 14, 28\}$ (see Fig. D.2 in Appendix D.3.2 for an illustration). We trained a model on MNIST with this set of covariances, where the noise variance in each patch was independent and log uniformly distributed in $[10^{-9}, 10^3]$. This allows generating image patches in any order in an autoregressive manner. We illustrate three different orders with different values of b in Fig. 7.5. Each path generates a different solution even with identical random seeds.

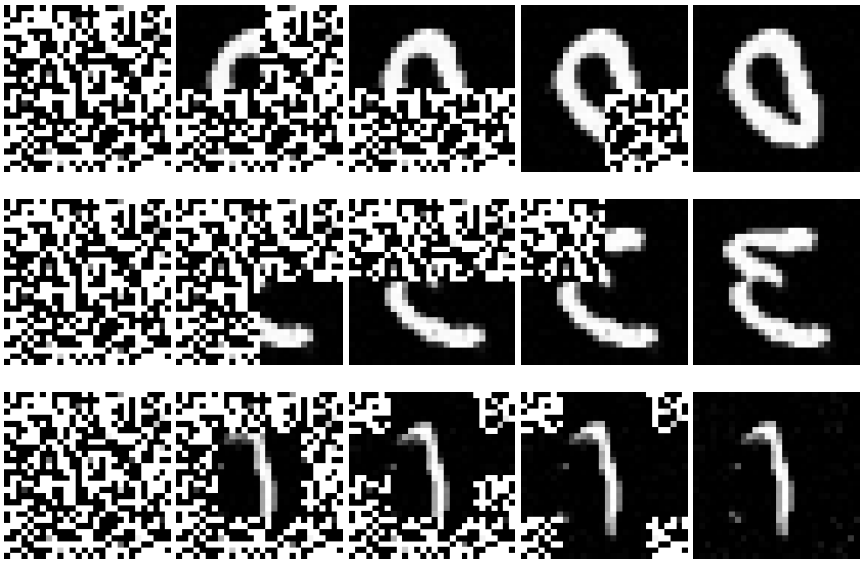


Figure 7.5 : Three autoregressive generations for a model trained on MNIST, with identical initial and injected noise. The first two rows generate the four quadrants of the image respectively in reading and reversed reading order. The third row generates the image in 16 patches, ordered from the center to the outer border.

Energy-guided adaptive sampler.

In theory, with a perfect score and an infinite number of sampling steps, all covariance schedules are equivalent. In practice, different schedules yield different approximations to the score and discretization errors. Which covariance paths lead to higher sampling quality, and how can we define them? We introduce a heuristically motivated *energy-guided* schedule that sets the covariance step $\boldsymbol{\delta\Sigma}_t$ adaptively by doing a descent step on the energy with respect to $\boldsymbol{\Sigma}_t$, using the covariance score $\nabla_{\boldsymbol{\Sigma}} U_\theta(\mathbf{x}_t, \boldsymbol{\Sigma}_t)$. Specifically, we set

$$\boldsymbol{\delta\Sigma}_t \propto \boldsymbol{\Sigma}_t \nabla_{\boldsymbol{\Sigma}} U_\theta(\mathbf{x}_t, \boldsymbol{\Sigma}_t) \boldsymbol{\Sigma}_t. \tag{7.12}$$

The multiplication by $\boldsymbol{\Sigma}_t$ on both sides can be thought of as a preconditioning of the gradient (notice that $\boldsymbol{\delta\Sigma}_t$ has the same units as $\boldsymbol{\Sigma}_t$) and can be motivated as a steepest descent in the Bregman geometry generated by $-\log\det\boldsymbol{\Sigma}_t$ (see Appendix D.2). 7.12 removes the need for choosing a discretization schedule during sampling and instead defines one automatically and adaptively to the current iterate $\mathbf{x}_t$, a property shared by the algorithm of [106] but in a different setting.

To illustrate the behavior of this sampler, we consider the task of reconstructing MNIST digits from a subset of k randomly selected pixels (unobserved pixels are replaced by high-variance noise). Fig. 7.6 shows the classification error over $1,000$ samples as a function of the number of measurements k (experimental details provided in Appendix D.3.1). Both samplers converge to clean-image baseline performance for large k, but the energy-guided sampler (7.12) achieves a better classification error than a geometric schedule $\boldsymbol{\delta\Sigma}_t \propto \boldsymbol{\Sigma}_t$ for smaller k. This illustrates the advantage of allowing the model to unmask different pixels at different rates in challenging inverse problems.

Preliminary experiments on a CelebA inpainting task indicate that the energy-guided sampler underperforms predefined schedules in this setting. We hypothesize that on more challenging tasks with more spatial variability, diagonal covariances in the spatial domain are not flexible enough to define meaningful paths. We believe that more general covariance spaces that are not restricted to a fixed basis should lead to gains similar to those we observe on MNIST, and leave exploration of those questions to future research.

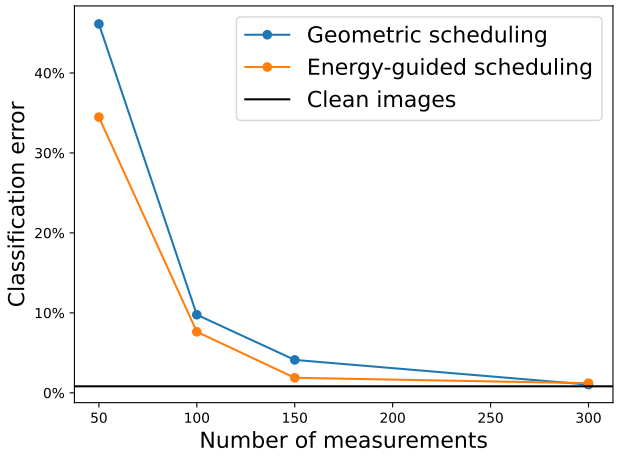


Figure 7.6 : Classification error as a function of the number of measurements. The energy-guided scheduling (7.12, orange) consistently achieves a lower error rate than the fixed geometric scheduling $\boldsymbol{\delta\Sigma} \propto \boldsymbol{\Sigma}$ (blue) with few measurements, with both methods reaching the baseline classifier error at 300 measurements.

Unbiased corrector steps with MALA

Typically, diffusion-based posterior samplers iterate between prediction and correction steps [7,49], where the correction is given by a version of the *Unadjusted Langevin*

Algorithm (ULA):

$$\mathbf{x}'_t = \mathbf{x}_t - \frac{\eta}{2}\left[\nabla_{\mathbf{y}} U_\theta(\mathbf{x}_t, \boldsymbol{\Sigma}_t) - \nabla_{\mathbf{x}_t} \log p(\mathbf{x}_T|\mathbf{x}_t)\right] + \sqrt{\eta}\mathbf{v}'_t. \tag{7.13}$$

where $\mathbf{x}_T = \mathbf{y}$ and $p(\mathbf{x}_T|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_T; \mathbf{x}_t, \boldsymbol{\Sigma}_T - \boldsymbol{\Sigma}_t)$. Notice that we need an additional term in the correction step as we aim to sample from $p(\mathbf{x}_t|\boldsymbol{\Sigma}_t, \mathbf{y})$ rather than just $p(\mathbf{x}_t|\boldsymbol{\Sigma}_t)$. Since the energy model provides $U_\theta(\mathbf{x}_t, \boldsymbol{\Sigma}_t) \approx -\log p(\mathbf{x}_t|\boldsymbol{\Sigma}_t)$, the likelihood term $\nabla_{\mathbf{x}_t} \log p(\mathbf{x}_T = \mathbf{y}|\mathbf{x}_t)$ acts as guidance to keep the iterates consistent with the observation $\mathbf{y}$ at covariance $\boldsymbol{\Sigma}_T$. It has been shown that using better MCMC samplers than ULA can boost the performance of diffusion-based samplers [48]. In this context, an advantage of our energy formulation is that it provides access to the density $p(\mathbf{x}_t|\boldsymbol{\Sigma}_t)$, in addition to the score, at every step. This enables the use of *unbiased* correctors such as the *Metropolis-Adjusted Langevin Algorithm* (MALA) [137], which treats (7.13) as a proposal $q(\mathbf{x}'_t|\mathbf{x})$ that is accepted with probability $\min\left(1, \frac{p(\mathbf{x}'_t|\boldsymbol{\Sigma}_T, \mathbf{y})\, q(\mathbf{x}_t|\mathbf{x}'_t)}{p(\mathbf{x}_t|\boldsymbol{\Sigma}_T, \mathbf{y})\, q(\mathbf{x}'_t|\mathbf{x}_t)}\right)$. We emphasize that the ability to compute this acceptance probability during *posterior* sampling is a distinctive feature of our anisotropic energy model, which explicitly provides access to the density $p(\mathbf{y}|\boldsymbol{\Sigma})$, setting it apart from prior isotropic energy-based diffusion approaches [48, 188].

We report in Table 7.3 a comparison between ULA and MALA as a function of the number of corrector steps. We observe that increasing the number of MALA steps consistently improves reconstruction quality as measured by LPIPS with the ground truth, whereas ULA does not yield further gains. A visual comparison is provided in Fig. D.22.

7.4.3 Blind inverse problems

We show that our normalized energy model can be used to solve *blind inverse problems*, where the noise covariance $\boldsymbol{\Sigma}$ (or equivalently, the measurement operator $\mathbf{H}$) is

Table 7.3 : LPIPS distance between ground truth $\mathbf{x}$ and reconstruction $\hat{\mathbf{x}}$ for different correction schemes on a CelebA inpainting task.

Sampler	**Number of corrector steps**		
	1	5	8
ULA corrector	0.093	0.093	0.093
MALA corrector	0.093	0.091	0.089

unknown. Our proposed strategy is straightforward: we estimate $\boldsymbol{\Sigma}$ by maximizing the posterior probability $\log p(\boldsymbol{\Sigma}|\mathbf{y}) = \log p(\mathbf{y}|\boldsymbol{\Sigma}) + \log p(\boldsymbol{\Sigma}) + \text{cst}$, given a prior on covariance matrices $p(\boldsymbol{\Sigma})$. Here, we choose a uniform prior over a set $\mathcal{S}$ of covariances, leading to

$$\hat{\boldsymbol{\Sigma}} = \underset{\boldsymbol{\Sigma}\in\mathcal{S}}{\operatorname{argmax}} \log p_\theta(\mathbf{y}|\boldsymbol{\Sigma}). \tag{7.14}$$

We illustrate this procedure on an inpainting task where the box size is unknown and is estimated through (7.14). More precisely, we observe $\mathbf{y} = \mathbf{x} + \boldsymbol{\Sigma}_s^{1/2}\mathbf{v}$, where $\boldsymbol{\Sigma}_s$ is a block-diagonal covariance with noise standard deviation σ_1 inside a central $s \times s$ box and $\sigma_2 = 10^{-4}$ outside of it. Fig. 7.7 shows that our energy model estimates accurately both the box size s and the noise level σ_1. This success critically relies on the normalization of the energy values across covariances $\boldsymbol{\Sigma}$: an energy model trained without the covariance score matching objective fails at this task.

7.5 Discussion

We introduced a framework for learning *normalized* conditional densities for linear inverse problems, built on Anisotropic Covariance Score Matching (A-CSM), a regularization that enforces consistency across covariance matrices via the Fokker–Planck equation. The resulting energy models match the reconstruction quality of stan-

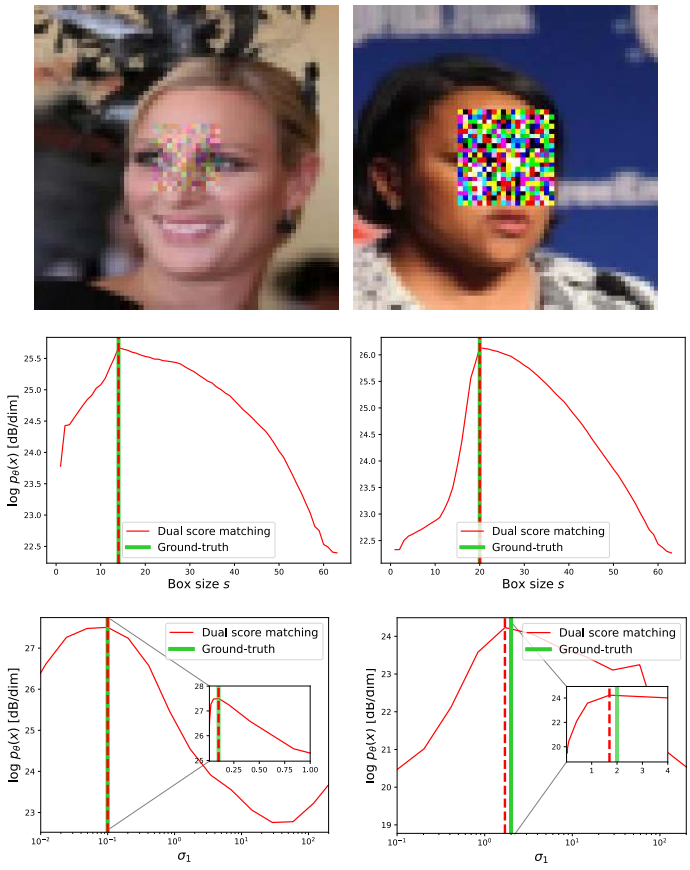


Figure 7.7 : Blind reconstruction experiment. **Top:** two observations corrupted by noise of standard deviation σ_1 inside a central $s \times s$ square, with $(\sigma_1, s) \in \{(0.1, 14), (2, 20)\}$. **Middle and bottom:** Log probability $\log p_\theta(\mathbf{y}|\mathbf{\Sigma}_s)$ as a function of box size s and σ_1, respectively. The noise covariance parameters estimated by the energy model trained with dual score matching (red dashed vertical lines) accurately track their ground truth values (solid green vertical lines).

dard diffusion solvers while unlocking capabilities the score-only literature could not deliver: blind degradation estimation, energy-guided adaptive sampling, unbiased MCMC correction, and principled comparisons across posterior samplers. Viewed alongside the plug-and-play/regression distinction of Section 7.2.2, our approach sits between the two: it retains the flexibility of a prior-plus-likelihood factorization (through Bayes' rule) while inheriting the trainability of direct-regression methods (through the Fokker–Planck consistency the A-CSM loss enforces).

Two chapter-specific limitations are worth flagging. First, training with the dual (A-DSM + A-CSM) objective is more computationally intensive than plain score matching; sliced score matching [215] could replace the extra backpropagation with forward-mode Jacobian–vector products. Second, our diagonal-covariance parameterization (spatial or spectral) covers a range of degradations but does not exhaust them: extending to general covariances would broaden applicability at additional memory and compute cost. Beyond these, promising uses of learned normalized densities include measurement design via posterior entropy or mutual information (a more direct alternative to [216]), and estimation of local curvature or "tangent subspaces" of the prior in the vicinity of individual images. Chapter 9 discusses broader cross-part combinations.

Chapter 8

Flow map denoisers: traversing the distortion–perception plane

Modern flow-matching and diffusion models are trained to learn the *instantaneous velocity* of a stochastic interpolant, forcing many-step integration at inference and, in the inverse-problem setting, a hard trade-off between distortion and perception at every function-evaluation budget. *Flow maps* instead learn an *average velocity* between arbitrary time pairs, transporting samples in a single evaluation. In this chapter we train a flow map and use its average/two-time *denoiser* form as a plug-and-play denoiser for image inverse problems. Our main observation is that the lookahead of the trained flow-map denoiser directly indexes a continuous family of estimators along the distortion–perception plane: the lookahead becomes a *knob* that traverses the DP frontier with a single trained model. We characterize this frontier theoretically in the Gaussian case, validate the plug-and-play sampler on image inverse problems, and show that flow-map denoisers dominate their diffusion-based counterparts across the plane.

This chapter is adapted from [14].

8.1 Introduction

Image restoration is highly ill-posed: a single degraded observation corresponds to a vast set of plausible clean images, and reconstruction methods must therefore navigate

the distortion–perception (DP) trade-off formalized in Section 2.1.2 [1, 26]. Where a reconstruction lands on the DP plane is application-dependent: medical imaging demands low distortion, creative photography favours high perceptual quality, and safety-critical settings may need something in between. This chapter develops a solver that traces out the DP plane at inference time using *a single trained model*, pulling the "training target for velocity" knob of Section 2.7 to swap the instantaneous velocity of standard flow matching for an average velocity.

Most existing solvers commit to a single operating point on the DP frontier. Regression-based methods target minimum distortion [60, 203, 217]. Generative posterior samplers [2, 63, 106, 218] target minimum perceptual divergence at the cost of higher per-pixel error. A few recent works based on data-dependent flow models attempt to traverse the frontier [164, 166, 175] by explicitly interpolating between a conditional-mean estimator and the clean image, but the traversal relies on external mechanisms — switching between models, adjusting discretization, or retraining — rather than being intrinsic to a single learned object.

In this chapter we take a different approach: we *train* a flow map [56–58, 219] for image restoration and observe that its two-time structure already encodes a continuum of estimators along the DP plane. While the flow-map literature so far has focused on accelerating sampling [220–222], we identify the *lookahead* t of a trained flow map as a knob over an *average denoiser*: an average/two-time generalization of the Tweedie denoiser that predicts the clean signal from a noisy input at time s using lookahead t. Varying t interpolates between the MMSE estimator (at $t = s$) and a posterior sample (at $t \to 1$); for Gaussian targets we prove that intermediate lookaheads trace the *optimal* DP curve of (2.6). Embedded in a Plug-and-Play (PnP) framework [44], this yields a single-model solver that spans the DP plane on CelebA

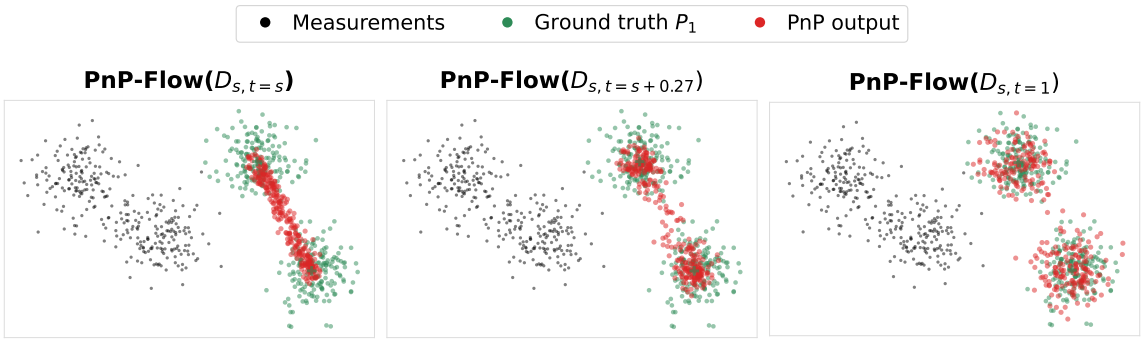


Figure 8.1 : PnP with average denoisers on a 2D mixture-of-Gaussians inverse problem ($\mathbf{y} = \mathbf{H}\mathbf{x} + \sigma\boldsymbol{\epsilon}$, with $\mathbf{H}$ a rotation+scaling operator and $\sigma = 0.3$). $D_{s,t}$ denotes the average denoiser (defined formally in Section 8.3.1). The lookahead t controls the DP trade-off: at $t = s$ the denoiser is MMSE and PnP reconstructions concentrate at the posterior mean, which for this symmetric bimodal posterior lies between the two modes, yielding low distortion but poor perceptual quality; as $t \to 1$, the lookahead progressively restores the bimodal structure, recovering samples from both modes at the cost of higher distortion; details of this experiment are in Appendix E.3.4.

and AFHQ inverse problems — inpainting, motion deblurring, super-resolution, and Gaussian deblurring — matching specialized baselines at both endpoints and uniquely tracing a smooth curve between them. Fig. 8.1 illustrates this behavior on a 2D Gaussian mixture model.

Notation.

Bold symbols (e.g., $\mathbf{x}$) denote vectors, and bold uppercase symbols (e.g., $\mathbf{H}$) denote matrices. Mappings such as the flow map $X_{s,t}$ and denoisers $D_{s,t}$ are written in standard (non-bold) font.

8.2 Background

8.2.1 Flow matching and flow maps

Stochastic interpolants.

The flow-matching framework is developed in the beginning of Part II and Section II. Throughout this chapter, unless stated otherwise, we work with the linear interpolant and denote the trained network by $\mathbf{v}_\theta(\mathbf{x}, t)$. For notational consistency with the flow-map generalization below, we retain the instantaneous-denoiser label (5.14) as (8.1) locally:

$$D_t(\mathbf{x}) := \mathbb{E}[\mathbf{x}_1 \mid \mathbf{x}_t = \mathbf{x}] = \mathbf{x} + (1-t)\mathbf{v}(\mathbf{x}, t). \tag{8.1}$$

A detailed characterization of denoisers can be found in [29].

Flow maps.

Sampling from p_1 using the learned velocity field requires numerical integration of the ODE, typically with many function evaluations. Flow maps [56–58] bypass this integration by training a model to directly predict the average over a finite interval $[s, t]$:

$$X_{s,t}(\mathbf{x}) = \mathbf{x} + (t-s)\,\mathbf{v}(\mathbf{x}, s, t), \tag{8.2}$$

where $\mathbf{v}(\mathbf{x}, s, t)$ is the *average velocity* over the interval $[s, t]$. A network $\mathbf{v}_\theta(x, s, t)$ is trained to predict this average velocity, conditioned on both the current time s and the target time t. While there are different alternatives for the loss for training flow maps [56], in this work we consider the Lagrangian self-distillation (LSD) given by

$$\mathcal{L}_{\mathrm{LSD}}(\hat{v}) = \int_0^1 \int_0^t \mathbb{E}_{\mathbf{x}_0, \mathbf{x}_1} \left[\left\| \partial_t \hat{X}_{s,t}\left(\mathbf{x}_s\right) - \mathbf{v}_\theta\left(\hat{X}_{s,t}\left(\mathbf{x}_s\right), t\right) \right\|^2 \right] ds dt. \tag{8.3}$$

We consider the LSD as it achieved the most stable training in practice and a good visual result; we include further background in Appendix E.1.1.

8.2.2 Inverse problems and Plug-and-Play methods

We adopt the general forward model $\mathbf{y} = \mathcal{A}(\mathbf{x}) + \mathbf{w}$ with $\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \sigma_0^2 \mathbf{I})$ introduced in Section 2.1 (Eq. (2.1)), where $\mathcal{A}(\cdot)$ is a known, possibly non-linear forward operator. Section 2.6 reviewed the two dominant families of methods that combine flow or diffusion priors with data-consistency updates — guidance-based methods and plug-and-play (optimization-based) methods — and their point-estimate, projection-based, variational, and asymptotically-exact instantiations. This chapter builds on the plug-and-play/variational family (see Section 2.6).

Forward–backward splitting and PnP.

Plug-and-Play (PnP) algorithms [43, 44] were originally derived from proximal splitting for problems of the form $\min_{\mathbf{x}} g(\mathbf{x}) + h(\mathbf{x})$, where g is a data-fidelity term and h a regularizer. Forward–backward splitting (FBS) alternates between a gradient step on g and a proximal step on h:

$$\mathbf{z}^k = \mathbf{x}^k - \lambda_t \nabla g(\mathbf{x}^k), \tag{8.4}$$

$$\mathbf{x}^{k+1} = \mathrm{prox}_{\lambda h}(\mathbf{z}^k), \qquad \mathrm{prox}_{\lambda h}(\mathbf{z}) := \arg\min_{\mathbf{x}} \tfrac{1}{2}\|\mathbf{x} - \mathbf{z}\|^2 + \lambda h(\mathbf{x}). \tag{8.5}$$

Since the proximal operator of a regularizer is itself a denoising problem, PnP replaces $\mathrm{prox}_{\lambda h}$ with a learned denoiser D_t [10, 50, 122, 223–227] — concretely, the Tweedie denoiser of a pre-trained diffusion or flow model (Section 2.4, Eq. (2.16)). Existing instantiations typically optimize distortion at the expense of perceptual quality, producing over-smoothed reconstructions. Section 8.3 shows that replacing the standard *instantaneous* denoiser with the flow map's *average* denoiser turns this fixed operating point into a continuous DP-plane knob.

8.2.3 Distortion–Perception Trade-off

The formal Blau–Michaeli [26] and Freirich [1] characterization of the distortion–perception (DP) trade-off, including the closed-form optimal curve (2.6) for MSE distortion with W_2 perceptual divergence and the Pareto-optimal linear interpolation (2.7) between the MMSE estimator $\mathbf{x}^\star$ and any perfectly perceptual estimator, is developed in Section 2.1.2. Existing approaches to traversing this frontier require explicit interpolation between multiple models or task-specific retraining to access different operating points [164, 166]. This chapter seeks a *single* model that directly parameterizes (2.7), enabling continuous traversal of the DP plane at inference time.

8.3 Average denoisers: traversing the DP plane

In this section we define and analyze the *average denoiser* associated to flow maps and its connection to the DP trade-off (Section 8.3.1).

8.3.1 Flow maps as denoisers

Recall we seek a family of estimators parametrized by a *single model* that interpolates between P^* (lowest distortion) and $P = 0$ (lowest perception). A natural candidate is the denoiser implicitly defined by the flow map (Section 8.2.1). The same object was independently introduced by [177] for one-step language modeling on discrete data; here we study it in the continuous setting and show it parametrizes the DP frontier.

Definition 8.1 (Average denoiser) Given a flow map with average velocity $\mathbf{v}(\cdot, s, t)$*, the* average denoiser *(or two-time denoiser) is:*

$$D_{s,t}(\mathbf{x}) := \mathbf{x} + (1 - s)\mathbf{v}(\mathbf{x}, s, t). \tag{8.6}$$

This definition has the same form as the instantaneous denoiser (8.1), but replaces the instantaneous velocity $\mathbf{v}(\cdot, s)$ with the average velocity $\mathbf{v}(\cdot, s, t)$ over the interval $[s, t]$. This modification has an important consequence: the parameter t now controls where $D_{s,t}(\mathbf{x})$ sits on the DP plane, i.e., varying t traces a continuous path between the low-distortion and low-perception extremes. Importantly, both s and t are inputs of $\mathbf{v}_\theta(\mathbf{x}, s, t)$, so a *single trained model* produces a whole family of denoisers with different characteristics, as we show in the following section.

8.3.2 Traversing the DP plane with flow maps

Warm-up: Gaussian case (1D analysis)

We start with the Gaussian case, which renders a closed-form expression of the average denoiser in (8.6). For clarity, we derive the results in the scalar case ($x \in \mathbb{R}$); all results extend directly to the multivariate setting by applying the analysis component-wise. Let $p_1 = \mathcal{N}(0, \sigma_p^2)$, with $\sigma_p^2 \leq 1$; following the stochastic interpolant formulation in Section 8.2.1, the marginal variance at noise level t is $\sigma_t^2 = t^2\sigma_p^2 + (1-t)^2$, and the instantaneous denoiser is *linear*, given by $\mathbb{E}[x_1 \mid x_t = x] = A_t x$, with $A_t = t\sigma_p^2/\sigma_t^2$. Using Tweedie's relationship in (8.1), we also have a linear instantaneous velocity field $v(x, t) = B_t x$ with $B_t = [t(\sigma_p^2 + 1) - 1]/\sigma_t^2$.

Flow map and average denoiser.

The linearity of the instantaneous velocity yields a scalar flow map $x_t = \Phi(t, s)\, x_s$ with $\Phi(t, s) = \sigma_t/\sigma_s$, obtained by integrating $\int_s^t B_\tau \, d\tau = \log(\sigma_t/\sigma_s)$ (see Appendix E.2.2). Following Definition 8.1, the average denoiser reduces to $D_{s,t}(x_s) = \Lambda(s, t)\, x_s$ where:

$$\Lambda(s, t) = \frac{(1-s)\sigma_t - (1-t)\sigma_s}{\sigma_s(t-s)}. \tag{8.7}$$

This gain satisfies $\Lambda(s,s) = A_s$ (lowest distortion) and $\Lambda(s,1) = \sigma_p/\sigma_s$ (lowest perception), and is strictly increasing in t (see Appendix E.2.2).

Optimal DP estimators in the Gaussian case.

For a Gaussian target, the optimal DP estimator in (2.7) is also linear, given by $\hat{x}_P = \Gamma(\alpha)\, x_s$, with the gain given by

$$\Gamma(\alpha) = \alpha\, A_s + (1-\alpha)\, \sigma_p/\sigma_s. \tag{8.8}$$

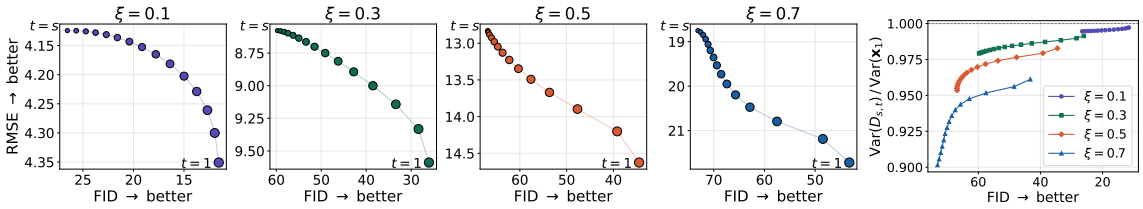
The parameter $\alpha = P/P^* \in [0,1]$ parametrizes the perception level, interpolating linearly between the MMSE gain A_s and the best perception gain σ_p/σ_s. Given this particular instance of the DP estimator, we can establish a direct relationship between the flow map and the DP estimator in the Gaussian case:

Theorem 2 (Exact optimality) *Let $p_1 = \mathcal{N}(0, \sigma_p^2)$ with $\sigma_p^2 \leq 1$, and let $\Lambda(s,t)$ be the average denoiser gain (8.7) associated with the true flow map. Then for every $s \in (0,1)$ and $t \in [s,1]$, there exists a perception level*

$$\alpha(s,t) = \frac{\sigma_p/\sigma_s - \Lambda(s,t)}{\sigma_p/\sigma_s - A_s} \in [0,1] \tag{8.9}$$

such that $\Lambda(s,t) = \Gamma(\alpha(s,t))$. Moreover, $\alpha(s,t)$ decreases monotonically from 1 at $t = s$ (MMSE, $P = P^*$) to 0 at $t = 1$ (perfect perception, $P = 0$). Consequently, the average denoiser $D_{s,t}$ traces the optimal DP curve exactly as t varies in $[s,1]$.

The proof is deferred to Appendix E.2.2. In summary, both $\Lambda(s,t)$ and the optimal DP gain $\Gamma(\alpha)$ are continuous monotonic functions that share the same endpoints: A_s (MMSE, lowest distortion) and σ_p/σ_s (lowest perception). Hence, inverting the affine relation $\Gamma(\alpha) = \Lambda(s,t)$ yields a bijection $\alpha(s,t)$ between the lookahead and the perception level, so the average denoiser exactly realizes the optimal DP estimator



(a) DP trade-off curves **(b)** Var. vs FID

Figure 8.2 : Analysis of the perception-distortion DP trade-off and variance restoration on CelebA 128×128, quantified as RMSE/Var ratio vs FID. **(a)** DP trade-off at different noise levels. Each curve is swept by varying the lookahead parameter t using a single trained model. **(b)** The near-collapse of curves across noise levels indicates a universal relationship between variance restoration and perceptual quality (FID) improvement.

at every intermediate operating point. We now extend this insight to more general distributions.

8.3.3 Beyond the Gaussian case: image denoising experiments

For general image distributions, the average denoiser no longer admits a closed-form characterization. While exact DP optimality is not expected beyond the Gaussian case, the structural insight underlying Theorem 2 should still hold: as t varies from s to 1, the output of $D_{s,t}$ is expected to interpolate smoothly between the MMSE estimator (low distortion, low perception quality) and a perceptually-aligned estimator (higher distortion, improved perception). Consequently, we expect (i) smooth, monotonic DP curves parameterized by t, and (ii) convex trajectories consistent with the optimal DP function (Section 8.2.3), which we verify empirically below. While this section focuses on image data, we further support this intuition for a 2D mixture of Gaussians (MoG) in Appendix E.2.3, where the DP curve can be computed explicitly.

Setup.

We train a flow map network $\mathbf{v}_\theta(\mathbf{x}, s, t)$ on CelebA 128×128 (Appendix E.3) and evaluate $D_{s,t}(\mathbf{x}_s)$ at multiple lookahead values $t \in [s, 1]$, where $t = s$ recovers the MMSE denoiser D_s and $t = 1$ reaches the endpoint of the flow. Since the stochastic interpolant (5.10) at time s takes the form $\mathbf{x}_s = s\,\mathbf{x}_1 + (1 - s)\,\boldsymbol{\epsilon}$ with $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I})$, applying the average denoiser $D_{s,t}$ to $\mathbf{x}_s$ corresponds to Gaussian denoising: recovering the clean signal $\mathbf{x}_1$ from an observation where the signal is scaled by s and corrupted by noise of standard deviation $1-s$. We parameterize the noise level by $\xi := 1-s \in (0, 1)$, where larger ξ means heavier noise.

DP traversal.

Fig. 8.2a shows the empirical DP curves obtained by sweeping the lookahead t from s (MMSE) to 1 (full perception) across four noise levels $\xi \in \{0.1, 0.3, 0.5, 0.7\}$. As expected, the average denoiser traces smooth, monotone curves in the DP plane: increasing t improves perception (lower FID) at the cost of distortion (higher RMSE). The resulting curves exhibit the convex shape predicted by the optimal DP function. Unlike prior methods that provide only discrete or heuristic control, our approach yields a smooth and continuous parameterization of the DP frontier from a single trained model.

We also observe that the traversal range increases with the noise level. For $\xi = 0.1$, the range is narrow since the MMSE estimator is already close to the data distribution, whereas for $\xi = 0.7$, the lookahead spans a wide range of operating points. This is consistent with [1], as the perception gap P^* grows with noise, allowing a larger trade-off between distortion and perception. Fig. E.1 illustrates this effect: as t increases from s to 1, reconstructions transition from smooth averages to sharp, detailed images.

Variance restoration.

A natural question is *why* increasing the lookahead improves perceptual quality. A simple explanation follows from the law of total variance: for any noisy observation $\mathbf{x}_s$, we have $\mathrm{Var}(\mathbf{x}_1) = \mathrm{Var}\big(\mathbb{E}[\mathbf{x}_1 \mid \mathbf{x}_s]\big) + \mathbb{E}\big[\mathrm{Var}(\mathbf{x}_1 \mid \mathbf{x}_s)\big]$. Since the second term is strictly positive, the MMSE estimator $D_s(\mathbf{x}_s) = \mathbb{E}[\mathbf{x}_1 \mid \mathbf{x}_s]$ has lower variance than the true data, i.e., $\mathrm{Var}(D_s(\mathbf{x}_s)) < \mathrm{Var}(\mathbf{x}_1)$. This *variance gap* leads to reduced contrast and over-smoothed reconstructions, and is precisely what the optimal DP estimator (2.7) corrects via interpolation. As the lookahead t increases, the average denoiser progressively restores this variance, effectively traversing the same trade-off within a single model. Fig. 8.2b supports this empirically: the relationship between variance ratio and FID is nearly identical across noise levels, suggesting that variance restoration underlies the observed perceptual improvements.

8.4 Plug-and-play with average denoisers

We now embed the average denoiser (8.6) into the PnP framework from Section 8.2.2 to solve general inverse problems with continuous DP control. Importantly, the DP interpretation applies at the level of the denoiser $D_{s,t}$, while the overall PnP solution reflects its interaction with the forward model. As a result, in general inverse problems, the lookahead controls a trade-off that combines perceptual alignment with data-consistency effects, rather than exactly reproducing the denoising DP frontier.

Algorithm.

Given a trained flow map $\mathbf{v}_\theta(\cdot, s, t)$, observations $\mathbf{y}$, and a chosen lookahead $t \in [s, 1]$, our method alternates between a gradient descent step on the data-fidelity term

$g(\mathbf{x}) = \frac{1}{2}\|\mathbf{y} - \mathcal{A}(\mathbf{x})\|^2$ and a proximal step utilizing the average denoiser. Following [223], we include a renoising step prior to applying the denoiser. This reprojects the gradient-updated iterate onto the support of the distribution at noise level s, ensuring that the average denoiser $D_{s,t}$ processes inputs at its intended operational noise level. The complete procedure is summarized in Algorithm 6.

Algorithm 6 PnP with Flow Map Denoisers

Require: Trained flow map $\mathbf{v}_\theta(\cdot, s, t)$, observations $\mathbf{y}$, forward operator $\mathcal{A}(\cdot)$, noise levels $\{s_k\}_{k=0}^{K-1}$, lookahead $t_k \in [s_k, 1]$, step sizes $\{\lambda_k\}_{k=0}^{K-1}$, iterations K

Initialize $\mathbf{x}^0$

for $k = 0, \ldots, K-1$ **do**

  $\mathbf{z}^k = \mathbf{x}^k - \lambda_k \nabla g(\mathbf{x}^k)$ ▷ Gradient step (data fidelity)

  Sample $\epsilon \sim \mathcal{N}(0, \mathbf{I})$

  $\tilde{\mathbf{z}}^k = (1 - s_k)\,\epsilon + s_k\,\mathbf{z}^k$ ▷ Renoising (reproject to noise level s)

  $\mathbf{x}^{k+1} = \tilde{\mathbf{z}}^k + (1 - s_k)\,\mathbf{v}_\theta(\tilde{\mathbf{z}}^k, s_k, t_k)$ ▷ Flow map denoiser

end for

return $\mathbf{x}^K$

Role of the renoising step.

The stochastic renoising step is what makes the lookahead a DP knob in the inverse-problem setting. With renoising, $D_{s,t}$ operates at its intended noise level s, placing it in the regime where Theorem 2 characterizes its behavior, and the injected noise restores the sample diversity required at the perceptual end of the frontier. Without renoising, the iteration reduces to a fixed-point Tikhonov estimator and the lookahead degenerates into a bias-variance knob; we treat this case in Appendix E.2.4.

Computational cost and connection to PnP-Flow.

Algorithm 6 requires one forward pass through $\mathbf{v}_\theta$ per iteration. The lookahead t is merely a conditioning input, adding zero overhead and preserving efficiency over methods requiring ODE backpropagation or trace computations.

8.5 Related work

Consistency models and flow maps for inverse problems.

Consistency models [228] learn to map any point on a probability flow ODE to its origin, enabling single-step generation. Consistency trajectory models [229] and flow maps [57] generalize this to arbitrary pairs (s, t) by learning the ODE solution operator. More recent works repurposed these models for controlled generation, either as guided samplers [221, 222, 230, 231], as priors [220], or for optimizing the noise initialization [232]. Most recently, and concurrently to this work, [233] also leverage flow maps for guidance, formulating reward-based generation as an optimal control problem and using the flow map for fast guided sampling. However, all these approaches primarily target accelerated sampling at a fixed operating point and do not provide explicit control over the restoration trade-off. In contrast, we treat the lookahead t as a continuous parameter, turning a single flow map into a family of estimators that spans the entire distortion–perception (DP) frontier.

Methods that traverse the DP plane.

Recent approaches explore DP control by learning data-dependent flows between degraded and clean distributions. Inversion by Direct Iteration (InDI) [166] and stochastic interpolants [175] learn flows from measurements to clean images, while

Posterior-mean Rectified Flow [164] replaces the input with an approximation of the MMSE estimator. These methods traverse the DP plane by varying the number of integration steps. However, they require supervised training, are typically tied to a specific degradation, and may rely on multiple models (e.g., a separate MMSE estimator). In contrast, our method uses a *single*, degradation-agnostic model that continuously parameterizes the DP frontier via the lookahead t, without retraining or auxiliary networks, and generalizes to arbitrary forward operators.

8.6 Numerical results

We now focus on the numerical results. Throughout the experiments, we aim to (i) demonstrate competitive performance with state-of-the-art methods, and (ii) show that our method uniquely enables continuous traversal of the DP plane with a single model.

Setup.

We evaluate all methods in two datasets: CelebA 128×128 and AFHQ 256×256; we also consider AFHQ-cats 192×192. For each dataset, we train a flow map model $\mathbf{v}_\theta(\mathbf{x}, s, t)$, parametrized by the Song UNet architecture [45, 234], from scratch using the off-diagonal loss of LSD described in (8.3). More training details can be found in Appendix E.3.1. We assess the reconstruction quality of the different samplers with mean squared error (in terms of PSNR $= -10 \log_{10}$ MSE) and two perceptual metrics, LPIPS [23] and DISTS [214]. In addition, we compute the FID [24] between ensembles of test set images and posterior samples. We compute the average of 100 samples, and for all the methods we consider Gaussian noise with $\sigma = 0.05$, unless random inpainting with $\sigma = 0.01$; also, in Appendix E.4.2 we include an experiment

Table 8.1 : Quantitative results for random-mask inpainting (90%, $\sigma = 0.01$) and motion deblurring ($\sigma = 0.05$) on **CelebA** 128×128. Best results are in **bold**; second-best underlined.

Sampler	Inpainting				Motion Deblurring			
	PSNR↑	**LPIPS↓**	**FID↓**	**DISTS↓**	**PSNR↑**	**LPIPS↓**	**FID↓**	**DISTS↓**
Flow-priors	26.50	0.089	59.43	0.132	25.07	0.151	114.41	0.188
D-Flow	<u>26.55</u>	<u>0.066</u>	50.27	0.110	<u>29.77</u>	**0.071**	73.15	**0.112**
OT-ODE	22.16	0.158	84.92	0.179	21.07	0.198	106.78	0.202
DPS-ODE	25.60	0.070	**45.74**	**0.106**	27.36	0.101	57.02	0.133
PnP-Flow ($t = s$)	**27.29**	0.073	48.88	0.113	**29.79**	0.114	<u>53.65</u>	0.130
PnP-Flow ($t = 1$)	26.03	**0.065**	<u>46.79</u>	<u>0.109</u>	29.42	<u>0.082</u>	**52.61**	<u>0.119</u>

Table 8.2 : Quantitative results for random-mask inpainting (90%, $\sigma = 0.01$) and motion deblurring ($\sigma = 0.05$) on **AFHQ** 256×256. Best results are in **bold**; second-best underlined.

Sampler	Inpainting				Motion Deblurring			
	PSNR↑	**LPIPS↓**	**FID↓**	**DISTS↓**	**PSNR↑**	**LPIPS↓**	**FID↓**	**DISTS↓**
Flow-priors	<u>23.94</u>	0.193	18.96	0.164	23.14	**0.207**	27.73	<u>0.154</u>
D-Flow	22.80	0.192	18.31	0.137	**25.52**	<u>0.230</u>	<u>19.92</u>	0.156
OT-ODE	21.14	0.263	16.03	0.181	16.73	0.348	77.53	0.234
DPS-ODE	23.08	<u>0.127</u>	**8.69**	<u>0.098</u>	20.49	0.292	36.71	0.179
PnP-Flow ($t = s$)	**24.99**	0.162	13.49	0.135	<u>25.40</u>	0.347	20.37	0.201
PnP-Flow ($t = 1$)	23.23	**0.122**	<u>9.11</u>	**0.095**	23.31	**0.207**	**14.04**	**0.137**

with Poisson noise.

We compared with recent methods that leverage pre-trained flow models, namely OT-ODE [218], Flow Priors [235], D-Flow [236], DPS-ODE [2] and PnP-Flow [223], which corresponds to our proposed method for $t = s$. For all the models, we performed a search grid of hyperparameters to obtain the best performance possible, and we use the same trained network for the prior/regularizer; we defer to Appendix E.3.2 for further details.

8.6.1 Inverse problems

We evaluate on the following tasks: (i) motion deblurring with random kernel generated with the code* and size 31×31 (CelebA) and 61×61 (AFHQ) and $\sigma_b = 1$; (ii) random inpainting with 90% missing pixels; (iii) super-resolution ($\times 4$); and (iv) Gaussian deblurring, which are deferred to Appendix E.4.2.

Quantitative results are reported in Tables 8.1 and 8.2 for CelebA and AFHQ respectively (with error bars in Appendix E.4.1), and visual results in Appendix E.4.7. PnP-Flow with $t = s$ achieves the best or near-best PSNR across all four (dataset, task) settings, confirming that the average denoiser at small lookaheads behaves as a low-distortion estimator. At the other extreme, PnP-Flow with $t = 1$ achieves the best or near-best LPIPS, DISTS, and FID on every task; on inpainting, it matches DPS-ODE on FID while substantially improving LPIPS and DISTS. Competing methods either commit to a single operating point (DPS-ODE favors perception, D-Flow favors distortion) or fall short on both, whereas our approach attains both regimes from a single trained network. Beyond accuracy, PnP-Flow is also the fastest method among those compared as shown in Appendix E.3.3.

*https://github.com/LeviBorodenko/motionblur

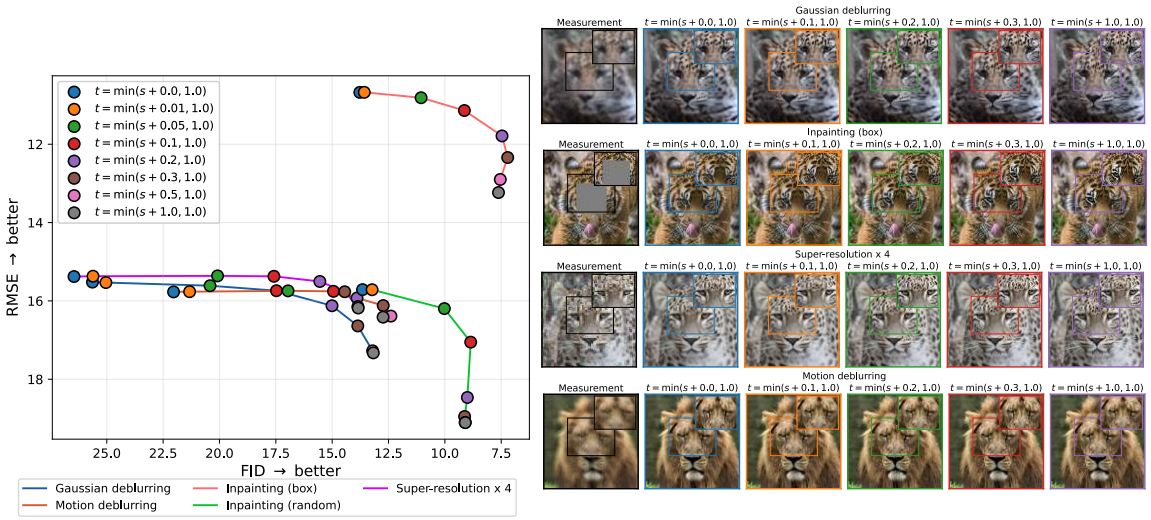


Figure 8.3 : Across five linear inverse problems on AFHQ (Gaussian/motion deblurring, box/random inpainting, SR×4), we sweep the lookahead t within a PnP solver and report RMSE (distortion) versus FID (perception). Each marker corresponds to a value of t, with small lookaheads favoring low distortion and large ones favoring low FID. A single trained model spans the full frontier across all tasks, without retraining or guidance tuning; reconstructions on the right illustrate the perceptual change as t increases (larger version in Fig. E.27).

8.6.2 Analysis of the DP trade-off

We now analyze how the lookahead parameter t controls the characteristics of our reconstructions by moving in the DP plane. We evaluate the average denoiser $\{D_{s,t}\}_t$ at multiple lookahead values within a PnP solver and report RMSE (distortion) against FID (perception); for box inpainting, we use a mask of size 80×80.

Results are in Fig. 8.3, and three observations stand out. First, varying t produces a smooth, (almost) monotonic trajectory in the DP plane: small lookaheads yield minimum RMSE, large lookaheads minimum FID, and intermediate values smoothly

interpolate between the two. This empirically confirms our analysis from Section 8.3.2 and shows that the behavior extends to natural images and general degradations. Second, the same qualitative trend appears across all five inverse problems — Gaussian deblurring, motion deblurring, random/box inpainting, and super-resolution — despite the very different forward operators, suggesting that the DP control given by t is degradation-agnostic. Third, no single t is universally best: the optimal lookahead depends on the user's preference between distortion and perception, which is precisely the use case our framework is designed to support.

Non-linear inverse problems.

Finally, we evaluate the DP trade-off on JPEG compression, a non-linear degradation, with quantization factor 10. We study the effect of the number of PnP iterations; the results are shown in Fig. 8.4 for AFHQ; see Appendix E.4.2 for comparison with baselines. PnP with flow maps consistently achieves better perceptual quality. However, increasing the number of iterations eventually degrades FID, consistent with observations in other inverse problems.

8.7 Discussion

We introduced *flow-map denoisers*, an average/two-time generalization of the Tweedie denoiser whose lookahead t acts as an intrinsic DP-plane knob: within a single trained flow map, varying t interpolates between the MMSE estimator ($t = s$) and a posterior sample ($t \to 1$), provably tracing the optimal DP curve of (2.6) in the Gaussian case and empirically doing so on CelebA and AFHQ inverse problems. Three chapter-specific limitations point to natural next steps. First, the formal optimality result is restricted to the Gaussian case; the observed non-monotonic DP behavior on box

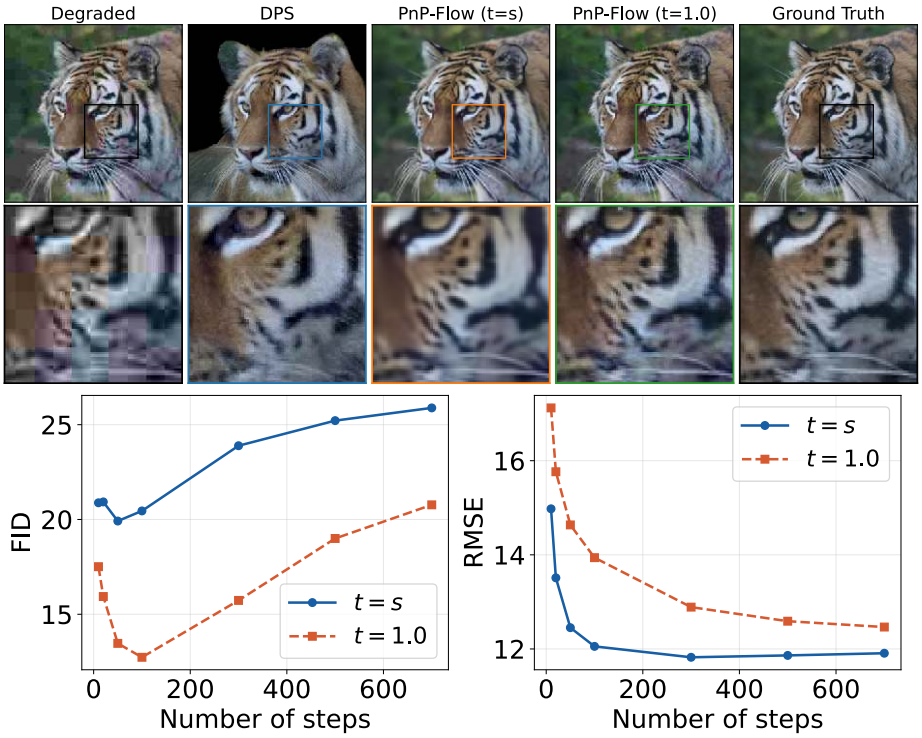


Figure 8.4 : DP behavior on JPEG compression.

inpainting (Fig. 8.3) shows this matters. Second, flow-map training is more expensive than standard flow matching and may underperform on pure distortion metrics such as PSNR; training the average denoiser directly via SURE-based or consistency losses [206, 237, 238] is an obvious alternative. Third, our experiments are pixel-space (CelebA 128, AFHQ 256); the natural extensions are latent-space and text-to-image settings, and more general restoration priors [186, 227]. Chapter 9 outlines the broader "flow maps as universal denoisers" program.

Chapter 9

Discussion and future directions

This thesis has developed a suite of methods that make diffusion-based inverse-problem solvers faster, more diverse, more reliably able to enforce hard constraints, and better matched to the structure of the task. This closing chapter summarizes the main contributions, distills the cross-cutting themes, and outlines methodological and application-facing directions that follow naturally from the work.

9.1 Summary of contributions

The unifying claim of the thesis is that a diffusion-based inverse-problem solver has four design knobs (Section 2.7) — the sampling algorithm, the source distribution of the generative process, the training target of denoising score matching (score alone vs. score-plus-density), and the training target for velocity (instantaneous vs. average) — and that each can be pulled independently to address a distinct failure mode of the pre-trained diffusion or flow-matching prior. Part I pulls the sampling-algorithm knob; Part II pulls the remaining three.

Part I: Inference-time algorithms on pre-trained priors. The three inference-time algorithms leave the pre-trained diffusion or flow prior untouched and modify only the sampling process:

- **Higher-order annealed Langevin diffusion** (Chapter 3) augments over-damped Langevin sampling with momentum variables and a preconditioner,

reducing the iteration budget by an order of magnitude on massive MIMO detection, channel estimation, and image restoration while preserving accuracy.

- **Repulsive latent score distillation** (Chapter 4) recasts posterior sampling in the latent space of a pre-trained latent diffusion model as a variational optimization problem and injects a repulsive coupling across particles that provably prevents the mode collapse typical of variational latent samplers.

- **Shielded Langevin Monte Carlo** (Chapter 5) borrows the Rimon–Koditschek navigation potential from robot motion planning to modify the Langevin drift and diffusion so that trajectories stay in a non-convex punctured support while still targeting the correct restricted stationary distribution.

Part II: Training priors for improved inference. The three training-time methods each pull one of the remaining knobs:

- **Prior-informed flow matching** (Chapter 6) modifies the *source distribution* of a flow-matching model, replacing the generic Gaussian source with a task-informed source built from a coarse graph predictor and training the flow to complete the reconstruction.

- **Energy-based models with anisotropic noise** (Chapter 7) modify the *training target*: standard denoising score matching learns only the score, but extending it to matrix-valued anisotropic noise via a dual score-matching objective yields a companion covariance score that renders the density normalization tractable and unlocks blind inverse problems, energy-guided sampling, and uncertainty quantification.

- **Flow-map denoisers** (Chapter 8) modify the *training target for velocity*: instead of the instantaneous velocity of a stochastic interpolant, we train a flow map that outputs the average velocity across arbitrary time pairs and use its two-time denoiser form as a plug-and-play primitive. The lookahead of the trained flow map becomes an intrinsic knob for traversing the distortion–perception plane with a single model.

9.2 Open questions and future directions

9.2.1 Methodological directions

Combining inference-time and training-time approaches. The two parts of the thesis have been developed in isolation. A natural next step is to combine them: running higher-order Langevin sampling on top of an anisotropic energy prior; applying repulsive score distillation on a flow-matching prior that has been prior-informed for the target task; or embedding the shielded LMC drift inside a flow-map-based sampler to enforce hard constraints on top of a DP-plane knob. Preliminary experiments suggest that such combinations inherit both the fast convergence of Part I and the reliability of Part II, but the theoretical picture — which regimes benefit most, and when combinations become redundant — is open.

Guarantees for shielded LMC. The shielded LMC sampler of Chapter 5 provably keeps trajectories in the free space and, empirically, targets the correct restricted stationary distribution, but formal non-asymptotic convergence rates are still open. Existing results for constrained Langevin methods assume convex feasible sets and break down on the non-convex punctured domains that motivated the sampler in the first place; extending mixing-time and Wasserstein-convergence guarantees to

the shielded dynamics would give practitioners principled ways to trade compute for reliability, and would clarify how the navigation-potential parameters interact with the friction and step-size choices.

Flow maps as universal denoisers. The flow-map denoisers of Chapter 8 raise the possibility of a single generative model that is simultaneously a one-step sampler, a many-step sampler, and a whole *family* of denoisers indexed by the lookahead t. Fully realizing this vision requires training objectives that do not favor any single time pair (the Lagrangian self-distillation loss of Chapter 8 weights the diagonal heavily), a better theoretical understanding of how the average denoiser traces the distortion–perception curve outside the Gaussian regime, and empirical evidence that a single trained flow map matches the specialized single-step and iterative denoisers it aims to unify. This unified denoiser would collapse the distinction between fast one-step samplers, iterative diffusion samplers, and plug-and-play denoisers into a single interface parameterized by t.

Safety and controllability. Beyond the barrier-based safety of Chapter 5, many real deployments demand privacy, copyright, or fairness constraints on generated content. Extending navigation-potential ideas to these settings, and combining them with the inference-time acceleration of Part I, is a promising direction with immediate practical impact.

9.2.2 Scientific applications

The methods in this thesis have been demonstrated primarily on imaging, wireless communications, and small-scale graph problems, but the design principles carry over to any inverse problem in which a physical forward operator meets a learned gener-

ative prior. Two scientific application areas are particularly promising: (i) medical and biological imaging modalities where reliability is at a premium (e.g., magnetic resonance imaging, cryogenic electron microscopy, computational lensing) and (ii) molecular and materials discovery, where the graph-flow techniques of Chapter 6 generalize naturally to reconstruction and completion tasks over molecular graphs.

We close by noting that the pace at which diffusion and flow-based generative models are advancing suggests that many of the specific algorithms in this thesis will be superseded. What we hope will persist are the design principles: exploit the extra structure of continuous-time generative models, prefer tractable posterior approximations, and treat the schedule and coupling of the generative process as first-class design variables.

Part III

Appendix

Appendix A

Supplement for Chapter 3: Higher-order annealed Langevin diffusion

A.1 Proof of Proposition 1

We define the following matrices

$$\mathbf{D} = \begin{bmatrix} 0 & 0 \\ 0 & \gamma\mathbf{M} \end{bmatrix}, \qquad \mathbf{Q}_t = \begin{bmatrix} 0 & -\mathbf{C}_t \\ \mathbf{C}_t & 0 \end{bmatrix}. \tag{A.1}$$

Given these matrices, we rewrite the SDE in (3.14) as follows

$$\mathrm{d}\mathbf{X}_t = -(\mathbf{D} + \mathbf{Q}_t)\nabla H(\mathbf{X}_t)\mathrm{d}t + \sqrt{2\tau\mathbf{D}}\,\mathrm{d}\mathbf{W}_t. \tag{A.2}$$

In this way, we have boiled down our dynamic to the general SDE formulation considered in [72]. Thus we can apply [72, Theorem 1] to show that the invariant distribution of the process is given by (3.4). Furthermore, leveraging our Assumptions 1 and 2, we have that $(\mathbf{D} + \mathbf{Q}_t)\nabla H(\mathbf{X}_t)$ is *Lipschitz continuous for all* t, so we can apply [239, Theorem 5.2.1] to show that the invariant distribution is unique. Hence, the only thing left to show is that the Hamiltonian of the pre-conditioned ULD is given by $H(\mathbf{x}, \mathbf{v}) = U(\mathbf{x}) + \frac{1}{2}\mathbf{v}^\top\mathbf{M}^{-1}\mathbf{v}$. We show this next.

We start by writing the Fokker-Planck equation of (3.14), which describes the transition probability density of $[\mathbf{x}_t, \mathbf{v}_t]$ denoted by $\rho(\mathbf{x}, \mathbf{v}, t)$, and is given by

$$\frac{\partial\rho}{\partial t} = \nabla \cdot \left(-\mathbf{b}(\mathbf{x}, \mathbf{v}, t)\rho + \frac{1}{2}\nabla \cdot (\boldsymbol{\Sigma}(\mathbf{x}, \mathbf{v})\rho) \right). \tag{A.3}$$

The term $\mathbf{b}(\mathbf{x}, \mathbf{v}, t)$ is the drift vector field, $\mathbf{\Sigma}(\mathbf{x}, \mathbf{v}, t)$ is the diffusion matrix, which is symmetric and non-negative. Then, the drift vector field is given by the deterministic components of the SDE in (A.2)

$$\mathbf{b}(\mathbf{x}, \mathbf{v}, t) = \begin{bmatrix} \mathbf{C}_t \mathbf{M}^{-1} \mathbf{v} \\ -\mathbf{C}_t \nabla U(\mathbf{x}) - \gamma \mathbf{v} \end{bmatrix} \tag{A.4}$$

and the diffusion matrix is given by the matrix that multiplies the random part in (A.2)

$$\mathbf{\Sigma}(\mathbf{x}, \mathbf{v}, t) = \begin{bmatrix} \mathbf{0} \\ 2\gamma\tau\mathbf{M} \end{bmatrix} \tag{A.5}$$

Thus, we have

$$\begin{aligned} \frac{\partial \rho}{\partial t} &= [\nabla_{\mathbf{x}}, \nabla_{\mathbf{v}}] \cdot \Bigg(- \begin{bmatrix} \mathbf{C}_t \mathbf{M}^{-1} \mathbf{v} \\ -\mathbf{C}_t \nabla U(\mathbf{x}) - \gamma \mathbf{v} \end{bmatrix} \rho + \\ &\qquad \frac{1}{2} \nabla \cdot \left(\begin{bmatrix} \mathbf{0} \\ 2\gamma\tau\mathbf{M} \end{bmatrix} \rho \right) \Bigg) \\ &= \underbrace{\nabla_{\mathbf{v}} \cdot (\mathbf{C}_t \nabla U(\mathbf{x}) \rho) - \nabla_{\mathbf{x}} \cdot \left(\mathbf{C}_t \mathbf{M}^{-1} \mathbf{v} \rho \right)}_{\text{Liouville operator of Hamiltonian dynamics}} + \\ &\qquad \underbrace{\nabla_{\mathbf{v}} \cdot (\gamma \mathbf{v} \rho) + \frac{1}{2} \nabla_{\mathbf{v}} \cdot \nabla_{\mathbf{v}} \left(2\gamma\tau\mathbf{M}\rho \right)}_{\mathcal{F}_0 \text{of the Ornstein–Uhlenbeck process}} \end{aligned} \tag{A.6}$$

To show that $\pi(\mathbf{x}, \mathbf{v})$ defined in (3.4) is the invariant distribution, we need to show that it belongs to the kernel of the Fokker-Planck operator. Consequently, we replace

$\rho(\mathbf{x}, \mathbf{v}, t) = \pi(\mathbf{x}, \mathbf{v})$ in (A.6); we start with the first term of (A.6)

$$\begin{aligned}
&\nabla_{\mathbf{v}} \cdot (\mathbf{C}_t \nabla U(\mathbf{x}) \rho) - \nabla_{\mathbf{x}} \cdot \left(\mathbf{C}_t \mathbf{M}^{-1} \mathbf{v} \rho\right) \\
&\quad = \left(\mathbf{C}_t \nabla U(\mathbf{x})\right)^{\top} \cdot \nabla_{\mathbf{v}} (\rho) - \left(\mathbf{C}_t \mathbf{M}^{-1} \mathbf{v}\right)^{\top} \cdot \nabla_{\mathbf{x}} (\rho) \\
&\quad = -\left(\mathbf{C}_t \nabla U(\mathbf{x})\right)^{\top} \cdot \left(\tau^{-1} \mathbf{M}^{-1} \mathbf{v}\right) \rho + \\
&\qquad\qquad \left(\mathbf{C}_t \mathbf{M}^{-1} \mathbf{v}\right)^{\top} \cdot \left(\tau^{-1} \nabla U(\mathbf{x})\right) \rho \\
&\quad = \tau^{-1} \big[- \nabla U(\mathbf{x})^{\top} \mathbf{C}_t^{\top} \cdot \left(\mathbf{M}^{-1} \mathbf{v}\right) + \\
&\qquad\qquad \mathbf{v}^{\top} (\mathbf{M}^{-1})^{\top} \mathbf{C}_t^{\top} \cdot \left(\nabla U(\mathbf{x})\right) \big] \rho.
\end{aligned} \tag{A.7}$$

Given that $\mathbf{C}_t$ is symmetric, we get

$$- \nabla U(\mathbf{x})^{\top} \mathbf{C}_t^{\top} \cdot \left(\mathbf{M}^{-1} \mathbf{v}\right) + \mathbf{v}^{\top} (\mathbf{M}^{-1})^{\top} \mathbf{C}_t^{\top} \cdot \left(\nabla U(\mathbf{x})\right) = 0 \tag{A.8}$$

Now focus on the second term in (A.6):

$$\begin{aligned}
&\nabla_{\mathbf{v}} \cdot (\gamma \mathbf{v} \rho) + \frac{1}{2} \nabla_{\mathbf{v}} \cdot \nabla_{\mathbf{v}} \left(2 \gamma \tau \mathbf{M} \rho\right) \\
&\qquad = \gamma \nabla_{\mathbf{v}} \cdot (\mathbf{v} \rho) - \gamma \nabla_{\mathbf{v}} \cdot \left(\tau \mathbf{M} \mathbf{M}^{-1} \tau^{-1} \mathbf{v} \rho\right) \\
&\qquad = \gamma \nabla_{\mathbf{v}} \cdot (\mathbf{v} \rho) - \gamma \nabla_{\mathbf{v}} \cdot (\mathbf{v} \rho) \\
&\qquad = 0.
\end{aligned} \tag{A.9}$$

Combining (A.8) and (A.9), we see that $\rho(\mathbf{x}, \mathbf{v}, t) = \pi(\mathbf{x}, \mathbf{v})$ is in the kernel of the Fokker-Planck equation, completing the proof.

A.2 Proof of Proposition 2

The proof is similar to that of Proposition 1. We define the following matrices

$$\mathbf{D} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & \alpha \mathbf{M} \end{bmatrix}, \quad \mathbf{Q}_t = \begin{bmatrix} 0 & -\mathbf{C}_t & 0 \\ \mathbf{C}_t & 0 & -\lambda \mathbf{M} \\ 0 & \lambda \mathbf{M} & 0 \end{bmatrix}, \tag{A.10}$$

and follow the same procedure as Appendix A.1.

We need to show that the Hamiltonian of the pre-conditioned third-order dynamic is given by $H(\mathbf{x},\mathbf{v},\mathbf{z}) = U(\mathbf{x}) + \frac{1}{2}\mathbf{v}^\top\mathbf{M}^{-1}\mathbf{v} + \frac{1}{2}\mathbf{z}^\top\mathbf{M}^{-1}\mathbf{z}$. Therefore, we start describing the Fokker-Planck equation (A.3) of (3.14). The drift vector is given by

$$\mathbf{b}(\mathbf{x},\mathbf{v},\mathbf{z},t) = \begin{bmatrix} \mathbf{C}_t\mathbf{M}^{-1}\mathbf{v} \\ -\mathbf{C}_t\nabla U(\mathbf{x}) + \lambda\mathbf{z} \\ -\lambda\mathbf{v} - \alpha\mathbf{z} \end{bmatrix} \tag{A.11}$$

and the diffusion matrix is given by

$$\mathbf{\Sigma}(\mathbf{x},\mathbf{v},\mathbf{z},t) = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ 2\tau\alpha\mathbf{M} \end{bmatrix}. \tag{A.12}$$

Thus, we have

$$\frac{\partial\rho}{\partial t} = [\nabla_{\mathbf{x}},\nabla_{\mathbf{v}},\nabla_{\mathbf{z}}]\cdot\left(-\begin{bmatrix} \mathbf{C}_t\mathbf{M}^{-1}\mathbf{v} \\ -\mathbf{C}_t\nabla U(\mathbf{x}) + \lambda\mathbf{z} \\ -\lambda\mathbf{v} - \alpha\mathbf{z} \end{bmatrix}\rho + \frac{1}{2}\nabla\cdot\left(\begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ 2\tau\alpha\mathbf{M} \end{bmatrix}\rho\right)\right) \tag{A.13}$$

$$\begin{aligned} = &\underbrace{\nabla_{\mathbf{v}}\cdot(\mathbf{C}_t\nabla U(\mathbf{x})\rho) - \nabla_{\mathbf{x}}\cdot\left(\mathbf{C}_t\mathbf{M}^{-1}\mathbf{v}\rho\right)}_{\text{Liouville operator of Hamiltonian dynamics}} \\ &- \nabla_{\mathbf{v}}\cdot(\lambda\mathbf{z}\rho) + \nabla_{\mathbf{z}}\cdot(\lambda\mathbf{v}\rho) \\ &+ \underbrace{\nabla_{\mathbf{z}}\cdot(\alpha\mathbf{z}\rho) + \frac{1}{2}\nabla_{\mathbf{z}}\cdot\nabla_{\mathbf{z}}\left(2\tau\alpha\mathbf{M}\rho\right)}_{\mathcal{F}_0 \text{ of the Ornstein–Uhlenbeck process}} \end{aligned} \tag{A.14}$$

Both terms associated with the Liouville operator and the Ornstein–Uhlenbeck process are the same as in Appendix A.1, so they are 0. Then, we have

$$-\nabla_{\mathbf{v}}(\lambda \mathbf{z}\rho) + \nabla_{\mathbf{z}}(\lambda \mathbf{v}\rho) = \tag{A.15}$$
$$-\lambda[\mathbf{z}^\top \cdot (\tau^{-1}\mathbf{M}^{-1}\mathbf{v})]\rho + \lambda[\mathbf{v}^\top \cdot (\tau^{-1}\mathbf{M}^{-1}\mathbf{z})]\rho.$$

This is 0 due to the symmetry of $\mathbf{M}$, completing the proof.

Appendix B

Supplement for Chapter 4: Repulsive latent score distillation

B.1 Technical proofs

B.1.1 Data augmentation

In Section 4.4.2 we consider an augmented variational distribution $q_\rho(\mathbf{x}_0, \mathbf{z}_0|\mathbf{y})$ such that

$$q_\rho(\mathbf{x}_0|\mathbf{y}) = \int q_\rho(\mathbf{x}_0, \mathbf{z}_0|\mathbf{y}) \mathrm{d}\mathbf{z}_0. \tag{B.1}$$

Therefore, we seek a joint distribution such that Property B.1 holds.

Property B.1 For all $\mathbf{x}_0 \in \mathbb{R}^N$*, it holds* $\lim_{\rho \to 0} \pi_\rho(\mathbf{x}_0) = \pi(\mathbf{x}_0)$.

Notice that our approach using data augmentation resembles some recent methods introduced in the bibliography. In [122], the authors propose to decouple data and diffusion model. More recently, a Bayesian version of the RED method was proposed in [240]. This method shares similarities with our method, in the sense that both are augmented versions that resemble RED. However, our method has three main differences: 1) it leverages latent diffusion models, which allows us to solve large-scale inverse problems (512×512 and beyond), 2) it uses the diffused trajectory to regularize the solution, and 3) it promotes diversity via the coupling term.

B.1.2 Proof of proposition 1

We expand the KL objective as follows

$$\begin{aligned}
\mathrm{KL}(q(\mathbf{z}_0, \mathbf{x}_0|\mathbf{y}) \| p(\mathbf{z}_0, \mathbf{x}_0|\mathbf{y})) &= \int q(\mathbf{z}_0, \mathbf{x}_0|\mathbf{y}) \log \frac{q(\mathbf{z}_0, \mathbf{x}_0|\mathbf{y})}{p(\mathbf{z}_0, \mathbf{x}_0|\mathbf{y})} \mathrm{d}\mathbf{z}_0 \mathrm{d}\mathbf{x}_0 \\
&= \int q(\mathbf{z}_0, \mathbf{x}_0|\mathbf{y}) \log \frac{q(\mathbf{x}_0|\mathbf{z}_0, \mathbf{y}) q(\mathbf{z}_0|\mathbf{y}) p(\mathbf{y})}{p(\mathbf{y} \mid \mathbf{x}_0) p(\mathbf{x}_0|\mathbf{z}_0) p(\mathbf{z}_0)} \mathrm{d}\mathbf{z}_0 \mathrm{d}\mathbf{x}_0 \\
&= \underbrace{\int q(\mathbf{z}_0, \mathbf{x}_0|\mathbf{y}) \log q(\mathbf{x}_0|\mathbf{z}_0, \mathbf{y}) \mathrm{d}\mathbf{z}_0 \mathrm{d}\mathbf{x}_0}_{(i)} \\
&\quad - \underbrace{\int q(\mathbf{z}_0, \mathbf{x}_0|\mathbf{y}) \log p(\mathbf{y} \mid \mathbf{x}_0) \mathrm{d}\mathbf{z}_0 \mathrm{d}\mathbf{x}_0}_{(ii)} \\
&\quad - \int \underbrace{q(\mathbf{z}_0, \mathbf{x}_0|\mathbf{y}) \log p(\mathbf{x}_0|\mathbf{z}_0) \mathrm{d}\mathbf{z}_0 \mathrm{d}\mathbf{x}_0}_{(iii)} \\
&\quad + \underbrace{\int q(\mathbf{z}_0, \mathbf{x}_0|\mathbf{y}) \frac{q(\mathbf{z}_0|\mathbf{y})}{p(\mathbf{z}_0)} \mathrm{d}\mathbf{z}_0 \mathrm{d}\mathbf{x}_0}_{(iv)} + \log p(\mathbf{y}).
\end{aligned} \tag{B.2}$$

Based on the augmented posterior, we have for (ii) and (iii) that

$$(ii) = \int q(\mathbf{z}_0, \mathbf{x}_0|\mathbf{y}) \log p(\mathbf{y} \mid \mathbf{x}_0) \mathrm{d}\mathbf{z}_0 \mathrm{d}\mathbf{x}_0 = \mathbb{E}_{q(\mathbf{x}_0, \mathbf{z}_0|\mathbf{y})} \left[\frac{1}{2\sigma_0^2} \|\mathbf{y} - \mathcal{A}(\mathbf{x}_0)\|^2 \right] \tag{B.3}$$

and

$$(iii) = \int q(\mathbf{z}_0, \mathbf{x}_0|\mathbf{y}) \log p(\mathbf{x}_0 \mid \mathbf{z}_0) \mathrm{d}\mathbf{z}_0 \mathrm{d}\mathbf{x}_0 = \mathbb{E}_{q(\mathbf{x}_0, \mathbf{z}_0|\mathbf{y})} \left[\frac{1}{2\rho^2} \|\mathbf{x}_0 - \mathcal{D}_\phi(\mathbf{z}_0)\|^2 \right]. \tag{B.4}$$

Regarding the first term, we can write as

$$(i) = \int q(\mathbf{z}_0|\mathbf{y}) \left[q(\mathbf{x}_0 \mid \mathbf{z}_0, \mathbf{y}) \log q(\mathbf{x}_0|\mathbf{z}_0, \mathbf{y}) \right] \mathrm{d}\mathbf{z}_0 \mathrm{d}\mathbf{x}_0 = \mathbb{E}_{q(\mathbf{z}_0|\mathbf{y})} \left[H(q(\mathbf{x}_0 \mid \mathbf{z}_0, \mathbf{y}) \right]. \tag{B.5}$$

Finally, the last term can be obtained by following theorem 2 in [41], assuming that the score is learned exactly, namely $\epsilon_\theta(\mathbf{z}_t; t) = -\sigma_t \nabla_{\mathbf{z}_t} \log p(\mathbf{z}_t)$, and under some mild

assumptions on the growth of $\log q\left(\mathbf{z}_t \mid \mathbf{y}\right)$ and $p\left(\mathbf{z}_t\right)$ at infinity, we have

$$\mathrm{KL}(q(\mathbf{z}_0|\mathbf{y})\|p(\mathbf{z}_0)) = \int_0^T \frac{\beta(t)}{2}\omega(t)\mathbb{E}_{q(\mathbf{z}_t|\mathbf{y})}\left[\|\nabla_{\mathbf{z}_t}\log q\left(\mathbf{z}_t \mid \mathbf{y}\right) - \nabla_{\mathbf{z}_t}\log p\left(\mathbf{z}_t\right)\|_2^2\right] dt \tag{B.6}$$

over the denoising diffusion trajectory $\{\mathbf{z}_t\}$ for positive values $\{\beta(t)\}$. This essentially implies that a weighted score-matching over the continuous denoising diffusion trajectory is equal to the KL divergence.

B.1.3 Proof of proposition 2

The first two terms are straightforward. We focus here on the regularization term. The regularization term in Proposition 3 corresponds to the score matching loss defined in [41] For general weighting schemes $\omega(t)$, we have the following Lemma from [41]

Lemma B.1 The time-derivative of the KL *divergence at timestep t obeys*

$$\frac{d\,\mathrm{KL}\left(q\left(\mathbf{z}_t \mid \mathbf{y}\right)\|p\left(\mathbf{z}_t\right)\right)}{dt} = -\frac{\beta(t)}{2}\mathbb{E}_{q(\mathbf{z}_t|\mathbf{y})}\left[\|\nabla_{\mathbf{z}_t}\log q\left(\mathbf{z}_t \mid \mathbf{y}\right) - \nabla_{\mathbf{z}_t}\log p\left(\mathbf{z}_t\right)\|^2\right].$$

Then, under the condition $\omega(0) = 0$, the integral in Proposition 3 can be written as (see [41])

$$\begin{aligned}\int_0^T \frac{\beta(t)}{2}\omega(t)\,\mathbb{E}_{q(\mathbf{z}_t|\mathbf{y})}\left[\|\nabla_{\mathbf{z}_t}\log q\left(\mathbf{z}_t \mid \mathbf{y}\right) - \nabla_{\mathbf{z}_t}\log p\left(\mathbf{z}_t\right)\|^2\right] dt &= -\int_0^T \omega(t)\frac{d\mathrm{KL}\left(q\left(\mathbf{z}_t \mid \mathbf{y}\right)\|p\left(\mathbf{z}_t\right)\right)}{dt}dt \\ &\overset{(a)}{=} \underbrace{-\omega(t)\mathrm{KL}(q\left(\mathbf{z}_t \mid y\right)\|p\left(\mathbf{z}_t\right))]_0^T}_{=0} + \int_0^T \omega'(t)\,\mathrm{KL}(q\left(\mathbf{z}_t \mid \mathbf{y}\right)\|p\left(\mathbf{z}_t\right))\,dt \\ &= \int_0^T \omega'(t)\,\mathbb{E}_{q(\mathbf{z}_t|\mathbf{y})}\left[\log\tfrac{q(\mathbf{z}_t|\mathbf{y})}{p(\mathbf{z}_t)}\right] dt,\end{aligned}$$

where $\omega'(t) := \frac{d\omega(t)}{dt}$. The equality holds because $\omega(t)\mathrm{KL}\left(q\left(\mathbf{z}_t \mid \mathbf{y}\right)\|p\left(\mathbf{z}_t\right)\right)]_0^T$ is zero at $t = 0$ and $t = T$. This is because $\omega(t) = 0$ by assumption at $t = 0$, and x_T becomes a pure Gaussian noise at the end of the diffusion process which makes $p\left(\mathbf{z}_T\right) = q\left(\mathbf{z}_T \mid \mathbf{y}\right)$ and thus $\mathrm{KL}\left(q\left(\mathbf{z}_T \mid \mathbf{y}\right)\|p\left(\mathbf{z}_T\right)\right) = 0$.

Now, we consider our proposed variational distribution defined in (4.7). with N particles and the pairwise kernel. For each particle i, we apply the forward diffusion $\mathbf{z}_t^{(i)} = \alpha_t \mathbf{z}_0^{(i)} + \sigma_t \boldsymbol{\epsilon}$, which yields the distribution $q\left(\mathbf{z}_t^{(i)} \mid \mathbf{y}\right) = \mathcal{N}\left(\alpha_t \mathbf{z}_0^{(i)}, \sigma_t^2 I\right)$, and thus $\nabla_{\mathbf{z}_t} \log q_t\left(\mathbf{z}_t \mid \mathbf{y}\right) = -\left(\mathbf{z}_t^{(i)} - \alpha_t \mathbf{z}_0^{(i)}\right)/\sigma_t^2 = -\frac{\epsilon^{(i)}}{\sigma_t}$. By applying the reparameterization trick, we obtain

$$
\begin{aligned}
&\nabla_{\mathbf{z}_0^{(i)}} \operatorname{reg}(\mathbf{z}_t^{(1)}, \cdots, \mathbf{z}_t^{(N)}) \\
&\quad = \int_0^T \omega'(t)\, \mathbb{E}_{\epsilon \sim \mathcal{N}(0,1)} \left[\left(-\nabla_{\mathbf{z}_t^{(i)}} \gamma \log \sum_{j=1}^N k\left(\mathbf{z}_t^{(i)}, \mathbf{z}_t^{(j)}\right) + \nabla_{\mathbf{z}_t^{(i)}} \log q_t(\mathbf{z}_t^{(i)} \mid \mathbf{y}) - \nabla_{\mathbf{z}_t} \log p(\mathbf{z}_t^{(i)}) \right)^\top \frac{d\mathbf{z}_t^{(i)}}{d\mathbf{z}_0^{(i)}} \right] dt \\
&\quad = \int_0^T \omega'(t)\, \mathbb{E}_{\epsilon \sim \mathcal{N}(0,1)} \left[\left(-\nabla_{\mathbf{z}_t^{(i)}} \gamma \log \sum_{j=1}^N k\left(\mathbf{z}_t^{(i)}, \mathbf{z}_t^{(j)}\right) - \frac{\epsilon}{\sigma_t} + \frac{\epsilon_\theta(\mathbf{z}_t^{(i)}; t)}{\sigma_t} \right)^\top \alpha_t \mathbf{I} \right] dt \\
&\quad = \int_0^T \omega'(t) \tfrac{\alpha_t}{\sigma_t}\, \mathbb{E}_{\epsilon \sim \mathcal{N}(0,1)} \left[-\nabla_{\mathbf{z}_t^{(i)}} \gamma \sigma_t \log \sum_{j=1}^N k\left(\mathbf{z}_t^{(i)}, \mathbf{z}_t^{(j)}\right) - \epsilon + \epsilon_\theta(\mathbf{z}_t^{(i)}; t) \right] dt.
\end{aligned}
$$

We can rearrange terms to arrive at the following compact form

$$
\begin{aligned}
&\nabla_{\mathbf{z}_0^{(i)}} \operatorname{reg}(\mathbf{z}_t^{(1)}, \cdots, \mathbf{z}_t^{(N)}) = \\
&\qquad \mathbb{E}_{t \sim \mathcal{U}[0,T], \epsilon \sim \mathcal{N}(0,1)} \left[\lambda_t \left(-\nabla_{\mathbf{z}_t^{(i)}} \gamma \sigma_t \log \sum_{j=1}^N k\left(\mathbf{z}_t^{(i)}, \mathbf{z}_t^{(j)}\right) - \epsilon + \epsilon_\theta(\mathbf{z}_t^{(i)}; t) \right) \right]
\end{aligned} \tag{B.7}
$$

for $\lambda_t := T\omega'(t)\alpha_t/\sigma_t$. When considering the measurement matching term and the error between the ambient and the augmented variable, we obtain $\lambda_t := T\omega'(t)\alpha_t/\sigma_t 4\sigma_0^2\rho^2$.

B.2 Additional experiments

B.2.1 Implementation details of baselines

For every baseline we use the authors' released implementation with their recommended hyperparameters; the specific pre-trained models, prior families, and small adjustments are summarized in Table B.1. RED-Diff additionally uses $\lambda = 0.25$ and $l_r = 0.1$; the other baselines use their default settings.

We tried running ReSample with Stable Diffusion (the released implementation does not natively support it), but obtained worse results than with LDM-VQ-4, so

Table B.1 : Baselines used in the RLSD experiments. Pre-trained diffusion models: FFHQ from [2]; ImageNet from [3]. Latent RED-Diff uses the same variational sampler as RED-Diff but with a latent diffusion prior (Section 4.3).

Method	**Prior**	**Reference**
PSLD / Latent-DPS	Stable Diffusion	[112]
DPS	pixel-space diffusion	[2]
RED-Diff	pixel-space diffusion	[50]
Latent RED-Diff	latent diffusion	[50] (this thesis)
FPS-SMC	pixel-space diffusion	[241]
ΠGDM	pixel-space diffusion	[53]
ReSample	LDM-VQ-4 (FFHQ)	[53]

we report the LDM-VQ-4 numbers.

NonAug-RLSD. This method is our variant using the particle-based variational approximation (4.7) without the augmentation of Section 4.4.2. For clarity, we show it in Alg. 7.

Algorithm 7 Non-augmented RLSD for solving inverse problems

Require: $\mathbf{y}, \mathcal{A}(\cdot), L, \boldsymbol{\epsilon}_\theta(\mathbf{z}_t, t), \mathcal{D}_\phi(\cdot), \{\lambda, \gamma, \tilde{\rho}, l_{r_z}\}$

for $l = 1$ to L **do**

Initialize $\{\mathbf{z}_{i,0}^0\}_{i=1}^N$

$t = T - \frac{\ell}{L}T$ and $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I})$

$\lambda_t = \lambda(\sigma_t/\alpha_t)$

$\mathbf{z}_{i,t}^\ell = \alpha_t \mathbf{z}_{i,0}^\ell + \sigma_t \boldsymbol{\epsilon}$

$\mathcal{L}_z = \sum_{i=1}^n \|\mathbf{y} - \mathcal{A}(\mathcal{D}_\phi(\mathbf{z}_i^l))\|^2 + \lambda_t \left(\mathrm{sg}\left[\boldsymbol{\epsilon}_\theta(\mathbf{z}_{i,t}^\ell, t) - \boldsymbol{\epsilon} - \gamma \nabla_{\mathbf{z}_t^{(i)}} \sigma_t \log \sum_{j=1}^N k\left(\mathbf{z}_t^{(i)}, \mathbf{z}_t^{(j)}\right) \right] \right)^\top \mathbf{z}_{i,0}^\ell$

$\mathbf{z}_0^\ell = \mathrm{OptimizerStep}_{\mathbf{z}_0^\ell}(\mathcal{L}_z, l_{r_z})$

end for

return $\{\mathbf{x}_{i,0}^L = \mathcal{D}_\phi(\mathbf{z}_{i,0}^L)\}_{i=1}^n$

RLSD. For our augmented formulation, we use Stable diffusion trained in the LAION [242] dataset as its pre-trained model. For the kernel function, we consider a RBF $k(\mathbf{z}_i, \mathbf{z}_j) = \exp(-\frac{||g_{\text{DINO}}(\mathbf{z}_i) - g_{\text{DINO}}(\mathbf{z}_j)||^2}{h_t})$ where $h_t = m_t^2 / \log N$, m_t is the median particle distance [109] and g_{DINO} is a pre-trained neural network [118]. Notice that NonRepuls-RLSD corresponds to $\gamma = 0$.

Regarding the similarity metric, we consider the cosine similarity in the range of DINO defined as follow

$$\text{Sim}(\mathbf{x}_1, \cdots, \mathbf{x}_N) = \frac{1}{n(n-1)} \sum_{i \neq j} \frac{g_{\text{DINO}}(\mathbf{x}_i)^T g_{\text{DINO}}(\mathbf{x}_j)}{\|g_{\text{DINO}}(\mathbf{x}_i)\|_2 \|g_{\text{DINO}}(\mathbf{x}_j)\|_2}, \tag{B.8}$$

Based on this metric, we define diversity as

$$\text{Div}(\mathbf{x}_1, \cdots, \mathbf{x}_N) = 1 - \text{Sim}(\mathbf{x}_1, \cdots, \mathbf{x}_N). \tag{B.9}$$

Lastly, we consider a decreasing annealing schedule. It has been notice in previous works [50, 243] that a decreasing timestep works better than sampling uniformly at random. As a consequence, we consider this scheme.

B.2.2 Super Resolution

We consider super resolution from $\times$ 8 downsampled images. In this case we use $\lambda = 0.2$, $\tilde{\rho} = 0.05$, $l_{r_x} = 0.4$ and $l_{r_z} = 0.6$. The results are shown in Table B.2. For additional comparison, we compare also with solvers that generate samples at 256×256. For this comparison, similar to Section 4.5.1, we downsample the result of RLSD to 256 and compare at that resolution. The results are in Table B.3. This example illustrates the key difference between RLSD and PSLD for solving inverse problems. To be more specific, PSLD generates images of faces that have the typical artifacts when sampling with Stable Diffusion. On the other hand, RLSD leverages Stable Diffusion as multiple denoisers at different scale.

Table B.2 : SR $\times$ 8 with $\sigma_0 = 0.001$ - FFHQ 512. The best method for each metric and experiment is bolded.

Sampler	**PSNR [dB]** $\uparrow$	**LPIPS** $\downarrow$	**FID** $\downarrow$
PSLD	24.82	0.314	81.31
Latent RED-diff	26.07	0.439	76.07
NonRepuls-RLSD	**28.39**	**0.286**	**65.42**

Table B.3 : SR $\times$ 8 with $\sigma_0 = 0.001$ - FFHQ 256. The best method for each metric and experiment is bolded.

Sampler	**PSNR [dB]** $\uparrow$	**LPIPS** $\downarrow$	**FID** $\downarrow$
NonRepuls-RLSD	**28.4**	**0.149**	**65.42**
RED-Diff	25.69	0.264	104.59
DPS	23.83	0.175	90.45
ΠGDM	24.44	**0.128**	82.01

For completeness, we also compare with RED-Diff when solving SR $\times$ 4, as it has the same resolution as input than RLSD with SR $\times$ 8 (RED-Diff handles images of size 256x256). The results is Table B.4.

Table B.4 : SR $\times$ 8 with $\sigma_0 = 0.001$ - FFHQ 256. The best method for each metric and experiment is bolded.

Sampler	PSNR [dB] ↑	LPIPS ↓	FID ↓
NonRepuls-RLSD	28.4	**0.149**	**65.42**
RED-Diff (SR $\times$ 4)	**28.91**	0.157	78.41
DPS (SR $\times$ 4)	27.14	0.128	71.8
FPS-SMC (SR $\times$ 4)	27.36	0.21	120.49

B.2.3 Motion deblurring

We consider Motion Blurring. In this case we use $\lambda = 0.007$, $\tilde{\rho} = 0.01$, $l_{r_x} = 0.4$ and $l_{r_z} = 0.3$, and $L = 500$. In particular, we follow [2], where we convolve the image with a 61×61 motion kernel that is randomly sampled with intensity 0.3^2. The results are shown in Table B.5. For additional comparison, we compare also with solvers that generate samples at 256×256. For this comparison, similar to Section 4.5.1, we downsample the result of RLSD to 256 and compare at that resolution. The results are in Table B.6.

Table B.5 : Motion Blurring with $\sigma_0 = 0.001$ - FFHQ 512. The best method for each metric and experiment is bolded.

Sampler	PSNR [dB] ↑	LPIPS ↓	FID ↓
PSLD	25.17	0.389	135.22
Latent RED-diff	27.85	0.329	118.09
NonRepuls-RLSD	**30.4**	**0.23**	**56.79**

Table B.6 : Motion Blurring with $\sigma_0 = 0.001$ - FFHQ 256. The best method for each metric and experiment is bolded.

Sampler	PSNR [dB] ↑	LPIPS ↓	FID ↓
NonRepuls-RLSD	**30.47**	**0.095**	**56.79**
RED-Diff	30.27	0.15	103.17
DPS	24.7	0.22	90.45
ReSample	26.82	0.115	72.74

B.2.4 Inpainting with masked box (half-face)

We consider here additional baselines for box inpainting (half-face). See results in Table B.7.

Table B.7 : Box inpainting (half face) with $\sigma_0 = 0.001$ - FFHQ 256. The best method for each metric and experiment is bolded.

Sampler	PSNR [dB] ↑	LPIPS ↓	FID ↓
NonAug-RLSD ($\gamma = 50$)	23.34	0.164	98.65
NonRepuls-RLSD	**24.98**	<u>0.079</u>	**29.18**
RLSD ($\gamma = 50$)	24.69	0.111	31.41
Hybrid-RLSD	<u>24.72</u>	0.096	30.48
ΠGDM	23.74	0.077	33.8
FPS-SMC	24.91	0.086	59.59

B.2.5 Inpainting with random mask

We consider random inpainting where we drop 80% of the pixels. In this case we use $\lambda = 0.009$, $\tilde{\rho} = 0.08$, $l_{r_x} = 0.4$ and $l_{r_z} = 0.8$. The numerical results are shown in Table B.8, and visual examples are shown in Fig. B.3.

Table B.8 : Random inpainting with (80%) mask and with ($\sigma_0 = 0.001$) - FFHQ 512. In bold is the best method for each metric and experiment.

Sampler	**PSNR [dB]** ↑	**LPIPS** ↓	**FID** ↓
PSLD	28.53	0.212	65.14
NonRepuls-RLSD	**30.56**	**0.145**	**41.11**

Table B.9 : Random inpainting with (80%) mask and with ($\sigma_0 = 0.001$) - FFHQ 256. In bold is the best method for each metric and experiment.

Sampler	**PSNR [dB]** ↑	**LPIPS** ↓	**FID** ↓
NonRepuls-RLSD	**30.56**	0.073	**41.11**
RED-Diff	28.55	0.074	62.87
DPS	26.48	0.15	95.44
ReSample	27.49	**0.062**	54.42

B.2.6 Inpainting free mask

We use the free masks ($10\% - 20\%$) from [4]. For this experiment, we consider ImageNet [211] to demonstrate that our method outperforms its baselines on other

datasets. We consider the $\lambda = 0.15$, $\tilde{\rho} = 0.15$, $l_{r_z} = 0.8$ and $l_{r_x} = 0.4$, and we consider 500 steps (instead of the full trajectory of 1000 steps) for both RLSD and PSLD. For PSLD, we use the parameters from their experiments with box inpainting (similar to the case of half face). The quantitative results are shown in Table B.10, and qualitative results in Fig. B.4.

Table B.10 : Free mask [4] with ($\sigma_0 = 0.001$) - ImageNet 512. In bold is the best method for each metric and experiment.

Sampler	**PSNR [dB]** ↑	**LPIPS** ↓	**FID** ↓
PSLD	22.56	0.154	69.43
NonRepuls-RLSD	**26.77**	**0.075**	**60.53**

Results when considering the best sample across particles. If instead of focusing on the average performance, we focus on the performance achieved by the best image among the particle ensemble, then the conclusion is different in favor of the RLSD; see Table B.11. This can be explained as follows: while some of the modes obtained with RLSD might not be as good as NonRepuls-RLSD, others might be better.

Table B.11 : Box inpainting (half face) with $\sigma_0 = 0.001$ - FFHQ 512. We consider the best particle across each batch of them.

Sampler	**PSNR [dB] ↑**	**LPIPS ↓**	**FID ↓**
PSLD (mean)	21.34	0.10	57.7
PSLD (max)	22.72	0.082	57.7
NonRepuls-RLSD (mean)	24.98	0.079	**29.18**
NonRepuls-RLSD (max)	<u>25.82</u>	<u>0.071</u>	29.18
RLSD ($\gamma = 50$) (mean)	24.69	0.111	31.41
RLSD ($\gamma = 50$) (max)	**25.84**	**0.069**	27.84

B.2.7 Qualitative results



(a) NonAug-RLSD ($\gamma = 30$).



(b) Latent RED-Diff.

Figure B.1 : Phase Retrieval. Adding repulsion allows to sample from different modes.

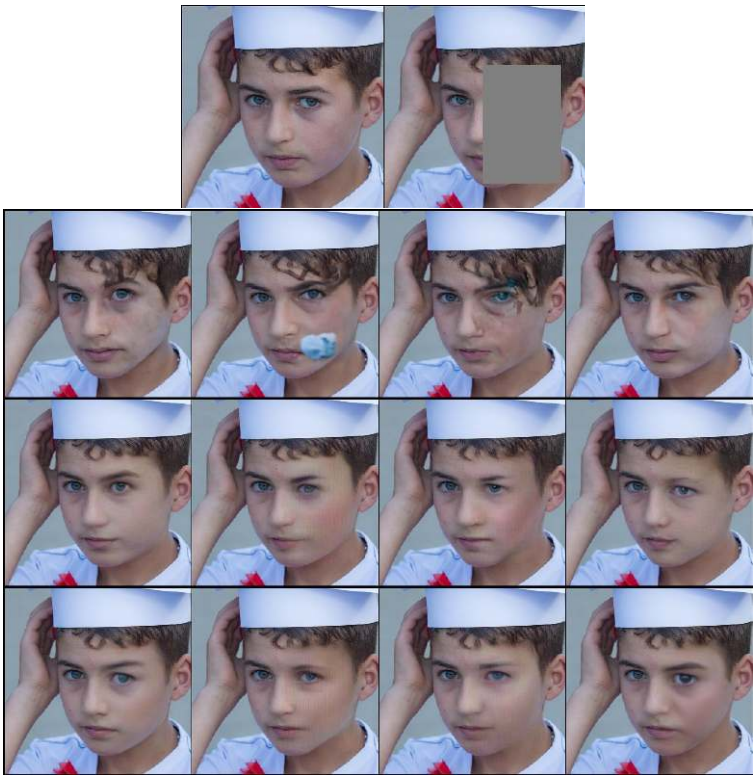


Figure B.2 : Inpainting half face using (from top to bottom): Ground truth and Measurement, PSLD, NonRepuls-RLSD, and RLSD ($\gamma = 50$). We generate four samples for each method from a different initialization; for RLSD, they interact through the repulsion term. First, NonRepuls-RLSD, and RLSD outperforms PSLD for all four images. Second, image 4 from RLSD (last row) looks different (has brown eye), while the four samples of NonRepuls-RLSD have blue eye; this illustrates that RLSD promotes diversity.



Figure B.3 : Random inpainting (80%): PSLD (top) and NonRepuls-RLSD (down), and four different samples for each method. Notice that NonRepuls-RLSD generates a better reconstruction of the background.

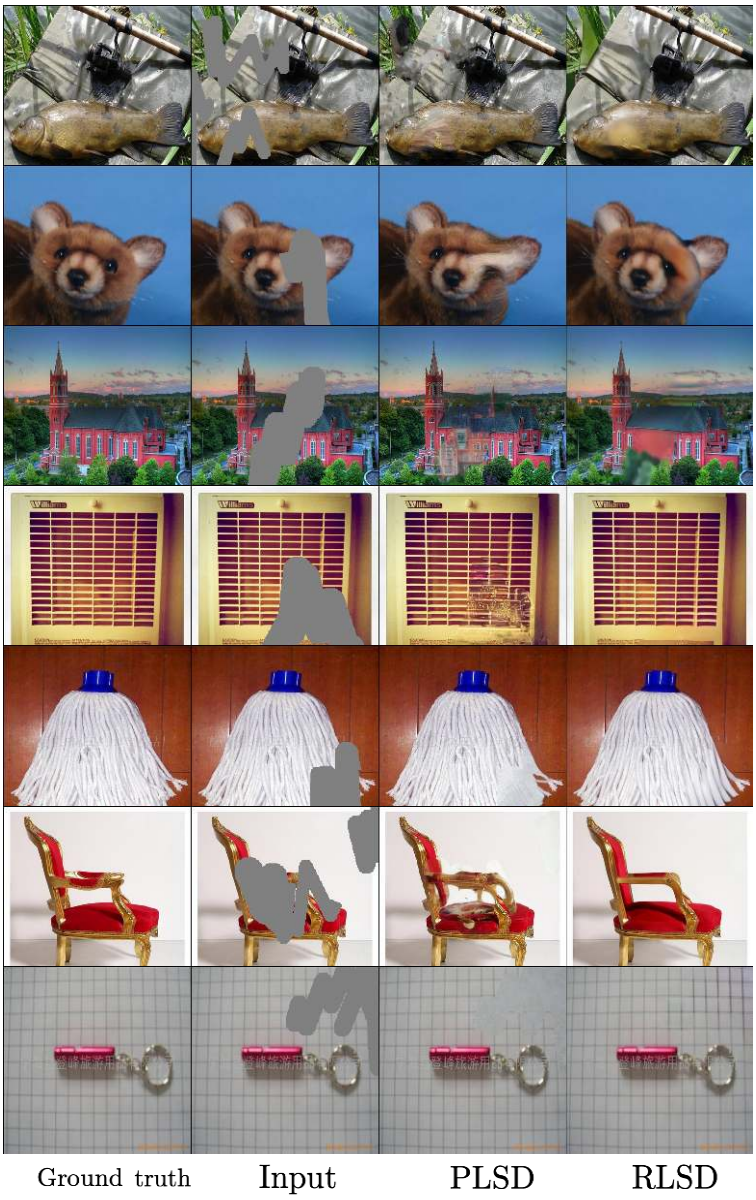


Figure B.4 : Qualitative examples of Inpainting with free mask on Imagenet. We consider PSLD (third column) and NonRepuls-RLSD (forth column). For rows 2, 4-7, RLSD outperforms PSLD by a large margin. Furthermore, notice that RLSD can reconstruct backgrounds that might be difficult, such as the one row 4. On the other hand, RLSD struggles to reconstruct fine-details, as it is shown in row 3.

B.2.8 Ablation of RLSD

Running Time/Number of Steps

We ablate the running time between RLSD and PSLD when generating two samples for super resolution $\times 8$. We ran the experiments on the same NVIDIA A100 GPU with 80GB. When running the full trajectory (1000 steps), the running time of RLSD without particles is 8.8 minutes, while PSLD is 9.2 minutes. When running 200 steps, the running time of RLSD without particles is 1.75 minutes, while PSLD is 1.9 minutes. The results for this case are shown in Fig. B.5. Notice that although the running time difference between RLSD and PSLD is not large, our formulation leverages the diffused trajectory as a denoiser. Therefore, we require fewer steps to obtain a high-fidelity estimation; for instance, for super resolution we need just 200 steps. This showcases that our method works fine with just a few number of steps; see also Fig. B.6.



(a)



(b)



(c)

Figure B.5 : Reconstruction for different number of steps. a) Ground-truth (left) and Measurement (right). b) The two on the left are the reconstruction using PSLD with 200 steps (running time = 1.9 min), while the two of the right are with RLSD ($\gamma = 0$) (running time = 1.75 min). c) The two on the left are the reconstruction using PSLD with 1000 steps (running time = 9.2 min), while the two of the right are with RLSD ($\gamma = 0$) (running time = 8.8 min).

Compared to ReSample [114], RLSD is also computationally more efficient. Specifically, ReSample involves an inner optimization within the sampling. Consequently, when using LDM (256x256 resolution), it takes 4 minutes in one A100 GPU to generate 1 image. Moreover, when we tried Stable Diffusion, it took 16 minutes (considering the same hyperparameter configuration). We hypothesize that ReSample takes a long time with Stable Diffusion due to the optimization step, where it might struggle

to converge at every step. While this might be improved, it demonstrates that the method needs more fine-tuning than ours. On the other hand, our method, using Stable Diffusion as prior, can generate 4 samples with just 200 steps in less than 4 minutes (without repulsion, 1 image per minute) and 5 minutes (with full repulsion).



Figure B.6 : Reconstruction as a function of L (number of steps in the optimization). Increasing L modifies the number of denoisers that are used (the limits of the interval are the same, and it changes the step-size). Also, we need to decrease l_{r_z} (from 1 to 0.8) as well the coupling term $\tilde{\rho}$, from 0.15 to 0.1.

Repulsion term

Effect of γ. We showcase an example of the effect of γ. We consider $N = 4$ particles and 200 steps, $\tilde{\rho} = 0.1, \lambda_t = 0.15, l_{r_x} = 0.4$ and $l_{r_z} = 1$. It is important to remark that when considering fewer steps, we need a higher γ (compared to the example in Section 4.5.1; the reason why this happens is due to the annealing factor in the repulsion term (4.12), given by $\alpha(t)$. The example for $\gamma = 0, 100, 150, 200$ is shown in Fig. B.7. Also, in Table B.12 we present quantiative results.

Table B.12 : Ablation γ for inpainting with free-mask.

γ	**PSNR [dB]** ↑	**LPIPS** ↓	FID	Diversity ↑
0	30.03	0.064	22.62	0.004
25	**30.04**	**0.064**	**22.21**	**0.005**
50	30	0.064	22.31	0.005

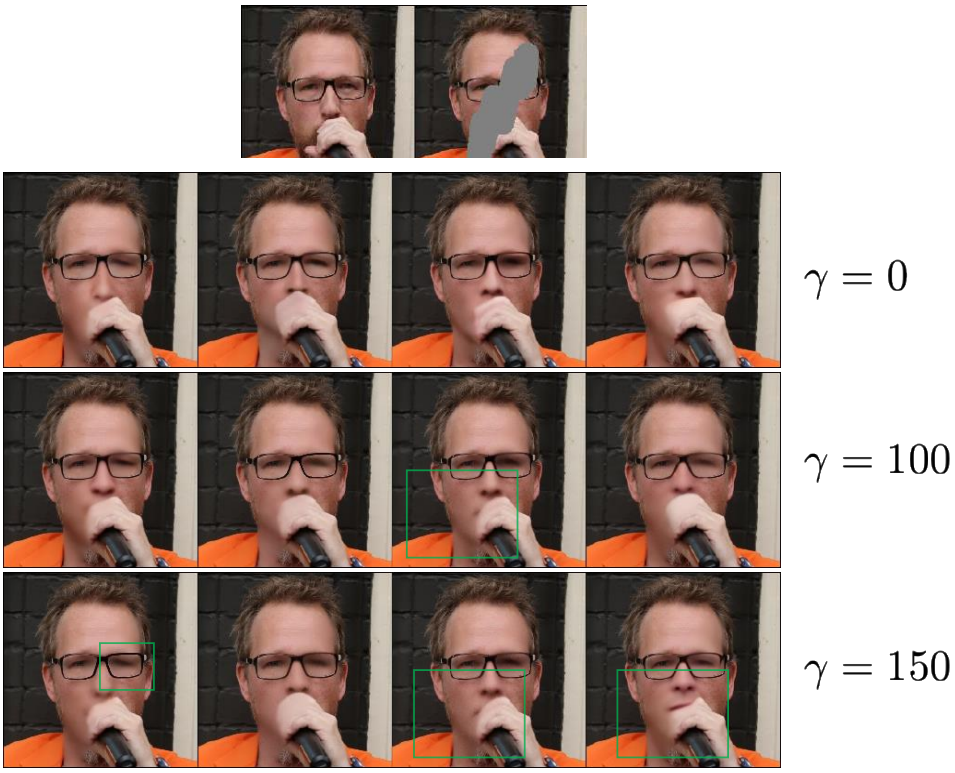


Figure B.7 : Reconstruction as a function of γ (number of repulsion), for $L = 200$ and $N = 4$. Increasing γ enhances diversity: for $\gamma = 150$, 2 out of 4 samples show the mouth, while for $\gamma = 0$ all samples have the mouth hidden.

Effect of repulsion on a fixed interval. Depending on the downstream task, we can include repulsion only on a fixed interval. For instance, this is the case of

Phase Retrieval, where modes are discrete and isolated. Therefore, once each particle get around one of the mode (different), we can turn-off the repulsion, which yields a faster solver. For the case of inpainting, we show in Fig. B.8 an ablation changing t_{repul} where $t \in [0, t_{\text{repul}}]$ and in Fig. B.9 when $t \in [t_{\text{repul}}, T]$ (with $T = 200$).

We observe that for diversity, it is more important to have repulsion at the beginnig of the sampling, which corresponds to the higher noise levels. Intuitively, it is more important to impose repulsion in the high-level semantics of the image instead of the fine-details. However, adding some repulsion later in the sampling process might improve some details (see the case $t_{\text{repul}} \in [100, 200]$ in Fig. B.9).

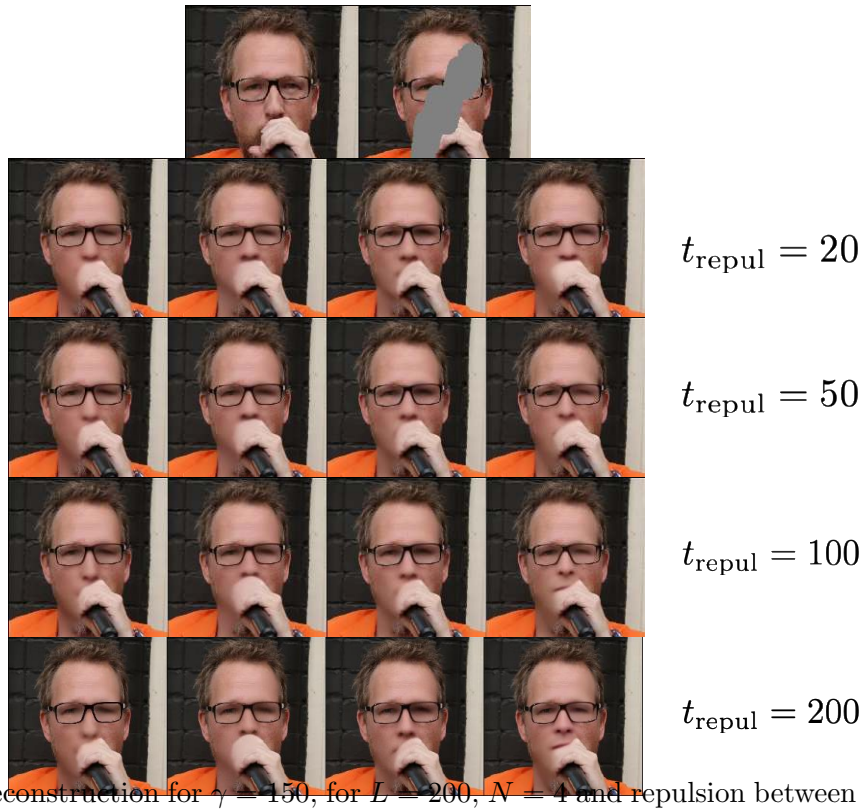


Figure B.8 : Reconstruction for $\gamma = 150$, for $L = 200$, $N = 4$ and repulsion between 0 and t_{repul}: when $t_{\text{repul}} = T$, we have repulsion in all the trajectory, while $t_{\text{repul}} = 0$ corresponds to NonRepuls-RLSD.

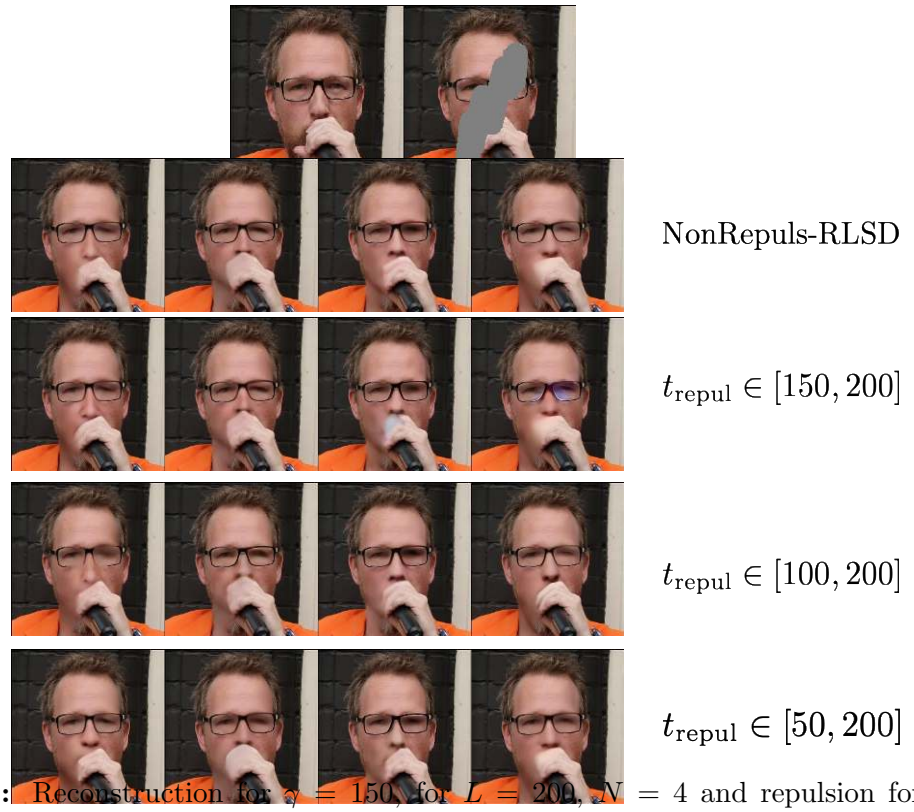


Figure B.9 : Reconstruction for $\gamma = 150$, for $L = 200$, $N = 4$ and repulsion for $t \in [t_{\text{repul}}, 200]$; notice that $t_{\text{repul}} = T$ corresponds to NonRepuls-RLSD.

Effect of number of particles. Lastly, we do an ablation when considering more particles in the repulsion term. Given four particles, we consider three settings: $N = 0$ (four independent particles), $N = 2$ (two independent particles and two interacting particles) and $N = 4$ (four interacting particles). We use again the free mask [4], and $L = 200$. In Table B.13 we show quantitative results, while in Figs. B.10 and B.11 we show two qualitative results. In both cases for $N = 2$, the last two columns correspond to the two i.i.d. particles. Consequently, those two images can change when considering $N = 4$, while the first two columns can change when moving from $N = 0$ to two. In particular, in Fig. B.10 with $N = 2$ we incerase diversity, while in Fig. B.11 we need at least $N = 4$.

Table B.13 : Ablation when changing the number of particles in the repulsion term with $\sigma_0 = 0.001$ - FFHQ 512.

Sampler	PSNR [dB] ↑	LPIPS ↓	FID ↓	Diversity
NonRepuls-RLSD ($N = 0$)	**30.03**	0.064	**65.42**	0.002
RLSD ($N = 2$)	<u>29.99</u>	0.064	71.8	0.004
RLSD ($N = 4$)	29.99	**0.063**	69.91	0.005



Figure B.10 : Reconstruction when increasing the number of particles in the repulsion, for $N = 0, 2$ and 4. Notice that $N = 0$ corresponds to the NonRepuls-RLSD, $N = 2$ to Hybrid case, and $N = 4$ to RLSD.

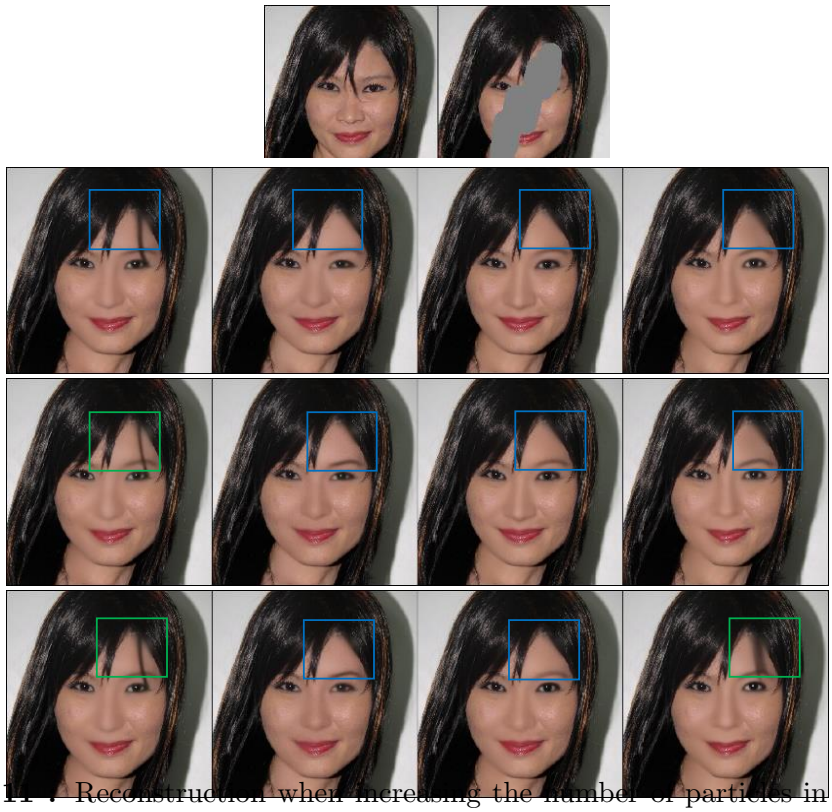
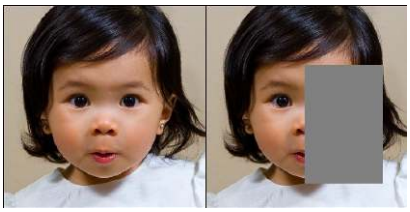


Figure B.11 : Reconstruction when increasing the number of particles in the repulsion, for $N = 0, 2$ and 4. Notice that $N = 0$ corresponds to the NonRepuls-RLSD, $N = 2$ to Hybrid case, and $N = 4$ to RLSD. In this case, we need full repulsion to increase diversity w.r.t. the NonRepuls-RLSD case.

Comparison between NonAug-RLSD vs RLSD

We compare in Fig. B.12 NonAug-RLSD vs RLSD. Notice that RLSD achieves a sharper reconstructed image; in particular, the shirt for the case of NonAug-RLSD has not fine-detail, while RLSD has.



(a)



(b)

Figure B.12 : Comparison between RLSD (top row of Fig. b) and NonAug-RLSD (last row of Fig. b). RLSD achieves a sharper reconstructed image; in particular, the shirt for the case of NonAug-RLSD has not fine-detail, while RLSD has.

Comparison with images from ImageNet between RLSD and RED-Diff with diffusion prior trained on FFHQ

Here we show some examples of reconstruction using out-of-distribution images; we use samples from ImageNet [211]. In Fig. B.13 we show an example from ImageNet, where we compare RLSD with RED-diff using FFHQ. Clearly, the performance of

RLSD is better than RED-diff. This is expected given that the diffusion model of RED-diff is with FFHQ. However, this demonstrates that using more powerful diffusion model as prior enables to deploy our model for different type of images.



(a)



(b)



(c)

Figure B.13 : Results of the reconstruction of an image from the validation set of ImageNet using RLSD (b) and RED-diff with diffusion prior trained on FFHQ (c). This example showchases that using a large-pre trained model such as Stable Diffusion enables to use our method with images from very different classes (Imagenet and FFHQ).

Coupling parameter

We do an ablation of the coupling parameter $\tilde{\rho}$. We consider three values: $\tilde{\rho} = \{0.03, 0.07, 0.12\}$. The qualitative results for this case are shown in Fig. B.14, and the quantiative in Table B.14 As expected, increasing the coupling parameter yields

a more diverse set of images. The reason why this happens is because the prior (through $\mathbf{z}$) has more weight when estimating the image $\mathbf{x}$. However, this increase in diversity is penalized by a lower performance. On the other hand, a lower ρ generates four images that look very similar. It is important to note that our repulsion term is a better trade-off in terms of diversity/quality.

Table B.14 : Ablation ρ for inpainting with free-mask - $\gamma = 0$

ρ	**PSNR [dB]** ↑	**LPIPS** ↑	FID	Diversity ↑
0.03	29.94	0.069	25.57	0.01
0.07	**30.03**	**0.064**	**22.62**	0.002
0.12	28.6	0.065	23.25	**0.006**

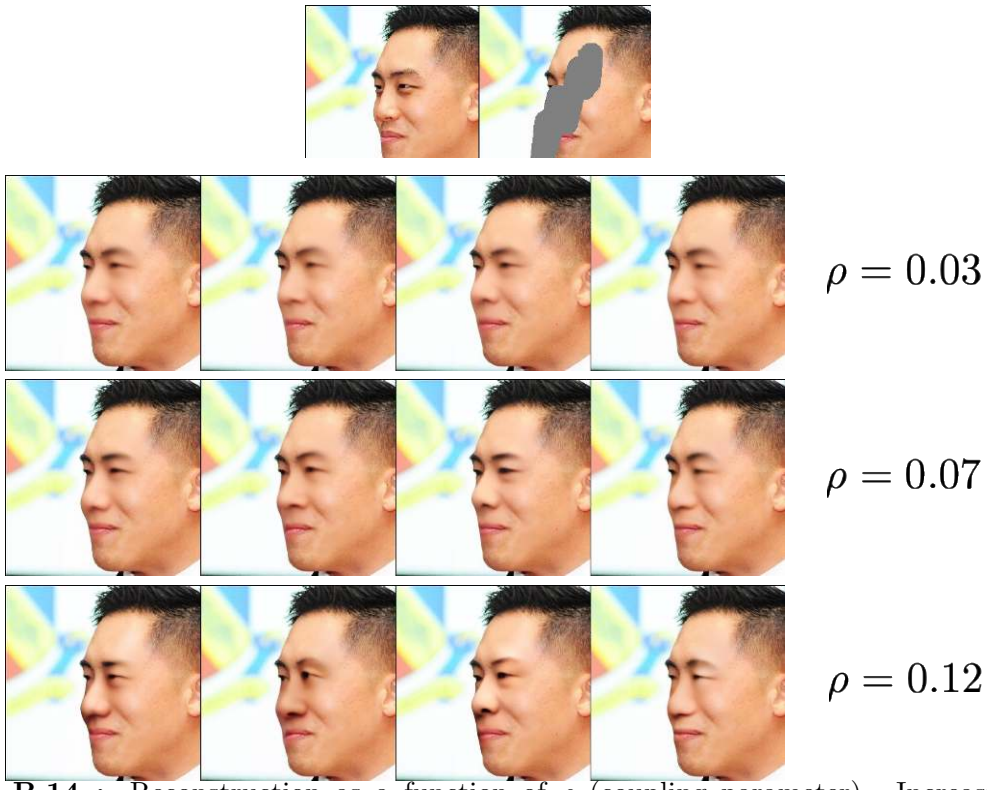


Figure B.14 : Reconstruction as a function of ρ (coupling parameter). Increasing ρ enhances diversity but reduces quality: for $\rho = 0.12$, the 4 samples looks different but with a low quality, while with a lower ρ, the diversity is lower the performance is better.

Appendix C

Supplement for Chapter 6: Prior-informed flow matching

C.1 Proofs

C.1.1 Proof for Theorem 1

Proof. Our goal is to show that for any permutation matrix $\mathbf{P}$, our estimated density satisfies $\log p(\mathbf{P}^\top \mathbf{A}_1 \mathbf{P}) = \log p(\mathbf{A}_1)$. First, we notice that $\operatorname{tr}\left(\frac{\partial v_\theta(\mathbf{A}_t, t)}{\partial \mathbf{A}_t}\right) = \langle v_\theta(\mathbf{A}_t, t), d\mathbf{A}_t \rangle_F$ Let's define a permuted graph $\mathbf{A}_1' = \mathbf{P}^\top \mathbf{A}_1 \mathbf{P}$ and a similarly permuted mask $\xi' = \mathbf{P}^\top \xi \mathbf{P}$; to simplify notation, we denote $F(.) = \log p(.)$.

First, we establish the equivariance of the initial state $\mathbf{A}_0$. Let $\mathbf{A}_0'$ be the initial state constructed from the permuted graph $\mathbf{A}_1'$ and mask ξ'.

$$
\begin{aligned}
\mathbf{A}_0' &= \xi' \odot \mathbf{A}_1' + (1 - \xi') \odot f_{\text{prior}}(\xi' \odot \mathbf{A}_1') \\
&= (\mathbf{P}^\top \xi \mathbf{P}) \odot (\mathbf{P}^\top \mathbf{A}_1 \mathbf{P}) + (1 - \mathbf{P}^\top \xi \mathbf{P}) \odot f_{\text{prior}}((\mathbf{P}^\top \xi \mathbf{P}) \odot (\mathbf{P}^\top \mathbf{A}_1 \mathbf{P})) \\
&= \mathbf{P}^\top (\xi \odot \mathbf{A}_1) \mathbf{P} + \mathbf{P}^\top (1 - \xi) \mathbf{P} \odot f_{\text{prior}}(\mathbf{P}^\top (\xi \odot \mathbf{A}_1) \mathbf{P}) \quad (\text{since } \odot \text{ distributes over } \mathbf{P})
\end{aligned}
$$

The key requirement for this proof is the permutation equivariance of the prior estimator, f_{prior}. This condition is satisfied by both prior models used in our work. The SIGL-based prior is permutation equivariant by design, as it uses a GNN encoder to learn the graphon structure. The node2vec-based prior enforces permutation equivariance by first mapping nodes to a canonical ordering based on the principal

components of their embeddings, ensuring that any permutation of an input graph is processed identically.

With the permutation equivariance of f_{prior} established, such that $f_{\text{prior}}(\mathbf{P}^\top \mathbf{X}\mathbf{P}) = \mathbf{P}^\top f_{\text{prior}}(\mathbf{X})\mathbf{P}$, we can apply this property:

$$\begin{aligned}
\mathbf{A}'_0 &= \mathbf{P}^\top(\xi \odot \mathbf{A}_1)\mathbf{P} + \mathbf{P}^\top(1-\xi)\mathbf{P} \odot \\
&\qquad (\mathbf{P}^\top f_{\text{prior}}(\xi \odot \mathbf{A}_1)\mathbf{P}) \\
&= \mathbf{P}^\top(\xi \odot \mathbf{A}_1)\mathbf{P} + \mathbf{P}^\top\big((1-\xi) \odot f_{\text{prior}}(\xi \odot \mathbf{A}_1)\big)\mathbf{P} \\
&= \mathbf{P}^\top\Big(\xi \odot \mathbf{A}_1 + (1-\xi) \odot f_{\text{prior}}(\xi \odot \mathbf{A}_1)\Big)\mathbf{P} \\
&= \mathbf{P}^\top \mathbf{A}_0 \mathbf{P}
\end{aligned}$$

Thus, the initial state $\mathbf{A}_0$ is permutation-equivariant.

Next, we examine the flow path $\mathbf{A}'_t$ corresponding to the permuted graph $\mathbf{A}'_1$:

$$\begin{aligned}
\mathbf{A}'_t &= (1-t)\mathbf{A}'_0 + t\mathbf{A}'_1 \\
&= (1-t)(\mathbf{P}^\top \mathbf{A}_0 \mathbf{P}) + t(\mathbf{P}^\top \mathbf{A}_1 \mathbf{P}) \\
&= \mathbf{P}^\top\big((1-t)\mathbf{A}_0 + t\mathbf{A}_1\big)\mathbf{P} \\
&= \mathbf{P}^\top \mathbf{A}_t \mathbf{P}.
\end{aligned}$$

The path itself is equivariant. The differential element also transforms equivariantly: $d\mathbf{A}'_t = \mathbf{P}^\top d\mathbf{A}_t \mathbf{P}$.

Now, we evaluate the scalar function $F(\mathbf{A}'_1)$ by integrating along the permuted path $\mathbf{A}'_t$:

$$F(\mathbf{A}'_1) = -\int_0^1 \langle v_\theta(\mathbf{A}'_t, t), d\mathbf{A}'_t \rangle_F dt + C$$

Substituting the equivariant forms for the path and its differential:

$$F(\mathbf{A}'_1) = -\int_0^1 \langle v_\theta(\mathbf{P}^\top \mathbf{A}_t \mathbf{P}, t), \mathbf{P}^\top d\mathbf{A}_t \mathbf{P} \rangle_F dt + C$$

By the assumed permutation equivariance of the velocity field v_θ, we have $v_\theta(\mathbf{P}^\top \mathbf{A}_t \mathbf{P}, t) = \mathbf{P}^\top v_\theta(\mathbf{A}_t, t)\mathbf{P}$. Substituting this in:

$$F(\mathbf{A}'_1) = -\int_0^1 \langle \mathbf{P}^\top v_\theta(\mathbf{A}_t, t)\mathbf{P}, \mathbf{P}^\top d\mathbf{A}_t \mathbf{P} \rangle_F dt + C$$

The Frobenius inner product $\langle \mathbf{A}, \mathbf{B} \rangle_F = \mathrm{tr}(\mathbf{A}^\top \mathbf{B})$ is invariant to unitary transformations. Specifically, for any orthogonal matrix $\mathbf{P}$ (where $\mathbf{P}^\top \mathbf{P} = \mathbf{I}$):

$$\begin{aligned}
\langle \mathbf{P}^\top \mathbf{X} \mathbf{P}, \mathbf{P}^\top \mathbf{Y} \mathbf{P} \rangle_F &= \mathrm{tr}\big((\mathbf{P}^\top \mathbf{X} \mathbf{P})^\top (\mathbf{P}^\top \mathbf{Y} \mathbf{P})\big) \\
&= \mathrm{tr}\big(\mathbf{P}^\top \mathbf{X}^\top \mathbf{P} \mathbf{P}^\top \mathbf{Y} \mathbf{P}\big) \\
&= \mathrm{tr}\big(\mathbf{P}^\top \mathbf{X}^\top \mathbf{Y} \mathbf{P}\big) \\
&= \mathrm{tr}(\mathbf{X}^\top \mathbf{Y}) \\
&= \langle \mathbf{X}, \mathbf{Y} \rangle_F.
\end{aligned}$$

Applying this property, the integrand simplifies:

$$\langle \mathbf{P}^\top v_\theta(\mathbf{A}_t, t)\mathbf{P}, \mathbf{P}^\top d\mathbf{A}_t \mathbf{P} \rangle_F = \langle v_\theta(\mathbf{A}_t, t), d\mathbf{A}_t \rangle_F$$

The integrand is identical for both the original and permuted inputs. Therefore, the integrals are equal:

$$F(\mathbf{P}^\top \mathbf{A}_1 \mathbf{P}) = -\int_0^1 \langle v_\theta(\mathbf{A}_t, t), d\mathbf{A}_t \rangle_F dt + C = F(\mathbf{A}_1)$$

This confirms that the scalar function F is permutation-invariant. □

C.2 Experimental details

C.2.1 Details about the architecture

Our model adopts a modified version of the adjacency score network architecture introduced in GDSS [151]. The network is a permutation-equivariant graph neural

network designed to approximate the scores $\nabla_{\mathbf{A}_t} \log p_t(X_t, \mathbf{A}_t)$ and $\nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t, \mathbf{A}_t)$ at each diffusion step; in this paper, we use only score w.r.t. $\mathbf{A}_t$. Concretely, the architecture consists of stacked message-passing layers followed by a multi-layer perceptron. Each layer propagates node and edge information through adjacency-based aggregation, ensuring equivariance under node relabeling. Time information t is incorporated by scaling intermediate activations with the variance of the forward diffusion process, following the practice in continuous-time score models. Residual connections and normalization layers are used to stabilize training. The final output is an $N \times N$ tensor matching the adjacency dimension. This design provides the required permutation-equivariance and expressive power while remaining computationally tractable for mid-sized benchmark graphs.

The modification that incorporates is a module to build an embedding for the variable t and a FiLM style modulation to incorporate noise conditioning. In particular, we incorporate the following modules:

- A positional encoding based on a sinusoidal embedding following [47]
- An MLP layer with SiLU activation per attention layer
- A modulation at each attention layer, where we scale the hidden features by an adaptive RMS norm operation [244]

C.2.2 Details about the datasets

In Table C.1 we report the statistics of the datasets used in the main text.

Table C.1 : Statistics of the datasets used for evaluation.

Dataset	# Graphs	Avg. Nodes	Avg. Edges	# Classes	Domain
ENZYMES	600	32.63	62.14	6	Bioinformatics
PROTEINS	1,113	39.06	72.82	2	Bioinformatics
IMDB-B	1,000	19.77	96.53	2	Social Network

C.2.3 Hyperparameters

Flow-based baselines

We report the hyperparameters of our model, both the architecture and training. All three baselines use the same rectified-flow architecture and optimizer family; the only substantive differences are the prior settings.

PIFM - Link Prediction, 10% masked	
`batch_size`	64 (IMDB-B & ENZYMES), 32 (PROTEINS)
`optimizer`	Adam
`learning_rate`	2e-4
`dropout`	0.2
`hidden_dim`	32
`num_layers`	5
`num_linears`	2
`channels`	{c_init: 2, c_hid: 8, c_final: 4}
`train/val/test_noise_std`	0.05 (IMDB-B), 0.1 (PROTEINS & ENZYMES)
`ode_steps` (Euler, K)	1 to 100
`prior`	GraphSAGE (default hyperparameters)

PIFM - Link Prediction, 50% masked	
`batch_size`	64 (IMDB-B & ENZYMES), 32 (PROTEINS)
`optimizer`	Adam
`learning_rate`	2e-4
`dropout`	0.2
`hidden_dim`	32
`num_layers`	5
`num_linears`	2
`channels`	{c_init: 2, c_hid: 8, c_final: 4}
`train/val/test_noise_std`	0.05 (IMDB-B & PROTEINS), 0.1 (ENZYMES)
`ode_steps` (Euler, K)	1 to 100
`prior`	GraphSAGE (default hyperparameters)

PIFM (VGAE) – Link Prediction, 50% masked	
`batch_size`	64 (IMDB-B & ENZYMES), 32 (PROTEINS)
`optimizer`	Adam
`learning_rate`	2e-4
`dropout`	0.2
`hidden_dim`	32
`num_layers`	5
`num_linears`	2
`channels`	{c_init: 2, c_hid: 8, c_final: 4}
`train/val/test_noise_std`	0.01 (IMDB-B), 0.0 (PROTEINS & ENZYMES)
`ode_steps` (Euler, K)	1 to 100
`prior`	VGAE (epochs: 200, lr: 0.01, hidden: 32/16)

Flow w/ Gaussian Prior	**Value**
`epochs`	1000
`batch_size`	64 (default), 32 (PROTEINS)
`optimizer`	Adam
`learning_rate`	2e-4
`dropout`	0.2
`hidden_dim`	32
`num_layers`	5
`num_linears`	2
`channels`	{c_init: 2, c_hid: 8, c_final: 4}
`train_noise_std` (masked t=0)	0.00
`val_noise_std` (masked t=0)	0.00
`ode_steps` (Euler, K)	1 to 100
`prior`	None (masked entries initialized from $\mathcal{N}(0.5, 1)$)

C.2.4 Metrics Calculation

We evaluate performance only on the set of masked (unknown) edges in the upper triangle of the adjacency matrix. For each test graph, all metrics are computed on these entries and then averaged across graphs.

Metrics Used in Tables We report the following four metrics in the main results:

- **Area Under the ROC Curve (AUC).** Computed on the raw predicted scores (when available). AUC measures the probability that a randomly chosen true edge receives a higher predicted score than a randomly chosen non-edge. Larger AUC indicates stronger ranking performance.

- **Average Precision (AP).** Computed from the precision–recall curve induced by ranking the predictions. AP summarizes how well the model recovers true edges across all possible thresholds, with higher values indicating better precision–recall trade-offs.

- **False Positive Rate (FPR).** After thresholding predictions at 0.5, the FPR is defined as

$$\mathrm{FPR} = \frac{\mathrm{FP}}{\mathrm{FP} + \mathrm{TN}},$$

- **False Negative Rate (FNR).** After thresholding predictions at 0.5, the FNR is defined as

$$\mathrm{FNR} = \frac{\mathrm{FN}}{\mathrm{FN} + \mathrm{TP}},$$

- **MMD.** A kernel-based method that measures the difference between two probability distributions by embedding them in a feature space and finding the maximum difference between their mean embeddings.

AUC and AP are threshold-independent metrics (computed directly on the provided scores), while FPR and FNR are threshold-dependent error rates (obtained after binarizing at 0.5). All values reported in the tables are averaged over test graphs and expressed in percentage.

C.2.5 Training cost

All models are trained on NVIDIA A100-SXM4-80GB GPUs. Once the prior is fixed, the flow model used in PIFM has essentially the same training loop and computational cost as the flow model initialized with a Gaussian prior. Thus, the main additional cost of PIFM comes from fitting the prior itself, not from the subsequent

flow-matching stage. This observation is consistent with our empirical training runs, where we did not observe noticeable differences in optimization behavior between Gaussian-prior and prior-informed flow variants across edge-drop settings. Table C.2 reports training time of the flow model for 100 epochs on each dataset, with PROTEINS taking the most time while the other 2 datasets remain similar training costs.

Table C.2 : Training time of the flow model for 100 epochs on each dataset under 50% mask rate.

Dataset	**Training Time (s)**
ENZYMES	206.4
PROTEINS	646.7
IMDB-B	243.7

C.2.6 Inference cost

Since PIFM reconstructs graphs by integrating a learned velocity field over K Euler steps, its runtime grows approximately linearly with the number of Euler steps integrated. We observe a cost of about 0.03 seconds per step per graph, so increasing K improves refinement quality at the expense of proportionally higher latency.

Table C.3 reports the per-graph inference time of PIFM as a function of the number of Euler steps. The results show near-linear scaling, from approximately 0.03 seconds in the lowest-step setting to about 2.8 seconds when using 100 steps.

Table C.3 : Per-graph inference time of PIFM as a function of the number of Euler steps.

Setting	**Inference Time per Graph (s)**
$K = 1$	0.03
$K = 10$	0.28
$K = 50$	1.40
$K = 100$	2.80

To contextualize these costs, Table C.4 compares PIFM against the standalone prior predictors on each dataset. As expected, the one-shot priors are faster because they do not require iterative refinement. However, PIFM with $K = 1$ remains relatively inexpensive while retaining smallest reconstruction error, with per-graph inference times below 0.06 seconds on all three benchmarks. This makes the single-step setting attractive in regimes where distortion is the main objective, while larger values of K are better suited to settings where perceptual refinement is more important.

Table C.4 : Per-graph inference time comparison between standalone priors and PIFM.

Method	**PROTEINS (s)**	**ENZYMES (s)**	**IMDB-B (s)**
Node2Vec Prior	0.0016	0.0020	0.0010
GraphSAGE Prior	0.0003	0.0003	0.0002
PIFM ($K = 1$)	0.0425	0.0532	0.0440
PIFM ($K = 100$)	2.8207	2.8419	2.8660

Appendix D

Supplement for Chapter 7: Energy-based models with anisotropic noise

D.1 Dual-score matching for anisotropic noise

We expand on the dual-score matching formulation for anisotropic noise introduced in Section 7.3.2. Recall that we want to learn a covariance-dependent energy function, $U_\theta(\mathbf{y}, \mathbf{\Sigma}) = -\log p(\mathbf{y}|\mathbf{\Sigma})$, to approximate the negative log-likelihood (NLL) of the noisy image distribution, for a family of covariance matrices $\mathbf{\Sigma}$. We model this energy as follows

$$U_\theta(\mathbf{y}, \mathbf{\Sigma}) = -\log p(\mathbf{y}|\mathbf{\Sigma}) = -\log\left(\int p(\mathbf{x})p(\mathbf{y}|\mathbf{x}, \mathbf{\Sigma})\mathrm{d}\mathbf{x}\right) \tag{D.1}$$

To learn this model, we use dual score matching, which requires computing the gradients w.r.t. $\mathbf{y}$ and $\mathbf{\Sigma}$.

Data score. We can use the Miyasawa-Tweedie identity [205, 206] given by

$$\nabla_{\mathbf{y}} U_\theta(\mathbf{y}, \mathbf{\Sigma}) = \mathbb{E}_{\mathbf{x}}\left[\mathbf{\Sigma}^{-1}(\mathbf{y} - \mathbf{x})\,|\mathbf{y}\right]. \tag{D.2}$$

We define the anisotropic denoising score matching loss as

$$\tilde{\ell}_{\mathrm{A-DSM}} = \mathbb{E}_{\mathbf{x},\mathbf{y}}\left[\|\nabla_{\mathbf{y}} U_\theta(\mathbf{y}, \mathbf{\Sigma}) - \mathbf{\Sigma}^{-1}(\mathbf{y} - \mathbf{x})\|_2^2\right] \tag{D.3}$$

To have a scale-invariant loss, we reweight by $\mathbf{\Sigma}^{\frac{1}{2}}$:

$$\ell_{\mathrm{A-DSM}} = \mathbb{E}\left[\|\mathbf{\Sigma}^{\frac{1}{2}}\nabla_{\mathbf{y}} U_\theta(\mathbf{y}, \mathbf{\Sigma}) - \mathbf{\Sigma}^{-\frac{1}{2}}(\mathbf{y} - \mathbf{x})\|^2\right]. \tag{D.4}$$

Covariance score. To obtain this second score, we differentiate the energy in (D.1) w.r.t. $\mathbf{\Sigma}$. Given that $p(\mathbf{y}|\mathbf{x}, \mathbf{\Sigma}) = e^{-\frac{1}{2}(\mathbf{y}-\mathbf{x})^\top \mathbf{\Sigma}^{-1}(\mathbf{y}-\mathbf{x}) - \frac{d}{2}\log(2\pi) - \frac{1}{2}\log\det(\mathbf{\Sigma})}$, we obtain

$$\begin{aligned}\nabla_{\mathbf{\Sigma}} U_\theta(\mathbf{y}, \mathbf{\Sigma}) &= \mathbb{E}_{\mathbf{x}}\left[-\nabla_{\mathbf{\Sigma}} \log p(\mathbf{y}|\mathbf{x}, \mathbf{\Sigma}) | \mathbf{y}, \mathbf{\Sigma}\right] \\ &= \mathbb{E}_{\mathbf{x}}\left[-\nabla_{\mathbf{\Sigma}}\left(-\frac{1}{2}(\mathbf{y}-\mathbf{x})^\top \mathbf{\Sigma}^{-1}(\mathbf{y}-\mathbf{x}) - \frac{d}{2}\log(2\pi) - \frac{1}{2}\log\det(\mathbf{\Sigma})\right) \middle| \mathbf{y}, \mathbf{\Sigma}\right].\end{aligned} \tag{D.5}$$

The two gradient terms can be simplified as follows:

$$\nabla_{\mathbf{\Sigma}}\left[-\frac{1}{2}(\mathbf{y}-\mathbf{x})^\top \mathbf{\Sigma}^{-1}(\mathbf{y}-\mathbf{x})\right] = \frac{1}{2}\mathbf{\Sigma}^{-1}(\mathbf{y}-\mathbf{x})(\mathbf{y}-\mathbf{x})^\top \mathbf{\Sigma}^{-1} \tag{D.6}$$

$$\nabla_{\mathbf{\Sigma}}\left[-\frac{1}{2}\log\det\mathbf{\Sigma}\right] = -\frac{1}{2}\mathbf{\Sigma}^{-1}. \tag{D.7}$$

Substituting (D.6) and (D.7) into (D.5) gives

$$\nabla_{\mathbf{\Sigma}} U_\theta(\mathbf{y}, \mathbf{\Sigma}) = \mathbb{E}_{\mathbf{x}}\left[\frac{1}{2}\mathbf{\Sigma}^{-1} - \frac{1}{2}\mathbf{\Sigma}^{-1}(\mathbf{y}-\mathbf{x})(\mathbf{y}-\mathbf{x})^\top \mathbf{\Sigma}^{-1} \middle| \mathbf{y}, \mathbf{\Sigma}\right]. \tag{D.8}$$

Finally, we define the anisotropic covariance score matching loss as

$$\tilde{\ell}_{\text{A}-\text{CSM}} = \mathbb{E}_{\mathbf{x},\mathbf{y}}\left[\left\| \nabla_{\mathbf{\Sigma}} U_\theta(\mathbf{y}, \mathbf{\Sigma}) - \left(\frac{1}{2}\mathbf{\Sigma}^{-1} - \frac{1}{2}\mathbf{\Sigma}^{-1}(\mathbf{y}-\mathbf{x})(\mathbf{y}-\mathbf{x})^\top \mathbf{\Sigma}^{-1}\right)\right\|_F^2\right], \tag{D.9}$$

Similarly to the data score, we weight the loss by $\mathbf{\Sigma}^{\frac{1}{2}}$ on both sides to be scale-invariant:

$$\ell_{\text{A}-\text{CSM}} = \mathbb{E}_{\mathbf{x},\mathbf{y},\mathbf{\Sigma}}\left[\left\| \mathbf{\Sigma}^{\frac{1}{2}} \nabla_{\mathbf{\Sigma}} U_\theta(\mathbf{y}, \mathbf{\Sigma}) \mathbf{\Sigma}^{\frac{1}{2}} - \frac{1}{2}\mathbf{I} + \frac{1}{2}\mathbf{\Sigma}^{-\frac{1}{2}}(\mathbf{y}-\mathbf{x})(\mathbf{y}-\mathbf{x})^\top \mathbf{\Sigma}^{-\frac{1}{2}}\right\|_2^2\right]. \tag{D.10}$$

Final loss. Now that we have the dual-score matching loss, we define an overall objective by integrating over all possible covariance matrices $\mathbf{\Sigma}$. Directly extending the combined loss from the paper yields

$$\ell(\theta, \mathbf{\Sigma}) = \mathbb{E}_{\mathbf{\Sigma}}\left[k_1 \ell_{\text{A}-\text{DSM}}(\theta, \mathbf{\Sigma}) + k_2 \ell_{\text{A}-\text{CSM}}(\theta, \mathbf{\Sigma})\right] \tag{D.11}$$

where k_1, k_2 are constants that normalize each term with respect to the ambient dimension d. In particular, we consider $k_1 = 1/d$ and $k_2 = 1/d^2$:

$$\ell(\theta, \mathbf{\Sigma}) = \mathbb{E}_{\mathbf{\Sigma}} \left[\frac{1}{d} \ell_{\text{A-DSM}}(\theta, \mathbf{\Sigma}) + \frac{1}{d^2} \ell_{\text{A-CSM}}(\theta, \mathbf{\Sigma}) \right] \tag{D.12}$$

While this loss is appealing, it is impractical due to the size of the covariance. As mentioned in Section 7.3.2, we consider a diagonal matrix instead.

Diagonal $\mathbf{\Sigma}$ with independent parameters. We consider diagonal matrices where $\mathbf{\Sigma} = \text{diag}(\mathbf{\Phi})$ with $\mathbf{\Phi} = [\Phi_1, \ldots, \Phi_d]^\top$. Each parameter is independent with prior $p(\mathbf{\Phi}) = \prod_{i=1}^d p(\Phi_i)$, where $p(\Phi_i) \propto \Phi_i^{-1}$ for $\Phi_i \in [\Phi_{\min}, \Phi_{\max}]$.

To compute the gradient with respect to $\mathbf{\Phi}$, we utilize the chain rule. Note that $\frac{\partial \operatorname{vec}(\mathbf{\Sigma})}{\partial \Phi_i}$ is a vector of length d^2 with a 1 at the position corresponding to the i-th diagonal element Σ_{ii}, and 0 elsewhere. For a generic function $U_\theta(\mathbf{y}, \mathbf{\Sigma})$, the gradient component i is:

$$\left[\nabla_{\mathbf{\Phi}} U_\theta(\mathbf{y}, \mathbf{\Sigma}(\mathbf{\Phi}))\right]_i = \operatorname{vec}\left(\nabla_{\mathbf{\Sigma}} U_\theta(\mathbf{y}, \mathbf{\Sigma})\right)^\top \cdot \frac{\partial \operatorname{vec}(\mathbf{\Sigma})}{\partial \Phi_i} \tag{D.13}$$

$$= \left[\nabla_{\mathbf{\Sigma}} U_\theta(\mathbf{y}, \mathbf{\Sigma})\right]_{ii} \tag{D.14}$$

Specifically, for the log-determinant term in (D.5), the gradient is:

$$\left[\nabla_{\mathbf{\Phi}} \left(-\frac{1}{2} \log\det \mathbf{\Sigma}\right)\right]_i = \operatorname{vec}\left(-\frac{1}{2}\mathbf{\Sigma}^{-1}\right)^\top \cdot \frac{\partial \operatorname{vec}(\mathbf{\Sigma})}{\partial \Phi_i} \tag{D.15}$$

$$= -\frac{1}{2}(\mathbf{\Sigma}^{-1})_{ii} = -\frac{1}{2\Phi_i} \tag{D.16}$$

Thus, in vector form, we obtain the simplified result:

$$\nabla_{\mathbf{\Phi}} \left[-\frac{1}{2} \log\det \mathbf{\Sigma}\right] = -\frac{1}{2}\mathbf{\Phi}^{-1} \tag{D.17}$$

For the quadratic term, we have the following (assuming $\mathbf{z} = (\mathbf{y} - \mathbf{x})$)

$$\nabla_{\mathbf{\Phi}} \left[-\frac{1}{2}(\mathbf{y}-\mathbf{x})^\top \mathbf{\Sigma}^{-1}(\mathbf{y}-\mathbf{x}) \right] = \operatorname{vec}\left(\frac{1}{2}\mathbf{\Sigma}^{-1}\mathbf{z}\mathbf{z}^\top \mathbf{\Sigma}^{-1} \right) \cdot \frac{\partial \operatorname{vec}(\mathbf{\Sigma})}{\partial \mathbf{\Phi}}, \tag{D.18}$$
$$= \frac{1}{2} \operatorname{vec}\left((\mathbf{\Phi}^{-1} \circ \mathbf{z})(\mathbf{\Phi}^{-1} \circ \mathbf{z})^\top \right) \cdot \frac{\partial \operatorname{vec}(\mathbf{\Sigma})}{\partial \mathbf{\Phi}},$$

where $\circ$ is the element-wise product. The i-th element is

$$\left[\nabla_{\mathbf{\Phi}} \left[-\frac{1}{2}(\mathbf{y}-\mathbf{x})^\top \mathbf{\Sigma}^{-1}(\mathbf{y}-\mathbf{x}) \right] \right]_i = \left[\frac{1}{2} \operatorname{vec}\left((\mathbf{\Phi}^{-1} \circ \mathbf{z})(\mathbf{\Phi}^{-1} \circ \mathbf{z})^\top \right) \right]_{ii}. \tag{D.19}$$

Combining both (D.19) and (D.17), we get

$$\nabla_{\mathbf{\Phi}} U_\theta(\mathbf{y}, \mathbf{\Sigma}(\mathbf{\Phi})) = \mathbb{E}_{\mathbf{x}} \left[\frac{1}{2}\mathbf{\Phi}^{-1} - \frac{1}{2}(\mathbf{\Phi}^{-1} \circ \mathbf{z})^2 \middle| \mathbf{y}, \mathbf{\Sigma}(\mathbf{\Phi}) \right]. \tag{D.20}$$

The training algorithm is described in Alg. 8. Notice we are using $\lambda = d^{-\frac{1}{2}}$ in (D.12), which corresponds to the loss in the main paper.

D.1.1 Algorithms

Algorithm 8 Training

repeat

- $\mathbf{x} \sim p(\mathbf{x})$ (This is the dataset given beforehand)
- $\mathbf{\Phi} \sim p(\mathbf{\Phi})$
- $\mathbf{v} \sim \mathcal{N}(0, \mathbf{I})$
- $\mathbf{y} = \mathbf{x} + \mathbf{\Sigma}(\mathbf{\Phi})^{1/2}\mathbf{v}$
- Compute $\ell_{\text{A}-\text{DSM}}(\theta, \mathbf{\Sigma}(\mathbf{\Phi})) = \frac{1}{N}\sum_{\mathbf{x},\mathbf{y}} \|\nabla_{\mathbf{y}} U_\theta(\mathbf{y}, \mathbf{\Sigma}(\mathbf{\Phi})) - \mathbf{\Phi}^{-1} \circ (\mathbf{y}-\mathbf{x})\|^2_{\frac{\mathbf{\Phi}}{d}}$
- Compute $\ell_{\text{A}-\text{CSM}}(\theta, \mathbf{\Sigma}(\mathbf{\Phi})) = \frac{1}{N}\sum_{\mathbf{x},\mathbf{y}} \left\| \nabla_{\mathbf{\Phi}} U_\theta(\mathbf{y}, \mathbf{\Sigma}(\mathbf{\Phi})) - \left(\frac{1}{2}\mathbf{\Phi}^{-1} - \frac{1}{2}(\mathbf{\Phi}^{-1} \circ (\mathbf{y}-\mathbf{x}))^2 \right) \right\|_{\frac{\mathbf{\Phi}^2}{d^2}}$
- Combine $\ell(\theta) = \ell_{\text{A}-\text{DSM}}(\theta, \mathbf{\Sigma}(\mathbf{\Phi})) + \ell_{\text{A}-\text{CSM}}(\theta, \mathbf{\Sigma}(\mathbf{\Phi}))$
- Perform one gradient descent step on $\ell(\theta)$ w.r.t. θ

until Convergence

Algorithm 9 Predictor–Corrector Sampling

Require: Noise schedule $\{\mathbf{\Sigma}_0, \dots, \mathbf{\Sigma}_T\}$, number of corrector steps n_{steps}, energy model U_θ, measurement $\mathbf{y}$, binary matrix with the structure of the degradation $\mathbf{\Sigma}_{id} = \text{clamp}(\mathbf{H}^{-1}, 0, 1)$, r, temperature η

Initialize $\mathbf{x}_T \leftarrow \mathbf{y}$
for $c = T - 1$ to 0 **do**
  // PREDICTOR STEP.
  Compute score gradient $\mathbf{g}_c = -\nabla_{\mathbf{x}_c} U_\theta(\mathbf{x}_c, \mathbf{\Sigma}_c)$
  if Schedule = Adaptive **then**
    Compute covariance gradients $\mathbf{g}_{\Sigma_c} = -\nabla_{\mathbf{\Sigma}} U_\theta(\mathbf{x}_{c,0}, \mathbf{\Sigma}_c)$
    Covariance gradient descent + PSD constraint:

$$\mathbf{\Sigma}_{c+1} \leftarrow \max(\mathbf{\Sigma}_c - \eta_\Sigma \, \mathbf{\Sigma} \circ \mathbf{g}_\Sigma \circ \mathbf{\Sigma}, \varepsilon \mathbf{I})$$

    Define $\Delta\mathbf{\Sigma} = \mathbf{\Sigma}_c \circ \mathbf{g}_{\Sigma_c} \circ \mathbf{\Sigma}_c$
  else if Schedule = Fixed **then**
    $\Delta\mathbf{\Sigma} = \mathbf{\Sigma}_c - \mathbf{\Sigma}_{c+1}$
  end if
  Sample noise $\mathbf{z} \sim \mathcal{N}(0, \mathbf{I})$

$$\mathbf{x}_{c+1,0} \leftarrow \mathbf{x}_c + \Delta\mathbf{\Sigma} \circ \mathbf{g} + \Delta\mathbf{\Sigma}^{1/2} \circ \mathbf{z}$$

  // CORRECTOR STEP.
  for $s = 0$ to n_{steps} **do**
    Compute score gradient $\mathbf{g}_{c+1,s} = -\nabla_{\mathbf{x}_{c+1,s}} U_\theta(\mathbf{x}_{c+1,s}, \mathbf{\Sigma}_{c+1})$
    Sample noise $\mathbf{z} \sim \mathcal{N}(0, \mathbf{I})$
    if Domain = pixel **then**
      Compute step size $\epsilon \leftarrow \left(\frac{\text{r} \cdot \|\mathbf{\Sigma}_{id} \circ \mathbf{z}\|}{\|\mathbf{\Sigma}_{id} \circ \mathbf{g}\|}\right)^2 \cdot 2$
      Construct element-wise step size $\mathbf{\Sigma}_\epsilon = \mathbf{\Sigma}_{id} \circ \epsilon$

$$\mathbf{x}_{c+1,s+1} \leftarrow \mathbf{x}_{c+1,s} + \mathbf{\Sigma}_\epsilon \circ \mathbf{g}_{c+1,s} + \sqrt{2\,T}\, \mathbf{\Sigma}_\epsilon^{1/2} \circ \mathbf{z}$$

    else if Domain = frequency **then**
      Compute step size $\epsilon \leftarrow \left(\frac{\text{r} \cdot \|\mathbf{z}\|}{\|\mathbf{g}\|}\right)^2 \cdot 2$

$$\mathbf{x}_{c+1,s+1} \leftarrow \mathbf{x}_{c+1,s} + \epsilon \left(\mathbf{g}_{c+1,s} + \nabla_{\mathbf{x}_t} \log p(\mathbf{x}_T = y | \mathbf{x}_{c+1,s})\right) + \sqrt{2\,T\,\epsilon}\mathbf{z}$$

    end if
    if MALA **then**
      Compute Metropolis–Hastings acceptance probability $\alpha = \min\left(1, \frac{\exp\left(-U_\theta(\mathbf{x}^\star, \mathbf{\Sigma}_{c+1})\right) q(\mathbf{x}_{c+1,s} | \mathbf{x}^\star)}{\exp\left(-U_\theta(\mathbf{x}_{c+1,s}, \mathbf{\Sigma}_{c+1})\right) q(\mathbf{x}^\star | \mathbf{x}_{c+1,s})}\right)$
      Sample $u \sim \mathcal{U}(0, 1)$ and accept/reject

$$\mathbf{x}_{c+1,s+1} \leftarrow \begin{cases} \mathbf{x}^\star, & u < \alpha \\ \mathbf{x}_{c+1,s}, & \text{otherwise} \end{cases}$$

    end if
  end for
end for
Return $\mathbf{x}_{0,n_{steps}}$

D.2 Energy-guided sampling as a Bregman mirror descent

While the gradient $\nabla_{\mathbf{\Sigma}} U_\theta(\mathbf{y}, \mathbf{\Sigma})$ corresponds to the steepest descent with respect to the Euclidean norm, it is more natural here to consider the steepest descent with respect to the Bregman divergence generated by the negative log determinant:

$$D(\mathbf{\Sigma} \,\|\, \mathbf{\Sigma}') = \operatorname{tr}(\mathbf{\Sigma}'^{-1}\mathbf{\Sigma}) - \log\det(\mathbf{\Sigma}'^{-1}\mathbf{\Sigma}) - d. \tag{D.21}$$

Note that this Bregman divergence corresponds to twice the KL divergence between two zero-mean Gaussian distributions with covariance $\mathbf{\Sigma}$ and $\mathbf{\Sigma}'$. For a stepsize parameter $\gamma > 0$, a Bregman mirror descent step computes

$$\operatorname*{argmin}_{\mathbf{\Sigma}'} U_\theta(\mathbf{y}, \mathbf{\Sigma}') + \frac{1}{\gamma} D(\mathbf{\Sigma}' \,\|\, \mathbf{\Sigma}) = (\mathbf{\Sigma}^{-1} + \gamma \nabla_{\mathbf{\Sigma}} U_\theta(\mathbf{y}, \mathbf{\Sigma}))^{-1} \tag{D.22}$$

$$= \mathbf{\Sigma} - \gamma \mathbf{\Sigma} \nabla_{\mathbf{\Sigma}} U_\theta(\mathbf{y}, \mathbf{\Sigma}) \mathbf{\Sigma} + o(\gamma), \tag{D.23}$$

where the last step is a first-order Taylor approximation for small stepsizes $\gamma \to 0$. This leads to the update

$$\delta\mathbf{\Sigma} = \mathbf{\Sigma} \nabla_{\mathbf{\Sigma}} U(\mathbf{y}, \mathbf{\Sigma}) \mathbf{\Sigma}. \tag{D.24}$$

D.3 Implementation

D.3.1 Architecture and training hyperparameters

Our architecture is a modification of the SongNet NCSNPP network [7] implemented in [47]. We use the same hyperparameters as the original implementation, which are reported in Table D.1; the only modification is the noise embedding, which we describe below. We also describe in Table D.2 the hyperparameters of the training. The MNIST classifier used in 7.6 is a CNN with 3 layers and ReLU activation functions.

Table D.1 : SongUNet Architecture Summary

Variable/Module	**Configuration**
Network Parameters	
Image Resolution	64×64
Input/Output Channels	3
Model Channels	128
Channel Multipliers	[1, 2, 2, 2]
Embedding Channels	512 (128×4)
Number of Blocks per Resolution	4
Attention Resolutions	[16]
Dropout	0.10
Embedding Configuration	
Embedding Type	Positional
Noise Channel Multiplier	2
Noise Embedding Channels	256 (128×2)
Anisotropic Noise	True
Anisotropic Gamma Embedding Network	
Fourier Dimension	256
Base Channels	128
Residual Blocks	4
Output Channels	512
Time Range	$[10^{-9}, 10^{3}]$
Encoder Architecture	
Resolution Levels	64, 32, 16, 8
Channels per Level	128, 256, 256, 256
Encoder Type	Standard
Decoder Architecture	
Decoder Type	Standard
Skip Connections	Yes
Adaptive Scaling	False
Resampling Filter	[1, 3, 3, 1]
UNet Block Configuration	
Attention Heads	1 (at 16×16)
Skip Scale	$\sqrt{0.5}$
Normalization	GroupNorm ($\epsilon = 10^{-6}$)

Noise Embedding Modulation. Given the diagonal noise of the covariance $\text{diag}(\mathbf{\Sigma}_t)$, we first compute embeddings:

$$\mathbf{e}_{\text{spatial}} \leftarrow f_{\text{spatial}}(\text{diag}(\mathbf{\Sigma}_t)) \in \mathbb{R}^{B \times c_{\text{emb}} \times H \times W}, \tag{D.25}$$

$$\mathbf{e}_{\text{frequency}} \leftarrow f_{\text{frequency}}(\text{diag}(\mathbf{\Sigma}_t)) \in \mathbb{R}^{B \times c_{\text{emb}} \times 1 \times 1}, \tag{D.26}$$

where $f_{\text{spatial}}(.)$ and $f_{\text{frequency}}(\mathbf{\Sigma}_t)$ are two residual architectures (ResNet) [245], consisting of 3×3 convolutions paired with Group Normalization and SiLU activations, and a residual connection. Given that the main architecture (a UNet) performs down and up-sampling operations both in the channel and in the spatial domain, we need to accommodate the dimensions of the embedding network to each layer ℓ. For the channel dimension, we use 1×1 convolutional layers as follows:

$$\mathbf{e}_{\ell,\text{spatial}} \leftarrow \text{Conv}_{1\times1}(\mathbf{e}_{\text{spatial}}) \in \mathbb{R}^{B \times c_\ell \times H \times W} \tag{D.27}$$

$$\mathbf{e}_{\ell,\text{frequency}} \leftarrow \text{Conv}_{1\times1}(\mathbf{e}_{\text{frequency}}) \in \mathbb{R}^{B \times c_\ell \times 1 \times 1} \tag{D.28}$$

For the spatial dimensions, we use the following operations

$$\mathbf{e}_{\ell,\text{spatial}} \leftarrow \text{Interpolate}(\mathbf{e}_{\ell,\text{spatial}}) \in \mathbb{R}^{B \times c_\ell \times H_\ell \times W_\ell} \tag{D.29}$$

Given the embedding vector for each layer, we apply a multiplicative conditioning mechanism, given by

$$\mathbf{x}_\ell = \text{SiLU}\left(\text{GroupNorm}(\mathbf{x}_\ell) \odot (1 + \mathbf{e}_{\ell,\text{spatial/frequency}})\right) \tag{D.30}$$

where GroupNorm(.) is the group normalization, with a number of channels in each group of $\min(32, c_\ell/4)$.

Table D.2 : Training hyperparameters for CelebA and ImageNet64

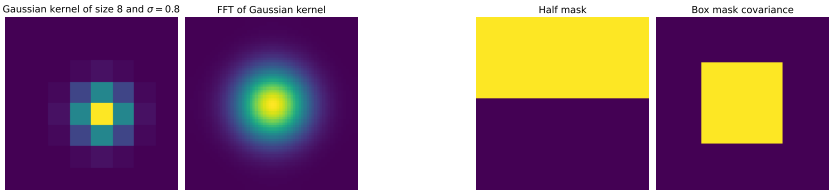
Hyperparameter	**Value**
Number of GPUs	6
Dataset	CelebA/ImageNet64
Training batch size	128
σ^2_{min}	1×10^{-9}
σ^2_{max}	1×10^{3}
Learning rate	2×10^{-4}
Total steps	400,000
LR decay	100,000 steps
Warmup steps	1,000
Grad. clipping	Norm = 20

Table D.3 : Training hyperparameters for AFHQ

Hyperparameter	**Value**
Number of GPUs	8
Dataset	AFHQ (192×192)
Training batch size	64
σ^2_{min}	1×10^{-9}
σ^2_{max}	1×10^{3}
Learning rate	2×10^{-4}
Total steps	100,000
Warmup steps	1,000
Grad. clipping	Norm = 20

D.3.2 Distribution covariances used for training

As described in Section 7.3.3, we used four types of covariances: Gaussian deblurring, shown in Fig. D.1a, super-resolution (a combination of Gaussian kernel with a down-sampling operation), and box and half-mask inpainting, illustrated in Fig. D.1b. The family of autoregressive patch-based covariances used in the MNIST experiments in Section 7.4.2 are shown in Fig. D.2.



(a) Gaussian Kernel **(b)** Box and half-mask inpainting

Figure D.1 : Examples of particular $\mathbf{\Sigma}_t$ for different kernels and masks.

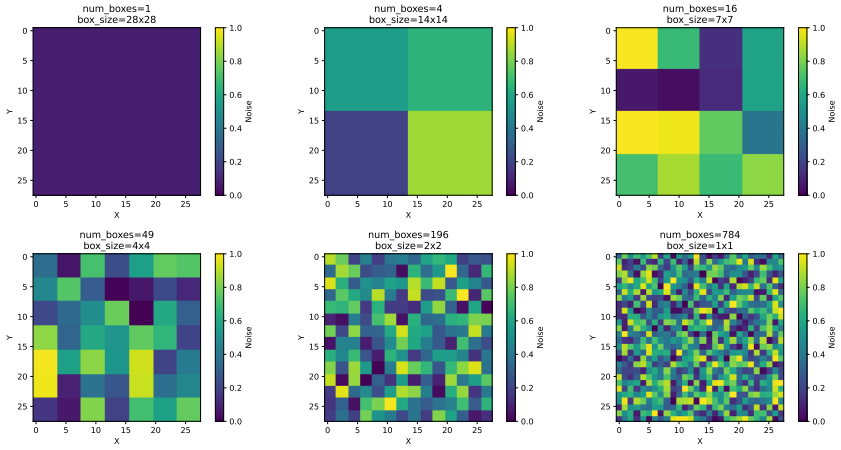


Figure D.2 : Examples of autoregressive patch-based covariances $\mathbf{\Sigma}_t$ used for training. We first sample the number of boxes, and then the variance of each box.

D.3.3 Comparison with previous energy-based models

Our method is more computationally intensive to train than standard score-based models. This overhead primarily comes from double backpropagation, which is inherent to energy-based formulations and is widely acknowledged in prior work [48, 188, 246, 247]. In this sense, the increased cost is not specific to our method, but rather

a known trade-off when learning energy models. That said, we further contextualize below that our model has a computational cost comparable to prior work when accounting for model size and resolution.

1. We use the Song architecture from EDM ($\approx$ 70M parameters), which is substantially larger than the network used in the original DSM paper [189] ($\approx$ 13M parameters). The larger architecture naturally increases training time, while also yielding improved performance. For reference, EDM training at 64×64 takes approximately 4 days on 8 V100 GPUs ($\approx$ 4 A100s) with a batch size of 256 (see `https://github.com/NVlabs/edm`). Scaling to 192×192 increases the pixel count by $9\times$, which significantly increases the computational cost, either in terms of GPU count or total training time.

2. This computational challenge is shared by energy-based models more broadly:

   - Thornton [188] used 8 A100 GPUs at 64×64 resolution (without the dual objective).
   - Balcerak [246] trained at 64×64 on CelebA with 4 A100s and a batch size of 32; neither [188] nor [246] report total training time.
   - Boffi [247] required approximately 8 days for the flow matching teacher and 6 days for the shortcut model on ImageNet 64×64.
   - Du [48] trained at 128×128 using TPU clusters, making direct comparison difficult.

D.3.4 Details of baselines

We describe the setup for our method in Table D.4.

Table D.4 : Inpainting - Hyperparameters for the Predictor-Corrector Sampler with fixed corrector.

Parameter	**Value**	**Description**
σ_{begin}	25	Initial (maximum) noise standard deviation inside the mask.
σ_{end}	10^{-3}	Final (minimum) noise standard deviation inside the mask.
T	600	Number of discretized noise levels.
σ_{dist}	Geometric	Progression type of the noise schedule.
N_{steps}	1	Number of corrector steps per noise level.
r	0.15	step-size for the Langevin update.
ζ (temp)	0.9	Temperature scaling for score-based sampling.

Computational cost for inference. We expand on the running times for Gaussian deblurring across all methods, considering the running time for sampling a batch of 16 images at 64×64:

Sampler	Time [sec]	Computational cost (NFE)
Ours	139	(N_p) NFEs + (N_p) grad w.r.t. input for energy
DPS	167	N NFEs + N grad w.r.t. denoiser
RED-Diff	22	N NFEs

Importantly, the inference time of our method is comparable to DPS and standard diffusion samplers, so the computational overhead is concentrated entirely in training and does not affect sampling time. In contrast to these methods, our energy formualtion learns a full posterior density. We believe this makes the overhead modest

relative to the added capabilities our approach unlocks, such as blind estimation and normalization constant computation, which are not possible with DPS or RED-Diff.

For sampling, the additional $\nabla_y U_\theta$ gradient is similar in cost to DPS's backpropagation and architectural overhead is minimal. Thus, wall-clock time is approximately equal at the same NFEs. For AFHQ 192×192: DPS 55 sec, ours 74 sec.

D.4 Additional experiments

D.4.1 Additional inverse problems

In-distribution covariances

We include the results for **super-resolution** $\times 4$ and **horizontal half-mask inpainting** in Table D.5; visual results can be found in Appendix D.4.6.

For super-resolution, we do not use corrector steps; in terms of numerical results, our method generates outperforms plug-and-play by a significant margin, similarly to the deblurring case. As illustrated in Fig. D.19, the generated images present some artifacts. We attribute this partly to the known limitations of the frequency-domain conditioning: our frequency-domain degradations are defined with a Fourier transform with periodic boundary conditions, which can cause border effects. For practical use, a more careful implementation with more advanced boundary handling (e.g., a suitable discrete cosine transform) should be preferred.

Regarding half-mask inpainting, our method achieves better FID than baselines. However, in contrast to box inpainting, our method have a similar LPIPS than plug-and-play baselines.

Table D.5 : Experiments on CelebA 64×64: Comparison on super-resolution $\times 4$ and half-mask inpainting.

Sampler	Super-resolution $\times 4$			Half-mask Inpainting		
	PSNR↑	**LPIPS↓**	**FID↓**	**PSNR↑**	**LPIPS↓**	**FID↓**
DPS [2]	15.38	0.09	74.31	13.99	0.11	29.22
RED-Diff [50]	15.22	0.08	**64.75**	**15.65**	0.10	34.84
Ours	**22.41**	**0.01**	93.72	14.00	**0.10**	**28.15**

Out-of-distribution covariances

In this subsection, we illustrate how our method can handle out-of-distribution degradations, i.e., those inverse problems with an associated covariance $\mathbf{\Sigma}$ not used for training. In particular, we consider **random inpainting**, where we mask out a 70% of pixels, **vertical half mask inpainting**, with a mask of size of 64×30, and **motion deblurring**; for the latter, we considered the kernel from `https://github.com/LeviBorodenko/motionblur`of size 15×15 and intensity 0.3, and $\sigma_v = 2 \times 10^{-3}$; and for scheduling, we use a Gaussian deblurring kernel of size 15×15 and intensity 0.8, which serves as an approximation. The results are shown in Table D.6 and Table D.7. Remarkably, we observe that our models achieves a similar performance that plug-and-play methods.

Table D.6 : Experiments on CelebA 64 × 64: Comparison on random inpainting (0.7%) and vertical half-mask inpainting.

Sampler	Random Inpainting (0.7%)			Vertical-half Mask		
	PSNR↑	**LPIPS↓**	**FID↓**	**PSNR↑**	**LPIPS↓**	**FID↓**
DPS [2]	29.58	**0.01**	**21.83**	14.21	0.085	**31.54**
RED-Diff [50]	**30.25**	**0.01**	22.20	**15.80**	**0.083**	38.78
Ours	28.90	**0.01**	23.54	13.26	0.090	32.22

Table D.7 : Experiments on CelebA 64 × 64: Comparison on motion deblurring.

Sampler	**PSNR [dB]↑**	**LPIPS↓**	**FID↓**	**DISTS↓**
DPS	28.63	0.02	45.65	0.094
RED-Diff	**31.00**	0.02	**44.33**	0.085
Ours	29.42	**0.007**	50.6	**0.083**

Large-scale experiments

In this subsection, we include experiments on AFHQ-Cat 192 × 192, demonstrating that our proposed model can be trained on large-scale images. To simplify training, we consider only inpainting with a center-box of sizes $s \times s$, with values from $s = 20$ to $s = 50$, and half-mask with sizes $s \times 64$, for the same values of s.

Comparison with baselines. We compare with DPS for a box inpainting task; the result is shown in Tables D.9 and D.11.

Table D.8 : Box 30×30

Sampler	**Inpainting** (box of 30×30)			
	PSNR [dB] ↑	**LPIPS** ↓	**FID** ↓	**DISTS** ↓
Ours	33.78	**0.005**	**3.04**	**0.0095**
DPS	31.91	0.007	3.59	0.0113
RED-Diff	**37.58**	0.008	4.81	0.0128

Table D.9 : Experiments on CelebA 64×64: Comparison on box inpainting 30×30.

Table D.10 : Box 50×50

Sampler	**Inpainting** (box of 30×30)			
	PSNR [dB] ↑	**LPIPS** ↓	**FID** ↓	**DISTS** ↓
Ours	26.45	**0.021**	**9.43**	**0.029**
DPS	25.21	0.023	10.71	0.031
RED-Diff	**28.22**	0.028	13.28	0.037

Table D.11 : Experiments on CelebA 64×64: Comparison on box inpainting 50×50.

Blind experiment. We also consider a blind experiment, following the setup from Section 7.4.3. In particular, we consider $\sigma_2 = 10^{-4}$, and $\sigma_1, s = \{1, 30\}$. The result is illustrated in Fig. D.4; the value of the estimated σ_1 is 0.2.

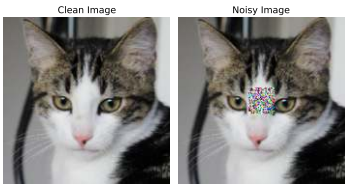


Figure D.3 : Noisy observation with a box covariance of size 30×30 and $\sigma_1 = 1$.

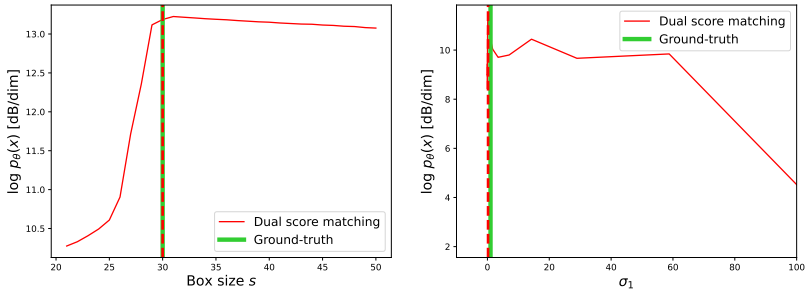


Figure D.4 : Blind reconstruction experiment. Log probability $\log p_\theta(\mathbf{y}|\mathbf{\Sigma}_s)$ as a function of box size s and σ_1, respectively. The energy model trained with dual score matching correctly identifies the true box size (dashed red for estimated size and solid green vertical lines).

D.4.2 Comparison between dual and single score matching

Blind experiment. Validating the correctness of the dual score matching is difficult since we do not have access to the true density. However, we can use some experiments to assess this, like the blind experiment in Section 7.4.3. In particular, this experiment aims to find the covariance that best explains a given measurement, relying on a proper density across different covariances. While in Section 7.4.3 we ony compared the dual score matching with the ground truth, here we add the single objective to show that, in this additional case, the estimation fails; this is illustrated in Fig. D.5.

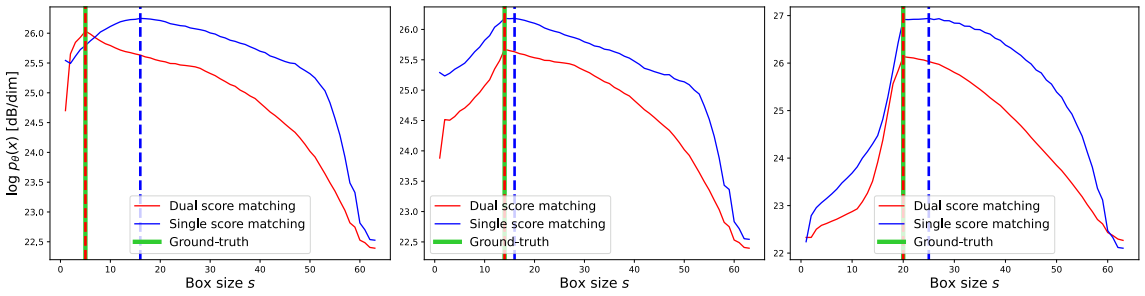


Figure D.5 : Log probability $\log p_\theta(\mathbf{y}|\mathbf{\Sigma}_s)$ as a function of box size s for $s = \{5, 14, 20\}$ with $\sigma_1 = \{2, 0.1, 2\}$ respectively. The energy model trained with dual score matching correctly identifies the true box size (dashed red for estimated size and solid green vertical lines), which is not the case for the model trained with single score matching (dashed blue vertical line).

One-shot conditional MMSE estimator. One advantage of our proposed method is that we can compute $\mathbb{E}[\mathbf{x}|\mathbf{y}, \mathbf{\Sigma}]$ in one shot. We illustrate this for a denoising task with box of size 20×20 (the outside does not have noise), where we evaluate the one-shot denoiser for multiple noise level in the box. The ablation is shown in Fig. D.6, where we compute the MSE as a function of σ_1^2, the noise inside the box. In this experiment, we also consider our proposed energy model trained with the single objective (only A-DSM). From this experiment, we observe that the energy-model trained with the dual-objective yields a better model in terms of denoising performance, showcasing that the regularization not only allows learning a normalized density model, but also helps for improving the denoising performance. To complement this experiment, we illustrate a few visual examples in Fig. D.7 using our energy model trained with the dual objective, and for a covariance with box of size 10×10, $\sigma_1 = 0.5$ (inside the box) and $\sigma_2 = 0.1$ (outside the box).

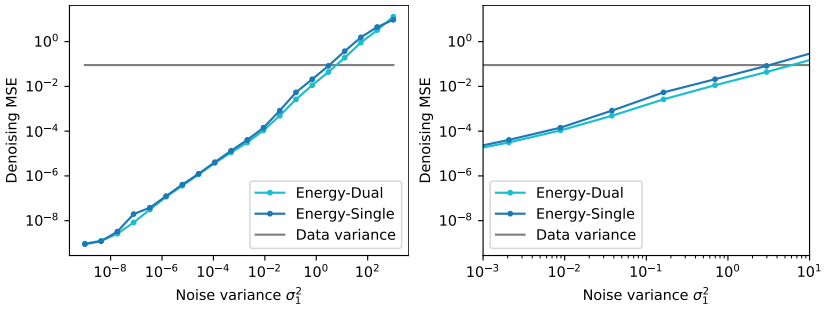


Figure D.6 : MMSE for one-shot denoising in box inpainting (10×10) as a function of the noise in the box.



Figure D.7 : MMSE estimator computed by our model, which corresponds to $\hat{\mathbf{x}} = \mathbb{E}[\mathbf{x}|\mathbf{y}, \boldsymbol{\Sigma}]$.

D.4.3 Analysis of the posterior distribution

CelebA. We include here additional results for the analysis of the posterior distribution described in Section 7.4.1. We start showing the distribution across multiple $(\mathbf{x}, \mathbf{y})$ in Fig. D.8, with generated images sorted from high to low probability in Fig. D.9. Notice that the behavior is similar to the single case, where our energy model is the one closer the posterior (left figure).

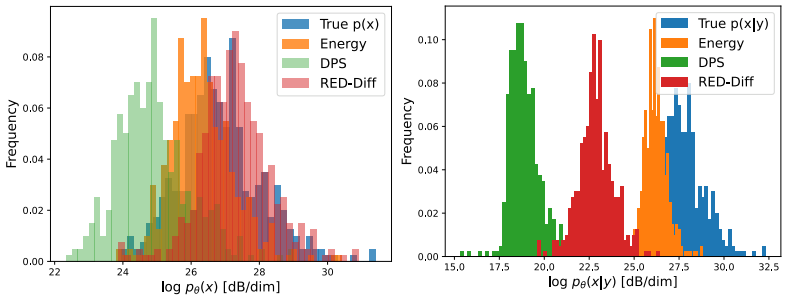


Figure D.8 : Histograms of $\log p_\theta(\hat{\mathbf{x}})$ and $\log p_\theta(\hat{\mathbf{x}}|\mathbf{y})$ for inpainting solutions $\hat{\mathbf{x}}$ generated by DPS, RED-Diff, and our energy model for different measurements $\mathbf{y}$ associated to different $\mathbf{x}$, along with the ground truths $\mathbf{x}$. Our energy model is well-calibrated with respect to both prior and posterior probabilities.

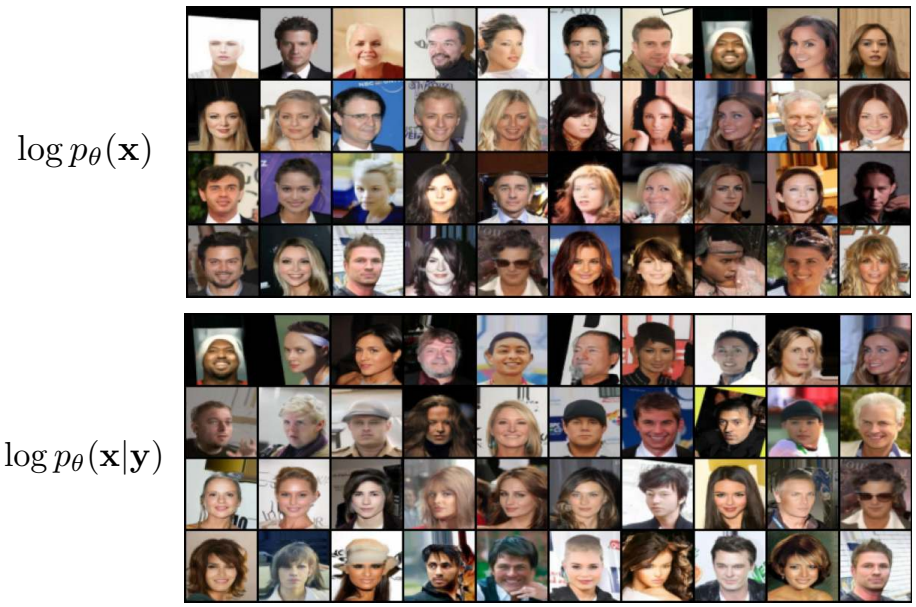


Figure D.9 : Images sorted from high to low probability (from left-up corner to right-down corner). Images with highest prior probability contains less details than those associated with higer probability.

ImageNet. We evaluate our model on ImageNet64, first validating that the learned prior aligns with the complexity-probability relationship described in [189]. As illustrated in Fig. D.10, images sorted by decreasing probability transition from sparse, low-detail structures to high-complexity textures; the highest-probability samples typically feature single objects on plain backgrounds, while the lowest-probability samples are densely textured.

We extend this analysis to samples sorted by the posterior distribution $\log p(\mathbf{x}|\mathbf{y})$ in Fig. D.11. We observe that the highest posterior probability does not necessarily correlate with the best reconstruction or visual quality. Furthermore, the complexity-probability trend observed in the prior persists in the posterior case.

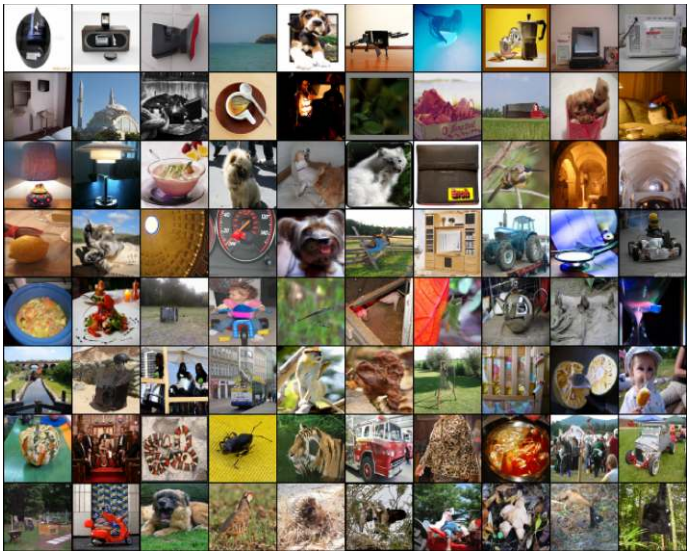


Figure D.10 : ImageNet posterior samples for a 21×21 center inpainting task, sorted by decreasing prior probability. Consistent with previous observations, samples with the highest probability feature uniform backgrounds and minimal detail, whereas lower-probability samples exhibit significantly higher structural complexity and dense detail.

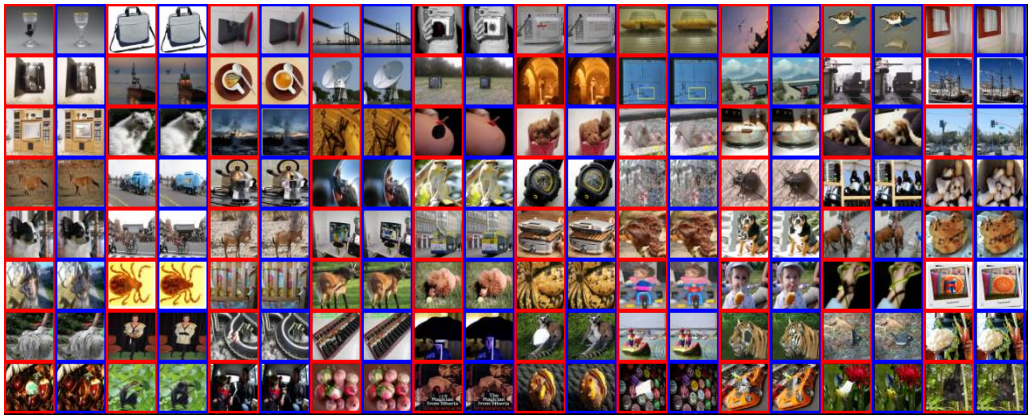


Figure D.11 : ImageNet posterior samples (red border) generated by sampling from measurements $\mathbf{y}$ generated using a box inpainting mask with a box of size 21×21, and sorted from high to low posterior probability, next to their corresponding ground truth (blue border).

D.4.4 Recovering the Gumbel distribution for the isotropic case

Our model generalizes the energy model of [189], which is recovered as a special case by setting $\mathbf{\Sigma}_t = \sigma_t^2 \mathbf{I}$ (isotropic covariance). Under this reduction, the anisotropic score matching objective reduces to the standard dual score matching of [2], and the Gumbel distribution result therefore carries over directly. We include a curve computed over 50k images alongside with some of the images, which can be found in Figs. D.12 and D.13

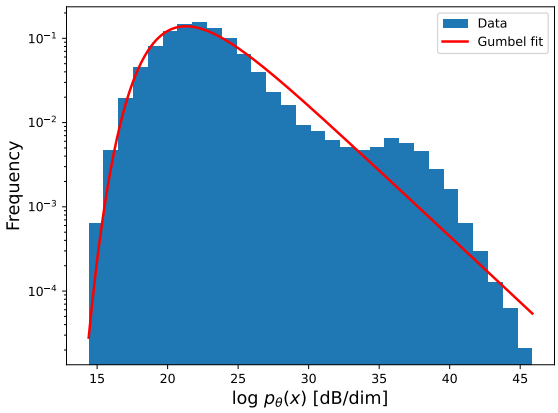


Figure D.12 : Histogram of log probabilities $\log p_\theta(\mathbf{x})$[dB/dim] for 50k in the ImageNet test set. The distribution is well-fit by a Gumbel distribution (red line), but there is a second mode (centered near 38 dB/dim) in contrast to the model in [1]. This additional mode is caused by grayscale images within ImageNet; these samples possess significantly higher probability under the model U_θ compared to standard RGB samples, creating an additional mode.

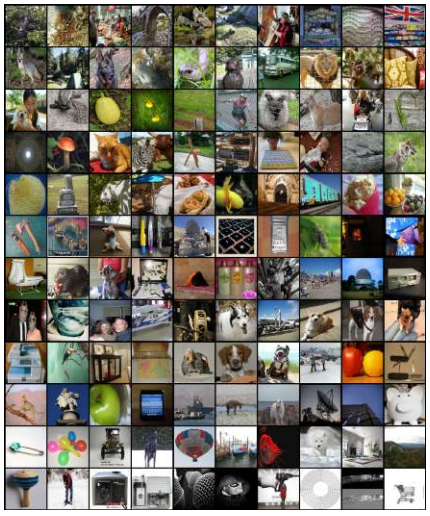


Figure D.13 : Images from the test set of ImageNet 64×64, sorted from low to high probability (left to right and top to down).

D.4.5 Validation of the normalization constant in synthethic experiment

Analogous to the experiments of [189], we generate $n = 100{,}000$ samples from a mixture of two Gaussian distributions, $\frac{1}{2}\mathcal{N}(0, \sigma_1^2 \mathrm{Id}) + \frac{1}{2}\mathcal{N}(0, \sigma_2^2 \mathrm{Id})$, with $\sigma_1 = 1$ and $\sigma_2 = 4$, in dimension $d = 1{,}000$. Each sample receives a random partition of the d dimensions into two equal groups A and B of size $d/2$, with two independent noise levels $t_{\mathrm{top}}, t_{\mathrm{bot}} \sim \mathrm{UniformLog}(t_{\min}, t_{\max})$, yielding per-dimension variance $\Sigma_{t,ii} = t_{\mathrm{top}}$ if $i \in A$ and $\Sigma_{t,ii} = t_{\mathrm{bot}}$ if $i \in B$. The true (normalized) energy is given by

$$U(y, \Sigma_t) = -\log\left(\sum_{i=1}^{2} \mathrm{e}^{-\frac{1}{2} y^\top (\sigma_i^2 \mathrm{Id} + \Sigma_t)^{-1} y - \frac{1}{2}\log\det\left(2\pi(\sigma_i^2 \mathrm{Id} + \Sigma_t)\right) - \log 2}\right). \tag{D.31}$$

We parameterize the energy as a mixture of quadratics

$$U_\theta(y, t_{\mathrm{top}}, t_{\mathrm{bot}}) = -\log\left(\sum_{i=1}^{2} \exp\Big(-a_i^{\mathrm{top}}(t)\, r_{\mathrm{top}}^2 - a_i^{\mathrm{bot}}(t)\, r_{\mathrm{bot}}^2 - b_i(t)\Big)\right), \tag{D.32}$$

where $r_{\mathrm{top}}^2 = \|y_A\|^2$ and $r_{\mathrm{bot}}^2 = \|y_B\|^2$ are the squared norms over each group, and the functions $a_i^{\mathrm{top}}, a_i^{\mathrm{bot}}, b_i$ are computed by a 5-layer MLP with a hidden dimension of 256 that takes the 2-dimensional schedule embedding $[\log(t_{\mathrm{top}} + t_{\min}), \log(t_{\mathrm{bot}} + t_{\min})]$ as input (reducing to the isotropic case when $t_{\mathrm{top}} = t_{\mathrm{bot}}$). This network is trained either with single (space) score matching or with dual score matching, both across noise levels $t_{\mathrm{top}}, t_{\mathrm{bot}} \in [t_{\min}, t_{\max}]$, where the dual loss includes one time-derivative term per noise level. Training is otherwise similar to the isotropic experiment, for 50,000 training steps with a batch size of 512 and an initial learning rate of 0.0001, over noise levels from $t_{\min} = 10^{-2}$ to $t_{\max} = 10^2$. The comparison is illustrated in Fig. D.14

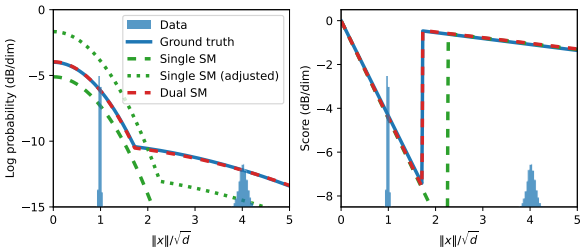


Figure D.14 : Comparison of single and dual score matching on a 1000-dimensional Gaussian scale mixture. Left: Dual score matching (red dashed) captures the true energy (blue solid), whereas single score matching (green) fails, even after global normalization. Right: Radial score components. Single score matching learns the score accurately within the data support (blue bar plot) but lacks accuracy outside of this region.

D.4.6 Reconstructed images

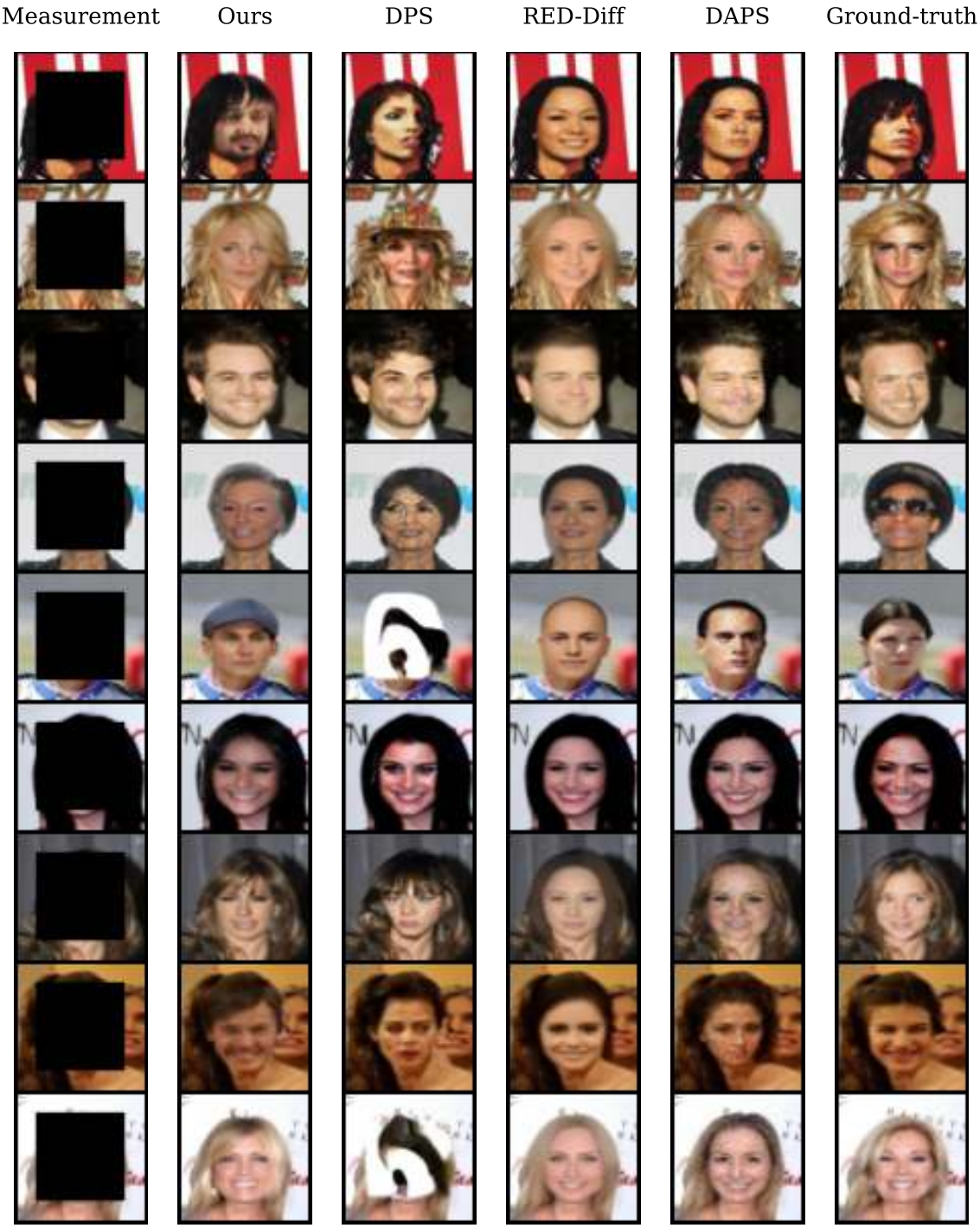


Figure D.15 : Examples of inpainting for CelebA.



Figure D.16 : Examples of inpainting for ImageNet.

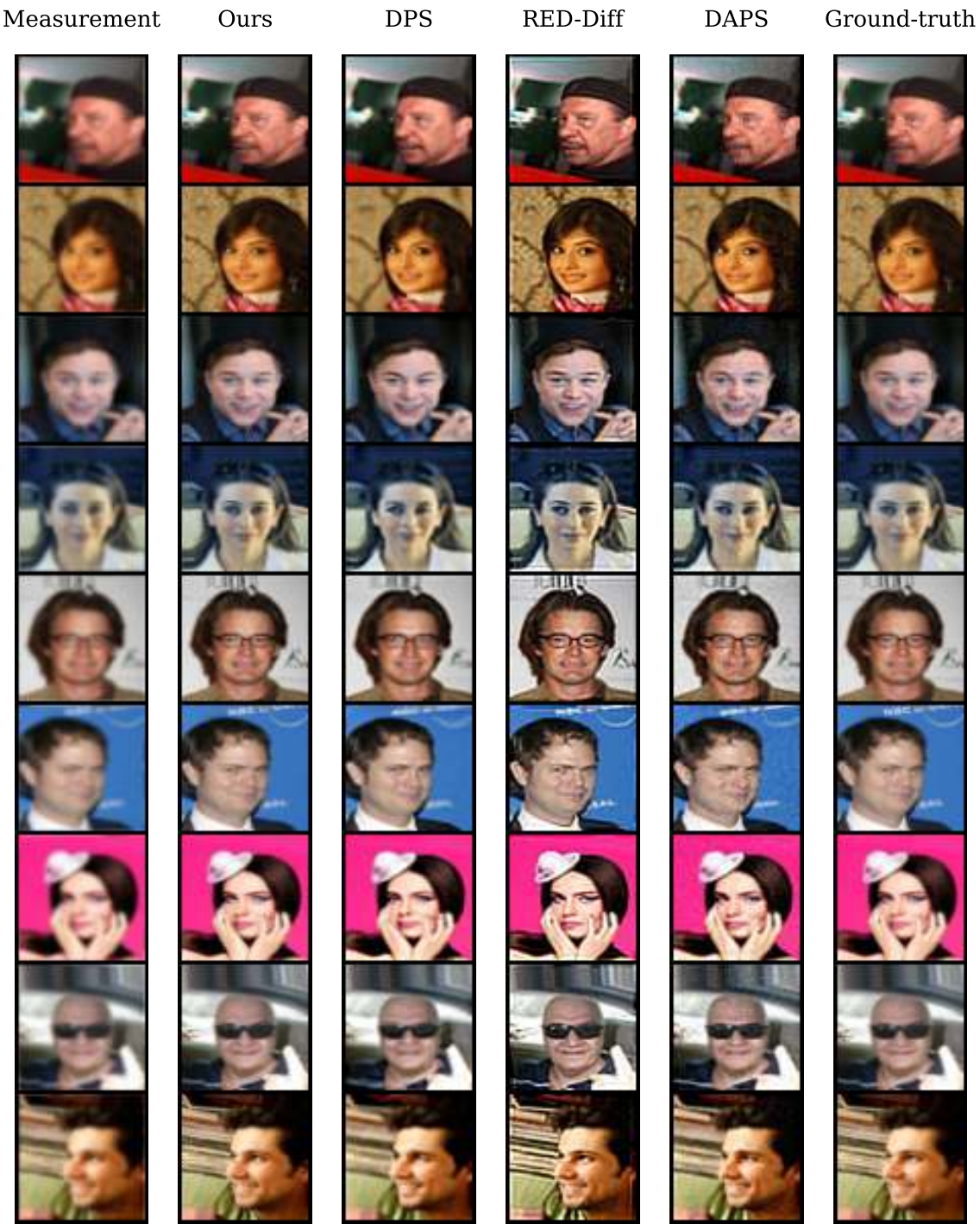


Figure D.17 : Examples of deblurring for CelebA



Figure D.18 : Examples of deblurring for ImageNet

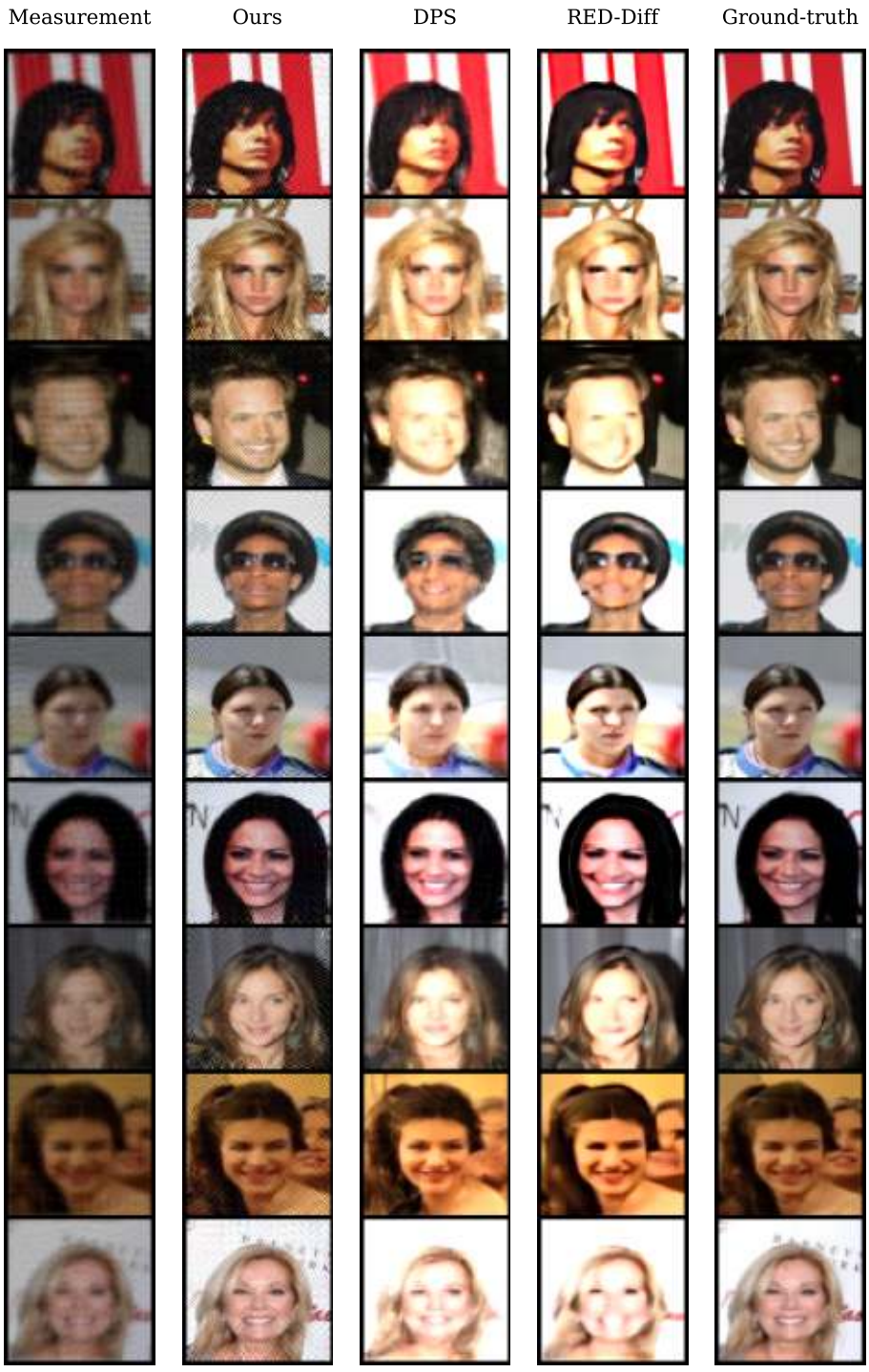


Figure D.19 : Examples of super-resolution for CelebA



Figure D.20 : Examples of inpainting for AFHQ-cats.

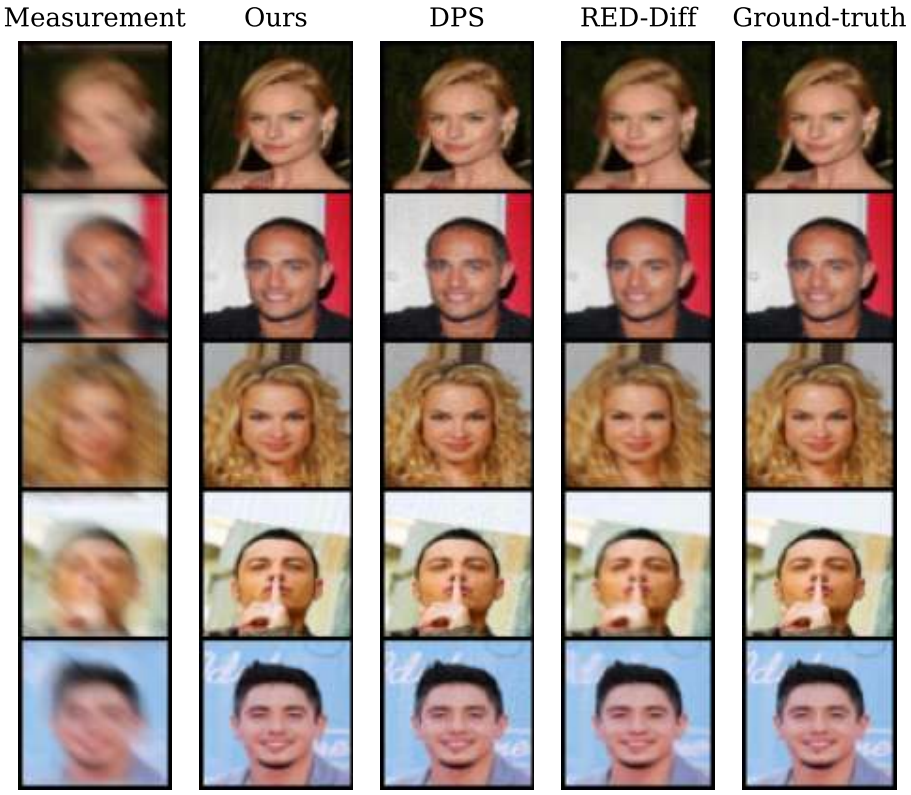


Figure D.21 : Examples of Motion Deblurring.

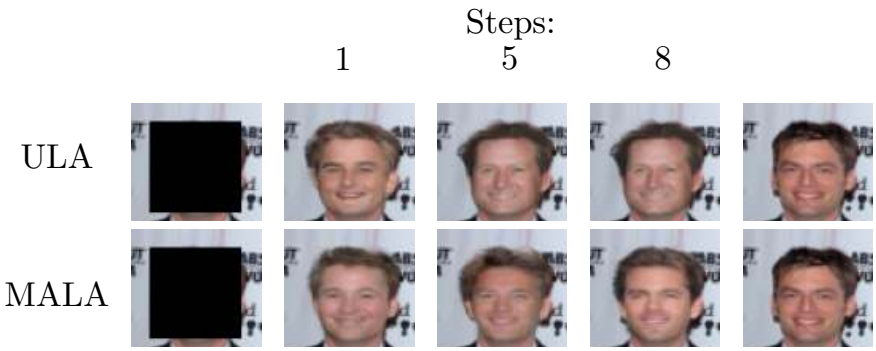


Figure D.22 : Comparison of the different corrector samplers a function of the number of steps.

Appendix E

Supplement for Chapter 8: Flow-map denoisers

E.1 Background

E.1.1 Flow Maps

Consider a probability flow defined by the ordinary differential equation

$$\dot{\mathbf{x}}_t = \mathbf{v}_t(\mathbf{x}_t), \quad \mathbf{x}_0 \sim p_0,$$

which transports samples from a simple distribution p_0 to a target distribution p_1. The associated *flow map* $\mathbf{X}_{s,t} : \mathbb{R}^d \to \mathbb{R}^d$ is defined as the solution operator of this ODE:

$$X_{s,t}(\mathbf{x}_s) = \mathbf{x}_t,$$

i.e., it maps a point at time s to its position at time t along the flow trajectory. In particular, $X_{0,1}$ directly maps samples from p_0 to p_1, enabling one-step generation without numerical integration. A convenient parameterization of the flow map is given by

$$X_{s,t}(\mathbf{x}) = \mathbf{x} + (t - s)\, \mathbf{v}(\mathbf{x}, s, t),$$

where $\mathbf{v}_{s,t}$ can be interpreted as the average velocity between times s and t. In the limit $t \to s$, this recovers the instantaneous velocity field:

$$\lim_{s \to t} \partial_t X_{s,t}(\mathbf{x}) = \mathbf{v}(\mathbf{x}, t),$$

which implies that $\mathbf{v}(\mathbf{x}, t, t) = \mathbf{v}(\mathbf{x}, t)$.

Characterization of the flow map. The flow map can be equivalently characterized in three ways:

- **Lagrangian form:**

$$\partial_t X_{s,t}(\mathbf{x}) = \mathbf{v}_t(X_{s,t}(\mathbf{x})),$$

  which describes evolution along trajectories;

- **Eulerian form:**

$$\partial_s X_{s,t}(\mathbf{x}) + \nabla X_{s,t}(\mathbf{x})\, \mathbf{v}_s(\mathbf{x}) = 0,$$

  which expresses conservation along the flow;

- **Semigroup property:**

$$X_{s,t}(\mathbf{x}) = X_{u,t}(X_{s,u}(\mathbf{x})),$$

  which encodes consistency across time intervals.

These equivalent formulations provide the basis for learning flow maps in practice [56].

Self-distillation framework. Instead of learning the instantaneous velocity field $\mathbf{v}(., t)$ and integrating the ODE, flow map models directly learn $\mathbf{v}(., s, t)$ by combining:

- the *diagonal loss* in (5.13) enforcing $\mathbf{v}(\mathbf{x}, t, t) \approx \mathbf{v}(\mathbf{x}, t)$,
- an *off-diagonal loss* enforcing consistency with one of the above three characterizations.

Notice that throughout this work we used the LSD loss in (8.3), which is the one associated to the Lagrangian condition. This approach, known as *self-distillation*, allows training a single model that approximates the full flow map $X_{s,t}$ without requiring a pre-trained teacher model.

E.2 Proofs and theoretical results

E.2.1 Average denoiser.

Recall the definition of the average denoiser

$$D_{s,t}(\mathbf{x}) = \mathbf{x} + (1-s)\,\mathbf{v}(\mathbf{x}, s, t), \tag{E.1}$$

where $\mathbf{v}(\mathbf{x}, s, t)$ denotes the average velocity of the flow map over $[s, t]$, and the flow map is parameterized as

$$X_{s,t}(\mathbf{x}) = \mathbf{x} + (t-s)\,\mathbf{v}(\mathbf{x}, s, t). \tag{E.2}$$

Proposition 6 (Basic properties) For all $0 \le s \le u \le t \le 1$, the average denoiser satisfies:

1. ***Flow-map relation.***

$$X_{s,t}(\mathbf{x}) = \frac{1-t}{1-s}\,\mathbf{x} + \frac{t-s}{1-s}\,D_{s,t}(\mathbf{x}). \tag{E.3}$$

2. ***Diagonal consistency (MMSE).***

$$D_{s,s}(\mathbf{x}) = D_s(\mathbf{x}) = \mathbb{E}[\mathbf{x}_1 \mid \mathbf{x}_s = \mathbf{x}]. \tag{E.4}$$

3. ***Semigroup structure.***

$$D_{s,t}(\mathbf{x}) = \gamma\, D_{s,u}(\mathbf{x}) + (1-\gamma)\, D_{u,t}(X_{s,u}(\mathbf{x})), \tag{E.5}$$

with

$$\gamma = \frac{(1-t)(u-s)}{(1-u)(t-s)} \in [0, 1]. \tag{E.6}$$

Proof. **(1) Flow-map relation.** Using the parameterization,

$$D_{s,t}(\mathbf{x}) - \mathbf{x} = (1-s)\,\mathbf{v}(\mathbf{x}, s, t), \quad X_{s,t}(\mathbf{x}) - \mathbf{x} = (t-s)\,\mathbf{v}(\mathbf{x}, s, t).$$

Eliminating $\mathbf{v}$ gives

$$X_{s,t}(\mathbf{x}) = \mathbf{x} + \frac{t-s}{1-s}\big(D_{s,t}(\mathbf{x}) - \mathbf{x}\big) = \frac{1-t}{1-s}\,\mathbf{x} + \frac{t-s}{1-s}\,D_{s,t}(\mathbf{x}).$$

(2) Diagonal consistency. By definition of the flow map,

$$\lim_{t\to s} \mathbf{v}(\mathbf{x}, s, t) = \mathbf{v}(\mathbf{x}, s),$$

the instantaneous velocity. Using Tweedie's relation,

$$D_s(\mathbf{x}) = \mathbf{x} + (1-s)\,\mathbf{v}(\mathbf{x}, s),$$

which matches $D_{s,t}$ at $t = s$.

(3) Semigroup structure. From the flow map semigroup property,

$$X_{s,t}(\mathbf{x}) = X_{u,t}(\mathbf{X}_{s,u}(\mathbf{x})).$$

Substitute the affine forms from (i) for each map and match coefficients of $\mathbf{x}$ and $D_{s,t}$. After rearrangement, this yields the convex combination:

$$D_{s,t}(\mathbf{x}) = \gamma\, D_{s,u}(\mathbf{x}) + (1-\gamma)\, D_{u,t}(X_{s,u}(\mathbf{x})),$$

with γ as stated.

E.2.2 Full derivation for the Gaussian case (Theorem 2)

We provide the complete derivation of the average denoiser gain and its connection to the optimal DP estimator for the scalar Gaussian target.

Setup

Let $p_1 = \mathcal{N}(0, \sigma_p^2)$ with $\sigma_p^2 \le 1$, and let $x_\tau = \tau x_1 + (1-\tau)\epsilon$ with $x_1 \sim p_1$, $\epsilon \sim \mathcal{N}(0,1)$, and $x_1 \perp \epsilon$. Then $x_\tau \sim \mathcal{N}(0, \sigma_\tau^2)$ with $\sigma_\tau^2 := \tau^2\sigma_p^2 + (1-\tau)^2$. The instantaneous denoiser and velocity are linear:

$$D_\tau(x) = A_\tau\, x, \quad A_\tau = \frac{\tau\sigma_p^2}{\sigma_\tau^2}, \qquad v_\tau(x) = B_\tau\, x, \quad B_\tau = \frac{A_\tau - 1}{1-\tau} = \frac{\tau(\sigma_p^2+1)-1}{\sigma_\tau^2}. \tag{E.7}$$

By linearity, the ODE $\dot{x}_\tau = v_\tau(x_\tau) = B_\tau x_\tau$ admits the closed-form solution $x_t = \Phi(t,s)\, x_s$ with $\Phi(t,s) = \exp\big(\int_s^t B_\tau\, d\tau\big)$. The average velocity over $[s,t]$ then satisfies $x_t = x_s + (t-s)\, v_{s,t}(x_s)$ with $v_{s,t}(x_s) = [\Phi(t,s) - 1]/(t-s) \cdot x_s$, and the average denoiser $D_{s,t}(x) = x + (1-s)\, v_{s,t}(x)$ becomes $D_{s,t}(x_s) = \Lambda(s,t)\, x_s$ with:

$$\Lambda(s,t) = \frac{(1-s)\,\Phi(t,s) - (1-t)}{t-s}. \tag{E.8}$$

We now introduce a few useful Lemmas.

Lemma E.1 (Closed form for $\Phi(t,s)$ and $\Lambda(s,t)$) $\Phi(t,s) = \sigma_t/\sigma_s$.

Proof. A direct computation gives $\frac{d}{d\tau}\frac{1}{2}\log\sigma_\tau^2 = \dot{\sigma}_\tau^2/(2\sigma_\tau^2) = [\tau(\sigma_p^2+1)-1]/\sigma_\tau^2 = B_\tau$. Hence $\int_s^t B_\tau\, d\tau = \log(\sigma_t/\sigma_s)$ and $\Phi(t,s) = \sigma_t/\sigma_s$. □

Substituting Lemma E.1 into (E.8) yields the closed form stated in (8.7):

$$\Lambda(s,t) = \frac{(1-s)\sigma_t - (1-t)\sigma_s}{\sigma_s(t-s)}. \tag{E.9}$$

Boundary values. At $t=1$, $\sigma_1 = \sigma_p$ gives directly $\Lambda(s,1) = \sigma_p/\sigma_s$. At $t = s$, applying L'Hôpital to (8.7) and using $\dot{\sigma}_\tau = [\tau(\sigma_p^2+1)-1]/\sigma_\tau$:

$$\Lambda(s,s) = \frac{(1-s)\dot{\sigma}_s + \sigma_s}{\sigma_s} = 1 + \frac{(1-s)[s(\sigma_p^2+1)-1]}{\sigma_s^2} = \frac{s\sigma_p^2}{\sigma_s^2} = A_s, \tag{E.10}$$

where the last equality uses $\sigma_s^2 + (1-s)[s(\sigma_p^2+1)-1] = s\sigma_p^2$, which one verifies by expanding $\sigma_s^2 = s^2\sigma_p^2 + (1-s)^2$.

Lemma E.2 (Monotonicity of $\Lambda(s,t)$) For fixed $s \in (0,1)$, $t \mapsto \Lambda(s,t)$ is strictly increasing on $[s,1]$.

Proof. Differentiating (8.7) in t and writing $\sigma_t - \sigma_s = \int_s^t \dot{\sigma}_u \, du$:

$$\partial_t \Lambda(s,t) = \frac{1-s}{\sigma_s (t-s)^2} \left[(t-s)\dot{\sigma}_t - \int_s^t \dot{\sigma}_u \, du \right]. \tag{E.11}$$

A direct calculation shows $\ddot{\sigma}_\tau > 0$ for all $\tau \in (0,1)$, so $\dot{\sigma}_\tau$ is strictly increasing. Therefore $\int_s^t \dot{\sigma}_u \, du < (t-s)\dot{\sigma}_t$ for $s < t$, giving $\partial_t \Lambda > 0$. □

Connection to the optimal DP estimator

The DP optimal estimator from [1]. The MMSE estimator $\hat{X}_0 = A_s x_s$ has standard deviation $\hat{\sigma} = A_s \sigma_s = s\sigma_p^2/\sigma_s$, and its perceptual gap to the prior is $P^* = W_2(p_{\hat{X}_0}, p_1) = |\sigma_p - \hat{\sigma}|$. The W_2-optimal transport map from $\mathcal{N}(0, \hat{\sigma}^2)$ to $p_1 = \mathcal{N}(0, \sigma_p^2)$ is the scaling $T^* = \sigma_p/\hat{\sigma}$. By [1, Theorem 3], the minimum-distortion deterministic estimator at perception level $P = \alpha P^*$ ($\alpha \in [0,1]$) is the displacement interpolant

$$\hat{X}_\alpha^{\text{det}} = \left[\alpha + (1-\alpha)T^*\right] A_s \, x_s =: \Gamma(\alpha) \, x_s, \tag{E.12}$$

with $\Gamma(1) = A_s$ (MMSE) and $\Gamma(0) = T^* A_s = \sigma_p/\sigma_s$ (perfect perception). The corresponding distortion–perception function is $D(P) = D^* + (P^* - P)^2$.

Proposition 7 (Average denoiser is exactly optimal) For any $s \in (0,1)$ and $t \in [s,1]$, $\Lambda(s,t) = \Gamma(\alpha(s,t))$ where

$$\alpha(s,t) = \frac{\sigma_p/\sigma_s - \Lambda(s,t)}{\sigma_p/\sigma_s - A_s} \tag{E.13}$$

decreases monotonically from 1 at $t = s$ to 0 at $t = 1$.

Proof. By (E.12), Γ is affine in α and maps $[0,1]$ bijectively onto $[A_s, \sigma_p/\sigma_s]$ (since $\sigma_p/\sigma_s > A_s$ whenever $s < 1$). By the boundary values of Λ and Lemma E.2, $\Lambda(s,\cdot)$

is continuous and strictly increasing on $[s, 1]$ with $\Lambda(s, s) = A_s = \Gamma(1)$ and $\Lambda(s, 1) = \sigma_p/\sigma_s = \Gamma(0)$. Inverting the affine relation $\Gamma(\alpha) = \Lambda(s, t)$ yields $\alpha(s, t)$ as stated, and the monotonicity follows because Γ is decreasing in α while Λ is increasing in t. □

Summary. As t increases from s to 1, the average denoiser gain $\Lambda(s, t)$ traces the affine segment from the MMSE gain A_s to the perfect-perception gain σ_p/σ_s. This segment parametrizes the W_2 geodesic from the MMSE output distribution $p_{\hat{X}_0}$ to the target p_1, and by [1, s], every point on it achieves the minimum distortion at its perception level. A visual comparison is shown in Fig. E.1.



Figure E.1 : Visual comparison for $\alpha = 0.3$. From left to right: clean image, noisy input, and reconstructions at increasing lookahead. The estimator smoothly transitions from a blurry conditional mean (MMSE) to a sharp sample, progressively restoring texture, contrast, and color saturation.

E.2.3 Experiment on mixture of Gaussians

MMSE denoiser for a mixture of Gaussians

Setup. Let $p_0 = \mathcal{N}(0, \mathbf{I})$ and $p_1 = \frac{1}{2}\mathcal{N}(\boldsymbol{\mu}_a, \sigma_c^2\mathbf{I}) + \frac{1}{2}\mathcal{N}(\boldsymbol{\mu}_b, \sigma_c^2\mathbf{I})$. Under the linear interpolant $\mathbf{x}_t = (1-t)\mathbf{x}_0 + t\mathbf{x}_1$, we seek to compute the MMSE denoiser:

$$D_t(\mathbf{x}) = \mathbb{E}[\mathbf{x}_1 \mid \mathbf{x}_t = \mathbf{x}]. \tag{E.14}$$

Step 1: Marginal distribution of $\mathbf{x}_t$. Conditioned on $\mathbf{x}_1$ originating from component $k \in \{a, b\}$, we have $\mathbf{x}_1 \sim \mathcal{N}(\boldsymbol{\mu}_k, \sigma_c^2\mathbf{I})$ and $\mathbf{x}_0 \sim \mathcal{N}(0, \mathbf{I})$. Thus:

$$\mathbf{x}_t \mid \text{component } k \;\sim\; \mathcal{N}\big(t\,\boldsymbol{\mu}_k,\; \sigma_t^2\mathbf{I}\big)\,, \quad \text{where } \sigma_t^2 = (1-t)^2 + t^2\sigma_c^2. \tag{E.15}$$

The marginal distribution is the mixture:

$$p(\mathbf{x}_t) = \frac{1}{2}\mathcal{N}(\mathbf{x}_t;\, t\boldsymbol{\mu}_a,\, \sigma_t^2\mathbf{I}) + \frac{1}{2}\mathcal{N}(\mathbf{x}_t;\, t\boldsymbol{\mu}_b,\, \sigma_t^2\mathbf{I}). \tag{E.16}$$

Step 2: Posterior weights. By Bayes' rule, the posterior probability $w_k(\mathbf{x}, t)$ that $\mathbf{x}_1$ originated from component k given $\mathbf{x}_t = \mathbf{x}$ is:

$$w_k(\mathbf{x}, t) = \frac{\mathcal{N}(\mathbf{x};\, t\boldsymbol{\mu}_k,\, \sigma_t^2\mathbf{I})}{\mathcal{N}(\mathbf{x};\, t\boldsymbol{\mu}_a,\, \sigma_t^2\mathbf{I}) + \mathcal{N}(\mathbf{x};\, t\boldsymbol{\mu}_b,\, \sigma_t^2\mathbf{I})}. \tag{E.17}$$

In log-space, this is proportional to $-\frac{\|\mathbf{x} - t\boldsymbol{\mu}_k\|^2}{2\sigma_t^2}$, and the weights are normalized via softmax such that $w_a + w_b = 1$.

Step 3: Conditional mean within components. Given that $\mathbf{x}_1$ comes from component k, the posterior mean of $\mathbf{x}_1$ given $\mathbf{x}_t = \mathbf{x}$ is:

$$\mathbb{E}[\mathbf{x}_1 \mid \mathbf{x}_t = \mathbf{x}, \text{component } k] = \boldsymbol{\mu}_k + A_t(\mathbf{x} - t\,\boldsymbol{\mu}_k), \tag{E.18}$$

where $A_t = \frac{t\,\sigma_c^2}{\sigma_t^2}$ is the regression coefficient (Tweedie's formula applied component-wise).

Step 4: Full MMSE denoiser. By the law of total expectation, the denoiser is the weighted average:

$$D_t(\mathbf{x}) = w_a(\mathbf{x}, t)\ [\boldsymbol{\mu}_a + A_t(\mathbf{x} - t\boldsymbol{\mu}_a)] + w_b(\mathbf{x}, t)\ [\boldsymbol{\mu}_b + A_t(\mathbf{x} - t\boldsymbol{\mu}_b)]\,. \tag{E.19}$$

This is a softmax-weighted average of the within-component posterior means. Its behavior is intuitive:

- When $\mathbf{x}$ is close to $t\boldsymbol{\mu}_a$, $w_a \approx 1$ and $D_t(\mathbf{x}) \approx \boldsymbol{\mu}_a + A_t(\mathbf{x} - t\boldsymbol{\mu}_a)$ (assigned to component a).
- When $\mathbf{x}$ is equidistant from $t\boldsymbol{\mu}_a$ and $t\boldsymbol{\mu}_b$, $w_a \approx w_b \approx 0.5$, and $D_t(\mathbf{x})$ averages the two, producing an output between the modes (the posterior averaging artifact).

We compute the average denoiser by ODE integration using the instantaneous velocity field. The nonlinearity introduced by the softmax weights makes the MoG fundamentally different from the single Gaussian case: the denoiser is non-linear in $\mathbf{x}$, the flow map lacks a closed-form solution, and the posterior averaging artifact (blurring between modes) emerges as a direct consequence of the mixture structure. However, given the simplicity of the instantaneous denoiser, we can approximate the average denoiser numerically.

DP analysis

In this section, we consider a controlled setting based on a 2D mixture of Gaussians. For this distsribution, the marginal velocity admits a closed-form expression, allowing both the average denoiser $D_{s,t}$ and the optimal DP estimator of [1] to be evaluated with high precision, bypassing the requirement of a learned model.

Setup. We consider $\boldsymbol{\mu}_a = (-2, 0)$, $\boldsymbol{\mu}_b = (2, 0)$, and $\sigma_c = 0.3$, and compute the average denoiser $D_{s,t}$ by integrating $\dot{x}_\tau = v(x_\tau, \tau)$ from $\tau = s$ to $\tau = t$ with a fourth-order Runge–Kutta scheme (200 steps).

Optimal DP construction. To construct the optimal DP curve, we follow [1, Theorem 3]. At perception level $P \in [0, P^*]$, the optimal estimator is given by (2.7).

We approximate this construction empirically by drawing $N = 5000$ samples, computing $\hat{\mathbf{x}}_0^{(i)} = D_s(\mathbf{x}_s^{(i)})$, and solving the discrete optimal transport problem between the empirical distributions $\{\hat{\mathbf{x}}_0^{(i)}\}$ and $\{\mathbf{x}_1^{(j)}\} \sim p_1$ using [248]. Let π denote the optimal coupling; we then form the interpolated samples as

$$\hat{\mathbf{x}}_P^{(i)} = \left(1 - \frac{P}{P^*}\right) \hat{\mathbf{x}}_0^{(i)} + \frac{P}{P^*} \mathbf{x}_1^{(\pi(i))}.$$

Distortion is measured as MSE with respect to the ground truth, while perception is quantified via the squared 2-Wasserstein distance $W_2^2(\hat{p}, p_1)$ between empirical distributions.

Results. Fig. E.2 compares the resulting curves at three noise levels $s \in 0.3, 0.5, 0.7$, corresponding to heavy, moderate, and light noise (recall that smaller s implies heavier noise, with $\xi := 1 - s$). Three observations stand out.

1. **Endpoints coincide at MMSE.** At $t = s$ (equivalently, $\alpha = 1$), both curves reduce to the MMSE estimator by construction, yielding identical distortion–perception coordinates across all panels.

2. **Near-optimal behavior in the interior.** For intermediate lookaheads, the average denoiser traces a DP curve that lies slightly above the Freirich optimum, with a small and nearly uniform gap. At $s = 0.3$ (heavy noise), this gap remains

below 0.02 in MSE across the full perception range. This discrepancy is not a numerical artifact: it persists when using higher-order integration (RK4) and when increasing the sample size from $N = 2000$ to $N = 5000$. Instead, it reflects a structural limitation—while the flow-map trajectory matches the W_2 geodesic in the Gaussian case, it deviates slightly for nonlinear velocity fields (here induced by the softmax-weighted mixture).

3. **Endpoint behavior at $t = 1$.** At the perceptual endpoint, the distribution $D_{s,1}(p_{x_s})$ closely approximates p_1 but does not match it exactly, resulting in a small residual W_2^2 gap. This gap decreases as noise increases (from 9.4×10^{-3} at $s = 0.5$ to 3.8×10^{-3} at $s = 0.3$, compared to a sampling floor of 2.2×10^{-3}). This is consistent with the intuition that the integrable $1/(1-\tau)$ singularity in the velocity field has a stronger effect when the final integration interval is short.

Taken together, these results show that the average denoiser is *near-optimal* in a controlled setting: the resulting curves are smooth, monotone, and convex, match the MMSE endpoint exactly, and exhibit only small, well-characterized deviations from the Freirich optimum. This complements the exact optimality established in the Gaussian case (Theorem 2) and supports the behavior observed at image scale (Fig. 8.2a).

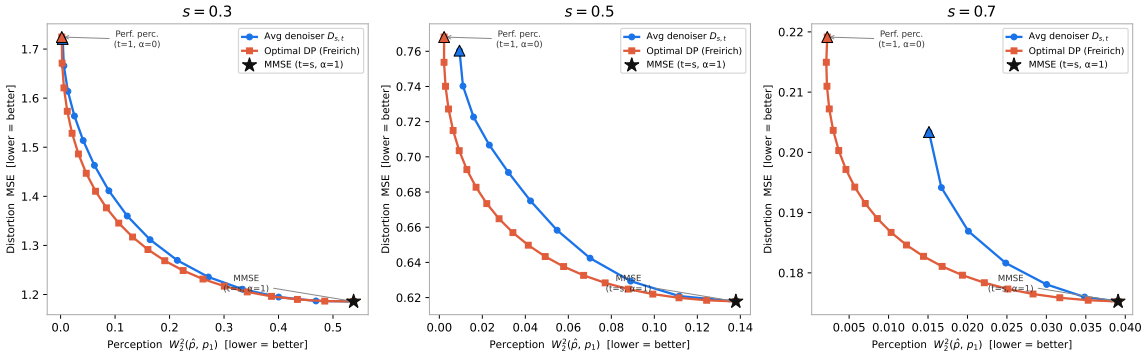


Figure E.2 : Distortion–perception comparison on a 2D mixture of Gaussians at three noise levels $s \in \{0.3, 0.5, 0.7\}$ (heavy, moderate, light noise). Blue: average denoiser $D_{s,t}$ swept over lookahead $t \in [s, 1]$. Red: [1] optimal DP estimator swept over perception level $\alpha \in [0, 1]$, computed via exact W_2-optimal transport between the MMSE pushforward and p_1. Both curves share the MMSE endpoint exactly (★); small residual gaps at the perfect-perception extreme (▲) reflect the velocity singularity near $\tau = 1$. The near-optimality of $D_{s,t}$ in the interior confirms that the flow-map trajectory closely approximates the W_2 geodesic even in a non-Gaussian setting, bridging Theorem 2 to the image experiments in Section 8.3.3.

E.2.4 Fixed point characterization

We characterize the fixed-point solution of the PnP iteration and show its dependence on the lookahead t. For simplicity, we consider linear inverse problems $\mathcal{A}(\mathbf{x}) = \mathbf{H}\mathbf{x}$ and analyze the deterministic variant of Algorithm 6 (i.e., $\epsilon = 0$). In this variant, the stochastic renoising step reduces to the deterministic map $R_s(\mathbf{z}) = s\,\mathbf{z}$, and the per-iteration operator becomes

$$T_t(\mathbf{x}) := D_{s,t}\big(R_s(F_\lambda(\mathbf{x}))\big), \quad \text{with} \quad F_\lambda(\mathbf{x}) = \mathbf{x} - \lambda \nabla g(\mathbf{x}), \quad R_s(\mathbf{z}) = s\,\mathbf{z}. \tag{E.20}$$

Let $L_D(t)$ denote the Lipschitz constant of $D_{s,t}$ and $L_F = \|\mathbf{I} - \lambda\,\mathbf{H}^\top\mathbf{H}\|$ that of F_λ. Global convergence would require T_t to be a contraction, i.e., $s\,L_D(t)\,L_F < 1$. The

gradient step satisfies $L_F \leq 1$ for any $\lambda \in (0,\, 2/\|\mathbf{H}^\top\mathbf{H}\|]$, with equality (i.e., $L_F = 1$) when $\mathbf{H}^\top\mathbf{H}$ has zero eigenvalues, as in inpainting. In such ill-posed cases, contraction must come entirely from the composition $s\, L_D(t)$. Empirically, however, $s\, L_D(t)\, L_F$ exceeds 1 across all (s,t) we tested as shown in Fig. E.3, so global contraction fails. Instead, we consider local convergence. Since $T_t = D_{s,t} \circ R_s \circ F_\lambda$, its local Lipschitz constant satisfies

$$L_{T_t}^{\mathrm{loc}} \;\leq\; s\, L_D^{\mathrm{loc}}(t)\, L_F. \tag{E.21}$$

While $L_F \leq 1$, the denoiser constant $L_D^{\mathrm{loc}}(t)$ typically increases with t, reflecting the transition from an averaging (MMSE) estimator at $t = s$ to a more mode-seeking estimator as $t \to 1$. Consequently, the contraction factor degrades with t, explaining the empirically observed loss of monotonic convergence for large lookaheads in Appendix E.4.5.

We assume there exists a closed neighborhood $U \subset \mathbb{R}^d$ of a fixed point $\mathbf{x}^\star(t)$ such that $T_t(U) \subseteq U$ and

$$\|T_t(\mathbf{x}) - T_t(\mathbf{x}')\| \leq \rho(t)\, \|\mathbf{x} - \mathbf{x}'\|, \quad \forall \mathbf{x}, \mathbf{x}' \in U, \tag{E.22}$$

with $\rho(t) \;\leq\; s\, L_D^{\mathrm{loc}}(t)\, L_F \;<\; 1$. This local contraction assumption is standard in PnP [249] and consistent with our empirical observations.

Theorem 3 (Local convergence to a lookahead-dependent fixed point) *Assume (E.22) holds. If the iteration enters U, then it converges to $\mathbf{x}^\star(t) \in U$ at a geometric rate:*

$$\|\mathbf{x}^k - \mathbf{x}^\star(t)\| \leq \rho(t)^k \|\mathbf{x}^0 - \mathbf{x}^\star(t)\|. \tag{E.23}$$

Moreover, $\mathbf{x}^\star(t)$ satisfies

$$\mathbf{x}^\star(t) = D_{s,t}\Big(R_s\big(F_\lambda(\mathbf{x}^\star(t))\big)\Big) = D_{s,t}\big(s\,(\mathbf{x}^\star(t) - \lambda \nabla g(\mathbf{x}^\star(t)))\big),$$

so different lookaheads yield distinct fixed points whenever $D_{s,t}$ is genuinely t-dependent.

The proof follows directly from the Banach fixed-point theorem applied to T_t on $(U, \|\cdot\|)$ and is given in Appendix E.2.4.

Proof of Theorem 3

Since U is a closed subset of the complete metric space $(\mathbb{R}^d, \|\cdot\|)$, it is itself complete. By assumption, T_t maps U into itself ($T_t(U) \subseteq U$) and, by Eq. E.22, is a strict contraction on U with constant $\rho(t) < 1$. The Banach fixed-point theorem then yields a unique $\mathbf{x}^\star(t) \in U$ with $T_t(\mathbf{x}^\star(t)) = \mathbf{x}^\star(t)$.

For the convergence rate, applying the contraction inequality with $\mathbf{x}' = \mathbf{x}^\star(t)$ gives

$$\|\mathbf{x}^{k+1} - \mathbf{x}^\star(t)\| = \|T_t(\mathbf{x}^k) - T_t(\mathbf{x}^\star(t))\| \leq \rho(t)\, \|\mathbf{x}^k - \mathbf{x}^\star(t)\|,$$

and iterating yields the bound (E.23).

Finally, the fixed-point identity $\mathbf{x}^\star(t) = T_t(\mathbf{x}^\star(t))$ together with the operator definition (E.20) gives

$$\mathbf{x}^\star(t) = D_{s,t}\big(R_s(F_\lambda(\mathbf{x}^\star(t)))\big) = D_{s,t}\big(s\,(\mathbf{x}^\star(t) - \lambda \nabla g(\mathbf{x}^\star(t)))\big),$$

which depends on t explicitly through $D_{s,t}$. Whenever $D_{s,t} \neq D_{s,t'}$ for $t \neq t'$, the corresponding fixed points $\mathbf{x}^\star(t)$ and $\mathbf{x}^\star(t')$ differ.

Empirical Lipschitz analysis. In Fig. E.3, we plot the composite bound $s\, L_D(t)\, L_F$ for the deterministic PnP operator $T_t = D_{s,t} \circ R_s \circ F_\lambda$ on CelebA 128×128 (Gaussian kernel of size 45, std 3.0, $\lambda = 1.0$, giving $L_F = 1.000$). The denoiser Lipschitz constant $L_D(t)$ is estimated by alternating power iteration on the Jacobian of $D_{s,t}$ at noisy samples $\mathbf{x}_s = s\,\mathbf{x}_1 + (1-s)\,\boldsymbol{\epsilon}$, taking the worst case over 20 samples. Across all (s,t) tested, the bound exceeds 1 (red line), so T_t is never a global contraction;

the algorithm's empirical convergence is therefore explained by the local-contraction analysis of Theorem 3 rather than by a global Banach argument.

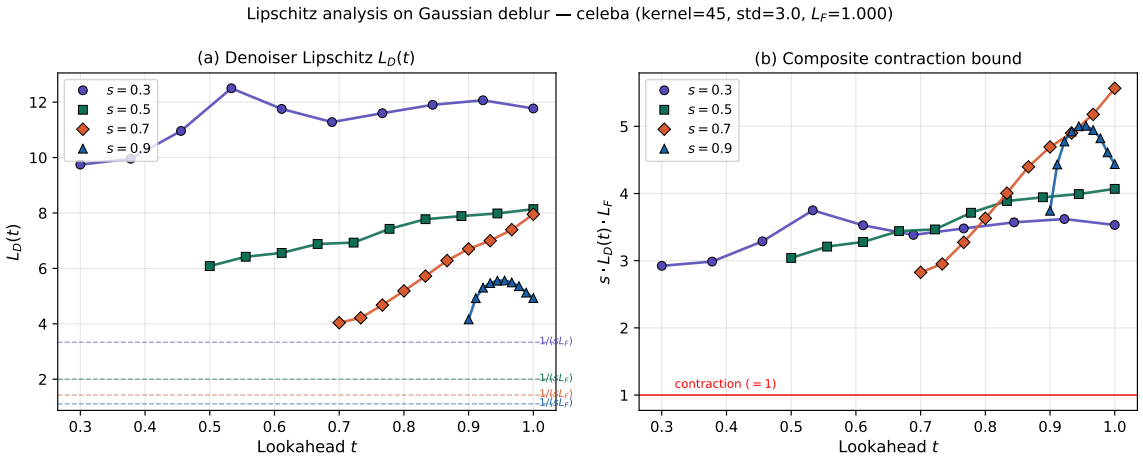


Figure E.3 : The composite bound $s\,L_D(t)\,L_F$ exceeds 1 across all (s, t), so T_t is never a global contraction — empirical convergence relies on the local analysis of Theorem 3. $L_D(t)$ is estimated via power iteration on $\partial D_{s,t}/\partial \mathbf{x}$ at noisy samples (worst case over 20). The bound increases with t, with steepest growth at low noise (large s).

On the validation of assumption (E.22)**.** Empirical validation of (E.22) and the t-dependence of fixed points is provided in the Gaussian linear-inverse setting (Appendix E.2.5, Fig. E.4), where T_t admits a closed-form analysis and the iterates converge to the predicted Tikhonov solution to machine precision. For image priors, the deterministic operator analyzed in Theorem 3 is of theoretical interest; the algorithm deployed in practice (Algorithm 6) uses stochastic renoising, which both stabilizes the iteration and induces the variance restoration that drives the empirical distortion–perception behavior (Section 8.3.3).

E.2.5 Gaussian case: global contraction.

We now generalize the analysis from Section 8.3.2 (where the denoiser was treated as a standalone estimator without a forward model) to general linear inverse problems. Focusing on the multivariate case with a Gaussian prior $p_1 = \mathcal{N}(\mathbf{0},\, \sigma_p^2\, \mathbf{I})$, the average denoiser remains a linear map $D_{s,t}(\mathbf{x}) = \Lambda(s,t)\, \mathbf{x}$ with the same scalar gain (8.7). In this case, T_t is a *global* contraction and (E.22) holds with $U = \mathbb{R}^d$.

Since the denoiser gain is a scalar, $L_D(t) = \Lambda(s,t)$ and $L_F = \|\mathbf{I} - \lambda\, \mathbf{H}^\top \mathbf{H}\|$. The contraction rate is $\rho(t) = m(t)\, \|\mathbf{I} - \lambda\, \mathbf{H}^\top \mathbf{H}\|$, where $m(t) := s\, \Lambda(s,t) < 1$ for all $t \in [s,1]$ and $s \in (0,1)$ (Appendix E.2.2). For any $\lambda \in (0,\, 2/\|\mathbf{H}^\top \mathbf{H}\|)$, we have $\|\mathbf{I} - \lambda\, \mathbf{H}^\top \mathbf{H}\| \leq 1$. Consequently, $\rho(t) < 1$ on all of $[s,1]$: the Gaussian case admits unconditional global convergence across all lookaheads, even for ill-posed forward models. Solving the fixed-point equation yields the closed form

$$\mathbf{x}^\star(t) \;=\; \big(c(t)\, \mathbf{I} + \lambda\, \mathbf{H}^\top \mathbf{H}\big)^{-1}\, \lambda\, \mathbf{H}^\top \mathbf{y}, \qquad c(t) \;=\; \frac{1}{m(t)} - 1, \tag{E.24}$$

i.e., a Tikhonov-regularized estimator whose regularization parameter $c(t)$ depends on both the forward model $\mathbf{H}$ and the lookahead t. Since $\Lambda(s,t)$ is strictly increasing in t (Appendix E.2.2), $c(t)$ is strictly decreasing. As the lookahead increases from s to 1, the Tikhonov regularization progressively weakens, yielding a continuum of estimators with decreasing shrinkage toward the prior mean. In contrast to the denoising setting, these fixed points do not, in general, correspond to the MMSE or lie on the optimal distortion–perception frontier; instead, the lookahead acts as a regularization parameter that controls the bias–variance trade-off of the reconstruction.

Proof of Eq. (E.24): Fixed Point for the Gaussian Case

By definition, the fixed point $\mathbf{x}^\star(t)$ satisfies $\mathbf{x}^\star(t) = T_t(\mathbf{x}^\star(t))$. For a linear inverse problem with data fidelity $g(\mathbf{x}) = \frac{1}{2}\|\mathbf{H}\mathbf{x}-\mathbf{y}\|^2$, the gradient is $\nabla g(\mathbf{x}) = \mathbf{H}^\top(\mathbf{H}\mathbf{x}-\mathbf{y})$. The gradient descent step is given by:

$$F_\lambda(\mathbf{x}) = s\big(\mathbf{x} - \lambda\mathbf{H}^\top(\mathbf{H}\mathbf{x} - \mathbf{y})\big).$$

In the multivariate Gaussian case, the average denoiser is the linear map $D_{s,t}(\mathbf{x}) = \Lambda(s,t)\mathbf{x}$. Substituting this into the fixed-point equation yields:

$$\begin{aligned}\mathbf{x}^\star(t) &= D_{s,t}\big(F_\lambda(\mathbf{x}^\star(t))\big)\\ \mathbf{x}^\star(t) &= \Lambda(s,t)\cdot s\big(\mathbf{x}^\star(t) - \lambda\mathbf{H}^\top(\mathbf{H}\mathbf{x}^\star(t) - \mathbf{y})\big).\end{aligned}$$

Defining the effective scaling factor $m(t) := s\,\Lambda(s,t)$, we can distribute $m(t)$ across the terms:

$$\mathbf{x}^\star(t) = m(t)\mathbf{x}^\star(t) - m(t)\lambda\mathbf{H}^\top\mathbf{H}\mathbf{x}^\star(t) + m(t)\lambda\mathbf{H}^\top\mathbf{y}.$$

Since $m(t) \in (0,1)$ across all valid lookaheads $t \in [s,1]$ (as shown in Appendix E.2.2), it is strictly positive. We can therefore divide both sides by $m(t)$:

$$\frac{1}{m(t)}\mathbf{x}^\star(t) = \mathbf{x}^\star(t) - \lambda\mathbf{H}^\top\mathbf{H}\mathbf{x}^\star(t) + \lambda\mathbf{H}^\top\mathbf{y}.$$

Subtracting $\mathbf{x}^\star(t)$ from both sides and rearranging to group all terms involving $\mathbf{x}^\star(t)$ on the left yields:

$$\left(\frac{1}{m(t)} - 1\right)\mathbf{x}^\star(t) + \lambda\mathbf{H}^\top\mathbf{H}\mathbf{x}^\star(t) = \lambda\mathbf{H}^\top\mathbf{y}.$$

Factoring out $\mathbf{x}^\star(t)$ gives:

$$\left[\left(\frac{1}{m(t)} - 1\right)\mathbf{I} + \lambda\mathbf{H}^\top\mathbf{H}\right]\mathbf{x}^\star(t) = \lambda\mathbf{H}^\top\mathbf{y}.$$

Finally, defining the regularization parameter as $c(t) := \frac{1}{m(t)} - 1$, we invert the matrix on the left-hand side to arrive at the closed-form solution:

$$\mathbf{x}^\star(t) = \big(c(t)\mathbf{I} + \lambda \mathbf{H}^\top \mathbf{H}\big)^{-1} \lambda \mathbf{H}^\top \mathbf{y},$$

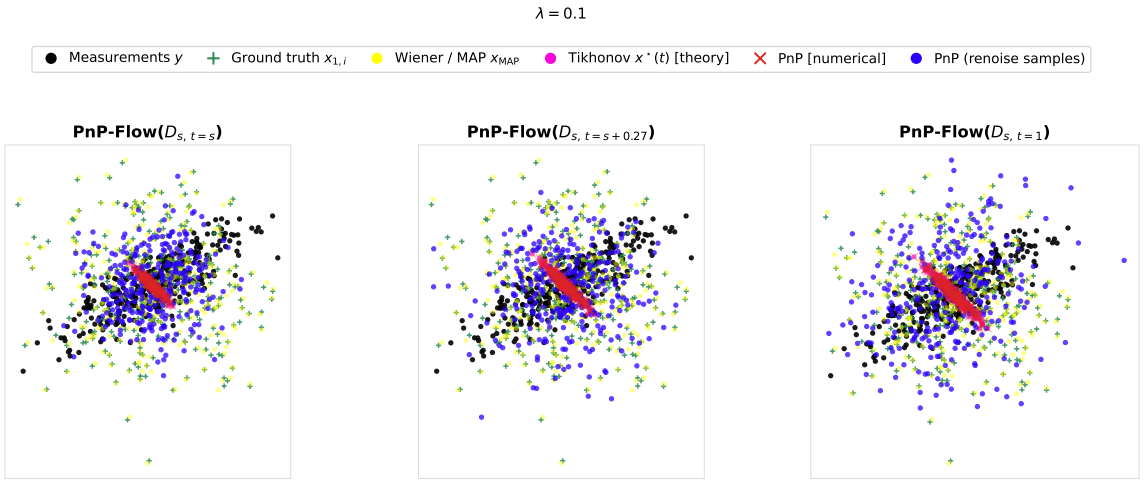
which completes the derivation.

Empirical validation. To corroborate the closed form (E.24), we consider a 2D Gaussian prior $p_1 = \mathcal{N}(\mathbf{0}, \sigma_p^2 \mathbf{I})$ with $\sigma_p = 2$, and two linear forward models,

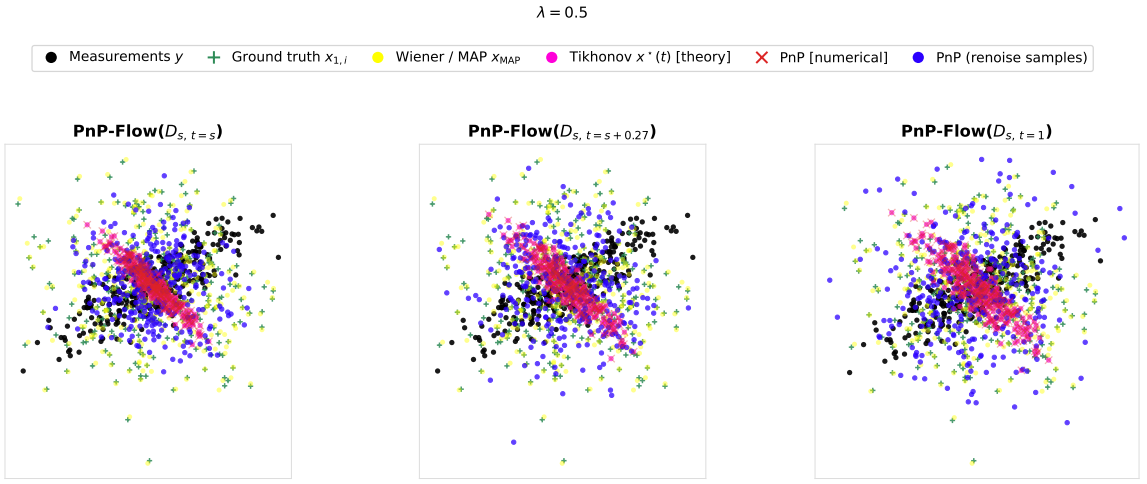
$$\mathbf{H}_1 = 0.8\, R(\pi/6), \qquad \mathbf{H}_2 = R(\pi/6)\, \mathrm{diag}(1,\, 0.3)\, R(\pi/4),$$

where the first is an isotropic rotation and the second is an anisotropic operator whose Gram matrix has eigenvalues $\{1,\, 0.09\}$ (and thus couples the coordinates). Observations are $\mathbf{y} = \mathbf{H}_k\, \mathbf{x}_1 + \eta$ with $\eta \sim \mathcal{N}(\mathbf{0}, \sigma_{\text{obs}}^2\, \mathbf{I})$, $\sigma_{\text{obs}} = 0.1$.

We run deterministic PnP-Flow ($\epsilon = 0$) with $\lambda = 0.1$ and 0.5, anchor $s = 0.45$, and lookaheads $t \in \{s,\, s + 0.27,\, 1\}$. The iterates converge to $\mathbf{x}^\star(t)$ to machine precision ($\|\hat{\mathbf{x}} - \mathbf{x}^\star(t)\|_{\text{mean}} < 5 \cdot 10^{-16}$), validating (E.24). As t increases, the effective regularization $c(t)/\lambda$ decreases ($0.93 \to 0.59 \to 0.43$), yielding a continuum of Tikhonov estimators with progressively weaker shrinkage toward the prior mean. The result is shown in Fig. E.4



(a)



(b)

Figure E.4 : Two forward models are tested: **(a)** isotropic rotation $\mathbf{H}_1 = 0.8\,R(\pi/6)$ with $\lambda = 0.1$, and **(b)** anisotropic operator $\mathbf{H}_2 = R(\pi/6)\,\mathrm{diag}(1.0, 0.3)\,R(\pi/4)$ with $\lambda = 0.5$. Black: measurements $\mathbf{y}$. Gold crosses: ground truth $\mathbf{x}_{1,i}$. Yellow: Wiener/MAP estimator. Red dots: Tikhonov fixed point $\mathbf{x}^\star(t)$ predicted by Eq. (E.24). Red $\times$: PnP iterates after convergence (matching theory to machine precision, $\|\hat{\mathbf{x}} - \mathbf{x}^\star(t)\|_{\text{mean}} < 5 \cdot 10^{-16}$). Blue dots: samples from the stochastic PnP variant, illustrating the variance restoration discussed in Section 8.2.2. As t increases (left to right), the Tikhonov regularization weakens and the iterates spread further from the prior mean toward the Wiener solution.

In this Gaussian inverse problem, the lookahead acts as a *regularization knob* rather than a parameterization of the distortion–perception frontier. Since the regime is strongly over-regularized, increasing t reduces bias and decreases the empirical MSE. This does not contradict the image experiments: exact distortion–perception optimality holds only in the denoising setting, while in inverse problems the interaction with the forward model breaks this guarantee. Here, the Gaussian case isolates the algorithmic role of the lookahead through its closed-form Tikhonov interpretation. This highlights that the exact DP interpretation holds in the denoising setting, while in inverse problems the lookahead acts as a forward-model-dependent regularization parameter.

Why $\mathbf{x}^\star(t)$ is not the MAP/Wiener estimator. One might expect PnP with $D_{s,t=s}$ (the MMSE denoiser) to recover the MAP estimator $\mathbf{x}_{\mathrm{MAP}}$, as in classical PnP with proximal operators. This is not the case. The flow-map denoiser $D_{s,t}(\mathbf{x}) = \Lambda(s,t)\,\mathbf{x}$ is not a proximal map: it operates at the intrinsic noise level σ_s of the interpolant $\mathbf{x}_s$, rather than the measurement noise σ_{obs}.

Matching (E.24) to $\mathbf{x}_{\mathrm{MAP}}$ would require $c(t)/\lambda = \sigma_{\mathrm{obs}}^2/\sigma_p^2$, which is incompatible with the contraction regime. Consequently, PnP-Flow operates at an effectively higher noise level, yielding over-regularized estimates. As the lookahead t increases, $c(t)$ decreases and $\mathbf{x}^\star(t)$ moves toward the Wiener solution, reducing bias. Thus, the lookahead controls a bias–variance trade-off induced by this noise mismatch, rather than traversing the optimal distortion–perception curve. In general, $\mathbf{x}^\star(t)$ should be interpreted as a lookahead-controlled Tikhonov estimator.

Effect of renoising in the Gaussian setting. To isolate the role of the renoising step, we repeat the experiment with the full stochastic PnP iteration. While the

deterministic case converges to a single Tikhonov solution, the stochastic iteration produces a distribution of estimates whose spread is largely independent of the step size λ.

Across different λ, the sample clouds consistently exhibit meaningful variance and align with the geometry of the data distribution, even when the mean estimate remains over-regularized. This indicates that the renoising step, together with the denoiser, primarily controls the variance of the estimator, while λ governs its mean through the data-consistency update.

As the lookahead increases, this variability is progressively amplified, yielding the distortion–perception behavior observed in the experiment.

E.3 Implementation

E.3.1 Training details

All models are trained unconditionally using the Flow Map objective [56] with the LSD loss (8.3).

Architecture. We use the SongUNet [7] – a UNet with self-attention – with a base channel width of 128 and channel multipliers $[1, 2, 3, 4]$, giving four resolution scales. Self-attention is applied at spatial resolutions 16×16 and 8×8. Each resolution level contains 3 residual blocks, dropout rate 0.1, and a cosine resampling filter $[1, 3, 3, 1]$.

Time conditioning. The network is conditioned on two scalars $(\alpha_r, \alpha_t) \in [0, 1]^2$ representing the interpolation times of the source and target. Both are encoded with a *positional (sinusoidal) embedding*. Following [56], the two conditioning scalars are embedded as $\mathrm{PE}(\alpha_t - \alpha_r)$ and $\mathrm{PE}(\alpha_r)$, summed, and passed through a two-layer

MLP with SiLU activations to produce a global conditioning vector injected into every residual block via adaptive group normalisation.

Loss weighting. For the off-diagonal term, we sample s, t uniformly from $\{(s,t) : 0 < s < t \leq 1\}$ as detailed in [56]. We follow [207] and learn a per-sample uncertainty weight for the LSD loss, we attach a small network that outputs a scalar log-variance $\log \sigma^2(\alpha_r, \alpha_t) \in \mathbb{R}$ per training example. The network takes the concatenated pair $[\alpha_r, \alpha_t] \in \mathbb{R}^{B\times 2\times H\times W}$ and generate a scalar used to weight the loss as $\mathcal{L} \propto e^{-\log\sigma^2}\,\ell + \log\sigma^2$.

Datasets and resolution.

- **CelebA-HQ** 128×128 [210]: 202,599 images; 90% train split.
- **AFHQ** 256×256 [212]: 16,130 images; 90% train split.

During training, 75% of the batch is used for the on-diagonal loss, while the remaining 25% for the off-diagonal one. Augmentation (horizontal flip, rotation, scaling, anisotropic scaling, translation) is applied with probability $p = 0.12$.

Optimization. All models are optimized with Adam [130] ($\beta_1 = 0.9$, $\beta_2 = 0.95$, $\varepsilon = 10^{-8}$) at a learning rate of 2×10^{-4} with no learning-rate schedule.

Compute. For training, we use 8 NVIDIA GPUs A100, while for inference we use a single GPU for all methods.

Table E.1 : Per-run training configuration.

Dataset	Resolution	Batch size	GPUs	Days
CelebA	128×128	128	8	10 ($\approx$ 700k iterations)
AFHQ	256×256	32	8	9 ($\approx$ 580k iterations)

E.3.2 Hyperparameters and baselines

For all baselines, unless otherwise specified, we use $\sigma = 0.05$ for Gaussian/motion deblurring and super-resolution, and $\sigma = 0.01$ for inpainting. Blur kernel size is set to 61 for AFHQ and 31 for CelebA.

Flow-Priors [235]. We follow the original parameterization with guidance weight λ and step size η. We use $\lambda = 10^5$, $\eta = 10^{-2}$, and $K = 1$ inner iteration in all settings. The number of outer iterations is fixed to $N = 100$.

D-Flow [236]. We tune the regularization parameter $\lambda \in \{0.1, 0.01, 0.001\}$ and the initialization blending parameter $\alpha \in \{0.1, 0.3, 0.5\}$, retaining $\lambda = 0.01$ and $\alpha = 0.1$. As recommended by the authors, the inner LBFGS optimization runs for 20 iterations and the ODE is solved with 6 Euler steps. The number of outer iterations is fixed to 20.

OT-ODE [218]. We tune the initial time $t_0 \in \{0.1, 0.2, 0.3, 0.4\}$ and the step-size schedule $\gamma \in \{\text{constant}, \sqrt{t}\}$. We use $\gamma = \text{constant}$ throughout. The retained values are $t_0 = 0.1$ for inpainting, $t_0 = 0.3$ for Gaussian deblurring and super-resolution, and $t_0 = 0.4$ for motion deblurring on AFHQ. The number of iterations is fixed to $N = 100$.

DPS-ODE [2]. This methods corresponds to the ODE (deterministic) version of the original DPS designed for diffusion models. We tune the guidance weight $\eta \in \{10^2, 10^3, 10^4\}$, retaining $\eta = 10^3$ as default, $\eta = 10^2$ on CelebA box inpainting, and $\eta = 10^2$ on AFHQ Gaussian deblurring. The number of iterations is fixed to $N = 100$.

PnP-Flow w/ $t = s$ [223] and w/ $t = 1$ (ours). We adjust the learning-rate exponent $\alpha \in \{0.05, 0.1\}$ and the number of time steps $N \in \{30, 50, 100, 300\}$. We use $\alpha = 0.05$ in most settings and $\alpha = 0.1$ for motion deblurring; $N = 100$ by default, $N = 300$ for hard random inpainting on AFHQ, and $N \in \{30, 50\}$ for motion deblurring.

E.3.3 Computational cost

Running time. Table E.2 reports per-image reconstruction times on a single A100 GPU. PnP-Flow is the fastest method we compare against, while reaching perceptual quality typically associated with posterior sampling. This illustrates a concrete advantage of using a flow map as a PnP denoiser: it inherits the speed of MAP-style approaches without sacrificing perceptual fidelity.

Table E.2 : Running time for each method. PnP with a flow map as a denoiser is the fastest method, while generating images that are closer to those from posterior sampling methods.

Sampler	Time [sec]
Flow-priors	34.8
D-Flow	322
OT-ODE	8.4
DPS-ODE	7.7
PnP-flow	4.5

Memory footprint. Overall, almost all the methods fit in a single A100 with a batch of 5 images for AFHQ 256x256; the only one that could not fit is D-flow, which needed to use a single image per batch.

E.3.4 Experiment of the teaser plot

We detail here the experiment underlying Fig. 8.1. The data distribution p_1 is a 2D isotropic mixture of Gaussians,

$$p_1 = \tfrac{1}{2}\mathcal{N}(\boldsymbol{\mu}_a, \sigma_c^2\mathbf{I}) + \tfrac{1}{2}\mathcal{N}(\boldsymbol{\mu}_b, \sigma_c^2\mathbf{I}), \quad \boldsymbol{\mu}_a = (4.5, 3.5),\ \boldsymbol{\mu}_b = (3.5, 5.5),\ \sigma_c = 0.35.$$

Observations are generated according to the linear forward model in (2.1), with $\mathcal{A}(\cdot) = \mathbf{H}$ a fixed rotation of angle $\pi/6$ followed by isotropic rescaling.

Similarly to the case in Appendix E.2.3, the MMSE denoiser $D_s(x) = \mathbb{E}[X_1 \mid X_s = x]$ admits a closed-form expression via the component posteriors. We use this

analytic denoiser to construct the flow-map denoiser

$$D_{s,t}(\mathbf{x}) = \mathbf{x} + (1-s)\,\mathbf{v}(\mathbf{x}, s, t), \quad \mathbf{v}(\mathbf{x}, s, t), = \frac{1}{t-s}\int_s^t \frac{D_\tau(\mathbf{x}_\tau) - \mathbf{x}_\tau}{1-\tau}\, d\tau,$$

where the average velocity is approximated by Euler integration of the flow $\dot{\mathbf{x}}_\tau = (D_\tau(\mathbf{x}_\tau) - \mathbf{x}_\tau)/(1-\tau)$ using 100 steps. Using this approximation of the average denoiser, we run PnP-Flow in Alg. 6 with $s = 0.45$ (which is fixed to isolate the behavior of the lookahead), step size $\gamma_k = \lambda\sqrt{1-k/K}$ with $\lambda = 0.4$, and $K = 50$ iterations. We initialize with $\mathbf{x}^{(0)} = \mathbf{H}^{-1}\mathbf{y}$ and run the algorithm independently for each sample. We consider three lookaheads $t \in \{s,\, s+0.27,\, 1\}$, corresponding to the MMSE, an intermediate, and the full lookahead regime.

The behavior follows the distortion–perception trade-off: at $t = s$, reconstructions collapse toward the inter-mode mean (low variance, high distortion), while at $t = 1$ the bimodal structure is recovered (higher perceptual quality). Intermediate lookaheads interpolate smoothly between these regimes, all using a single denoiser.

E.4 Additional experiments

E.4.1 Results with error bars

Here, we provide the results from Section 8.6 with the corresponding error bars.

Table E.3 : Quantitative results for random inpainting (90%) with $\sigma = 0.01$ on CelebA and AFHQ. Best results are in **bold**.

Sampler	CelebA 128×128				AFHQ 256×256			
	PSNR↑	LPIPS↓	FID↓	DISTS↓	PSNR↑	LPIPS↓	FID↓	DISTS↓
Flow-priors	26.50 ± 1.00	0.089 ± 0.011	59.43	0.132 ± 0.012	23.94 ± 0.79	0.193 ± 0.021	18.96	0.164 ± 0.011
D-Flow	<u>26.55</u> ± 2.79	<u>0.066</u> ± 0.031	50.27	0.110 ± 0.031	22.80 ± 3.80	0.192 ± 0.146	18.31	0.137 ± 0.083
OT-ODE	22.16 ± 0.74	0.158 ± 0.019	84.92	0.179 ± 0.016	21.14 ± 0.26	0.263 ± 0.019	16.03	0.181 ± 0.012
DPS-ODE	25.60 ± 1.32	0.070 ± 0.054	**45.74**	**0.106** ± 0.024	23.08 ± 0.73	<u>0.127</u> ± 0.021	**8.69**	<u>0.098</u> ± 0.014
PnP-Flow ($t = s$)	**27.29** ± 1.15	0.073 ± 0.073	48.88	0.113 ± 0.013	**24.99** ± 0.78	0.162 ± 0.013	13.49	0.135 ± 0.008
PnP-Flow ($t = 1$)	26.03 ± 1.03	**0.065** ± 0.009	<u>46.79</u>	<u>0.109</u> ± 0.012	23.23 ± 0.739	**0.122** ± 0.012	<u>9.11</u>	**0.095** ± 0.006

Table E.4 : Quantitative results for motion deblurring with $\sigma = 0.05$ on CelebA and AFHQ. Best results are in **bold**.

Sampler	CelebA 128×128				AFHQ 256×256			
	PSNR↑	LPIPS↓	FID↓	DISTS↓	PSNR↑	LPIPS↓	FID↓	DISTS↓
Flow-priors	25.07 ± 1.34	0.151 ± .035	114.41	0.188 ± 0.021	23.14 ± 2.26	**0.207** ± 0.063	27.73	<u>0.154</u> ± 0.028
D-Flow	**29.77** ± 1.67	**0.071** ± 0.022	73.15	**0.112** ± 0.019	**25.52** ± 2.60	<u>0.230</u> ± 0.085	<u>19.92</u>	0.156 ± 0.042
OT-ODE	21.07 ± 7.43	0.198 ± 0.157	106.78	0.202 ± 0.113	16.73 ± 5.51	0.348 ± 0.165	77.53	0.234 ± 0.122
DPS-ODE	27.36 ± 5.16	0.101 ± 0.063	57.02	0.133 ± 0.054	20.49 ± 3.66	0.292 ± 0.123	36.71	0.179 ± 0.60
PnP-Flow ($t = s$)	<u>29.79</u> ± 2.70	0.114 ± 0.056	<u>53.65</u>	0.130 ± 0.034	<u>25.40</u> ± 2.15	0.347 ± 0.076	20.37	0.201 ± 0.033
PnP-Flow ($t = 1$)	29.42 ± 2.80	<u>0.082</u> ± 0.042	**52.61**	<u>0.119</u> ± 0.029	23.31 ± 2.50	**0.207** ± 0.059	**14.04**	**0.137** ± 0.028

E.4.2 Additional inverse problems

Gaussian deblurring. We consider a kernel of size 31×31 for CelebA and 61×61 for AFHQ, with $\sigma_b = 3$. The quantitative results are in Table E.5.

Super-resolution. We consider super-resolution with a cubic method, and a factor of 4. The quantitative results are in Table E.6.

Box-inpainting. We consider a box of size 80×80 for AFHQ and 60×60 for CelebA. The quantitative results are in Table E.7.

Table E.5 : Quantitative results for Gaussian deblurring w/$\sigma = 0.05$ across CelebA and AFHQ datasets. Best results are in **bold**.

Sampler	**CelebA** 128×128				**AFHQ** 256×256			
	PSNR↑	**LPIPS↓**	**FID↓**	**DISTS↓**	**PSNR↑**	**LPIPS↓**	**FID↓**	**DISTS↓**
Flow-priors	**30.41**	**0.044**	80.53	0.125	22.73	0.182	12.87	0.124
D-Flow	26.59	0.081	66.93	**0.120**	22.80	0.175	12.31	0.121
OT-ODE	25.77	0.100	63.10	0.129	24.04	**0.168**	11.51	**0.114**
DPS-ODE	24.80	0.129	72.13	0.148	20.62	0.253	15.47	0.158
PnP-Flow ($t = s$)	27.25	0.233	70.80	0.183	**25.14**	0.453	25.67	0.235
PnP-Flow ($t = 1$)	26.60	0.157	**61.57**	0.152	23.41	0.191	**11.09**	0.122

Table E.6 : Quantitative results for super-resolution $\times 4$ w/$\sigma = 0.05$ across CelebA and AFHQ datasets. Best results are in **bold**.

Sampler	**CelebA** 128×128				**AFHQ** 256×256			
	PSNR↑	**LPIPS↓**	**FID↓**	**DISTS↓**	**PSNR↑**	**LPIPS↓**	**FID↓**	**DISTS↓**
Flow-priors	26.14	0.227	96.46	0.211	22.93	0.338	20.41	0.209
D-Flow	24.84	0.137	119.93	0.181	23.77	0.389	20.69	0.204
OT-ODE	24.79	0.142	81.18	0.173	20.36	0.400	44.28	0.247
DPS-ODE	26.38	**0.083**	**55.10**	**0.123**	24.18	**0.165**	**10.88**	**0.114**
PnP-Flow ($t = s$)	**26.94**	0.235	71.30	0.187	**25.10**	0.429	26.45	0.232
PnP-Flow ($t = 1$)	26.68	0.166	63.97	0.158	24.71	0.269	13.88	0.159

Table E.7 : Quantitative results for box inpainting w/$\sigma = 0.01$ across CelebA and AFHQ datasets. Best results are in **bold**.

Sampler	**CelebA** 128×128				**AFHQ** 256×256			
	PSNR↑	**LPIPS↓**	**FID↓**	**DISTS↓**	**PSNR↑**	**LPIPS↓**	**FID↓**	**DISTS↓**
Flow-priors	25.09	0.062	36.28	<u>0.068</u>	<u>26.51</u>	<u>0.048</u>	16.62	<u>0.052</u>
D-Flow	<u>26.36</u>	**0.056**	34.70	0.073	26.87	0.069	<u>11.08</u>	0.079
OT-ODE	24.65	0.065	<u>31.99</u>	0.073	20.80	0.085	25.97	0.095
DPS-ODE	24.13	0.115	58.64	0.132	21.07	0.069	19.48	0.071
PnP-Flow ($t = s$)	**26.77**	<u>0.059</u>	32.29	**0.062**	**28.33**	0.057	13.79	<u>0.052</u>
PnP-Flow ($t = 1$)	24.77	0.062	**26.07**	**0.062**	26.40	**0.031**	**7.61**	**0.033**

Non-linear

JPEG. We consider JPEG, following the setting from Section 8.6.2. Given that OT-ODE and Flow priors cannot handle non-linear degradations, we only compare with DPS-ODE; the result is shown in Table E.8 for AFHQ.

Table E.8 : Quantitative results for JPEG w/$\sigma = 0.01$ on AFHQ datasets. Best results are in **bold**.

Sampler	**AFHQ** 256×256			
	PSNR↑	**LPIPS↓**	**FID↓**	**DISTS↓**
DPS-ODE	18.76	0.287	19.05	0.184
PnP-Flow ($t = s$)	**26.79**	0.245	20.33	0.166
PnP-Flow ($t = 1$)	25.52	**0.143**	**12.84**	**0.124**

Colorization. We consider also Colorization; the result is shown in Table E.9 for AFHQ.

Table E.9 : Quantitative results for Colorization w/$\sigma = 0.001$ on AFHQ datasets. Best results are in **bold**.

Sampler	AFHQ 256×256			
	PSNR↑	LPIPS↓	FID↓	DISTS↓
DPS-ODE	18.60	0.413	33.79	0.236
PnP-Flow ($t = s$)	**26.30**	0.111	9.95	0.110
PnP-Flow ($t = 1$)	25.37	**0.105**	**9.47**	**0.094**

Gaussian deblurring with Poisson noise

Lastly, we consider Poisson noise, showing that our analysis goes beyond Gaussian noise. In particular, we normalized each image to $[0, 1]$, and consider an independent Poisson process normalised by a peak rate $\lambda > 0$:

$$y_i \,\big|\, \mathbf{x} \;\sim\; \text{Poisson}\big(\lambda\,(\mathbf{H}\mathbf{x})_i\big), \qquad i = 1, \ldots, N, \tag{E.25}$$

so that $\mathbb{E}[y_i \mid \mathbf{x}] = (\mathbf{H}\mathbf{x})_i$ and $\text{Var}[y_i \mid \mathbf{x}] = (\mathbf{H}\mathbf{x})_i/\lambda$. The signal-to-noise ratio scales as $\sqrt{\lambda}$, where lower λ corresponds to fewer photons and a noisier observation; we consider $\lambda = 20$. Given this noise model, the corresponding negative log-likelihood, used as the data-fidelity term in Alg. 6, is (up to a constant in $\mathbf{x}$)

$$g(\mathbf{x}) \;=\; \sum_{i=1}^{N} \big[\, (\mathbf{H}\mathbf{x})_i \;-\; y_i \log(\mathbf{H}\mathbf{x})_i \,\big]. \tag{E.26}$$

The results as a function of the number of steps is shown in Fig. E.5, while qualitative comparisons are in Fig. E.25.

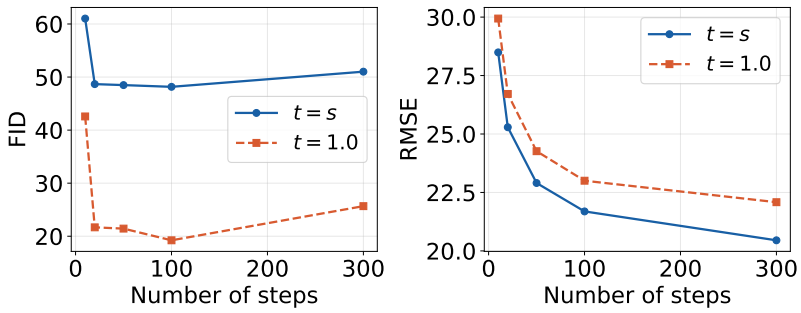


Figure E.5 : Ablation of PnP-flow as a function of the number of steps with Poisson noise and Gaussian deblurring

E.4.3 Visualization of the iterations for different lookaheads

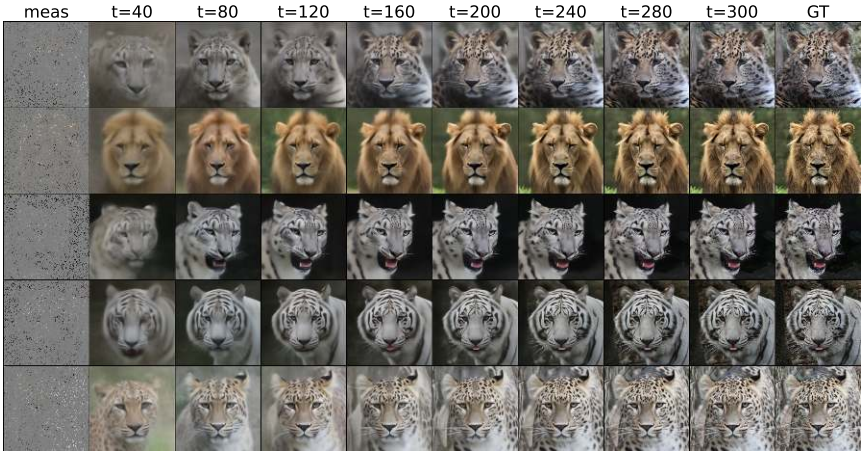


Figure E.6 : Progression of random inpainting with lookahead $t = s$

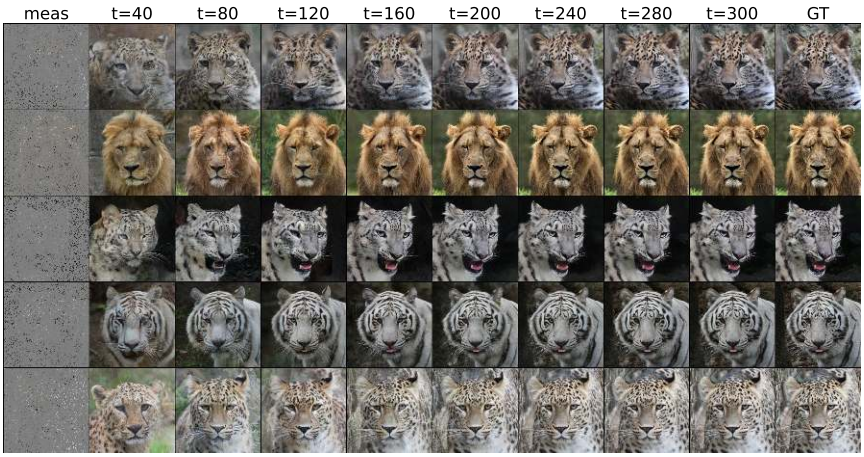


Figure E.7 : Progression of random inpainting with lookahead $t = 1$

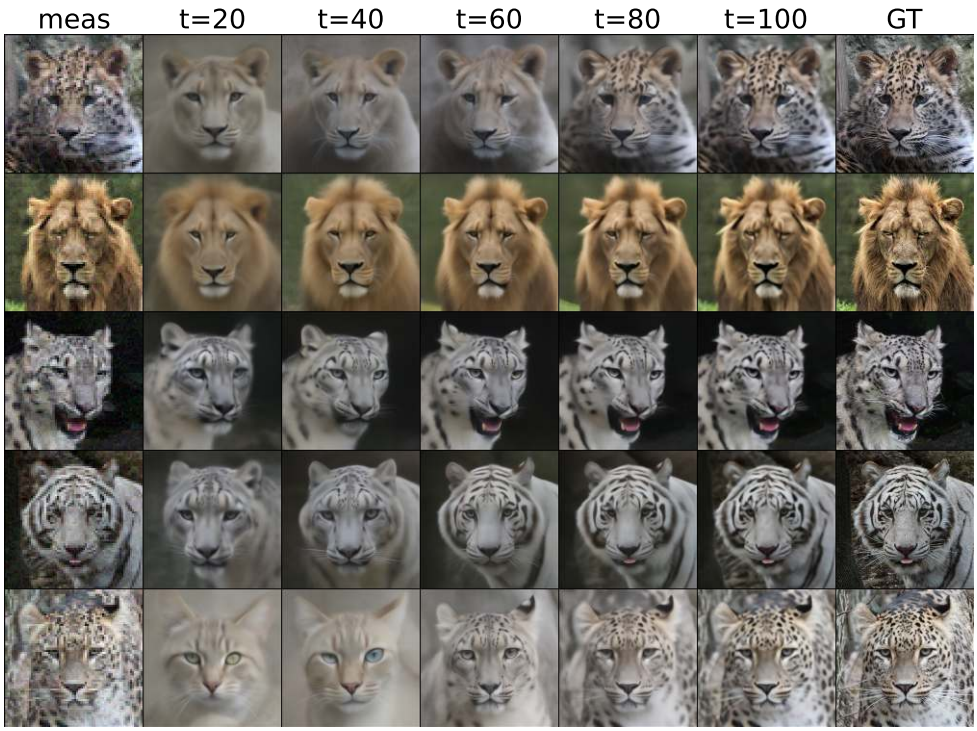


Figure E.8 : Progression of SR $\times 4$ with lookahead $t = s$

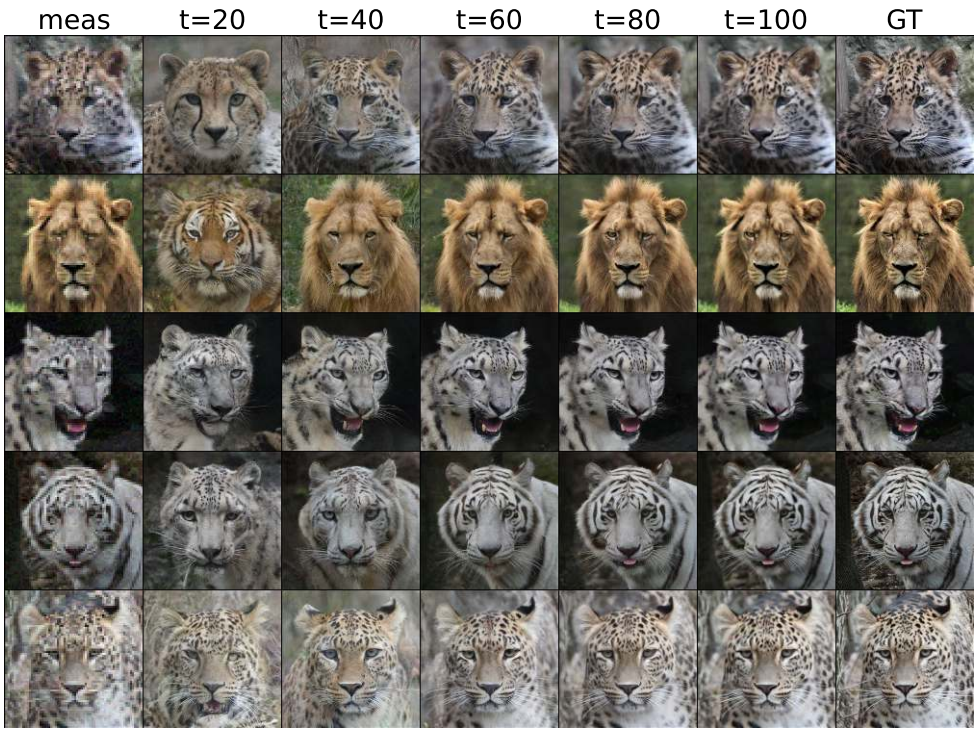


Figure E.9 : Progression of SR $\times 4$ with lookahead $t = 1$

E.4.4 Distortion–perception of CelebA

We include here the same plot as in Section 8.2.3; it is shown in Fig. E.10. Again, varying t produces a smooth, monotonic trajectory in the DP plane: small lookaheads minimize RMSE, large lookaheads minimize FID, and intermediate values smoothly interpolate between the two.

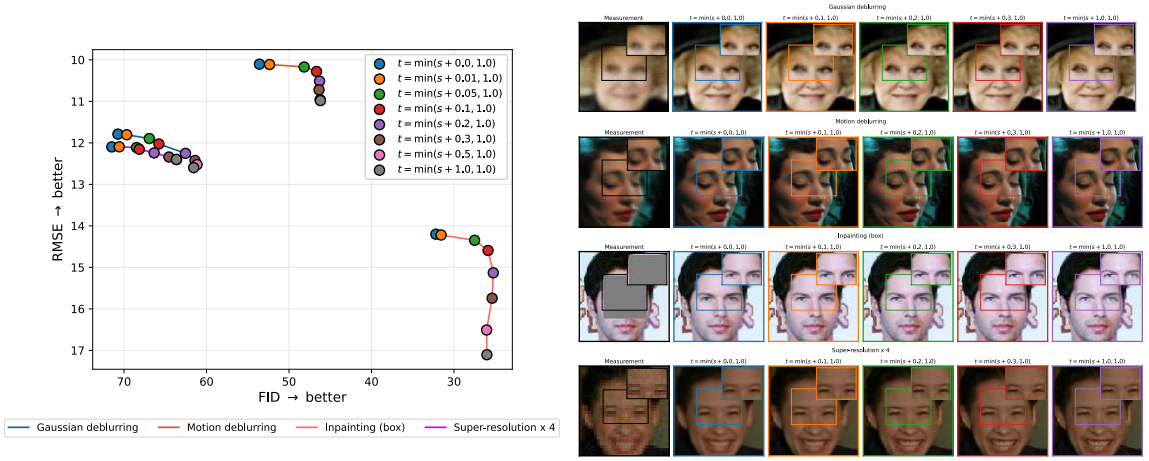


Figure E.10 : Distortion–perception trade-off traced by a single flow map on CelebA. Same setup as Fig. 8.3, but on CelebA 128 × 128. For each of four linear inverse problems (Gaussian deblurring, motion deblurring, box inpainting, and super-resolution), we sweep the lookahead t of our average denoiser within a PnP solver and report RMSE (distortion, ↓) against FID (perception, ↓). Each marker corresponds to a different value of t; small lookaheads cluster near the low-distortion regime (top of each curve) while large lookaheads cluster near the low-FID regime (right). The same trained model spans the entire frontier across all four tasks. Reconstructions on the right illustrate the perceptual change as t increases.

E.4.5 Performance as a function of the number of PnP iterations

We study how the two endpoint lookaheads, $t = s$ and $t = 1$, depend on the number of PnP iterations N. Results for AFHQ are reported on super-resolution ×4 (Fig. E.11), Gaussian deblurring (Fig. E.12) and box inpainting (Fig. E.13); we also include for CelebA in Fig. E.14.

The two regimes behave qualitatively differently. At $t = s$, both RMSE and FID decrease monotonically with N and plateau, consistent with the local-contraction analysis of Theorem 3. At $t = 1$, the perceptual trajectory is non-monotonic: FID

drops sharply within the first few tens of iterations, reaches its minimum, and then drifts back up as N continues to grow. The effect is most pronounced for Gaussian deblurring, where the best FID at $t = 1$ (≈ 11 at $N \approx 20$) is reached an order of magnitude earlier than the best FID at $t = s$ (≈ 23.5 at $N \approx 300$). This behavior is consistent with the lookahead-dependent contraction factor in Theorem 3. The local Lipschitz constant $L_D^{\mathrm{loc}}(t)$ grows with t, reflecting the transition from a contractive MMSE estimator at $t = s$ to a less regular, perceptually-oriented estimator at $t = 1$. RMSE remains well-behaved in both cases, indicating that the drift is along the perceptual axis.

The practical takeaway is that the two endpoints call for different iteration schemes: the perceptual endpoint ($t = 1$) is best run for fewer steps than the distortion endpoint ($t = s$). The values of N used in our main experiments (Appendix E.3.2) reflect this asymmetry. This behavior also explains why flow maps and consistency models are useful denoisers for solving restoration problem with few-steps as in [220, 222].

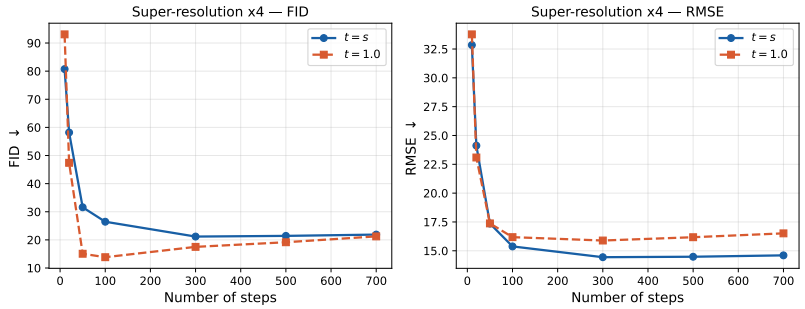


Figure E.11 : AFHQ - Super-resolution $\times 4$. Ablation of PnP-flow as a function of the number of steps.

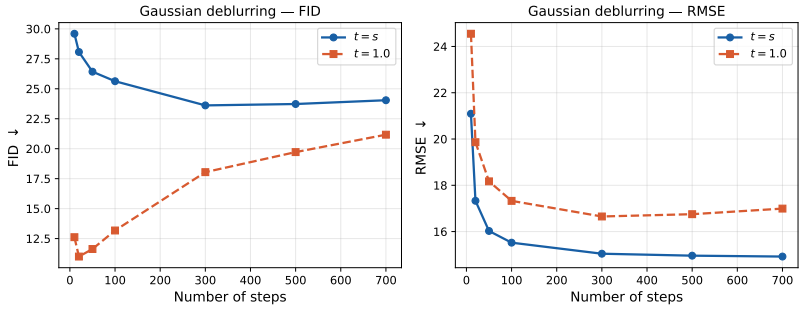


Figure E.12 : AFHQ - Gaussian deblurring. Ablation of PnP-flow as a function of the number of steps.

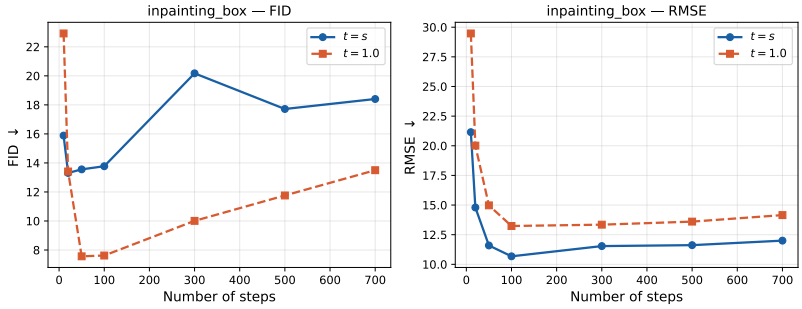


Figure E.13 : AFHQ - Box inpainting. Ablation of PnP-flow as a function of the number of steps.

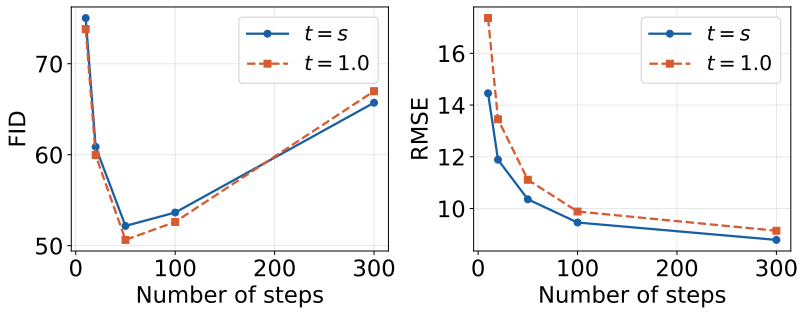


Figure E.14 : CelebA - Motion deblurring. Ablation of PnP-flow as a function of the number of steps.

E.4.6 Failure modes and limitations

While our contributions is using flow maps as denoisers in a PnP scheme, enhancing the perceptual quality of the estimated image, we observed that for some degradations, the DP traversing might not hold exactly. For instance, in phase retrieval with oversample of 4, we observe that PnP with flow maps outperform the instantaneous denoiser case ($t = s$) only after certain number of steps, as shown in Fig. E.15. Furthermore, the overall performance is not strong, as shown quantiative in Table E.10 and qualitative in Fig. E.16.

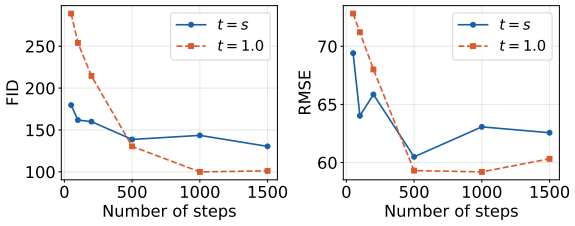


Figure E.15 : Performance as a function of the number of steps

Table E.10 : Quantitative results for Phase retrieval w/$\sigma = 0.01$ on AFHQ datasets. Best results are in **bold**.

Sampler	AFHQ 256×256			
	PSNR↑	LPIPS↓	FID↓	DISTS↓
PnP-Flow ($t = s$)	**17.36**	0.454	143.55	0.316
PnP-Flow ($t = 1$)	20.88	**0.317**	**99.95**	**0.240**

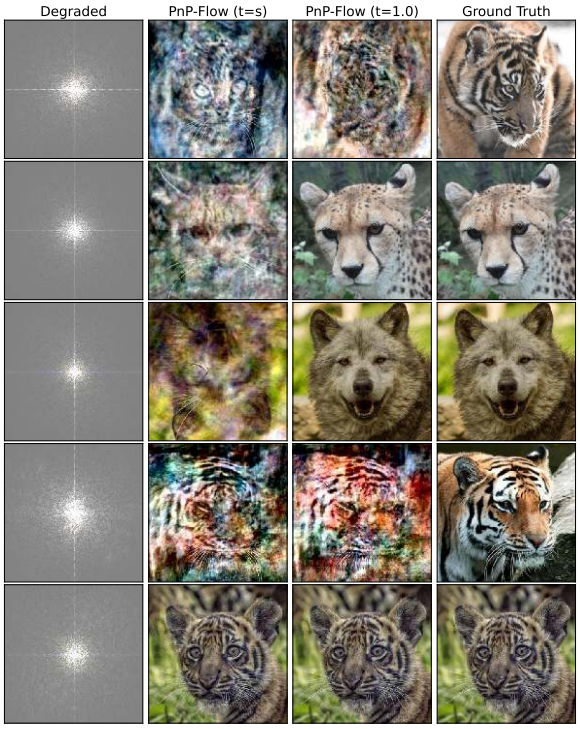


Figure E.16 : Reconstruction for phase retrieval

E.4.7 Visual results

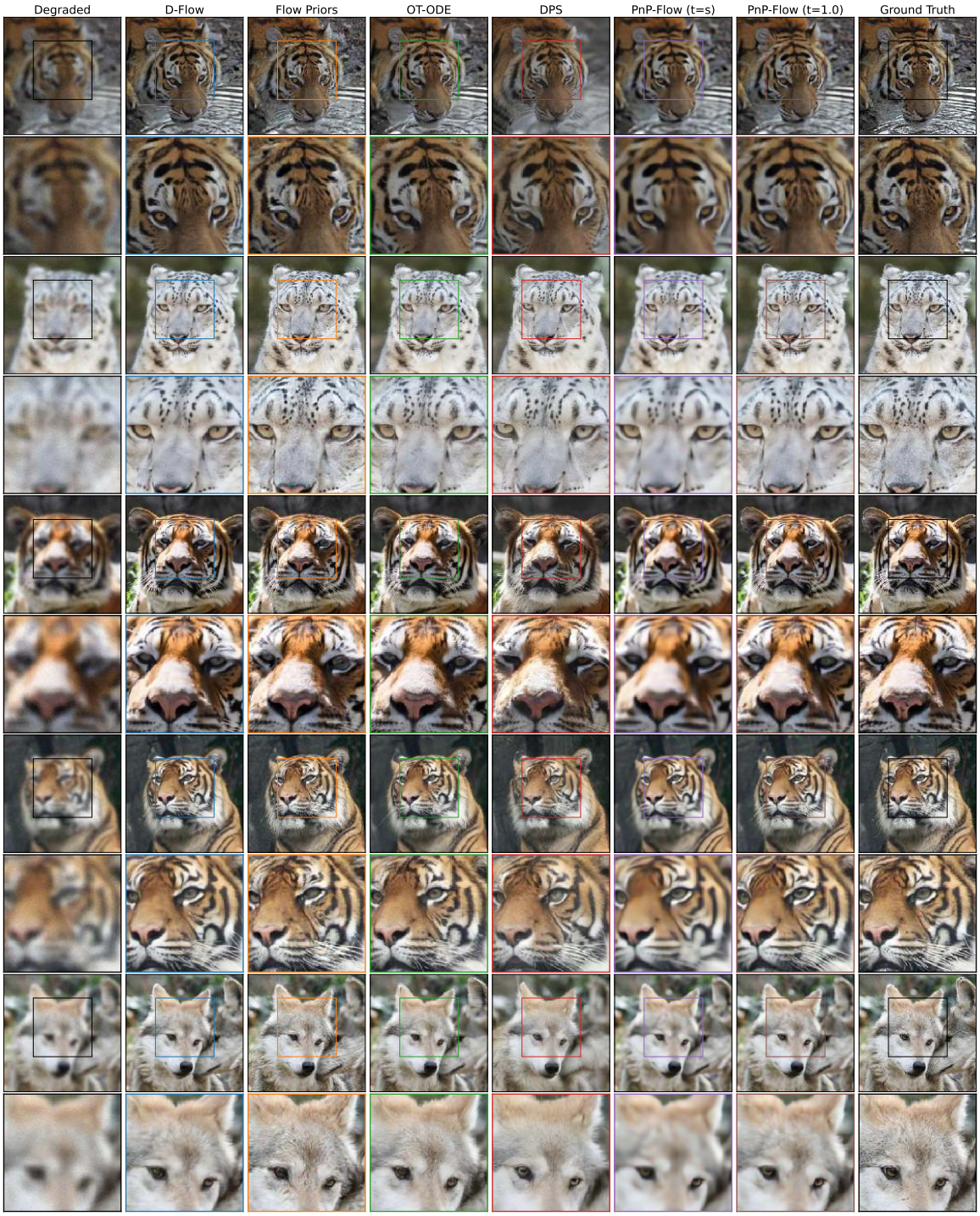


Figure E.17 : Gaussian deblurring with kernel size 61×61 and $\sigma_b = 3$, and $\sigma = 0.05$

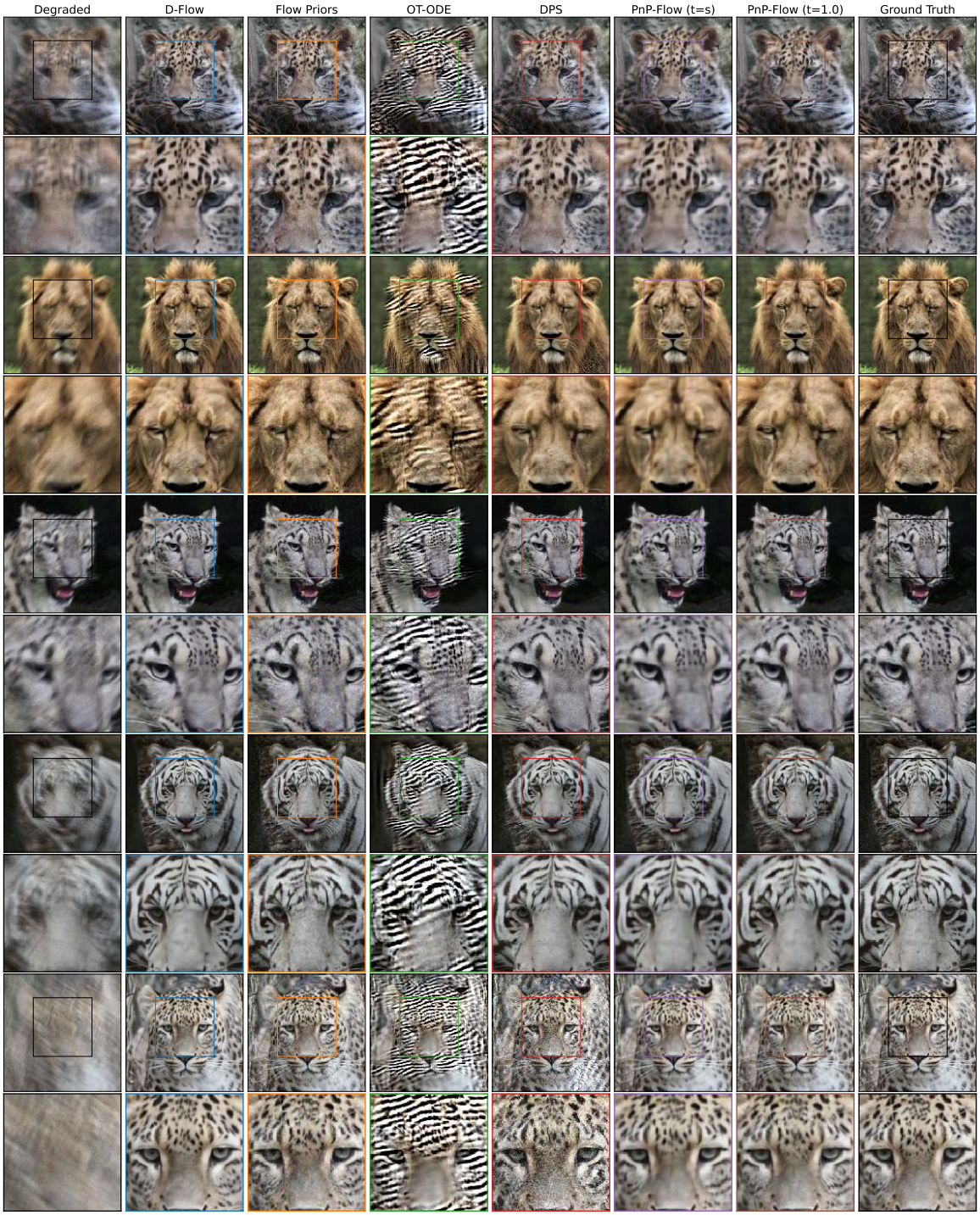


Figure E.18 : Motion deblurring with kernel size 61×61 and $\sigma_b = 1$, and $\sigma = 0.1$

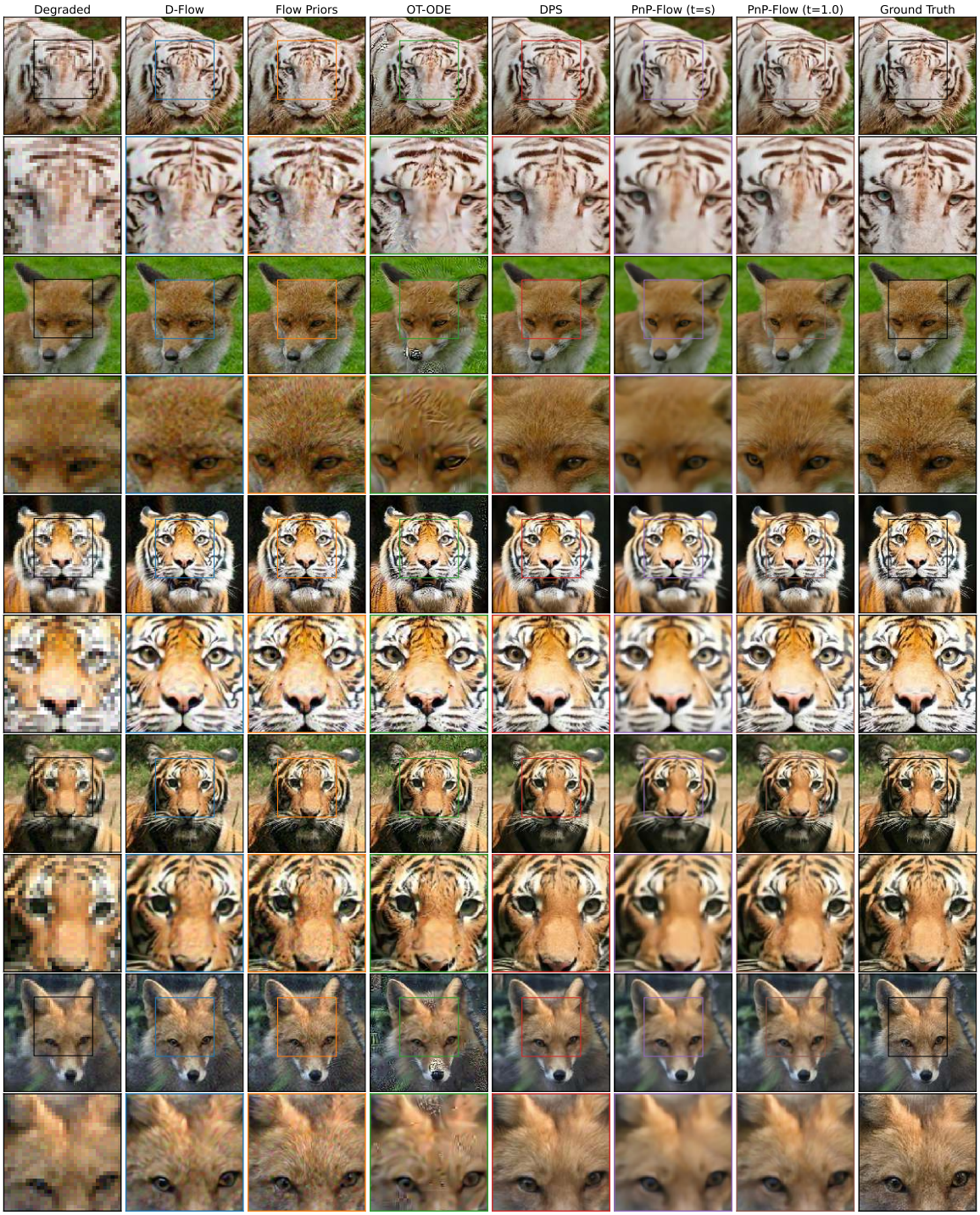


Figure E.19 : Super resolution $\times 4$, and $\sigma = 0.05$

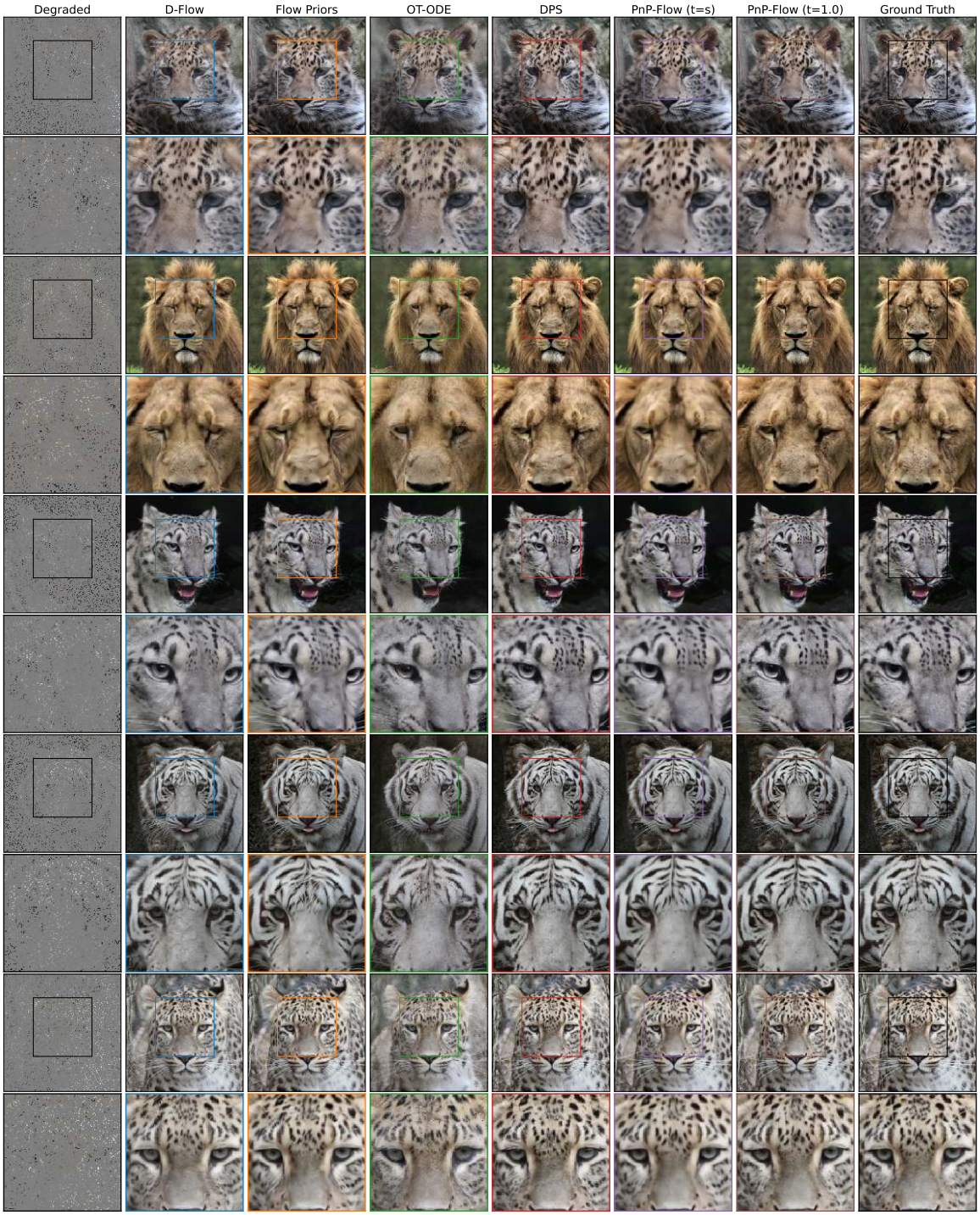


Figure E.20 : Random inpainting with 90% drop rate, and $\sigma = 0.01$

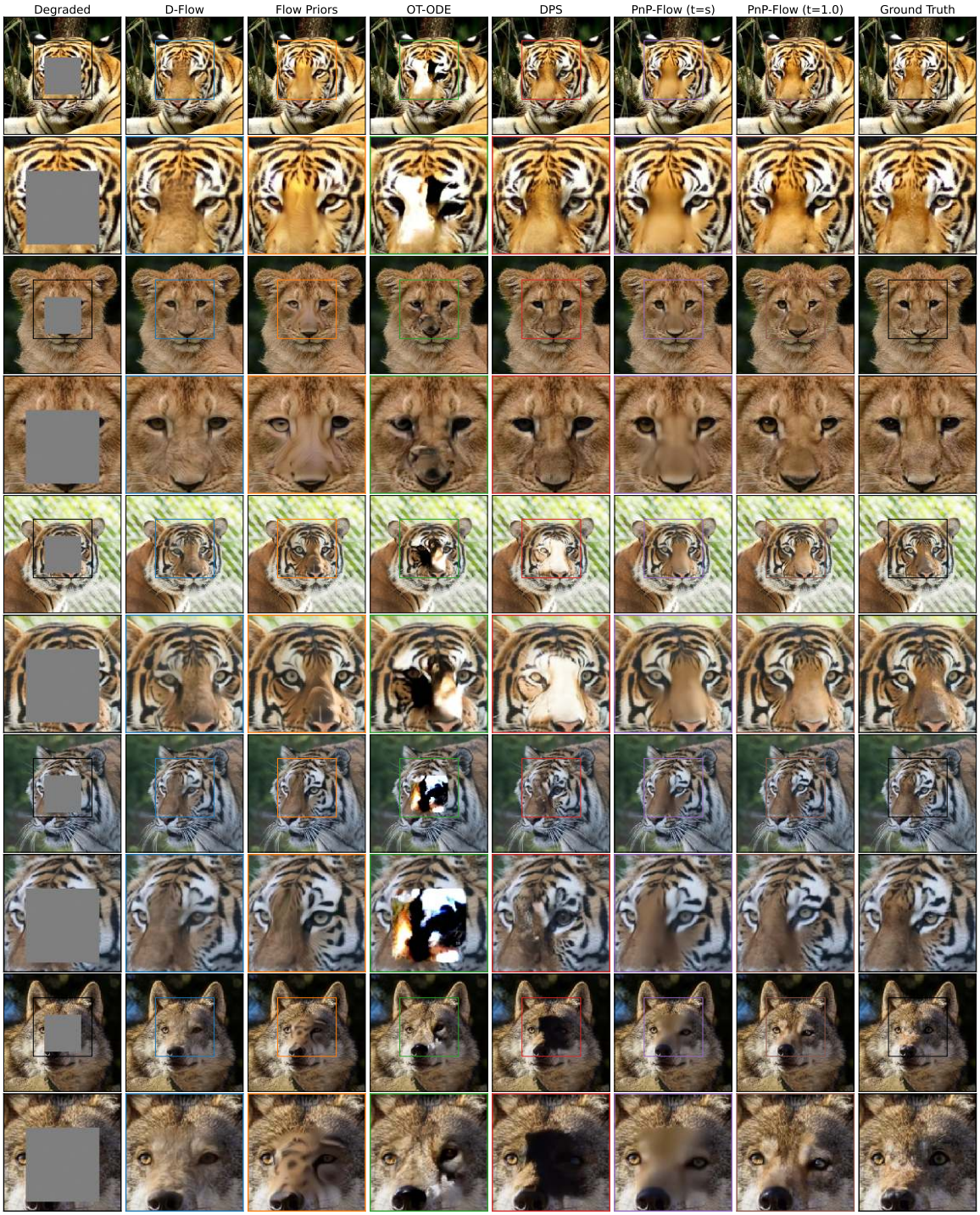


Figure E.21 : Box inpainting with mask 80×80, and $\sigma = 0.01$

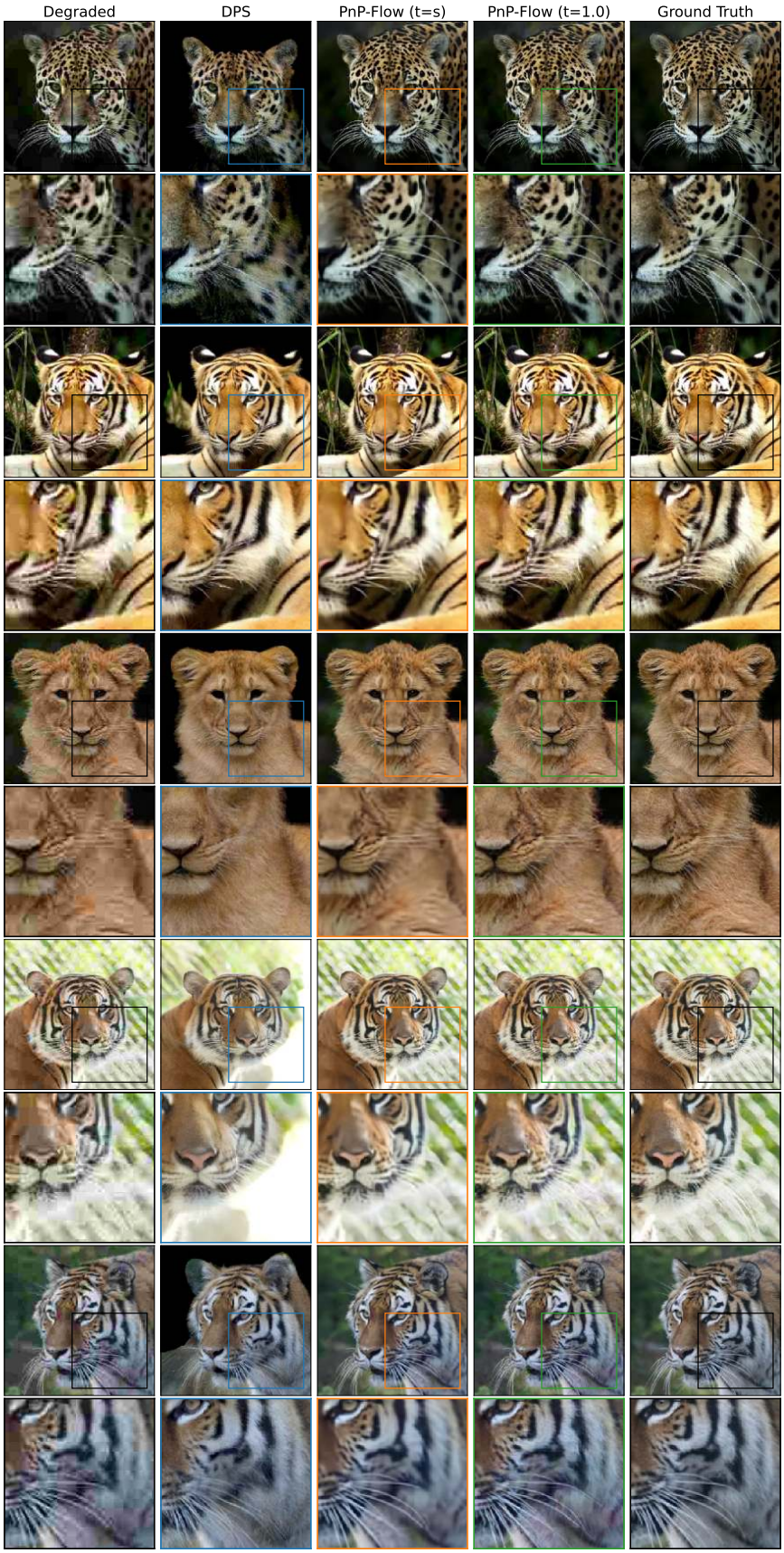


Figure E.22 : JPEG compresion

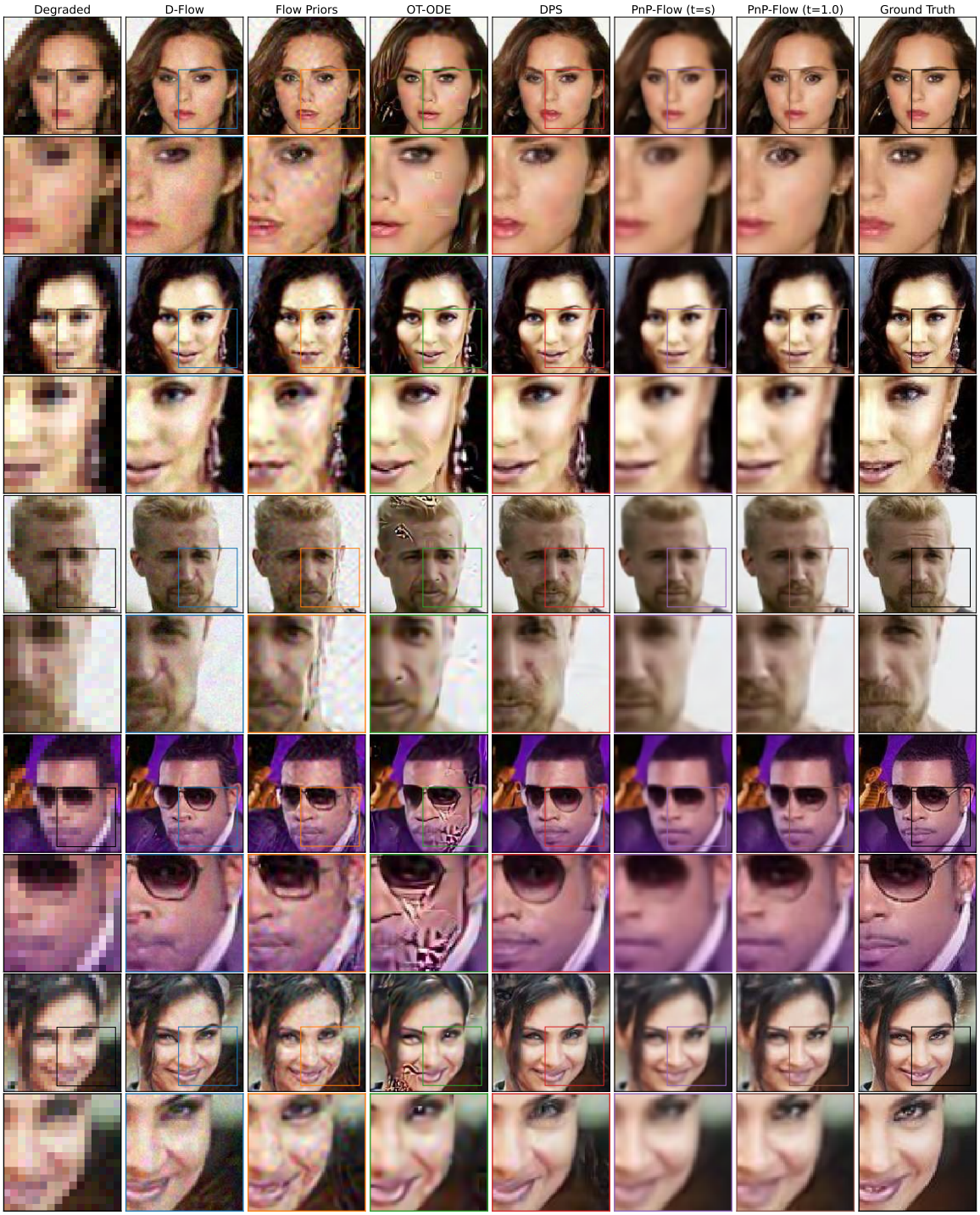


Figure E.23 : CelebA - Super resolution $\times 4$, and $\sigma = 0.05$

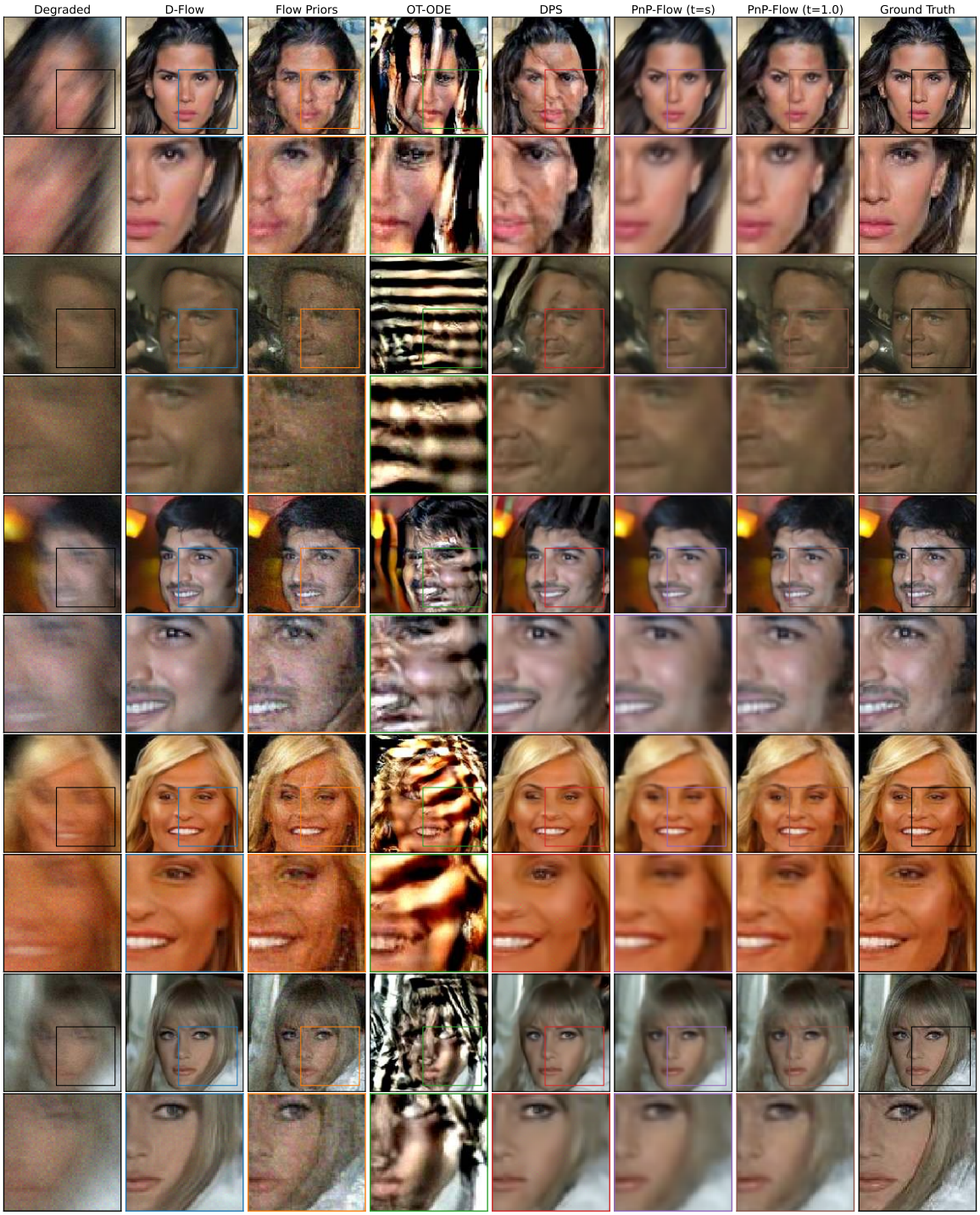


Figure E.24 : CelebA - Motion deblurring

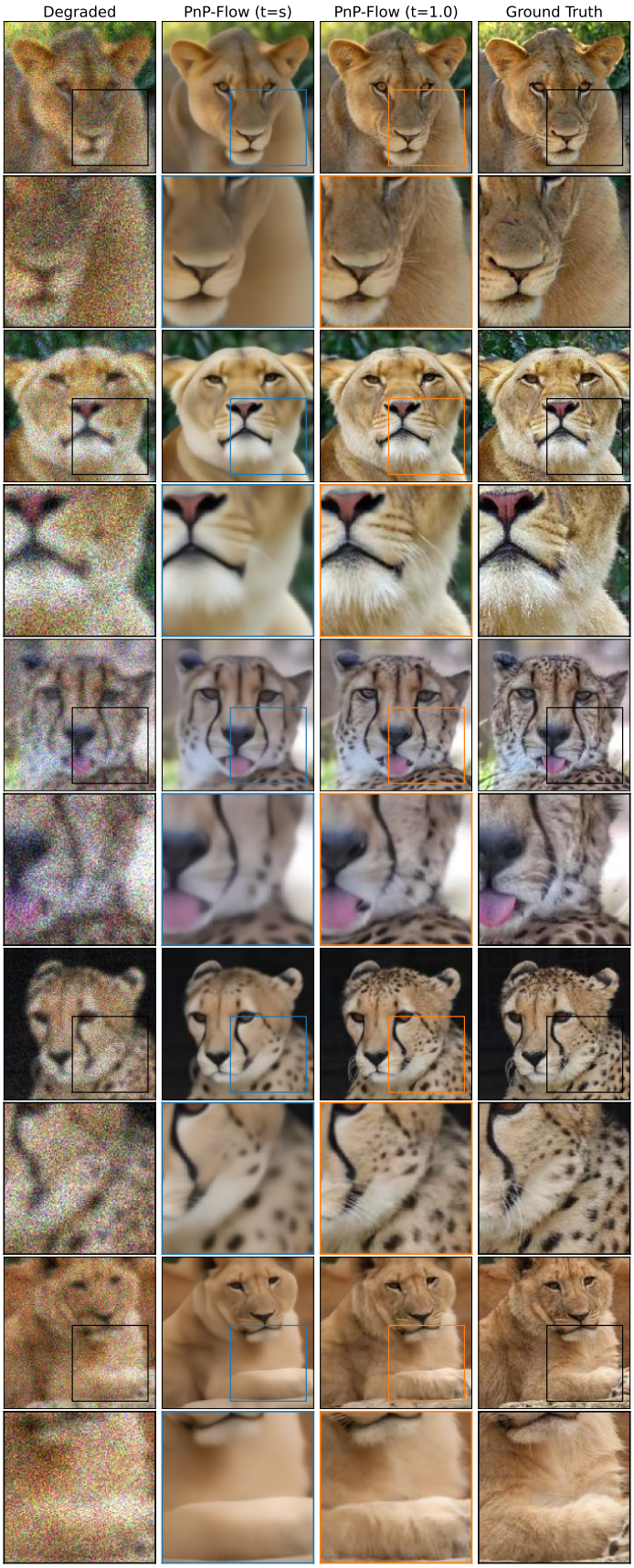


Figure E.25 : Gaussian deblurring with Poisson noise

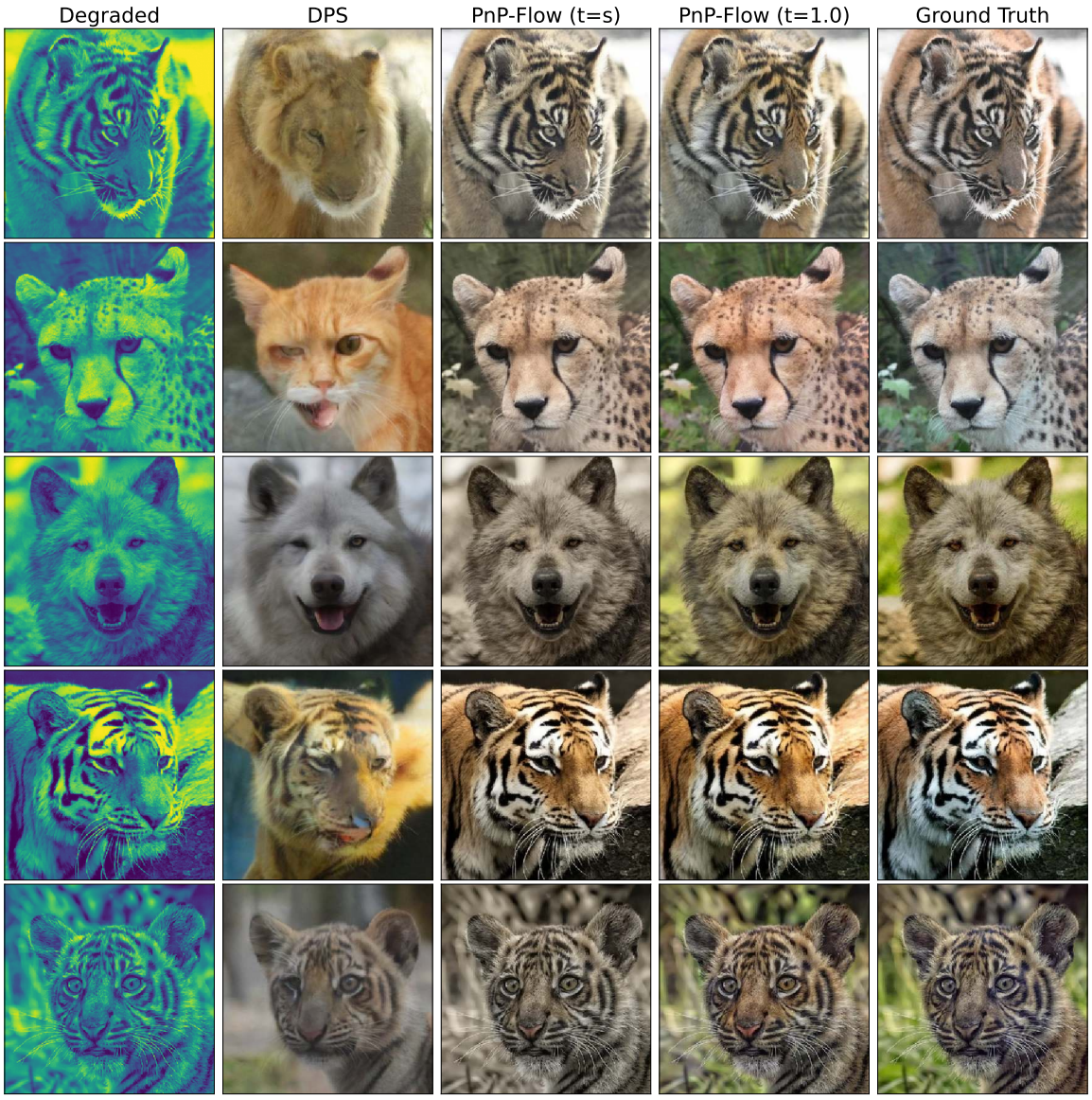


Figure E.26 : Colorization

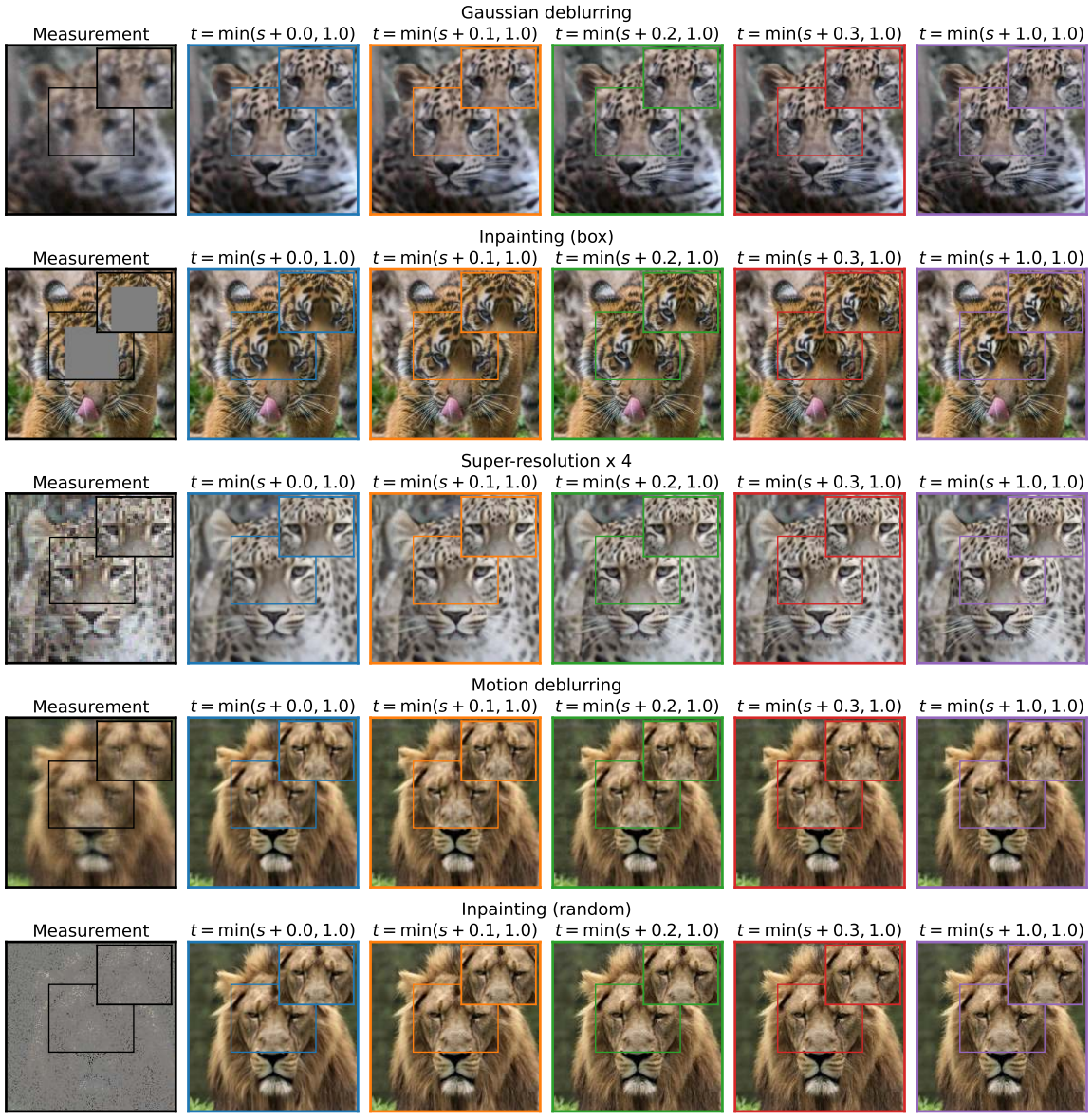


Figure E.27 : Larger version of the images in Fig. 8.3.

Bibliography

[1] D. Freirich, T. Michaeli, and R. Meir, "A theory of the distortion-perception tradeoff in wasserstein space," *Advances in Neural Inf. Process. Syst. (NIPS)*, vol. 34, pp. 25661–25672, 2021.

[2] H. Chung, J. Kim, M. T. Mccann, M. L. Klasky, and J. C. Ye, "Diffusion posterior sampling for general noisy inverse problems," in *Intl. Conf. Learn. Repr. (ICLR)*, 2023.

[3] P. Dhariwal and A. Nichol, "Diffusion models beat gans on image synthesis," *Advances in Neural Inf. Process. Syst. (NIPS)*, vol. 34, pp. 8780–8794, 2021.

[4] C. Saharia, W. Chan, H. Chang, C. Lee, J. Ho, T. Salimans, D. Fleet, and M. Norouzi, "Palette: Image-to-image diffusion models," in *ACM SIGGRAPH 2022 conference proceedings*, pp. 1–10, 2022.

[5] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli, "Deep unsupervised learning using nonequilibrium thermodynamics," in *Intl. Conf. on Machine Learning (ICML)*, pp. 2256–2265, PMLR, 2015.

[6] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," *Advances in Neural Inf. Process. Syst. (NIPS)*, vol. 33, pp. 6840–6851, 2020.

[7] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole, "Score-based generative modeling through stochastic differential equations," in

Intl. Conf. Learn. Repr. (ICLR), 2021.

[8] Y. Lipman, R. T. Chen, H. Ben-Hamu, M. Nickel, and M. Le, "Flow matching for generative modeling," in *Intl. Conf. Learn. Repr. (ICLR)*, 2023.

[9] N. Zilberstein, A. Sabharwal, and S. Segarra, "Solving linear inverse problems using higher-order annealed Langevin diffusion," *IEEE Trans. Signal Process.*, 2024.

[10] N. Zilberstein, M. Mardani, and S. Segarra, "Repulsive latent score distillation for solving inverse problems," in *Intl. Conf. Learn. Repr. (ICLR)*, 2025.

[11] N. Zilberstein, S. Segarra, and L. Chamon, "Sampling with shielded Langevin Monte Carlo using navigation potentials," in *Asilomar Conf. on Signals, Syst., and Comp.*, 2025.

[12] H. Chen, N. Zilberstein, and S. Segarra, "Prior-informed flow matching for graph reconstruction," *Accepted at Trans. Mach. Learn. Res. (also presented at SPIGM workshop at ICML 2026)*, 2026.

[13] N. Zilberstein, S. Segarra, E. P. Simoncelli, and F. Guth, "Learning normalized energy models for linear inverse problems," *Intl. Conf. on Machine Learning (ICML)*, 2026.

[14] N. Zilberstein, M. Mardani, E. P. Simoncelli, and S. Segarra, "Flow map denoisers: Traversing the distortion–perception plane for inverse problems," *Accepted at NeurIPS 2026 (also presented at SPIGM workshop at ICML 2026)*, 2026.

[15] A. Azizpour, N. Zilberstein, and S. Segarra, "Scalable implicit graphon learning," *Int. Conf. on Artif. Intell. and Stat.*, 2025.

[16] V. M. Tenorio, N. Zilberstein, S. Segarra, and A. G. Marques, "Graph guided diffusion: Unified guidance for conditional graph generation," *arXiv preprint arXiv:2505.19685*, 2025.

[17] F. Gama, N. Zilberstein, M. Sevilla, R. G. Baraniuk, and S. Segarra, "Unsupervised learning of sampling distributions for particle filters," *IEEE Trans. Signal Process.*, 2023.

[18] N. Zilberstein, A. Malhotra, S. Hamidi-Rad, and Y. Deenoo, "Topology preserving regularization for independent training of inter-operable models," in *UniReps: 2nd Edition of the Workshop on Unifying Representations in Neural Models*, 2024.

[19] M. Sevilla, N. Zilberstein, B. Cox, S. Pérez-Vieites, V. Elvira, and S. Segarra, "State and dynamics estimation with the kalman-langevin filter," in *Asilomar Conf. on Signals, Syst., and Comp.*, pp. 1372–1376, IEEE, 2023.

[20] F. Gama, N. Zilberstein, R. G. Baraniuk, and S. Segarra, "Unrolling particles: Unsupervised learning of sampling distributions," in *IEEE Intl. Conf. Acoust., Speech and Signal Process. (ICASSP)*, pp. 5498–5502, IEEE, 2022.

[21] N. Zilberstein, C. Dick, R. Doost-Mohammady, A. Sabharwal, and S. Segarra, "Accelerated massive MIMO detector based on annealed underdamped Langevin dynamics," in *IEEE Intl. Conf. Acoust., Speech and Signal Process. (ICASSP)*, 2023.

[22] A. Azizpour, N. Zilberstein, and S. Segarra, "Model-driven graph contrastive learning," *arXiv preprint arXiv:2506.06212*, 2025.

[23] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, "The unreasonable effectiveness of deep features as a perceptual metric," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[24] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "GANs trained by a two time-scale update rule converge to a local Nash equilibrium," *Advances in Neural Inf. Process. Syst. (NIPS)*, vol. 30, 2017.

[25] M. Bińkowski, D. J. Sutherland, M. Arbel, and A. Gretton, "Demystifying MMD GANs," in *Intl. Conf. Learn. Repr. (ICLR)*, 2018.

[26] Y. Blau and T. Michaeli, "The perception-distortion tradeoff," in *Proceedings of the IEEE/CVF Int. Conf. Comput. Vis. Pattern Recogn. (CVPR)*, 2018.

[27] J. Portilla, V. Strela, M. J. Wainwright, and E. P. Simoncelli, "Image denoising using scale mixtures of gaussians in the wavelet domain," *IEEE Transactions on Image processing*, vol. 12, no. 11, pp. 1338–1351, 2003.

[28] M. Elad and M. Aharon, "Image denoising via sparse and redundant representations over learned dictionaries," *IEEE Transactions on Image processing*, vol. 15, no. 12, pp. 3736–3745, 2006.

[29] P. Milanfar and M. Delbracio, "Denoising: a powerful building block for imaging, inverse problems and machine learning," *Philos. Trans. R. Soc. A*, vol. 383, no. 2299, 2025.

[30] R. Gao, E. Hoogeboom, J. Heek, V. De Bortoli, K. P. Murphy, and T. Salimans, "Diffusion models and gaussian flow matching: Two sides of the same coin," in *The Fourth Blogpost Track at ICLR 2025*, 2025.

[31] Y. Song and S. Ermon, "Generative modeling by estimating gradients of the data distribution," in *Advances in Neural Inf. Process. Syst. (NIPS)*, p. 11918–11930, 2019.

[32] B. D. Anderson, "Reverse-time diffusion equation models," *Stochastic Processes and their Applications*, vol. 12, no. 3, pp. 313–326, 1982.

[33] X. Liu, C. Gong, and Q. Liu, "Flow straight and fast: Learning to generate and transfer data with rectified flow," in *Intl. Conf. Learn. Repr. (ICLR)*, 2023.

[34] M. S. Albergo, N. M. Boffi, and E. Vanden-Eijnden, "Stochastic interpolants: A unifying framework for flows and diffusions," *J. Mach. Learn. Res.*, 2025.

[35] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models," in *Proceedings of the IEEE/CVF Int. Conf. Comput. Vis. Pattern Recogn. (CVPR)*, pp. 10684–10695, 2022.

[36] A. Vahdat, K. Kreis, and J. Kautz, "Score-based generative modeling in latent space," *Advances in Neural Inf. Process. Syst. (NIPS)*, vol. 34, pp. 11287–11302, 2021.

[37] D. Podell, Z. English, K. Lacey, A. Blattmann, T. Dockhorn, J. Müller, J. Penna, and R. Rombach, "Sdxl: Improving latent diffusion models for high-resolution image synthesis," in *Intl. Conf. Learn. Repr. (ICLR)*, vol. 2024, pp. 1862–1874, 2024.

[38] B. F. Labs, S. Batifol, A. Blattmann, F. Boesel, S. Consul, C. Diagne, T. Dockhorn, J. English, Z. English, P. Esser, S. Kulal, K. Lacey, Y. Levi, C. Li,

D. Lorenz, J. Müller, D. Podell, R. Rombach, H. Saini, A. Sauer, and L. Smith, "Flux.1 kontext: Flow matching for in-context image generation and editing in latent space," 2025.

[39] A. Hyvärinen and P. Dayan, "Estimation of non-normalized statistical models by score matching.," *Journal of Machine Learning Research*, vol. 6, no. 4, 2005.

[40] P. Vincent, "A connection between score matching and denoising autoencoders," *Neural Comput.*, vol. 23, no. 7, pp. 1661–1674, 2011.

[41] Y. Song, C. Durkan, I. Murray, and S. Ermon, "Maximum likelihood training of score-based diffusion models," *Advances in Neural Inf. Process. Syst. (NIPS)*, vol. 34, pp. 1415–1428, 2021.

[42] B. Efron, "Tweedie's formula and selection bias," *Journal of the American Statistical Association*, vol. 106, no. 496, pp. 1602–1614, 2011.

[43] Y. Romano, M. Elad, and P. Milanfar, "The little engine that could: Regularization by denoising (RED)," *SIAM J. Imag. Sciences*, vol. 10, no. 4, pp. 1804–1844, 2017.

[44] S. V. Venkatakrishnan, C. A. Bouman, and B. Wohlberg, "Plug-and-play priors for model based reconstruction," in *IEEE Global Conf. Signal and Info. Process. (GlobalSIP)*, pp. 945–948, IEEE, 2013.

[45] J. Song, C. Meng, and S. Ermon, "Denoising diffusion implicit models," in *Intl. Conf. Learn. Repr. (ICLR)*, 2020.

[46] C. Lu, Y. Zhou, F. Bao, J. Chen, C. Li, and J. Zhu, "Dpm-solver: A fast ode solver for diffusion probabilistic model sampling in around 10 steps," in

Advances in Neural Inf. Process. Syst. (NIPS), 2022.

[47] T. Karras, M. Aittala, T. Aila, and S. Laine, "Elucidating the design space of diffusion-based generative models," *Advances in Neural Inf. Process. Syst. (NIPS)*, vol. 35, pp. 26565–26577, 2022.

[48] Y. Du, C. Durkan, R. Strudel, J. B. Tenenbaum, S. Dieleman, R. Fergus, J. Sohl-Dickstein, A. Doucet, and W. S. Grathwohl, "Reduce, reuse, recycle: Compositional generation with energy-based diffusion models and MCMC," in *Intl. Conf. on Mach. Learn. (ICML)*, pp. 8489–8510, PMLR, 2023.

[49] A. Bradley and P. Nakkiran, "Classifier-free guidance is a predictor-corrector," *Trans. Mach. Learn. Res (TMLR).*, 2025.

[50] M. Mardani, J. Song, J. Kautz, and A. Vahdat, "A variational perspective on solving inverse problems with diffusion models," *Intl. Conf. Learn. Repr. (ICLR)*, 2024.

[51] B. Poole, A. Jain, J. T. Barron, and B. Mildenhall, "Dreamfusion: Text-to-3D using 2D diffusion," in *Intl. Conf. Learn. Repr. (ICLR)*, 2022.

[52] B. Kawar, M. Elad, S. Ermon, and J. Song, "Denoising diffusion restoration models," in *Advances in Neural Inf. Process. Syst. (NIPS)*, 2022.

[53] J. Song, A. Vahdat, M. Mardani, and J. Kautz, "Pseudoinverse-guided diffusion models for inverse problems," in *Intl. Conf. Learn. Repr.*, 2023.

[54] L. Wu, B. Trippe, C. Naesseth, D. Blei, and J. P. Cunningham, "Practical and asymptotically exact conditional sampling in diffusion models," *Advances in Neural Inf. Process. Syst. (NIPS)*, 2023.

[55] Z. Wu, Y. Sun, Y. Chen, B. Zhang, Y. Yue, and K. L. Bouman, "Principled probabilistic imaging using diffusion models as plug-and-play priors," *Advances in Neural Inf. Process. Syst. (NIPS)*, vol. 37, pp. 118389–118427, 2024.

[56] N. M. Boffi, M. S. Albergo, and E. Vanden-Eijnden, "How to build a consistency model: Learning flow maps via self-distillation," in *Advances in Neural Inf. Process. Syst. (NIPS)*, 2025.

[57] N. M. Boffi, M. S. Albergo, and E. Vanden-Eijnden, "Flow map matching with stochastic interpolants: A mathematical framework for consistency models," *Trans. Mach. Learn. Res (TMLR).*, 2025.

[58] Z. Geng, M. Deng, X. Bai, J. Z. Kolter, and K. He, "Mean flows for one-step generative modeling," in *Advances in Neural Inf. Process. Syst. (NIPS)*, 2025.

[59] J. G. Proakis, *Digital Communications 5th Edition*. McGraw Hill, 2007.

[60] G. Ongie, A. Jalal, C. A. Metzler, R. G. Baraniuk, A. G. Dimakis, and R. Willett, "Deep learning techniques for inverse problems in imaging," *IEEE J.Sel. Areas Inf. Theory.*, vol. 1, no. 1, pp. 39–56, 2020.

[61] S. Segarra, A. G. Marques, G. Mateos, and A. Ribeiro, "Network topology inference from spectral templates," *IEEE Trans. Signal and Info. Process. over Networks*, vol. 3, no. 3, pp. 467–483, 2017.

[62] X. Dong, D. Thanou, P. Frossard, and P. Vandergheynst, "Learning laplacian matrix in smooth graph signal representations," *IEEE Trans. Signal Process.*, vol. 64, no. 23, pp. 6160–6173, 2016.

[63] B. Kawar, G. Vaksman, and M. Elad, "SNIPS: Solving noisy inverse problems stochastically," in *Advances in Neural Inf. Process. Syst. (NIPS)*, vol. 34, pp. 21757–21769, 2021.

[64] M. Arvinte and J. I. Tamir, "MIMO channel estimation using score-based generative models," *IEEE Trans. Wireless Commun.*, vol. 22, no. 6, 2023 (Online Nov 2022).

[65] N. Zilberstein, C. Dick, R. Doost-Mohammady, A. Sabharwal, and S. Segarra, "Annealed Langevin dynamics for massive MIMO detection," *IEEE Trans. Wireless Commun.*, vol. 22, no. 6, 2023 (Online Nov 2022).

[66] X. Cheng, N. S. Chatterji, P. L. Bartlett, and M. I. Jordan, "Underdamped Langevin MCMC: A non-asymptotic analysis," in *Proc. Conf. Learning Theory (COLT)*, pp. 300–323, 2018.

[67] W. Mou, Y.-A. Ma, M. J. Wainwright, P. L. Bartlett, and M. I. Jordan, "High-order Langevin diffusion yields an accelerated MCMC algorithm," *J. Mach. Learn. Res.*, vol. 22, no. 42, pp. 1–41, 2021.

[68] T. Dockhorn, A. Vahdat, and K. Kreis, "Score-based generative modeling with critically-damped Langevin diffusion," in *Intl. Conf. Learn. Repr.*, 2022.

[69] M. Girolami and B. Calderhead, "Riemann manifold langevin and hamiltonian monte carlo methods," *J. R. Stat. Soc. Series B Stat. Methodol.*, vol. 73, no. 2, pp. 123–214, 2011.

[70] T. Fu, L. Luo, and Z. Zhang, "Quasi-Newton Hamiltonian Monte Carlo," in *Conf. Uncertain. Artf. Intel.*, 2016.

[71] U. Şimşekli, C. Yıldız, T. H. Nguyen, G. Richard, and A. Cemgil, "Asynchronous stochastic Quasi-Newton MCMC for non-convex optimization," in *Intl. Conf. on Machine Learning (ICML)*, 2018.

[72] Y.-A. Ma, T. Chen, and E. Fox, "A complete recipe for stochastic gradient MCMC," *Advances in Neural Inf. Process. Syst. (NIPS)*, vol. 28, 2015.

[73] N. Zilberstein, C. Dick, R. Doost-Mohammady, A. Sabharwal, and S. Segarra, "Detection by sampling: Massive MIMO detector based on Langevin dynamics," in *European Signal Process. Conf. (EUSIPCO)*, pp. 1651–1655, 2022.

[74] N. Zilberstein, A. Swami, and S. Segarra, "Joint channel estimation and data detection in massive mimo systems based on diffusion models," in *IEEE Intl. Conf. Acoust., Speech and Signal Process. (ICASSP)*, pp. 13291–13295, IEEE, 2024.

[75] G. A. Pavliotis, *Stochastic Processes and Applications: Diffusion Processes, the Fokker-Planck and Langevin Equations*. Springer, 2014.

[76] A. S. Dalalyan and L. Riou-Durand, "On sampling from a log-concave density using kinetic Langevin diffusions," *Bernoulli*, 2020.

[77] P. Monmarché, "High-dimensional MCMC with a standard splitting scheme for the underdamped Langevin diffusion," *Electron. J. Stat.*, 2021.

[78] M. Ceriotti, *A novel framework for enhanced molecular dynamics based on the generalized Langevin equation*. PhD thesis, ETH Zurich, 2010.

[79] B. Leimkuhler and C. Matthews, *Molecular Dynamics: With Deterministic and Stochastic Numerical Methods*. Springer, May 2015.

[80] R. Kubo, "The fluctuation-dissipation theorem," *Reports on progress in physics*, vol. 29, no. 1, p. 255, 1966.

[81] M. Ottobre and G. A. Pavliotis, "Asymptotic analysis for the generalized Langevin equation," *Nonlinearity*, vol. 24, no. 5, p. 1629, 2011.

[82] M. Chak, N. Kantas, and G. A. Pavliotis, "On the generalized langevin equation for simulated annealing," *SIAM/ASA Journal on Uncertainty Quantification*, vol. 11, no. 1, pp. 139–167, 2023.

[83] B. Leimkuhler, C. Matthews, and J. Weare, "Ensemble preconditioning for Markov chain Monte Carlo simulation," *Stat. Comput.*, vol. 28, pp. 277–290, 2018.

[84] A. Jain and B. Poole, "Journey to the BAOAB-limit: finding effective MCMC samplers for score-based models," in *Advances in Neural Inf. Process. Syst. (NIPS)*, 2022.

[85] A. D. Baczewski and S. D. Bond, "Numerical integration of the extended variable generalized Langevin equation with a positive Prony representable memory kernel," *J. Chem. Phys.*, vol. 139, no. 4, p. 044107, 2013.

[86] A. Jalal, M. Arvinte, G. Daras, E. Price, A. G. Dimakis, and J. Tamir, "Robust compressed sensing mri with deep generative priors," *Advances in Neural Inf. Process. Syst. (NIPS)*, vol. 34, pp. 14938–14954, 2021.

[87] W. H. Chin, A. G. Constantinides, and D. B. Ward, "Parallel multistage detection for multiple antenna wireless systems," *Electron. Lett.*, vol. 38, pp. 597–599, 2002.

[88] K. Pratik, B. D. Rao, and M. Welling, "RE-MIMO: Recurrent and permutation equivariant neural MIMO detection," *IEEE Trans. Signal Process.*, vol. 69, pp. 459–473, 2021.

[89] H. He, C.-K. Wen, S. Jin, and G. Y. Li, "A model-driven deep learning network for MIMO detection," in *IEEE Global Conf. Signal and Info. Process. (GlobalSIP)*, pp. 584–588, 2018.

[90] G. O. LLC., "Gurobi optimizer reference manual," 2021.

[91] S. L. Loyka, "Channel capacity of MIMO architecture using the exponential correlation matrix," *IEEE Commun. Lett.*, vol. 5, no. 9, pp. 369–371, 2001.

[92] K. Venugopal, A. Alkhateeb, N. González Prelcic, and R. W. Heath, "Channel estimation for hybrid architecture-based wideband millimeter wave systems," *IEEE J. Sel. Areas Commun.*, vol. 35, no. 9, pp. 1996–2009, 2017.

[93] B. N. Bhaskar, G. Tang, and B. Recht, "Atomic norm denoising with applications to line spectral estimation," *IEEE Trans. Signal Process.*, vol. 61, no. 23, pp. 5987–5999, 2013.

[94] B. Mamandipoor, D. Ramasamy, and U. Madhow, "Newtonized orthogonal matching pursuit: Frequency estimation over the continuum," *IEEE Trans. Signal Process.*, vol. 64, no. 19, pp. 5066–5081, 2016.

[95] C. Metzler, A. Mousavi, and R. Baraniuk, "Learned D-AMP: Principled neural network based compressive image recovery," *Advances in Neural Inf. Process. Syst. (NIPS)*, vol. 30, 2017.

[96] H. He, C.-K. Wen, S. Jin, and G. Y. Li, "Deep learning-based channel estimation for beamspace mmwave massive MIMO systems," *IEEE Wirel. Commun.*, vol. 7, no. 5, pp. 852–855, 2018.

[97] V. Monga, Y. Li, and Y. C. Eldar, "Algorithm unrolling: Interpretable, efficient deep learning for signal and image processing," *IEEE Signal Process. Mag.*, vol. 38, no. 2, pp. 18–44, 2021.

[98] 3GPP, "Study on channel model for frequencies from 0.5 to 100 GHz," Tech. Rep. 38.901, 2020.

[99] Y. Song and S. Ermon, "Improved techniques for training score-based generative models," *Advances in Neural Inf. Process. Syst. (NIPS)*, vol. 33, pp. 12438–12448, 2020.

[100] T. Karras, S. Laine, and T. Aila, "A style-based generator architecture for generative adversarial networks," in *Proceedings of the IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, pp. 4401–4410, 2019.

[101] J. Choi, S. Kim, Y. Jeong, Y. Gwon, and S. Yoon, "Ilvr: Conditioning method for denoising diffusion probabilistic models. in 2021 ieee," in *Proceedings of the IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, pp. 14347–14356, 2021.

[102] A. Chowdhury, G. Verma, C. Rao, A. Swami, and S. Segarra, "Unfolding WMMSE using graph neural networks for efficient power allocation," *IEEE Trans. Wireless Commun.*, vol. 20, no. 9, pp. 6004–6017, 2021.

[103] B. Li, G. Verma, and S. Segarra, "Graph-based algorithm unfolding for energy-aware power allocation in wireless networks," *IEEE Trans. Wireless Commun.*, vol. 22, no. 2, pp. 1359–1373, 2023.

[104] H. Chung, J. Kim, S. Kim, and J. C. Ye, "Parallel diffusion models of operator and image for blind inverse problems," in *Proceedings of the IEEE/CVF Int. Conf. Comput. Vis. Pattern Recogn. (CVPR)*, pp. 6059–6069, IEEE, 2023.

[105] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models. 2022 ieee," in *Proceedings of the IEEE/CVF Int. Conf. Comput. Vis. Pattern Recogn. (CVPR)*, pp. 10674–10685, 2021.

[106] Z. Kadkhodaie and E. Simoncelli, "Stochastic solutions for linear inverse problems using the prior implicit in a denoiser," *Advances in Neural Inf. Process. Syst. (NIPS)*, vol. 34, pp. 13242–13254, 2021.

[107] Z. Wang, C. Lu, Y. Wang, F. Bao, C. Li, H. Su, and J. Zhu, "Prolificdreamer: High-fidelity and diverse text-to-3D generation with variational score distillation," *Advances in Neural Inf. Process. Syst. (NIPS)*, vol. 36, 2024.

[108] S. Kim, K. Lee, J. S. Choi, J. Jeong, K. Sohn, and J. Shin, "Collaborative score distillation for consistent visual editing," in *Advances in Neural Inf. Process. Syst. (NIPS)*, vol. 36, pp. 73232–73257, 2023.

[109] Q. Liu and D. Wang, "Stein variational gradient descent: A general purpose Bayesian inference algorithm," *Advances in Neural Inf. Process. Syst. (NIPS)*, vol. 29, 2016.

[110] F. D'Angelo and V. Fortuin, "Annealed stein variational gradient descent," in *Third Symposium on Advances in Approximate Bayesian Inference*, 2021.

[111] J. Ba, M. A. Erdogdu, M. Ghassemi, S. Sun, T. Suzuki, D. Wu, and T. Zhang,

"Understanding the variance collapse of SVGD in high dimensions," in *Intl. Conf. Learn. Repr. (ICLR)*, 2021.

[112] L. Rout, N. Raoof, G. Daras, C. Caramanis, A. Dimakis, and S. Shakkottai, "Solving linear inverse problems provably via posterior sampling with latent diffusion models," *Advances in Neural Inf. Process. Syst. (NIPS)*, vol. 36, 2024.

[113] H. Chung, J. C. Ye, P. Milanfar, and M. Delbracio, "Prompt-tuning latent diffusion models for inverse problems," *Intl. Conf. on Machine Learning (ICML)*, 2024.

[114] B. Song, S. M. Kwon, Z. Zhang, X. Hu, Q. Qu, and L. Shen, "Solving inverse problems with latent diffusion models via hard data consistency," *Intl. Conf. Learn. Repr. (ICLR)*, 2024.

[115] W. Xia, Y. Zhang, Y. Yang, J.-H. Xue, B. Zhou, and M.-H. Yang, "Gan inversion: A survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 3, pp. 3121–3138, 2022.

[116] G. Daras, J. Dean, A. Jalal, and A. Dimakis, "Intermediate layer optimization for inverse problems using deep generative models," in *Intl. Conf. on Machine Learning (ICML)*, pp. 2421–2432, PMLR, 2021.

[117] F. D'Angelo and V. Fortuin, "Repulsive deep ensembles are Bayesian," *Advances in Neural Inf. Process. Syst. (NIPS)*, vol. 34, pp. 3451–3465, 2021.

[118] M. Caron, H. Touvron, I. Misra, H. Jégou, J. Mairal, P. Bojanowski, and A. Joulin, "Emerging properties in self-supervised vision transformers," in *Proceedings of the IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, pp. 9650–9660, 2021.

[119] D. Geman and C. Yang, "Nonlinear image recovery with half-quadratic regularization," *IEEE transactions on Image Processing*, vol. 4, no. 7, pp. 932–946, 1995.

[120] G. Daras, H. Chung, C.-H. Lai, Y. Mitsufuji, J. C. Ye, P. Milanfar, A. G. Dimakis, and M. Delbracio, "A survey on diffusion models for inverse problems," *arXiv preprint arXiv:2410.00083*, 2024.

[121] Z. Kong, W. Ping, J. Huang, K. Zhao, and B. Catanzaro, "DiffWave: A versatile diffusion model for audio synthesis," in *Intl. Conf. Learn. Repr. (ICLR)*, 2020.

[122] Y. Zhu, K. Zhang, J. Liang, J. Cao, B. Wen, R. Timofte, and L. Van Gool, "Denoising diffusion models for plug-and-play image restoration," in *Proceedings of the IEEE/CVF Int. Conf. Comput. Vis. Pattern Recogn. (CVPR)*, pp. 1219–1229, 2023.

[123] K. Zhang, Y. Li, W. Zuo, L. Zhang, L. Van Gool, and R. Timofte, "Plug-and-play image restoration with deep denoiser prior," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 10, pp. 6360–6376, 2021.

[124] E. J. Hu, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, W. Chen, *et al.*, "LoRA: Low-rank adaptation of large language models," in *Intl. Conf. Learn. Repr. (ICLR)*, 2021.

[125] M. Armandpour, H. Zheng, A. Sadeghian, A. Sadeghian, and M. Zhou, "Reimagine the negative prompt algorithm: Transform 2D diffusion into 3D, alleviate janus problem and beyond," *arXiv preprint arXiv:2304.04968*, 2023.

[126] P. Wang, D. Xu, Z. Fan, D. Wang, S. Mohan, F. Iandola, R. Ranjan, Y. Li, Q. Liu, Z. Wang, *et al.*, "Taming mode collapse in score distillation for text-

to-3d generation," in *Proceedings of the IEEE/CVF Int. Conf. Comput. Vis. Pattern Recogn. (CVPR)*, pp. 9037–9047, IEEE, 2024.

[127] J. Song, Q. Zhang, H. Yin, M. Mardani, M.-Y. Liu, J. Kautz, Y. Chen, and A. Vahdat, "Loss-guided diffusion models for plug-and-play controllable generation," in *Intl. Conf. on Machine Learning (ICML)*, pp. 32483–32498, PMLR, 2023.

[128] D. A. Van Dyk and X.-L. Meng, "The art of data augmentation," *J. Comp. Graph. Stat.*, vol. 10, no. 1, pp. 1–50, 2001.

[129] M. Vono, N. Dobigeon, and P. Chainais, "Asymptotically exact data augmentation: Models, properties, and algorithms," *J. Comp. Graph. Stat.*, vol. 30, no. 2, pp. 335–348, 2020.

[130] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Intl. Conf. Learn. Repr. (ICLR)*, 2015.

[131] F. Vargas, W. S. Grathwohl, and A. Doucet, "Denoising diffusion samplers," in *Intl. Conf. Learn. Repr.*, 2023.

[132] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *Int. J. Robot. Res.*, vol. 5, no. 1, pp. 90–98, 1986.

[133] S. Paternain, M. Calvo-Fullana, L. F. Chamon, and A. Ribeiro, "Safe policies for reinforcement learning via primal-dual methods," *IEEE Trans. Autom. Control*, vol. 68, no. 3, pp. 1321–1336, 2022.

[134] M. Janner, Y. Du, J. Tenenbaum, and S. Levine, "Planning with diffusion for flexible behavior synthesis," in *Intl. Conf. Mach. Learn.*, pp. 9902–9915, PMLR,

2022.

[135] A. Schneuing, C. Harris, Y. Du, K. Didi, A. Jamasb, I. Igashov, W. Du, C. Gomes, T. L. Blundell, P. Lio, *et al.*, "Structure-based drug design with equivariant diffusion models," *Nature Computational Science*, vol. 4, no. 12, pp. 899–909, 2024.

[136] M. Kim, D. Kim, A. Yusuf, S. Ermon, and M. Park, "Training-free safe denoisers for safe use of diffusion models," *Advances in Neural Inf. Process. Syst. (NeurIPS)*, 2025.

[137] G. O. Roberts and R. L. Tweedie, "Exponential convergence of Langevin distributions and their discrete approximations," *Bernoulli*, vol. 2, pp. 341–363, 1996.

[138] A. S. Dalalyan and A. Karagulyan, "User-friendly guarantees for the langevin monte carlo with inaccurate gradient," *Stoch. Process. Their Appl.*, vol. 129, no. 12, pp. 5278–5311, 2019.

[139] C. P. Robert and G. Casella, *Monte Carlo statistical methods*, vol. 2. Springer, 1999.

[140] S. Bubeck, R. Eldan, and J. Lehec, "Sampling from a log-concave distribution with projected langevin monte carlo," *Discr. & Comp. Geo.*, vol. 59, no. 4, pp. 757–783, 2018.

[141] Y.-P. Hsieh, A. Kavis, P. Rolland, and V. Cevher, "Mirrored langevin dynamics," *Advances in Neural Inf. Process. Syst. (NeurIPS)*, vol. 31, 2018.

[142] K. S. Zhang, G. Peyré, J. Fadili, and M. Pereyra, "Wasserstein control of mirror langevin monte carlo," in *Conf. on lear. theo.*, pp. 3814–3841, PMLR, 2020.

[143] A. Salim and P. Richtarik, "Primal dual interpretation of the proximal stochastic gradient langevin algorithm," *Advances in Neural Inf. Process. Syst. (NeurIPS)*, vol. 33, pp. 3786–3796, 2020.

[144] Y. Kook, Y.-T. Lee, R. Shen, and S. Vempala, "Sampling with Riemannian Hamiltonian Monte Carlo in a constrained space," *Advances in Neural Inf. Process. Syst. (NeurIPS)*, vol. 35, pp. 31684–31696, 2022.

[145] M. Noble, V. De Bortoli, and A. Durmus, "Unbiased constrained sampling with self-concordant barrier hamiltonian monte carlo," *Advances in Neural Inf. Process. Syst. (NeurIPS)*, vol. 36, pp. 32672–32719, 2023.

[146] M. Gurbuzbalaban, Y. Hu, and L. Zhu, "Penalized overdamped and underdamped Langevin Monte Carlo algorithms for constrained sampling," *J. Mach. Learn. Res.*, vol. 25, no. 263, pp. 1–67, 2024.

[147] L. F. Chamon, M. R. Karimi, and A. Korba, "Constrained sampling with primal-dual langevin monte carlo," *Advances in Neural Inf. Process. Syst. (NeurIPS)*, vol. 37, pp. 29285–29323, 2024.

[148] S. Paternain, D. E. Koditschek, and A. Ribeiro, "Navigation functions for convex potentials in a space with convex obstacles," *IEEE Trans. Autom. Control*, vol. 63, no. 9, pp. 2944–2959, 2017.

[149] D. E. Koditschek and E. Rimon, "Robot navigation functions on manifolds with boundary," *Adv. Appl. Math.*, vol. 11, no. 4, pp. 412–442, 1990.

[150] C. Niu, Y. Song, J. Song, S. Zhao, A. Grover, and S. Ermon, "Permutation invariant graph generation via score-based generative modeling," in *Int. Conf. on Artif. Intell. and Stat.*, pp. 4474–4484, PMLR, 2020.

[151] J. Jo, S. Lee, and S. J. Hwang, "Score-based generative modeling of graphs via the system of stochastic differential equations," in *Intl. Conf. on Machine Learning (ICML)*, pp. 10362–10383, PMLR, 2022.

[152] C. Vignac, I. Krawczuk, A. Siraudin, B. Wang, V. Cevher, and P. Frossard, "Digress: Discrete denoising diffusion for graph generation," in *Intl. Conf. Learn. Repr. (ICLR)*, 2023.

[153] Y. Qin, M. Madeira, D. Thanou, and P. Frossard, "Defog: Discrete flow matching for graph generation," *Intl. Conf. on Machine Learning (ICML)*, 2025.

[154] F. Eijkelboom, G. Bartosh, C. Andersson Naesseth, M. Welling, and J.-W. van de Meent, "Variational flow matching for graph generation," *Advances in Neural Inf. Process. Syst. (NIPS)*, vol. 37, pp. 11735–11764, 2024.

[155] N. Yang, H. Wu, K. Zeng, Y. Li, S. Bao, and J. Yan, "Molecule generation for drug design: a graph learning perspective," *Fundam. Res.*, 2024.

[156] A. Grover, A. Zweig, and S. Ermon, "Graphite: Iterative generative modeling of graphs," in *Intl. Conf. on Machine Learning (ICML)*, pp. 2434–2444, PMLR, 2019.

[157] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining.*, p. 855–864, 2016.

[158] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Advances in Neural Inf. Process. Syst. (NIPS)*, NIPS'17, p. 1025–1035, 2017.

[159] T. N. Kipf and M. Welling, "Variational graph auto-encoders," *arXiv preprint arXiv:1611.07308*, 2016.

[160] X. Wang, H. Yang, and M. Zhang, "Neural common neighbor with completion for link prediction," in *Intl. Conf. Learn. Repr. (ICLR)*, 2024.

[161] H. Shomer, Y. Ma, H. Mao, J. Li, B. Wu, and J. Tang, "Lpformer: An adaptive graph transformer for link prediction," in *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2024.

[162] P. Trivedi, R. A. Rossi, D. Arbour, T. Yu, F. Dernoncourt, S. Kim, N. Lipka, N. Park, N. K. Ahmed, and D. Koutra, "Editing partially observable networks via graph diffusion models," in *Intl. Conf. on Machine Learning (ICML)*, 2024.

[163] K. Sharma, S. Kumar, and R. Trivedi, "Diffuse, sample, project: plug-and-play controllable graph generation," in *Intl. Conf. on Machine Learning (ICML)*, 2024.

[164] G. Ohayon, T. Michaeli, and M. Elad, "Posterior-mean rectified flow: Towards minimum MSE photo-realistic image restoration," in *Intl. Conf. Learn. Repr. (ICLR)*, 2025.

[165] J. Austin, D. D. Johnson, J. Ho, D. Tarlow, and R. Van Den Berg, "Structured denoising diffusion models in discrete state-spaces," *Advances in Neural Inf. Process. Syst. (NIPS)*, vol. 34, pp. 17981–17993, 2021.

[166] M. Delbracio and P. Milanfar, "Inversion by direct iteration: An alternative to denoising diffusion for image restoration," *Trans. Mach. Learn. Res (TMLR).*, 2023.

[167] A. Wijesinghe, S. Kandanaarachchi, D. M. Steinberg, and C. S. Ong, "Flowette: Flow matching with graphette priors for graph generation," *arXiv preprint arXiv:2602.23566*, 2026.

[168] A. Lugmayr, M. Danelljan, A. Romero, F. Yu, R. Timofte, and L. V. Gool, "Repaint: Inpainting using denoising diffusion probabilistic models," in *Proceedings of the IEEE/CVF Int. Conf. Comput. Vis. Pattern Recogn. (CVPR)*, 2022.

[169] M. E. J. Newman, "Clustering and preferential attachment in growing networks," *Phys. Rev. E*, vol. 64, no. 2, p. 025102, 2001.

[170] L. A. Adamic and E. Adar, "Friends and neighbors on the web," *Soc. Networks*, vol. 25, no. 3, pp. 211–230, 2003.

[171] T. Zhou, L. Lü, and Y.-C. Zhang, "Predicting missing links via local information," *The European Physical Journal B*, vol. 71, no. 4, pp. 623–630, 2009.

[172] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: online learning of social representations," in *ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining.*, p. 701–710, 2014.

[173] Z. Zhu, Z. Zhang, L.-P. Xhonneux, and J. Tang, "Neural bellman-ford networks: a general graph neural network framework for link prediction," in *Advances in Neural Inf. Process. Syst. (NIPS)*, (Red Hook, NY, USA), Curran Associates Inc., 2021.

[174] A. Tong, K. Fatras, N. Malkin, G. Huguet, Y. Zhang, J. Rector-Brooks, G. Wolf, and Y. Bengio, "Improving and generalizing flow-based generative models with minibatch optimal transport," *Trans. Mach. Learn. Res.*, 2024.

[175] M. S. Albergo, M. Goldstein, N. M. Boffi, R. Ranganath, and E. Vanden-Eijnden, "Stochastic interpolants with data-dependent couplings," in *Intl. Conf. on Mach. Learn. (ICML)*, pp. 921–937, PMLR, 2024.

[176] S. Dieleman, L. Sartran, A. Roshannai, N. Savinov, Y. Ganin, P. H. Richemond, A. Doucet, R. Strudel, C. Dyer, C. Durkan, *et al.*, "Continuous diffusion for categorical data," *arXiv preprint arXiv:2211.15089*, 2022.

[177] C. Lee, J. Yoo, M. Agarwal, S. Shah, J. Huang, A. Raghunathan, S. Hong, N. M. Boffi, and J. Kim, "Flow map language models: One-step language modeling via continuous denoising," *arXiv preprint arXiv:2602.16813*, 2026.

[178] D. Roos, O. Davis, F. Eijkelboom, M. Bronstein, M. Welling, I. I. Ceylan, L. Ambrogioni, and J.-W. van de Meent, "Categorical flow maps," *Intl. Conf. on Machine Learning (ICML)*, 2026.

[179] C. Morris, N. M. Kriege, F. Bause, K. Kersting, P. Mutzel, and M. Neumann, "Tudataset: A collection of benchmark datasets for learning with graphs," in *ICML Work. on Graph Rep. Learning*, 2020.

[180] A. K. McCallum, K. Nigam, J. Rennie, and K. Seymore, "Automating the construction of internet portals with machine learning," *Information Retrieval*, vol. 3, no. 2, pp. 127–163, 2000.

[181] Z. Yang, W. Cohen, and R. Salakhudinov, "Revisiting semi-supervised learning

with graph embeddings," in *Intl. Conf. on Machine Learning (ICML)*, pp. 40–48, PMLR, 2016.

[182] L. O'Bray, M. Horn, B. Rieck, and K. Borgwardt, "Evaluation metrics for graph generative models: Problems, pitfalls, and practical solutions," *Intl. Conf. Learn. Repr. (ICLR)*, 2022.

[183] J. You, R. Ying, X. Ren, W. L. Hamilton, and J. Leskovec, "Graphrnn: Generating realistic graphs with deep auto-regressive models," in *Intl. Conf. on Machine Learning (ICML)*, pp. 5708–5717, 2018.

[184] G.-H. Liu, A. Vahdat, D.-A. Huang, E. Theodorou, W. Nie, and A. Anandkumar, "I^2SB: Image-to-image Schrödinger bridge," in *Intl. Conf. on Mach. Learn. (ICML)*, pp. 22042–22062, PMLR, 2023.

[185] H. Negrel, F. Coeurdoux, M. S. Albergo, and E. Vanden-Eijnden, "Multitask learning with stochastic interpolants," in *Advances in Neural Inf. Process. Syst. (NIPS)*, 2025.

[186] N. Elata, H. Chung, J. C. Ye, T. Michaeli, and M. Elad, "InvFussion: Bridging supervised and zero-shot diffusion for inverse problems," *Advances in Neural Inf. Process. Syst. (NIPS)*, 2025.

[187] J. Bruna and J. Han, "Provable posterior sampling with denoising oracles via tilted transport," *Advances in Neural Inf. Process. Syst. (NIPS)*, vol. 37, pp. 82863–82894, 2024.

[188] J. Thornton, L. Béthune, R. Zhang, A. Bradley, P. Nakkiran, and S. Zhai, "Composition and control with distilled energy diffusion models and sequential monte carlo," in *Int. Conf. on Artif. Intell. and Stat.*, 2025.

[189] F. Guth, Z. Kadkhodaie, and E. P. Simoncelli, "Learning normalized image densities via dual score matching," *Advances in Neural Inf. Process. Syst. (NIPS)*, 2025.

[190] H. Yu, A. Klami, A. Hyvarinen, A. Korba, and O. Chehab, "Density ratio estimation with conditional probability paths," in *Intl. Conf. on Mach. Learn. (ICML)*, 2025.

[191] M. Plainer, H. Wu, L. Klein, S. Günnemann, and F. Noé, "Consistent sampling and simulation: Molecular dynamics with energy-based diffusion models," *Advances in Neural Inf. Process. Syst. (NIPS)*, 2025.

[192] G. Daras, M. Delbracio, H. Talebi, A. Dimakis, and P. Milanfar, "Soft diffusion: Score matching with general corruptions," *Trans. Mach. Learn. Res (TMLR).*, 2023.

[193] E. Hoogeboom and T. Salimans, "Blurring diffusion models," in *Intl. Conf. Learn. Repr. (ICLR)*, 2023.

[194] A. Bansal, E. Borgnia, H.-M. Chu, J. Li, H. Kazemi, F. Huang, M. Goldblum, J. Geiping, and T. Goldstein, "Cold diffusion: Inverting arbitrary image transforms without noise," *Advances in Neural Inf. Process. Syst. (NIPS)*, vol. 36, pp. 41259–41282, 2023.

[195] M. Gerdes, M. Welling, and M. C. Cheng, "GUD: Generation with unified diffusion," *arXiv preprint arXiv:2410.02667*, 2024.

[196] B. Chen, D. Martí Monsó, Y. Du, M. Simchowitz, R. Tedrake, and V. Sitzmann, "Diffusion forcing: Next-token prediction meets full-sequence diffusion," *Advances in Neural Inf. Process. Syst. (NIPS)*, vol. 37, pp. 24081–24125, 2024.

[197] T. Asthana, Y. Bao, A. Awad, and H. Krim, "Accelerated image-aware diffusion modeling," in *European Signal Process. Conf. (EUSIPCO)*, pp. 1862–1866, IEEE, 2025.

[198] G. Bartosh, D. Vetrov, and C. A. Naesseth, "Neural flow diffusion models: Learnable forward process for improved diffusion modelling," *Advances in Neural Inf. Process. Syst. (NIPS)*, vol. 37, pp. 73952–73985, 2024.

[199] S. Sahoo, A. Gokaslan, C. M. De Sa, and V. Kuleshov, "Diffusion models with learned adaptive noise," *Advances in Neural Inf. Process. Syst. (NIPS)*, vol. 37, pp. 105730–105779, 2024.

[200] R. Singhal, M. Goldstein, and R. Ranganath, "Where to diffuse, how to diffuse, and how to get back: Automated learning for multivariate diffusions," in *Intl. Conf. Learn. Repr. (ICLR)*, 2023.

[201] B. T. Feng, J. Smith, M. Rubinstein, H. Chang, K. L. Bouman, and W. T. Freeman, "Score-based diffusion models as principled priors for inverse imaging," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 10520–10531, 2023.

[202] Y. Hu, A. Peng, W. Gan, P. Milanfar, M. Delbracio, and U. S. Kamilov, "Stochastic deep restoration priors for imaging inverse problems," in *Intl. Conf. on Mach. Learn. (ICML)*, 2025.

[203] M. Terris, S. Hurault, M. Song, and J. Tachella, "Reconstruct anything model: a lightweight foundation model for computational imaging," *Intl. Conf. Learn. Repr. (ICLR)*, 2026.

[204] H. E. Robbins, "An empirical Bayes approach to statistics," in *Proc. Third Berkeley Symp. Math. Statist. Prob.*, pp. 388–394, Springer, 1956.

[205] K. Miyasawa *et al.*, "An empirical Bayes estimator of the mean of a normal population," *Bull. Inst. Internat. Statist*, vol. 38, no. 181-188, pp. 1–2, 1961.

[206] M. Raphan and E. P. Simoncelli, "Least squares estimation without priors or supervision," *Neural Comp.*, vol. 23, no. 2, pp. 374–420, 2011.

[207] T. Karras, M. Aittala, J. Lehtinen, J. Hellsten, T. Aila, and S. Laine, "Analyzing and improving the training dynamics of diffusion models," in *Proceedings of the IEEE/CVF Int. Conf. Comput. Vis. Pattern Recogn. (CVPR)*, pp. 24174–24184, 2024.

[208] S. Elfwing, E. Uchibe, and K. Doya, "Sigmoid-weighted linear units for neural network function approximation in reinforcement learning," *Neural Netw.*, vol. 107, pp. 3–11, 2018.

[209] B. Zhang, W. Chu, J. Berner, C. Meng, A. Anandkumar, and Y. Song, "Improving diffusion inverse problem solving with decoupled noise annealing," in *Proceedings of the IEEE/CVF Int. Conf. Comput. Vis. Pattern Recogn. (CVPR)*, pp. 20895–20905, 2025.

[210] Z. Liu, P. Luo, X. Wang, and X. Tang, "Deep learning face attributes in the wild," in *Proceedings of the IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, December 2015.

[211] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, *et al.*, "Imagenet large scale visual recognition challenge," *Int. J. Comp. Vision*, vol. 115, pp. 211–252, 2015.

[212] Y. Choi, Y. Uh, J. Yoo, and J.-W. Ha, "StarGAN v2: Diverse image synthesis for multiple domains," in *Proceedings of the IEEE/CVF Int. Conf. Comput. Vis. Pattern Recogn. (CVPR)*, 2020.

[213] Y. LeCun, C. Cortes, and C. Burges, "MNIST handwritten digit database," *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, vol. 2, 2010.

[214] K. Ding, K. Ma, S. Wang, and E. P. Simoncelli, "Image quality assessment: Unifying structure and texture similarity," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 5, pp. 2567–2581, 2020.

[215] Y. Song, S. Garg, J. Shi, and S. Ermon, "Sliced score matching: A scalable approach to density and score estimation," in *Uncertainty in artificial intelligence*, pp. 574–584, PMLR, 2020.

[216] L.-Q. Zhang, Z. Kadkhodaie, E. P. Simoncelli, and D. H. Brainard, "Generalized compressed sensing for image reconstruction with diffusion probabilistic models," *Trans. Mach. Learn. Res (TMLR).*, 2025.

[217] C. Dong, C. C. Loy, K. He, and X. Tang, "Image super-resolution using deep convolutional networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 2, pp. 295–307, 2015.

[218] A. Pokle, M. J. Muckley, R. T. Q. Chen, and B. Karrer, "Training-free linear image inverses via flows," *Trans. Mach. Learn. Res (TMLR).*, 2024.

[219] A. Sabour, S. Fidler, and K. Kreis, "Align your flow: Scaling continuous-time flow map distillation," in *Advances in Neural Inf. Process. Syst. (NIPS)*, 2025.

[220] M. Gülle, J. Yun, Y. U. Alçalar, and M. Akçakaya, "PnP-CM: Consistency models as plug-and-play priors for inverse problems," *Proceedings of the IEEE/CVF Int. Conf. Comput. Vis. Pattern Recogn. (CVPR)*, 2026.

[221] A. Sabour, M. S. Albergo, C. Domingo-Enrich, N. M. Boffi, S. Fidler, K. Kreis, and E. Vanden-Eijnden, "Test-time scaling of diffusions with flow maps," *arXiv preprint arXiv:2511.22688*, 2025.

[222] A. Spagnoletti, J. Prost, A. Almansa, N. Papadakis, and M. Pereyra, "Latino-pro: Latent consistency inverse solver with prompt optimization," in *Proceedings of the IEEE/CVF Int. Conf. Comput. Vis. Pattern Recogn. (CVPR)*, pp. 19597–19607, 2025.

[223] S. Martin, A. Gagneux, P. Hagemann, and G. Steidl, "PnP-flow: Plug-and-play image restoration with flow matching," in *Intl. Conf. Learn. Repr. (ICLR)*, 2025.

[224] M. Renaud, J. Prost, A. Leclaire, and N. Papadakis, "Plug-and-play image restoration with stochastic denoising regularization," in *Intl. Conf. on Mach. Learn. (ICML)*, 2024.

[225] R. Laumont, V. D. Bortoli, A. Almansa, J. Delon, A. Durmus, and M. Pereyra, "Bayesian imaging using plug & play priors: when langevin meets tweedie," *SIAM Journal on Imaging Sciences*, vol. 15, no. 2, pp. 701–737, 2022.

[226] R. Laumont, V. De Bortoli, A. Almansa, J. Delon, A. Durmus, and M. Pereyra, "On maximum a posteriori estimation with plug & play priors and stochastic gradient descent," *Journal of Mathematical Imaging and Vision*, vol. 65, no. 1, pp. 140–163, 2023.

[227] Y. Hu, M. Delbracio, P. Milanfar, and U. S. Kamilov, "A restoration network as an implicit prior," in *Intl. Conf. Learn. Repr. (ICLR)*, 2024.

[228] Y. Song, P. Dhariwal, M. Chen, and I. Sutskever, "Consistency models," in *Intl. Conf. on Mach. Learn. (ICML)*, pp. 32211–32252, PMLR, 2023.

[229] B. Kim, J. Kim, J. Kim, and J. C. Ye, "Generalized consistency trajectory models for image manipulation," in *Intl. Conf. Learn. Repr. (ICLR)*, 2025.

[230] T. Garber and T. Tirer, "Zero-shot image restoration using few-step guidance of consistency models (and beyond)," in *Proceedings of the IEEE/CVF Int. Conf. Comput. Vis. Pattern Recogn. (CVPR)*, pp. 2398–2407, 2025.

[231] P. Holderrieth, D. Chen, L. Eyring, I. Shah, G. Anantharaman, Y. He, Z. Akata, T. Jaakkola, N. M. Boffi, and M. Simchowitz, "Diamond maps: Efficient reward alignment via stochastic flow maps," *Intl. Conf. on Mach. Learn. (ICML)*, 2026.

[232] A. Mammadov, S. Takao, B. Chen, R. Baptista, M. Mardani, Y. W. Teh, and J. Berner, "Variational flow maps: Make some noise for one-step conditional generation," *Intl. Conf. on Mach. Learn. (ICML)*, 2026.

[233] J. Y. Huang, J. Lin, S. Shah, K. Nair, and N. M. Boffi, "How to guide your flow: Few-step alignment via flow map reward guidance," *Intl. Conf. on Mach. Learn. (ICML)*, 2026.

[234] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Int. Conf. Med. Image Comput. Comput.-Assist. Interv.*, pp. 234–241, Springer, 2015.

[235] Y. Zhang, P. Yu, Y. Zhu, Y. Chang, F. Gao, Y. N. Wu, and O. Leong, "Flow priors for linear inverse problems via iterative corrupted trajectory matching," *Advances in Neural Inf. Process. Syst. (NIPS)*, vol. 37, pp. 57389–57417, 2024.

[236] H. Ben-Hamu, O. Puny, I. Gat, B. Karrer, U. Singer, and Y. Lipman, "D-flow: Differentiating through flows for controlled generation," in *Intl. Conf. on Mach. Learn. (ICML)*, PMLR, 2024.

[237] G. Daras, A. Rodriguez-Munoz, A. Klivans, A. Torralba, and C. C. Daskalakis, "Ambient diffusion omni: Training good models with bad data," in *Advances in Neural Inf. Process. Syst. (NIPS)*, 2025.

[238] J. Tachella, M. E. Davies, and L. Jacques, "Unsure: self-supervised learning with unknown noise level and stein's unbiased risk estimate," in *Intl. Conf. Learn. Repr. (ICLR)*, 2025.

[239] B. Øksendal, *Stochastic differential equations*. Springer, 2003.

[240] E. C. Faye, M. D. Fall, and N. Dobigeon, "Regularization by denoising: Bayesian model and langevin-within-split gibbs sampling," *IEEE Trans. Image Process.*, vol. 34, pp. 221–234, 2024.

[241] Z. Dou and Y. Song, "Diffusion posterior sampling for linear inverse problem solving: A filtering perspective," in *Intl. Conf. Learn. Repr. (ICLR)*, 2024.

[242] C. Schuhmann, R. Beaumont, R. Vencu, C. Gordon, R. Wightman, M. Cherti, T. Coombes, A. Katta, C. Mullis, M. Wortsman, *et al.*, "LAION-5B: An open large-scale dataset for training next generation image-text models," *Advances in Neural Inf. Process. Syst. (NIPS)*, vol. 35, pp. 25278–25294, 2022.

[243] J. Zhu, P. Zhuang, and S. Koyejo, "HIFA: High-fidelity text-to-3d generation with advanced diffusion guidance," in *Intl. Conf. Learn. Repr. (ICLR)*, 2024.

[244] K. Crowson, S. A. Baumann, A. Birch, T. M. Abraham, D. Z. Kaplan, and E. Shippole, "Scalable high-resolution pixel-space image synthesis with hourglass diffusion transformers," in *Intl. Conf. on Mach. Learn. (ICML)*, 2024.

[245] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition. 2016 ieee conf," in *Proceedings of the IEEE/CVF Int. Conf. Comput. Vis. Pattern Recogn. (CVPR)*, pp. 770–778, 2015.

[246] M. Balcerak, T. Amiranashvili, A. Terpin, S. Shit, L. Bogensperger, S. Kaltenbach, P. Koumoutsakos, and B. Menze, "Energy matching: Unifying flow matching and energy-based models for generative modeling," in *Advances in Neural Inf. Process. Syst. (NIPS)*, 2025.

[247] X. Ai, Y. He, A. Gu, R. Salakhutdinov, J. Z. Kolter, N. M. Boffi, and M. Simchowitz, "Joint distillation for fast likelihood evaluation and sampling in flow-based models," *Intl. Conf. Learn. Repr. (ICLR)*, 2026.

[248] R. Flamary, N. Courty, *et al.*, "Pot: Python optimal transport," *J. Mach. Learn. Res.*, vol. 22, no. 78, pp. 1–8, 2021.

[249] E. Ryu, J. Liu, S. Wang, X. Chen, Z. Wang, and W. Yin, "Plug-and-play methods provably converge with properly trained denoisers," in *Intl. Conf. on Mach. Learn. (ICML)*, pp. 5546–5557, PMLR, 2019.