

NEURAL REPRESENTATIONS AND PROBABILISTIC INFERENCE FOR CONTINUOUS SIGNALS

AZMI A. HAIDER

THESIS SUBMITTED FOR THE DEGREE "DOCTOR OF PHILOSOPHY"
DISSERTATION BY PUBLICATIONS

University of Haifa
Faculty of Social Science
Department of Computer Science

OCTOBER, 2026



NEURAL REPRESENTATIONS AND PROBABILISTIC INFERENCE FOR CONTINUOUS SIGNALS

By: AZMI A. HAIDER
DR. DAN ROSENBAUM

THESIS SUBMITTED FOR THE DEGREE "DOCTOR OF PHILOSOPHY" DISSERTATION BY PUBLICATIONS

University of Haifa
Faculty of Social Science
Department of Computer Science

OCTOBER, 2026

Approved by: _____ Date: _____

(Supervisor)

Approved by: _____ Date: _____

(Chairperson of Advanced studies Committee)

Acknowledgements

I gratefully acknowledge the PhD funding provided by the Data Science Research Center (DSRC) at the University of Haifa, Israel.

I am deeply thankful to my advisor, Dr. Dan Rosenbaum, for his patience, guidance, and steady support throughout my PhD. His feedback, trust, and encouragement helped me navigate both the technical challenges and the long stretches of uncertainty that come with research.

I also wish to thank my M.Sc. advisor, Dr. Hagit Hel-Or, for her generosity with her time and her openness to discussions and questions. Her willingness to engage in thoughtful conversations had a lasting impact on how I approach research.

Finally, I am sincerely grateful to my family for their love, understanding, and support. Their encouragement made it possible for me to stay focused and persistent through the demanding years of this journey.

This doctoral thesis has been examined by a Committee of the Department of Chemistry as follows:

Professor [Enter name] .....
[Enter position]

Professor [Enter name] .....
[Enter position]

Professor [Enter name] .....
[Enter position]

Dr. Dan Rosenbaum .....
Supervisor

Contents

0.1 Introduction XXV

0.2 Thesis Overview XXIX

1 Probabilistic 3D Reconstruction via Latent Posterior Sampling 1

1.1 Introduction 2

1.2 Related Work 4

1.3 Background 6

1.3.1 Auto-decoding 3D representations 6

1.3.2 Posterior sampling with diffusion models 6

1.4 Method 7

1.4.1 Training the representation and reconstruction 7

1.4.2 Training the prior 10

1.4.3 Sampling from the posterior 11

1.5 Experiments 13

1.6 Conclusion 15

1.7 Appendix 16

1.7.1 Data 16

1.7.2 Implementation details 16

1.7.3 Training the Representation and Reconstruction model 16

1.7.4 Experiments 18

1.7.5 Posterior sampling 18

2 Tokenized Neural Fields: Structured Representations of Continuous Signals 21

2.1 Introduction . . . 22
2.2 Related Work . . . 24
2.3 Tokenized Neural Fields (TNF) . . . 28
2.3.1 Overview . . . 29
2.3.2 Coordinate-Conditioned Decoding . . . 29
2.3.3 Training and Inference . . . 30
2.4 Experiments . . . 31
2.4.1 1D Functions . . . 32
2.4.2 2D Physical Fields . . . 34
2.4.3 ERA5 Weather Reanalysis . . . 35
2.4.4 2D Image Modeling . . . 36
2.4.5 3D Scene Reconstruction . . . 38
2.5 Conclusion and outlook . . . 42
2.6 Appendix . . . 45
2.6.1 Decoder Implementation . . . 45
2.6.2 Cross-Attention Querying . . . 46
2.6.3 Experiment Details and More Results . . . 46

3 Flow Neural Processes: Efficient Probabilistic Inference via Amortized Probability Flows 53

3.1 Introduction . . . 54
3.2 Background . . . 56
3.2.1 Neural Processes . . . 56
3.2.2 Flow Matching . . . 58
3.3 Related Work . . . 59
3.3.1 Neural Processes . . . 59
3.3.2 Probabilistic Modeling of Functions . . . 60
3.3.3 Flow-Based Generative Modeling . . . 60
3.3.4 Efficient Generative Inference . . . 61
3.4 Flow Neural Processes . . . 62

3.4.1 The FlowNP Model . . . 62
3.4.2 Experiments . . . 65
3.4.3 Analysis . . . 70
3.5 Direct Flow Neural Processes . . . 73
3.5.1 Motivation . . . 73
3.5.2 Method . . . 74
3.5.3 Experimental Setup . . . 78
3.5.4 Analysis . . . 80
3.5.5 Application . . . 82
3.6 Conclusion . . . 85
3.7 Appendix . . . 86
3.7.1 Consistency of NP Models . . . 86
3.7.2 Training and Sampling . . . 87
3.7.3 Analysis . . . 88
3.7.4 Geometric Analysis . . . 90
3.7.5 Implementation Details for FlowNP and D-FlowNP . . . 91
3.7.6 Qualitative Sampling Grids . . . 96
3.7.7 Qualitative Comparison with Higher-Order Solvers . . . 98
3.7.8 Evaluation Metrics . . . 101
3.7.9 Wall-Clock Inference Analysis . . . 104
3.7.10 Bayesian Optimization . . . 105

List of Figures

1-1 Examples of various 3D prediction tasks performed by generating posterior samples with our method. For each task we show the observation, three samples of the scene and an uncertainty map computed from the variance of 10 samples. Two different views are shown in subsequent rows. **Top:** reconstruction from half of an image. The variance is high in the hidden half of the scene. **Middle:** reconstruction from only a few pixels (5% of a single image). **Bottom:** reconstruction from a few depth values (5% of a full depth image from a single direction). Samples and uncertainty map suggest sparse depth is enough to reconstruct the 3D shape and uncertainty remains only about color. 3

1-2 The reconstruction model mapping latent representations to images of a 3D scene. The latent decoder $D1$ maps the latent vector z_i corresponding to scene i into three multichannel planes (tri-planes) $\{T_{1i}, T_{2i}, T_{3i}\}$. Given an image and a camera position from which the image was taken, a ray is projected onto the scene from each pixel of the image, and multiple 3D points are sampled along the ray. Each 3D point p_j, is projected onto the multichannel tri-planes where each plane produces a feature vector f_j using bilinear interpolation. The three feature vectors are concatenated to form one feature vector f_j^*, and the decoder $D2$ is used to produce RGB and σ values for each 3D point along the ray. Volumetric rendering is then used to generate a single RGB value to be compared to the ground truth value of the pixel in the image. 8

1-3 Novel view reconstruction for held out 3D scenes. Each pair shows the ground truth image (left) and the reconstructed image (right). Top row: SRN cars. Bottom row: Objaverse chairs. 10

1-4 Samples from the trained diffusion model. Each row corresponds to a different sample of the latent representation, corresponding to a different 3D scene, and each column shows a different view reconstructed from the same scene. Left: SRN cars. Right: Objaverse-lvis chairs. 11

1-5 **Left:** The posterior sampling algorithm. **Right:** Illustration of a single step in the iterative process. Conditioned on the previous estimate z_t, the U-net predicts the noise, which is used to compute both z_{t-1} and $\tilde{z}_0$. The latter is fed to the reconstruction model to predict an image from the given view which is compared to the ground truth image y. The error is backpropagated through the frozen networks to compute a gradient which is then added to z_{t-1}. 12

1-6 Posterior samples given a single view for Objaverse chairs. Each row corresponds to a different sample of the scene, and each column shows a different view. The observation in the example on the left carries high information about the scene, resulting in very similar samples. The observations in the middle and right scenes are less informative, and therefore result in more diverse samples, where the chairs are completed with different possible configurations of legs, armrests and backrests. These example demonstrate a coherent merging of observed data and prior information. 13

1-7 3D reconstruction from noisy images. Reconstruction from 80 images without a prior (TensoRF) quickly deteriorates as noise increases. Using our prior to perform posterior sampling results in a much more robust method, significantly outperforming TensoRF even when using an order of magnitude less images (5 images). 13

1-8 Averaging latent samples results in higher PSNR scores but fails to capture the full posterior distribution. In the first row, the bottom half of an image is used as guidance, while in the second row, the top half is used. Since most generated samples closely match the ground truth, averaging leads to a higher PSNR but collapses the distribution into a single reconstruction, limiting the diversity of plausible outcomes. In the third row, guidance is provided using only a few pixels from a depth image. The generated samples vary in color, as color information is not available from the depth input. Averaging across multiple samples produces an intermediate color that achieves a higher PSNR but fails to reflect the full posterior, which contains diverse color variations. . . 20

2-1 TNF: An instance-specific token sequence T is queried by and input coordinate x via a shared decoder D_θ using cross-attention and feed-forward layers. The resulting prediction $\hat{f}(x)$ is compared to the ground truth $f(x)$ to compute a loss L, which is then used to update both the tokens and the shared decoder parameters. 29

2-2 1D function reconstruction using TNF. Ground truth functions (blue) are sampled from Gaussian Processes with RBF kernel (top row) and Matérn kernel (bottom row). TNF reconstructions are shown in red. Columns correspond to representation with increasing token counts (2×8, 4×8, 8×8, and 16×8), demonstrating improved fidelity with more tokens. 32

2-3 1D wave function reconstruction under variable density. Left: Reconstruction with 100% observability. Right: Reconstruction from 10% sparse observations (scattered points). TNF avoids the smoothing artifacts inherent to the grid baseline's interpolation step, accurately recovering the localized, high-frequency structures of the target signal even in the sparse regime. 32

2-4 Reconstruction error (MSE) across varying observation densities. Comparing TNF (red) against a grid-based baseline (blue) on 1D Wave, Black Hole, Navier-Stokes, and ERA5 datasets. TNF consistently outperforms the grid baseline at all sparsity levels, demonstrating superior robustness when reconstructing complete signals from highly sparse, non-uniform observations. 33

2-5 Black hole reconstructions from 28×28 simulated images using 16 tokens at different observation densities. Comparison of TNF against a grid-based baseline under full observation and 10% sparse observation (∼78 pixels). 37

2-6 Navier–Stokes vorticity reconstructions using 16 tokens at different observation densities. TNF and a grid-based baseline reconstruct 28×28 vorticity snapshots from full and 1% sparse observations (∼8 scattered points). At 1% density the grid baseline produces blocky, piecewise-constant artifacts from nearest-neighbor interpolation, while TNF leverages its learned continuous prior to produce smooth, physically plausible reconstructions. 38

2-7 ERA5 temperature (top) and wind (bottom) reconstructions on a 41×41 latitude–longitude grid using 20 tokens. Each row represents a different sample, showing the observed input, the TNF reconstruction, and the grid-baseline reconstruction under full observation (left block) and 2% sparse observation (right block). At full density TNF reproduces fine spatial gradients and flow structure while the grid baseline shows visible smoothing; at 2% density both methods recover large-scale patterns and fine-scale details become ambiguous. 39

2-8 Comparison of reconstruction performance. (Left) Quantitative results on CelebA and ImageNette. TNF achieves higher PSNR with much smaller representation and model size compared to prior methods. (Right) Qualitative reconstructions with 512×32 tokens: Left column: CelebA; right column: ImageNette. Image pairs show ground truth (left) vs. TNF output (right). 40

2-9 Cross-attention heatmaps for two representative tokens from a 16-token CelebA model. Each row tracks a single token across six different held-out images (columns). Without spatial supervision, individual tokens consistently attend to coherent semantic regions such as the area beneath the eyes and the mouth. Full per-token visualizations across all tokens and more held-out images are provided in Appendix 2.6.3. . 40

2-10 Unconditional image samples from a Diffusion Transformer (DiT) trained directly on compact (64×32) TNF token sets, decoded back to images through the shared decoder. The resulting samples are diverse and coherent, indicating that the learned token space forms an effective latent for downstream generative modeling. 40

2-11 3D reconstruction on the ShapeNet SRN car category. Left block: reconstruction of training viewpoints; right block: novel-view synthesis from held-out viewpoints. In each pair the ground-truth view is on the left and the TNF rendering on the right. All scenes share a single coordinate-conditioned network and differ only in their per-scene tokens, with no volumetric rendering, explicit geometry, or scene-specific encoders. 42

2-12 Linear interpolation in token space between two scenes. Shown are two examples (rows) from different viewpoints; in each, the scene on the left smoothly transitions through 10 interpolation steps into a different scene on the right, with gradual changes in geometry and appearance. 42

2-13 Unconditional 3D scene samples from a Diffusion Transformer (DiT) trained on compact (64×32) TNF scene tokens, decoded through the shared decoder. The samples exhibit coherent multi-view structure together with diverse geometry and appearance, demonstrating that the token space supports generative modeling of 3D scenes. 43

2-14 Visualization of token specialization in 3D. Each row corresponds to a different token, highlighting the regions of the car it attends to across multiple scenes (columns). The colored images on the left indicate the viewing angles of the corresponding scenes. 43

2-15 TNF: Our token-based coordinate-attention decoder. Each function f is represented by a different set of tokens $\{T_i\}^N$, and can be evaluated at point x using the decoder. 45

2-16 Generated token samples (right) and their nearest training-set reconstructions (left). The visual differences indicate that the model produces novel and diverse samples rather than memorizing the training data. . 49

2-17 Full attention heatmaps for 16 tokens (rows) across 12 test scenes (columns). Each token consistently focuses on similar regions across different images, demonstrating spatial specialization consistency. . . 51

2-18 Generated token samples (right) and their nearest training-set neighbors (left). The visual differences indicate the model generates novel and diverse samples rather than memorizing the training data. 52

3-1 FlowNP models the probability flow over target values conditioned on observed context points. Context tokens remain fixed at the data distribution ($t = 1$), while target tokens evolve along the probability flow. At each integration step, the model predicts the target velocity field conditioned on the context. 63

3-2 Samples from models trained on RBF (left) and Matérn-$\frac{5}{2}$ (right) kernels. ANP captures uncertainty mainly through local predictive variance (shaded area), whereas TNP, NDP, and FlowNP generate coherent posterior samples that capture global uncertainty. Unlike NDP, FlowNP performs conditional sampling directly without guidance. Unlike the autoregressive TNP, FlowNP generates all target points in parallel, resulting in smoother and faster sampling. 68

3-3 Conditional EMNIST samples generated by TNP, NDP, and FlowNP from 70 observed pixels. FlowNP produces sharp and diverse samples while requiring fewer function evaluations than TNP and no guidance for conditional generation, unlike NDP. 69

3-4 Visualization of temperature and wind prediction using conditional samples generated by FlowNP on two held-out data points from ERA5. The model generates coherent samples based on the context points. As more context points are given (bottom row) predictions become more accurate. 70

3-5 FlowNP vs. TNP for a random step function. FlowNP samples capture the sharp transition occurring in random positions, while TNP cannot model this function as each step in the autoregressive prediction is Gaussian. On the right: the marginal distribution of y predicted for a single point where $x = 0$ further demonstrates this and highlights FlowNP's capacity to capture multimodal distributions. 72

3-6 a) D-FlowNP architecture. The model extends FlowNP by replacing the original embedding module with a step-conditioned embedding (embed*), which additionally receives the desired integration step size Δt as input. The remainder of the architecture is identical to FlowNP. b) Training D-FlowNP via Step Amortization. The model is trained to minimize the discrepancy between a single coarse step of size $2\Delta t$ (student) and two fine steps of size Δt (teacher). This self-consistency objective $\mathcal{L}_{\text{step}}$ is added to the standard flow matching loss, enabling the model to learn accurate large-step transitions directly. 73

3-7 1D Gaussian Process Regression. Posterior samples from FlowNP and D-FlowNP conditioned on the same context sets. Rows correspond to different numbers of inference steps T, equivalent to the number of function evaluations (NFE) for the Euler solver used here. While FlowNP samples degrade into noise, lose coherence and collapse to the mean at low NFE, D-FlowNP maintains higher-quality, smoother and more diverse function samples even at $T = 1$ (NFE=1). 76

3-8 CelebA Image Completion. Target predictions conditioned on sparse context pixels. As the number of solver steps T decreases, standard FlowNP produces increasingly blurry samples and exhibits mode collapse. D-FlowNP preserves high-frequency detail and sample diversity even in the single-step setting. 78

3-9 Black Hole Imaging. Reconstruction of simulated black hole disks from sparse data. D-FlowNP successfully reconstructs coherent and diverse ring structures even at $T = 1$, whereas the baseline samples degrade significantly with lower T. 79

3-10 Efficiency-Quality Pareto Front (1D GP). D-FlowNP strictly dominates numerical solvers at low NFE. It maintains high Recall (diversity) even at $T = 1$, whereas baseline solvers suffer from integration drift or mode-collapse. 80

3-11 Cross-domain efficiency–quality tradeoffs on (a) 2D Gaussian processes, (b) Navier–Stokes reconstruction, and (c) black-hole imaging. Across domains, D-FlowNP provides improved posterior quality in the low-computational-budget regime. 83

3-12 Bayesian optimization regret over 1D functions. On standard GPs, D-FlowNP with a single step ($T = 1$) performs similarly to the Gaussian baseline (TNP) and generally outperforms FlowNP ($T = 1$). On the *Hidden Valley* task, which is designed to confound Gaussian predictors with a specific non-Gaussian posterior, TNP with EI strategy fails completely, while D-FlowNP successfully identifies the valley, and significantly outperforms TNP with Thompson Sampling and FlowNP-1 which are at random level. 84

3-13 Bayesian optimization regret on Navier Stokes Flows. D-FlowNP with one sampling step ($T = 1$) consistently achieves lower regret compared to both the Gaussian baseline (TNP) and standard FlowNP with one step. 84

3-14 A comparison of the first four steps of the Bayesian optimization process on the Navier Stokes dataset using FlowNP and D-FlowNP with low step counts ($T = 2$). FlowNP produces noisier, less coherent field predictions, causing Thompson Sampling to suggest poor candidate points. In contrast, D-FlowNP generates coherent predictions that allow the optimization loop to effectively target the minimum. 85

3-15 A visualization of the sampling process, showing the intermediate values $y_t^{\mathtt{tgt}}$ for different time steps. Each column is conditioned on a different context and each color represents a different random sample generated by FlowNP. 89

3-16 Log-likelihood as a function of number of ODE steps, computed using the midpoint ODE solver on the RBF GP data. 90

3-17 Comparison of Sampling Trajectories. We visualize the flow from the prior (noise) to the posterior for a 2-target GP. Standard FlowNP steps (red) diverge into low-probability regions when NFE is small due to the lack of curvature awareness. In contrast, D-FlowNP (blue) learns to follow the curved manifold, enabling accurate "shortcuts" in a single step. 91

3-18 CelebA Image Completion. Target predictions conditioned on sparse context pixels. As the number of solver steps T decreases, standard FlowNP exhibits increasing blurriness and mode collapse, while D-FlowNP preserves sharper high-frequency details and maintains greater sample diversity, even in the single-step regime. 96

3-19 Navier-Stokes Fluid Dynamics. Posterior samples of fluid velocity fields given sparse sensor measurements. D-FlowNP produces sharper reconstructions across all values of T. 97

3-20 Black Hole Imaging. Reconstruction of simulated black hole disks from very sparse data. D-FlowNP reconstructs coherent and diverse ring structures even at $T = 1$, whereas the baseline samples become increasingly averaged and degraded as T decreases. 97

3-21 Comparison of solvers across NFE (Task A). Standard solvers (Euler, Midpoint, RK4) fail to capture the GP smoothness and diversity at low T, while D-FlowNP remains robust. 99

3-22 Comparison of solvers across NFE (Task B). Even higher-order methods like RK4 require multiple steps to match the sample quality D-FlowNP achieves in a single step. 100

3-23 Wall-clock efficiency–quality tradeoffs on the 1D and 2D GP tasks. D-FlowNP reaches low MMD while maintaining stable Precision and Recall at substantially lower inference times than standard FlowNP solvers, demonstrating that its advantage in function evaluations translates into a practical latency improvement. 104

3-24 On standard 1D GP regression tasks with RBF and Matérn-5/2 kernels and on the HiddenValley function, D-FlowNP generally outperforms FlowNP at low step counts. While FlowNP with 4 steps achieves competitive results, reducing the number of steps to 2 or 1 yields significantly higher regret. 105

3-25 Regret on the Navier Stokes task comparing FlowNP and D-FlowNP across varying computational budgets ($T \in \{1, 2, 4, 16\}$). The plots generally show that D-FlowNP maintains lower regret at lower step counts compared to the baseline FlowNP, validating that step amortization allows for efficient and effective sequential decision-making. 106

List of Tables

1.1 Reconstruction from latent. 10

1.2 3D Consistency Comparison: To evaluate 3D consistency, we compare our model with 3Dim [Watson et al., 2022] and Zero-1-to-3 [Liu et al., 2023], trained on SRN Cars and Objaverse Chairs, respectively. Given an input image of a scene, each model generates multiple novel views, which are then used to train a TensoRF (NeRF) model. Since higher 3D consistency in the generated images facilitates NeRF training, models producing more consistent views enable NeRF to achieve a higher PSNR. Our results demonstrate that NeRF trained on images from our model attains the highest PSNR, highlighting the benefits of our model's built-in 3D structural understanding for improved 3D scene synthesis. 18

2.1 Architecture and optimization details across domains. N_t denotes the number of tokens and d_t the token dimension. CA denotes cross-attention blocks. Unless otherwise stated, all final MLPs use sinusoidal activations. The 3D SRN experiment uses ReLU activations. 47

3.1 Comparison of log-likelihood computed on the target set for various 1D GP datasets. We provide mean and standard deviation over five runs. FlowNP consistently achieves state-of-the-art performance, outperforming or matching all other models across all datasets. 67

3.2 Target log-likelihood (mean ± standard deviation) on the EMNIST, CelebA, and ERA5 datasets. FlowNP consistently achieves the best performance. 67

3.3 Target log-likelihood for ablations on three aspects differentiating FlowNP from NDP. 72
3.4 Comparing different ODE solvers on the RBF GP data. 90
3.5 Model hyperparameters used in all experiments. 94
3.6 FID scores on CelebA for varying number of solver steps T (lower is better). 97

0.1 Introduction

Many problems in machine learning and scientific computing involve signals that are fundamentally continuous. Images describe visual quantities over spatial coordinates, physical simulations represent fields that vary over space and time, and three-dimensional scenes define properties such as geometry and appearance throughout continuous space. Although these signals are typically observed through finite collections of measurements, the underlying object of interest can often be viewed as a function

$$f : \mathcal{X} \to \mathcal{Y},$$

where coordinates $x \in \mathcal{X}$ may represent spatial locations, time, viewing directions, or combinations thereof, and $f(x)$ describes the corresponding signal value.

Learning from such signals presents several closely related challenges. First, the signal must be represented in a form that is both compact and expressive. Second, this representation should allow reconstruction from observations that may be sparse, irregularly sampled, or noisy. Finally, because incomplete observations generally do not uniquely determine the underlying signal, a useful model should characterize the uncertainty over plausible reconstructions rather than producing only a single deterministic prediction. These questions—*representation, reconstruction, and probabilistic inference of continuous signals*—form the central theme of this thesis.

Traditional machine-learning systems often operate on signals after discretizing them onto fixed grids. Images are represented as arrays of pixels, three-dimensional data through voxels or point clouds, and physical quantities on simulation meshes. Such representations are convenient because they are compatible with established neural architectures, but they tie the representation to a particular resolution and sampling pattern. This can become limiting when measurements are sparse, irregularly distributed, or collected at resolutions that differ from those used during training, as frequently occurs in scientific and inverse problems.

A different perspective has emerged through *Implicit Neural Representations*

(INRs), also referred to as *Neural Fields* (NFs), which represent a signal directly as a continuous coordinate-conditioned function. Rather than explicitly storing its value at every discretized location, a neural network maps a queried coordinate to the corresponding signal value. Methods such as SIREN Sitzmann et al. [2020a] demonstrated that coordinate-based networks can represent complex, high-frequency signals, while Neural Radiance Fields (NeRFs) Mildenhall et al. [2020] showed the effectiveness of this paradigm for representing the geometry and appearance of three-dimensional scenes. Neural fields have since been adopted across image representation, scientific data, medical reconstruction, and three-dimensional modeling.

A major question arising from this paradigm is how to move from representing *individual signals* to learning reusable representations across entire collections of signals. Standard neural fields often require fitting a separate network for every instance. Conditional neural fields instead share a decoder while associating each instance with a lower-dimensional latent representation. Auto-decoding approaches such as Functa Dupont et al. [2022], as well as related methods for three-dimensional scene representation and generation, demonstrated that such latent spaces can provide compact representations suitable for subsequent learning tasks. More recent work has explored spatially structured latent representations Bauer et al. [2023] as well as attention-based approaches for reusable implicit representations Chen and Wang [2022], Kim et al. [2023].

At the same time, the success of Transformer architectures Vaswani et al. [2017a] has demonstrated the advantages of representing complex data through sets or sequences of learned tokens. Tokenized representations provide a structured interface through which information can be exchanged using attention and have become a fundamental building block of modern machine-learning systems. For continuous signals, however, tokenization is commonly inherited from an underlying discretization, for example through image patches or spatial grids. Extending the structured token-based view to continuous signals without first imposing such a discretization motivates one of the central representation problems considered in this thesis.

Representation alone, however, does not address an equally important property of

continuous reconstruction problems: *uncertainty*. Given only a subset of observations

$$\mathcal{D}_{\text{ctx}} = \{(x_i, y_i)\}_{i=1}^{N},$$

there may be many functions that agree with the observed values while differing substantially at unobserved locations. This ambiguity is particularly pronounced in inverse problems such as three-dimensional reconstruction from a small number of images, reconstruction of physical fields from sparse measurements, or prediction at unobserved spatial and temporal locations. Producing a single estimate may conceal this ambiguity, whereas probabilistic inference aims to characterize a distribution over plausible underlying signals.

Gaussian Processes (GPs) Rasmussen and Williams [2006] provide a classical Bayesian formulation of this problem by defining distributions directly over functions. Their principled treatment of uncertainty makes them particularly attractive for prediction from sparse observations. However, exact Gaussian Process inference becomes computationally expensive as the number of observations grows, and their behavior is strongly determined by the choice of kernel.

Neural Processes (NPs) Garnelo et al. [2018b,a] were introduced as learned alternatives that combine the probabilistic function-space perspective of Gaussian Processes with amortized neural prediction. Given a collection of observed context points, Neural Processes learn to predict distributions over function values at arbitrary target locations. Subsequent developments have increased their capacity through attention mechanisms Kim et al. [2019b], convolutional architectures Gordon et al. [2019], Gaussian predictive formulations, and more expressive autoregressive and Transformer-based models Bruinsma et al. [2023], Nguyen and Grover [2022]. These developments have established Neural Processes as a flexible framework for prediction and uncertainty estimation over families of continuous functions.

In parallel, generative modeling has undergone rapid development. Denoising diffusion models Ho et al. [2020] and score-based generative models Song et al. [2020] learn expressive probability distributions through iterative transformations involving

progressively corrupted versions of the data. Beyond unconditional generation, these models have also become powerful priors for inverse problems, where observations can be incorporated into the sampling procedure to generate multiple reconstructions consistent with the available evidence Chung et al. [2023], Kawar et al. [2022]. This perspective is particularly relevant when an observation admits several plausible explanations and therefore naturally connects modern generative models with probabilistic reconstruction.

More recently, these ideas have been extended from finite-dimensional data such as images to distributions over functions. Neural Diffusion Processes Dutordoir et al. [2023] and Diffusion-Augmented Neural Processes Bonito et al. [2023] incorporate diffusion-based generative modeling into the Neural Process framework. Infinite-dimensional diffusion models Pidstrigach et al. [2023] explore score-based generative modeling directly over continuous function spaces. These developments reflect a broader trend toward combining the flexibility of modern generative models with models whose basic objects are functions rather than fixed-dimensional vectors.

Flow-based generative modeling provides a closely related perspective. Flow Matching Lipman et al. [2022], Rectified Flow Liu et al. [2022], and stochastic-interpolant formulations Albergo et al. [2023] describe generation through continuous probability trajectories that transport samples from a simple reference distribution toward a complex data distribution. Instead of learning the score of a noisy distribution, these approaches can directly learn a velocity field governing the evolution of samples along the probability path. Once learned, the corresponding ordinary differential equation can be integrated to generate samples and, in suitable formulations, evaluate likelihoods. Functional Flow Matching Kerrigan et al. [2024] has further extended this perspective toward generative modeling over function spaces.

The expressive power of diffusion- and flow-based models comes with an important computational limitation. Sample generation normally requires an iterative denoising procedure or numerical integration of a probability-flow trajectory, involving repeated evaluations of a large neural network. Consequently, considerable recent work has focused on reducing the number of inference steps. Some approaches improve the

numerical integration procedure, whereas others alter the learned dynamics or train models specifically for coarse transitions. Consistency Models Song et al. [2023] learn mappings that permit high-quality generation using very few evaluations. Flow-map formulations Boffi et al. [2024] shift the learning problem from predicting only an instantaneous velocity to predicting transformations over finite portions of the trajectory. Shortcut Models Frans et al. [2025] condition the learned transformation on the desired step size and use self-consistency to learn increasingly large transitions, while MeanFlow Geng et al. [2025] provides another recent formulation aimed at one-step generative modeling.

These developments suggest that learning continuous signals can increasingly be approached through the interaction of three ideas: *continuous coordinate-based representations*, *structured learned representations shared across instances*, and *expressive generative models for probabilistic inference*. This thesis investigates different points at which these ideas interact, with the broader objective of developing neural models that can compactly represent continuous signals, reconstruct them from incomplete observations, and reason about the uncertainty that remains.

0.2 Thesis Overview

This thesis is organized around three paper-based chapters developed during the course of the PhD. The works comprising the thesis have been disseminated through peer-reviewed workshop publications and journal submissions. The work underlying **Chapter 1** was accepted and presented as a poster at the **ICLR 2025** workshop: *Frontiers in Probabilistic Inference: Sampling Meets Learning* (Singapore). The work presented in **Chapter 2** was accepted and presented as a poster at the **NeurIPS 2025** workshop: *Structured Probabilistic Inference & Generative Modeling* (San Diego, CA, USA), and was subsequently submitted in an extended form to ***Machine Learning: Science and Technology*** **(MLST)**. **Chapter 3** combines two related works: Flow Neural Processes (FlowNP), which was accepted and presented as a poster at **NeurIPS 2025** (San Diego, CA, USA), and Direct Flow Neural Processes (D-FlowNP), which

was accepted and presented as a poster at the **ICML 2026** workshop: *Structured Probabilistic Inference & Generative Modeling* (Seoul, South Korea). FlowNP and D-FlowNP were subsequently combined into an extended journal manuscript submitted to ***Machine Learning: Science and Technology*** **(MLST)**.

Together, the three chapters approach the modeling of continuous signals from complementary directions and progressively move from a domain-specific probabilistic reconstruction problem toward more general frameworks for continuous representation and probabilistic inference.

Chapter 1: Probabilistic 3D Reconstruction via Latent Posterior Sampling. The work presented in this chapter considers three-dimensional reconstruction as an inherently uncertain inference problem. Reconstructing a complete 3D scene from limited observations is ill-posed: multiple scenes may explain the same image, sparse set of pixels, or depth measurements. This ambiguity motivates moving beyond the conventional goal of predicting a single reconstruction.

The proposed approach combines a neural-field-based representation of 3D scenes with a learned generative prior over their latent representations. A shared conditional neural field reconstructs a scene from its latent representation, while a diffusion model Ho et al. [2020] learns the distribution of latent representations across the training data. At inference time, the generative prior is combined with the reconstruction likelihood to perform posterior sampling conditioned on the available observations. This allows the model to generate multiple plausible three-dimensional reconstructions and explicitly represent uncertainty in regions that are weakly constrained by the observations.

The chapter therefore combines two developments that had largely progressed in parallel: neural fields for continuous three-dimensional representation and diffusion models for expressive probabilistic modeling. Neural radiance fields Mildenhall et al. [2020] and conditional neural-field approaches had demonstrated powerful continuous representations of three-dimensional scenes, while diffusion models had shown that learned generative priors could support posterior inference in inverse problems

Graikos et al. [2022], Chung et al. [2023]. By performing probabilistic inference directly over a latent three-dimensional representation, this chapter demonstrates how these components can be combined to reason about the inherent ambiguity of 3D reconstruction.

This first study also exposes a broader representation question. The ability to perform effective reconstruction and probabilistic reasoning depends strongly on the space in which the underlying signal is represented. Compact global latent vectors are attractive for generative modeling, but aggressive compression may discard localized structure. Spatial representations preserve locality but frequently introduce assumptions about dimensionality, resolution, or geometry. These observations motivate the investigation of a more general and structured representation of continuous signals in the second chapter.

Chapter 2: Tokenized Neural Fields: Structured Representations of Continuous Signals. This chapter addresses the representation problem directly by introducing Tokenized Neural Fields (TNF), in which each continuous signal is represented by a compact set of learnable tokens together with a shared coordinate-conditioned decoder.

TNF combines ideas from conditional neural fields and token-based neural architectures. Like auto-decoding approaches such as Functa Dupont et al. [2022], the representation of each signal is learned jointly with a decoder shared across the dataset. However, rather than compressing the signal into a single global latent vector or imposing a predefined spatial latent structure Bauer et al. [2023], TNF represents each instance as a set of tokens. Coordinate queries interact with these tokens through cross-attention, allowing the decoder to reconstruct signal values at arbitrary continuous coordinates.

Importantly, the tokens are not obtained by dividing the signal into predefined image patches, grid cells, or voxels. Instead, they are directly optimized from the available observations. Their organization is therefore learned from the data rather than prescribed by the discretization of the input. This provides a representation that

can naturally operate on sparse and non-uniformly sampled observations while retaining the structured form that makes token representations attractive for downstream neural processing.

The chapter studies this representation across several continuous domains, including one-dimensional functions, two-dimensional physical fields, weather data, natural images, and three-dimensional scenes. Across these settings, the same basic formulation can be used despite differences in dimensionality and sampling structure. The learned tokens also provide a compact latent space on which generative models can operate, connecting continuous neural representations with the token-based generative architectures that have become increasingly prevalent in modern machine learning.

Whereas Chapter 1 focuses on performing probabilistic inference within a particular latent representation of three-dimensional scenes, Chapter 2 therefore takes a broader view and asks how continuous signals themselves should be represented. It establishes a structured representation that lies between two common extremes: an entire neural network fitted independently to every signal and a single global latent vector that summarizes the signal as a whole.

Chapter 3: Flow Neural Processes: Efficient Probabilistic Inference via Amortized Probability Flows. This chapter combines two closely related works developed sequentially during the PhD: *Flow Neural Processes* (FlowNP), The and *Direct Flow Neural Processes* (D-FlowNP).

The chapter returns to probabilistic inference, but at the general level of continuous functions rather than a particular three-dimensional latent representation. It builds on the Neural Process framework Garnelo et al. [2018b], in which a model receives arbitrary sets of observed context points and predicts a distribution over function values at arbitrary target locations.

Existing Neural Process formulations exhibit different trade-offs between computational efficiency and the expressiveness of their predictive distributions. Simple conditional models provide efficient inference but may represent uncertainty inadequately, while autoregressive approaches such as Transformer Neural Processes Nguyen

and Grover [2022] can represent richer dependencies at the cost of sequential generation. Diffusion-based Neural Processes Dutordoir et al. [2023], Bonito et al. [2023] demonstrate that modern generative models can provide more expressive function-space distributions, but introduce iterative inference.

The first part of the chapter introduces Flow Neural Processes (FlowNP), which formulate conditional function inference through Flow Matching Lipman et al. [2022]. Rather than directly parameterizing a simple predictive distribution, FlowNP learns a context-conditioned velocity field that transports an initial noise distribution toward the conditional distribution of target function values. Context observations remain fixed while target values evolve along the learned probability flow. This formulation supports coherent posterior sampling, uncertainty quantification, and likelihood evaluation within a single probabilistic framework while generating all target values jointly.

The flexibility of the model allows the same formulation to be applied to synthetic Gaussian Processes, image completion, weather fields, Navier–Stokes simulations, and black-hole imaging. In this sense, FlowNP provides a general probabilistic counterpart to the representation perspective developed in Chapter 2: rather than designing a probabilistic model for one particular signal modality, the objective is to model distributions over arbitrary continuous functions conditioned on sparse observations.

However, FlowNP inherits the primary computational limitation of continuous flow models. Generating a posterior sample requires numerically integrating the learned probability-flow ODE, and the quality of the resulting sample depends on the number of network evaluations used during integration. This becomes particularly restrictive in applications requiring repeated posterior queries, such as Bayesian optimization, active learning, and uncertainty-aware decision making.

The second part of the chapter introduces Direct Flow Neural Processes (D-FlowNP), which amortize the integration process itself. Inspired by recent work on learned flow maps Boffi et al. [2024] and Shortcut Models Frans et al. [2025], D-FlowNP conditions the model on the desired integration step size and introduces a self-consistency objective in which a single large transition is trained to reproduce

the result of multiple smaller transitions. The model consequently learns not only the instantaneous direction of the probability flow but also how to traverse larger portions of its trajectory directly.

This step amortization substantially reduces the number of function evaluations required for posterior generation, allowing accurate one-step and few-step sampling while preserving the expressive probabilistic behavior of FlowNP. The chapter therefore addresses both sides of a central trade-off in modern generative inference: increasing distributional expressiveness while retaining the computational efficiency necessary for practical downstream use.

Taken together, the three chapters study complementary components of a common problem. Chapter 1 demonstrates why probabilistic modeling is important when reconstructing complex continuous objects from incomplete observations and explores posterior inference over learned latent representations. Chapter 2 investigates how continuous signals can be represented compactly and structurally without relying on fixed discretizations. Chapter 3 develops a general framework for expressive probabilistic inference over functions and then addresses the computational cost introduced by such generative inference.

Collectively, the thesis advances neural approaches for *representing, reconstructing, and performing efficient probabilistic inference over continuous signals.* The progression from latent posterior inference in a specific continuous domain, through structured representations applicable across domains, to general and computationally efficient probabilistic function inference reflects a common objective: enabling neural models to operate on continuous data in a manner that is flexible with respect to sampling, expressive with respect to uncertainty, and practical in terms of computational cost.

Chapter 1

Probabilistic 3D Reconstruction via Latent Posterior Sampling

Abstract

The remarkable achievements of both generative models of 2D images and neural field representations for 3D scenes present a compelling opportunity to integrate the strengths of both approaches. In this work, we propose a methodology that combines a NeRF-based representation of 3D scenes with probabilistic modeling and reasoning using diffusion models. We view 3D reconstruction as a perception problem with inherent uncertainty that can thereby benefit from probabilistic inference methods. The core idea is to represent the 3D scene as a stochastic latent variable for which we can learn a prior and use it to perform posterior inference given a set of observations. We formulate posterior sampling using the score-based inference method of diffusion models in conjunction with a likelihood term computed from a reconstruction model that includes volumetric rendering. We train the model using a two-stage process: first we train the reconstruction model while auto-decoding the latent representations for a dataset of 3D scenes, and then we train the prior over the latents using a diffusion model. By using the model to generate samples from the posterior we demonstrate that various 3D reconstruction tasks can be performed, differing by the type of observation used as inputs. We showcase reconstruction from single-view, multi-view, noisy images, sparse pixels, and sparse depth data. These observations vary in the amount of information they provide for the scene and we show that our method can model the varying levels of inherent uncertainty associated with each task. Our experiments illustrate that this approach yields a comprehensive method capable of accurately predicting 3D structure from diverse types of observations.

1.1 Introduction

3D prediction using neural networks [Mildenhall et al., 2020, Sitzmann et al., 2019, Park et al., 2019] has garnered significant attention, tackling two main challenges: **3D reconstruction** (predicting 3D representations from limited observations) and **3D generation** (sampling new 3D scenes using generative models conditioned on signals like text or images). While 3D generation employs probabilistic generative models, 3D reconstruction is in most cases an ill-posed problem that requires incorporating prior knowledge and could therefore benefit from probabilistic inference methods.

In this work, we propose a probabilistic framework for 3D reconstruction. By combining a generative **prior** over latent 3D representations with a **likelihood** term from a reconstruction model, our approach predicts the full **posterior** distribution of a scene's 3D structure given sparse or noisy observations. This framework leverages a volumetric renderer based on a shared conditional neural field (CNF), trained in two

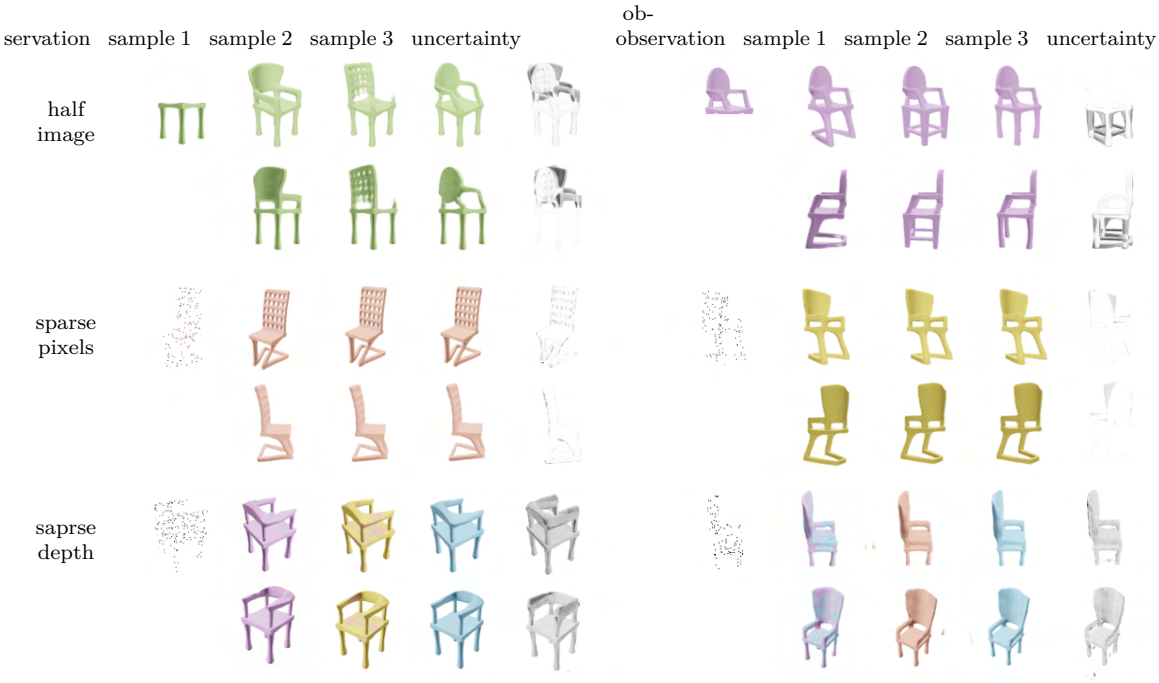


Figure 1-1: Examples of various 3D prediction tasks performed by generating posterior samples with our method. For each task we show the observation, three samples of the scene and an uncertainty map computed from the variance of 10 samples. Two different views are shown in subsequent rows. **Top:** reconstruction from half of an image. The variance is high in the hidden half of the scene. **Middle:** reconstruction from only a few pixels (5% of a single image). **Bottom:** reconstruction from a few depth values (5% of a full depth image from a single direction). Samples and uncertainty map suggest sparse depth is enough to reconstruct the 3D shape and uncertainty remains only about color.

stages:

1. **Auto-decoding** optimizes the shared CNF and latent representations for scenes in the training set.

2. **A diffusion model** captures the prior distribution over the latent representations.

Our method uses a tri-plane latent structure [Chan et al., 2021, Chen et al., 2022] for efficient representation, balancing global and local 3D information. Diffusion-based posterior sampling, guided by reconstruction gradients, enables probabilistic reasoning and uncertainty quantification. In contrast to previous work suggesting amortizing posterior inference [Kosiorek et al., 2021a] in a variational autencoder setting, which

experimentally demonstrated only limited results. By guiding the Langevin sampling with the gradient of the reconstruction model we combine the strength of two recently successful methods (1) iterative sampling with diffusion models and (2) gradient based optimization for translating observations to 3D representations.

We validate our method on tasks like single-view reconstruction, reconstruction from sparse pixels or depth, and noisy observations, demonstrating improved coverage of ground truth structures and generating uncertainty maps for unobserved regions.

Our contributions are as follows:

1. A probabilistic framework for 3D reconstruction, leveraging a diffusion prior and NeRF-based decoder.

2. A two-stage training approach: auto-decoding latent 3D representations and training a diffusion model as a prior.

3. Demonstrations on diverse 3D reconstruction tasks, showcasing robustness to sparse and noisy observations.

4. Enhanced reconstruction quality and uncertainty quantification through posterior sampling.

All code, models, and data will be released upon publication.

1.2 Related Work

Latent variable models over 3D scenes Early approaches like GQN [Eslami et al., 2018] used variational autoencoders for probabilistic reasoning in simple 3D scenes but lacked specialized 3D geometry. Later, NeRF-based models [Kosiorek et al., 2021b] integrated rendering pipelines but did not fully leverage NeRF's capacity for complex scenes. Our work advances this by replacing amortized inference with high-capacity diffusion models and Langevin posterior sampling. Other efforts [Shen et al., 2022, Sünderhauf et al., 2022, Goli et al., 2023] modeled uncertainty but lacked data-driven priors like ours.

Generating 3D with 2D Generative models Given limited 3D ground truth data, several works [Poole et al., 2022, Watson et al., 2022, Liu et al., 2023] use pretrained 2D diffusion models to infer 3D representations. In a similar approach, Liu et al. [2024] use a 2D prior to compute 3D uncertainty maps. While these approaches achieve impressive generative visual results, they do not explicitly reason about the 3D structure of the scene, which leads to less consistency in generation (see experiment in Sec. 1.7.4 in the appendix) and prevents them from performing the full range of 3D probabilistic reasoning tasks, e.g. reconstruction from depth information.

Generative models of observed 3D representations Despite data scarcity, some works [Shue et al., 2022, Erkoç et al., 2023] train diffusion models directly on 3D datasets using representations like tri-planes or neural fields. Our approach uniquely avoids reliance on 3D ground truth, using only 2D datasets to generate 3D scenes.

Generative models of latent 3D representations Inspired by [Dupont et al., 2022], we enhance conditional neural fields (CNFs) with a compressed tri-plane representation and diffusion-based posterior sampling. Similar models [Bautista et al., 2022, Yang et al., 2023] focus on generative tasks rather than reconstruction, while [Chen et al., 2023] unify training stages but lack our latent compression. Concurrent work [Le et al., 2024] explores full posterior inference but with different objectives.

The concurrent work of Le et al. [2024] shares a similar motivation to ours. It focuses on specific type of noisy observations using a 3D modeling of the corruption field, and extensively demonstrate the advantages of the full posterior distribution over the maximum only (MAP inference). In Zhang et al. [2024a] a 3D generative model is trained based on a Gaussian splatting representation which could also be combined with posterior sampling in future research.

1.3 Background

1.3.1 Auto-decoding 3D representations

Recent advances in 3D scene representation have leveraged deep neural networks. NeRF [Mildenhall et al., 2020] introduced a method to reconstruct 3D scenes by training a neural network on multi-view images, requiring separate models for each scene. Subsequent work, such as PixelNeRF [Yu et al., 2021] and IBRNet [Wang et al., 2021], developed generalizable models that integrate prior knowledge of 3D scenes, reducing the number of views needed. These models often rely on conditional neural fields (CNFs), where a shared neural field is conditioned on scene-specific representations.

Recent studies [Dupont et al., 2022, Bautista et al., 2022, Chen et al., 2023, Yang et al., 2023] proposed the use of CNFs to train representations of scenes that can later be used in downstream tasks. Such models use an *auto-decoding* approach [Bojanowski et al., 2019, Park et al., 2019], where the representations are optimized for each scene concurrently with the training of the shared CNF.

Tri-plane representations [Chan et al., 2021, Chen et al., 2022] have proven effective, maintaining spatial structure and balancing global and local information. These representations condition the CNF by interpolating queried 3D positions across orthogonal planes, as shown in Fig. 1-2.

1.3.2 Posterior sampling with diffusion models

Diffusion models, such as Denoising Diffusion Probabilistic Models (DDPM)[Ho et al., 2020], generate high-quality samples by reversing a forward diffusion process that progressively adds noise. Many different variants [Sohl-Dickstein et al., 2015, Song et al., 2020] use U-Net architectures [Ronneberger et al., 2015] to predict noise and iteratively refine samples. Given a noisy input x_t, the denoised sample x_{t-1} and the

clean estimate $\hat{x}_0$ are computed as:

$$\begin{aligned} x_{t-1} &\sim \mathcal{N}\left(\frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(x_t, t)\right),\ \tilde{\beta}_t I\right), \\ \hat{x}_0(x_t, t) &= \frac{1}{\sqrt{\bar{\alpha}_t}}\left(x_t - \sqrt{1-\bar{\alpha}_t}\epsilon_\theta(x_t, t)\right). \end{aligned} \tag{1.1}$$

Where, $\bar{\alpha}_t = \prod_{s=0}^{t} \alpha_s$ and $\tilde{\beta}_t$ is the noise variance. Diffusion models are widely used as priors for image restoration tasks (e.g., denoising, inpainting) [Choi et al., 2021, Chung et al., 2023, Kawar et al., 2022], with posterior sampling defined as:

$$\nabla_{x_t} \log p_t(x_t|y) = \nabla_{x_t} \log p_t(x_t) + \nabla_{x_t} \log p_t(y|x_t) \tag{1.2}$$

This approach combines the prior score $\nabla_{x_t} \log p_t(x_t)$ with the likelihood gradient $\nabla_{x_t} \log p_t(y|x_t)$, often referred to as guidance. While exact likelihoods typically depend on clean images x_0, approximations have been proposed, as detailed in Sec. 1.4.3.

1.4 Method

In this section we describe our method both at training time and at inference time. Training is based on two stages: (1) training the reconstruction model (RM) while optimizing the latent representation of the training scenes (auto-decoding), and (2) training a diffusion model over the latents as a prior. At inference time we use the trained prior and the reconstruction model to perform posterior sampling of the latents. For all implementation details please refer to Sec. 1.7.2 in the appendix.

1.4.1 Training the representation and reconstruction

The reconstruction model is a CNF followed by a volumetric renderer. Conditioned on a scene representation, the CNF predicts the values of 3D positions within the scene that are subsequently used by the volumetric renderer. The CNF is trained while concurrently auto-decoding the representation of each scene. The role of the reconstruction model is to form a mapping from the representation vectors to the

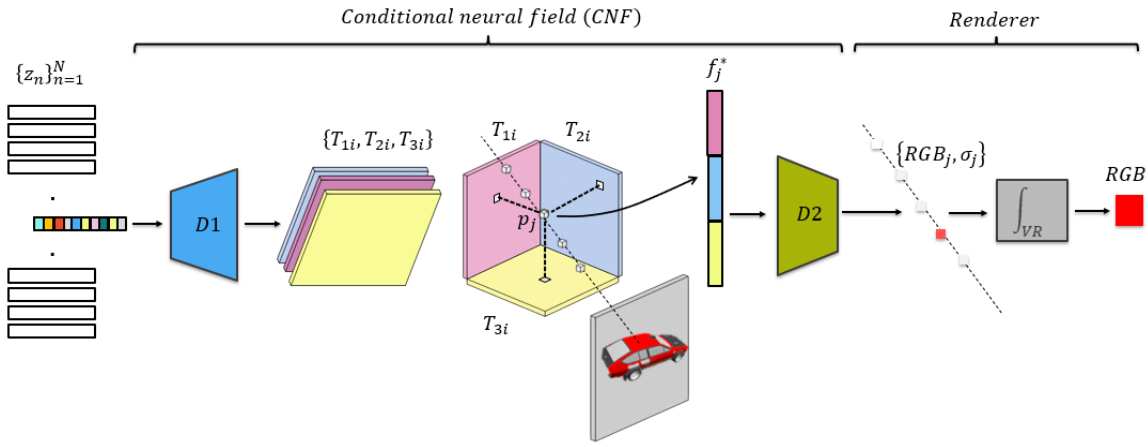


Figure 1-2: The reconstruction model mapping latent representations to images of a 3D scene. The latent decoder $D1$ maps the latent vector z_i corresponding to scene i into three multichannel planes (tri-planes) $\{T_{1i}, T_{2i}, T_{3i}\}$. Given an image and a camera position from which the image was taken, a ray is projected onto the scene from each pixel of the image, and multiple 3D points are sampled along the ray. Each 3D point p_j, is projected onto the multichannel tri-planes where each plane produces a feature vector f_j using bilinear interpolation. The three feature vectors are concatenated to form one feature vector f_j^*, and the decoder $D2$ is used to produce RGB and σ values for each 3D point along the ray. Volumetric rendering is then used to generate a single RGB value to be compared to the ground truth value of the pixel in the image.

values of the observations, i.e. image pixels, and also serve as the model through which the representation is optimized, effectively mapping the 3D scene observations back into the representation.

The model is depicted in Fig. 1-2. The latent vector $z_i \in \mathbb{R}^d$, corresponding to the i-th scene, is first reshaped into a 2D map of shape $r \times r \times c$. The latent decoder $D1$ decodes z_i into a 3D tensor $T \in \mathbb{R}^{R \times R \times 3C}$ using a series of ResNet blocks. T is reshaped to form a tri-plane representation $T_{1i}, T_{2i}, T_{3i} \in \mathbb{R}^{R \times R \times C}$. The tri-planes structure is used for reconstruction as follows: given an image of a scene, rays are projected from each pixel into the 3D scene, and multiple 3D points are sampled along each ray. Each 3D point is projected onto the tri-planes, and using bi-linear interpolation, each plane produces a single corresponding feature vector $f \in \mathbb{R}^C$. The three feature vectors are concatenated to form f^*. The decoder $D2$, an MLP, transforms f^* into RGB and σ values for the corresponding 3D position. This process is repeated for all 3D points along the ray and volumetric rendering is applied on

the ray's points to generate a single RGB value for the pixel from which the ray was projected into the scene.

The reconstruction model (RM) and the latents are trained using the auto-decoding approach as following: at each training iteration, a minibatch of scenes $\mathcal{B}$ is randomly selected along with the corresponding latent vectors, where for each scene a random set of images, and random set of pixels within the images are used. The minibatch is used to apply a forward pass of the reconstruction model on the latents, and backpropagate the loss between the model's output and ground-truth pixel values to all network weights and latent values.

$$\mathcal{L}_{rec} = \sum_{i \in \mathcal{B}} \sum_{x \in \mathcal{X}_i} \|x - RM_\phi(z_i)\|^2 \tag{1.3}$$

where $\mathcal{B}$ is a random minibatch of scenes, and $\mathcal{X}_i$ is a random set of pixels from a random set of images from each scene i. The network weights are updated using $\partial \mathcal{L}_{rec}/\partial \phi$, and the latents are updated using $\partial \mathcal{L}_{rec}/\partial z_i$. In this way the latent representation for each scene is optimized while the network weights converge to their final values. For all experiments in the paper we use a latent dimension of 1024, which forms a highly compressed representation of the scenes. For more implementation details, see Sec. 1.7.2 in the appendix.

Fig. 1-3 shows examples of reconstruction for a few selected scenes using two models that were trained on the SRN Cars [Sitzmann et al., 2019] and Objaverse-lvis chair category [Deitke et al., 2022]. See Sec. 1.7.1 in the appendix for details about the datasets.

After training the reconstruction models, 125 images of held-out test scenes are used to optimize the scene latents while freezing the reconstruction model's weights, and the latents are then used to reconstruct novel views of the scenes. The results show that the latent representation captures the 3D scenes with high fidelity. In Tab. 1.1 we compare the reconstruction accuracy of our compressed representation to Dupont et al. [2022]. Our results are favorable, and we argue that this is due to the spatial structure of the tri-plane representation.



	functa, as reported in [Dupont et al., 2022]	ours
latent dim	1024	1024
PSNR train	24.4	27.67
PSNR test	23.1	26.9

Table 1.1: Reconstruction from latent.

Figure 1-3: Novel view reconstruction for held out 3D scenes. Each pair shows the ground truth image (left) and the reconstructed image (right). Top row: SRN cars. Bottom row: Objaverse chairs.

1.4.2 Training the prior

The goal of the second stage is to obtain a prior over the latent representation. This is achieved by training a generative model based on a Denoising Diffusion Probabilistic Model (DDPM) [Ho et al., 2020] on the latent data obtained in the first stage. As is standard in diffusion models, the model is based on a U-net architecture [Ronneberger et al., 2015] that is trained to denoise the latent representations $\{z_i\}_{n=1}^N \in \mathbb{R}^d$. To comply with the U-net architecture, the latents are reshaped to be $\{z_i\}_{n=1}^N \in \mathbb{R}^{r\times r\times c}$. The training loss is computed by:

$$\mathcal{L}_{gen} = \mathbb{E}_{z\in\{z\},\epsilon\in\mathcal{N}(0,1),t\in U[0,T]} \left\|\epsilon_\theta\left(\sqrt{\bar{\alpha}_t}z + \sqrt{1-\bar{\alpha}_t}\epsilon, t\right) - \epsilon\right\|^2 \tag{1.4}$$

Our implementation is based on Graikos et al. [2022]. Fig 1-4 shows examples of random samples generated from the learned prior. Each image is generated by first sampling a latent from the prior, corresponding to a sampled scene (rows), and then using the reconstruction model to render images of the scene from different views (columns). The resulting samples show both coherence and diversity.



Figure 1-4: Samples from the trained diffusion model. Each row corresponds to a different sample of the latent representation, corresponding to a different 3D scene, and each column shows a different view reconstructed from the same scene. Left: SRN cars. Right: Objaverse-lvis chairs.

1.4.3 Sampling from the posterior

As described in Sec. 1.3.2, different methods have been proposed to sample from posterior distributions given a trained diffusion model as a prior. These methods consist of adding a likelihood term to each step in the iterative process of sampling from the prior. Here, the likelihood term comes from applying the reconstruction model (RM) on the estimated latent, and computing a squared loss compared to the given observation y , which corresponds to a Gaussian log-likelihood.

$$\log p(y|z) = -s\|y - RM_\phi(z)\|^2 + const. = -\mathcal{L}_{rec} + const. \tag{1.5}$$

where s is a scaling factor corresponding to the assumed variance of the reconstruction.

The method is depicted in Fig. 1-5, and described in Alg. 1. In more detail, at each step t the output of the U-net $\epsilon_\theta(z_t, t)$ is used to compute the one-step denoised latent z_{t-1} and a fully denoised estimate $\hat{z}_0$ (Eq. 1.1). The clean estimate is fed to the reconstruction model (RM) which outputs a prediction of the input views. A gradient of the log-likelihood with repsect to z_t can be computed by back-propagating the reconstruction error (Eq. 1.5) between the predicted images and the observed ground-truth images. However, this requires back-propagating through the U-net at each step. In order to accelerate inference, we approximate this gradient by computing

$\tilde{z}_0(z_{t-1}) = \frac{1}{\sqrt{\bar{\alpha}_t}}\left(z_{t-1} - \sqrt{1-\bar{\alpha}_t}\epsilon_\theta(z_t, t)\right)$, and the gradient with respect to z_{t-1}. When using many sampling steps we empirically observe that the difference between z_t and z_{t-1} is negligible and this approximation can be used to efficiently compute the posterior score:

$$\nabla_{z_t} \log p_t(z_t \mid y) \approx \nabla_{z_t} \log p_t(z_t) + \nabla_{z_{t-1}} \log p\Big(y \mid \tilde{z}_0(z_{t-1}, t)\Big), \tag{1.6}$$

Repeating this process for $t = T...1$ forms an approximated Langevin sampling process from the posterior distribution.

As the reconstruction loss is calculated with no regards to pixel order or quantity, this approach allows training a single prior model, and then use it to generate posterior samples for various types of conditioning signals. Examples include conditioning on many images, few images, or even a few random pixels per scene. Moreover, the desired inference task does not even need to be known at training time, as long as a corresponding reconstruction term can be formulated and differentiated at inference time.

Algorithm 1 Posterior Sampling

Input: images y, scale s

Initialize $z_T \sim N(0, 1)$

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

- $\epsilon \leftarrow \text{U-net}_\theta(z_t, t)$
- $z_{t-1} \sim \mathcal{N}\left(\frac{1}{\sqrt{\alpha_t}}\left(z_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon\right), \tilde{\beta}\right)$
- $\tilde{z}_0(z_{t-1}) = \frac{1}{\sqrt{\bar{\alpha}_t}}\left(z_{t-1} - \sqrt{1-\bar{\alpha}_t}\epsilon\right)$
- $\mathcal{L}_{rec} = \|y - RM_\phi\left(\tilde{z}_0(z_{t-1})\right)\|^2$
- $z_{t-1} \leftarrow z_{t-1} - s\partial\mathcal{L}_{rec}/\partial z_{t-1}$

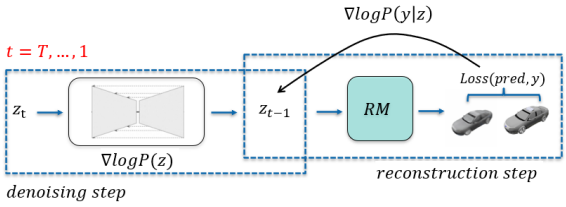


Figure 1-5: **Left:** The posterior sampling algorithm. **Right:** Illustration of a single step in the iterative process. Conditioned on the previous estimate z_t, the U-net predicts the noise, which is used to compute both z_{t-1} and $\tilde{z}_0$. The latter is fed to the reconstruction model to predict an image from the given view which is compared to the ground truth image y. The error is backpropagated through the frozen networks to compute a gradient which is then added to z_{t-1}.

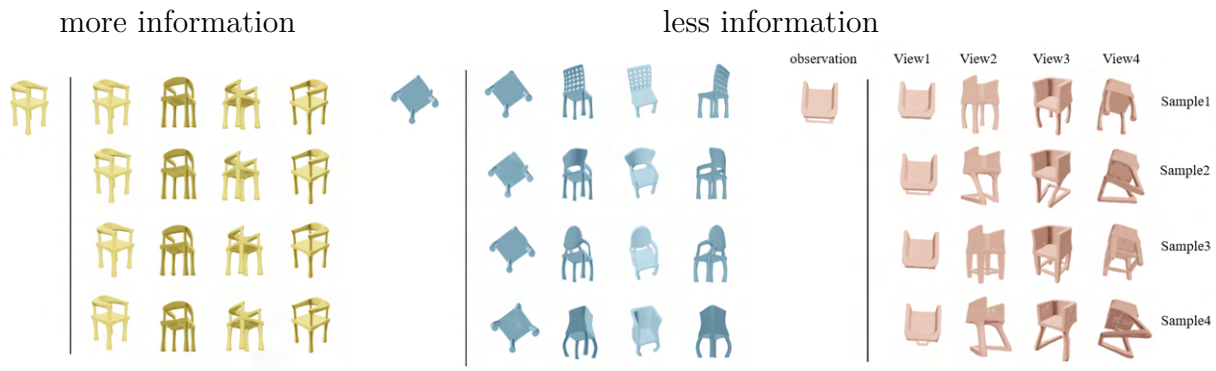


Figure 1-6: Posterior samples given a single view for Objaverse chairs. Each row corresponds to a different sample of the scene, and each column shows a different view. The observation in the example on the left carries high information about the scene, resulting in very similar samples. The observations in the middle and right scenes are less informative, and therefore result in more diverse samples, where the chairs are completed with different possible configurations of legs, armrests and backrests. These example demonstrate a coherent merging of observed data and prior information.

1.5 Experiments

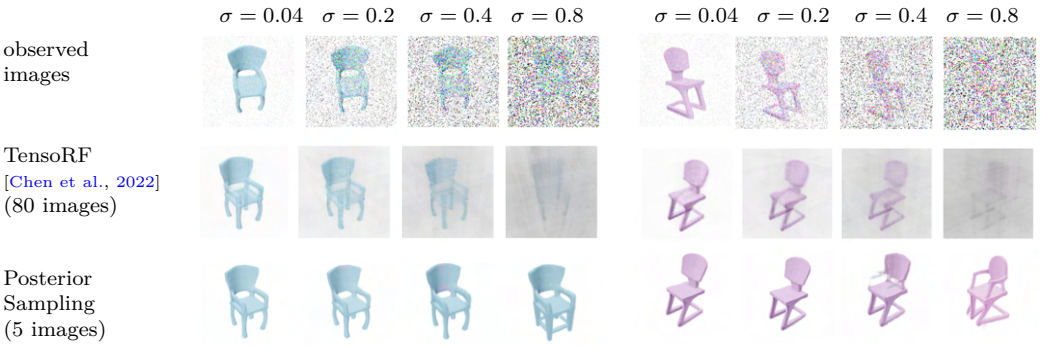


Figure 1-7: 3D reconstruction from noisy images. Reconstruction from 80 images without a prior (TensoRF) quickly deteriorates as noise increases. Using our prior to perform posterior sampling results in a much more robust method, significantly outperforming TensoRF even when using an order of magnitude less images (5 images).

For all experiments we use the same model and the same configuration.

Generating conditional samples

We show results of generating posterior samples given one observed image per scene. In Fig. 1-6, three examples from Objaverse chairs are shown. In the scene shown on the left, the given image contains enough information to predict any view of the

scene with certainty. This results in multiple samples (rows) that are almost identical. In the other examples the observed image is less informative and does not provide enough information about the scene from all angles. Therefore, samples from the posterior exhibit more diversity in the way they complete the missing information. More concretely, the chairs observed from uninformative views are predicted to have different possible leg, armrest and backrest configurations. Note that while the samples are different, the generated latent is a 3D representation, so each sample can be used to predict a coherent set of images from different views.

In Fig. 1-1, we demonstrate the ability of the method to perform more diverse probabilistic reasoning tasks. We show prediction from half-image inputs, from a sparse set of pixels of one image (5%), and from a sparse set of depth map pixels (5% of a depth map from a single view). For each scene we show three samples, showing two different views for each, and an uncertainty map. The uncertainty is computed by generating 10 samples of the scene, rendering corresponding 10 images for each view and computing the variance in the rendered images. Using our method for partial RGB observations (half-image or sparse pixels) is trivial to implement since the reconstruction model operates per pixel and can be used to predict any subset of pixels in the scene. In the case of depth data, we implement a different reconstruction loss comparing the predicted σ values to ground truth values without using the RGB prediction and the renderer in Fig. 1-2. Given a depth pixel value, the ground truth value of σ is set to 1 for the 3D point on the ray sampled at the given depth value, and 0 for all the other 3D points. We emphasize that this reconstruction model is formulated at inference time and is not used at training. The results show the different plausible predictions of the scene and the resulting uncertainty. For the first case we see that the uncertainty is high for the hidden half of the scene as expected. For the other two cases, samples generated from sparse pixel observations demonstrate a high degree of similarity, suggesting, perhaps surprisingly, that even just 5% of the pixels from a single view is sufficient for accurate 3D scene prediction. In case of the sparse depth data, the only uncertain aspect is the object color.

In Fig. 1-7, we evaluate the robustness of our method in 3D reconstruction from

noisy images by comparing it to TensoRF [Chen et al., 2022] without a prior. By generating samples from multiple scene images under increasing noise levels, we demonstrate that posterior sampling significantly improves resilience to noise. While TensoRF, trained on 80 images, experiences a sharp performance drop as noise increases, our method maintains stable performance even with only 5 input images.

1.6 Conclusion

In this work we introduced a methodology that combines the strengths of NeRF-based 3D reconstruction together with the probabilistic reasoning of diffusion models. Our method views 3D reconstruction as an ill-posed perception problem that requires reconciling the observed information with prior knowledge. We showed that (1) 3D scenes can be efficiently represented by compact latent vectors, using a reconstruction model that consists of a tri-plane representation, which preserves spatial structure within the 3D model; and (2) this representation is amenable to training a strong diffusion-model based prior that can later be used to solve various inference tasks. We highlight the importance of predicting the full posterior distribution rather than optimizing for an average sample with higher PSNR (see Sec. 1.7.5). Averaging tends to produce oversmoothed results that may score well numerically but fail to capture the full variability of plausible 3D structures. By emphasizing diverse posterior samples, our approach better represents the inherent uncertainty in 3D scene synthesis, leading to more robust and generalizable models and solving various 3D reconstruction tasks.

Limitations and future work: A main challenge that remains in 3D reconstruction is scaling to more complex and more diverse data towards developing methods that can reliably predict real 3D scenes from different levels of observations. Another challenge is the slow sampling time with diffusion models. While our results are demonstrated on small scale data, we believe that the compressed representation and the principled way of handling uncertainty that we propose, combined with recent developments in accelerating diffusion model sampling, are key for scaling up these models to larger and more complex datasets.

1.7 Appendix

1.7.1 Data

We use two datasets in our experiments. The first dataset is SRN Cars [Sitzmann et al., 2019], which comprises 3,200 scenes with 250 images each. We randomly divide the images in each scene evenly between training images and test images, and we use 3,000 scenes for training, holding out 200 scenes for testing. The second dataset we use is the Objaverse-lvis chair category [Deitke et al., 2022], which comprises 439 instances with 100 images generated for each scene. While this dataset is smaller, it is more diverse in terms of shapes. We use 80% of the images in each scene for training, and hold out 8 scenes for testing and visualizations. For both datasets we use image resolution of 128×128.

1.7.2 Implementation details

In this section we describe the implementation details of our model and experiments. All code, models and data will be made available upon publication.

1.7.3 Training the Representation and Reconstruction model

The weights of the reconstruction model ϕ are randomly initialized, while the latent representations z_i are initialized to zero. The size of the data set corresponds to the number of latent vectors, each latent representing a single scene $\{z_i\}_{i=1}^{N}$ (N scenes = N latents).

During training, the images of each scene optimize only its respective latent, while the entire model, including decoders, is jointly trained.

The latent representation z_i dimensions are $d = 1024$, $r = 16, c = 4$. D1 is constructed using a series of six ResNet blocks where at each block the number of channels is the following: [4, 32, 64, 96, 128, 192]. Blocks are followed by a self-attention layer and alternating upsampling. The resulting 3D tensor T is divided into two tensors responsible for generating RGB and density, $T_{RGB} \in \mathbb{R}^{R \times R \times 3C_{RGB}}$,

$T_\sigma \in \mathbb{R}^{R\times R\times 3C_\sigma}$, respectively. T_{RGB} is reshaped to form a tri-plane representation $T_{1i}, T_{2i}, T_{3i} \in \mathbb{R}^{R\times R\times C_{RGB}}$. Similarly, T_σ forms a tri-plane representation $T'_{1i}, T'_{2i}, T'_{3i} \in \mathbb{R}^{R\times R\times C_\sigma}$. Dimensions are $R = 128, C_{RGB} = 48, C_\sigma = 16$.

For each scene i, we randomly select 4096 rays from pixels in the training images. Along each ray, we sample 220 3D points and project them onto the tri-planes of both the RGB and density planes separately.

For each (RGB and density), this projection extracts three feature vectors from the three planes for further processing. Three vectors are concatenated into a single feature vector $f^*_{RGB} \in \mathbb{R}^{3C_{RGB}}$ for RGB and $f^*_\sigma \in \mathbb{R}^{3C_\sigma}$ for density. While the density feature vector f^*_σ produces density for 3D points by simply summing its elements, the RGB feature vector f^*_{RGB} is passed through D2 to produce a single RGB value. D2 is an MLP of 7 layers. Once all 3D points along the ray have RGB and density values, volumetric rendering, a parameterless process, produces a single RGB value to be compared with the pixel's color.

We train the model with a minibatch $\mathcal{B}$ size of 2 scenes, and with an Adam optimizer using three different learning rates: 1e-3 for the latents, 1e-4 for the D1 parameters and $1e-3$ for D2 parameters.

Our model is based on the code published in Chen et al. [2022].

At test time, a new latent (initialized to zeros) is coupled with the new scene and optimized using the learned/frozen decoders.

Training the Prior

As in Sec.1.4.1, latent representation z_i dimensions are $d = 1024$, $r = 16, c = 4$. The diffusion model used is implemented by Graikos et al. [2022] with the following parameters: The noise scheduler is a linear schedule with parameters $T = 1000, \beta_0 = 1e^{-4}, \beta_T = 2e^{-2}$. The U-net parameters are $model_channels = 64$, $num_resnet_blocks = 2$, $channel_mult = (1, 2, 3, 4)$, $attention_resolutions = [8, 4]$, $num_heads = 4$. We train the model with a minibatch $\mathcal{B}$ size of 32 scenes, and with an Adam optimizer with learning rate equal to 1e-3.

The reconstruction model and the diffusion model were trained on an NVIDIA

GeForce RTX 4090 for Approximately one day each.

1.7.4 Experiments

Generating 3D with 2D generative models

As mentioned in Sec. 1.2, 2D generative models for 3D generation approaches do not fully capture the underlying 3D structure of a scene. To evaluate the impact of explicit 3D structure reasoning, we trained a Neural Radiance Field (NeRF) on images generated by both standard 2D diffusion models and our proposed model, which incorporates an inherent understanding of 3D structure. Tab. 1.2 results indicate that NeRF trained on images from our model produces more consistent 3D reconstructions, highlighting the importance of explicit 3D reasoning in generative models for robust 3D scene synthesis.

Method	**PSNR↑**	**SSIM↑**
3Dim (SRN cars)	28.53	0.96
Ours (SRN cars)	34.7	0.98
zero123 (Objaverse chairs)	26.8	0.925
Ours (Objaverse chairs)	43.4	0.99

Table 1.2: 3D Consistency Comparison: To evaluate 3D consistency, we compare our model with 3Dim [Watson et al., 2022] and Zero-1-to-3 [Liu et al., 2023], trained on SRN Cars and Objaverse Chairs, respectively. Given an input image of a scene, each model generates multiple novel views, which are then used to train a TensoRF (NeRF) model. Since higher 3D consistency in the generated images facilitates NeRF training, models producing more consistent views enable NeRF to achieve a higher PSNR. Our results demonstrate that NeRF trained on images from our model attains the highest PSNR, highlighting the benefits of our model's built-in 3D structural understanding for improved 3D scene synthesis.

1.7.5 Posterior sampling

Posterior sampling involves two types of computations: 1) denoising, using the diffusion as a prior to generate a plausible latent, and 2) reconstruction, using the reconstruction model to align the latent with the observed views.

For all experiments we use the same model using the same inference process. We generate posterior samples using 1000 iterations as described in Alg. 1 with the same scale factor s =5e-3 for all experiments. The only exception is the experiment with noisy data in Fig 1-7, where the scale factor for most extreme noise level $\sigma = 0.8$ was decreased to a value of s =3e-3, corresponding to the high noise variance in the observation.

Full posterior vs higher psnr

In Fig. 1-8, we present two examples (rows) of conditional posterior sampling, where the leftmost image serves as the observed image. The reconstructions are displayed alongside their corresponding PSNR values. We generate 20 samples from the posterior distribution and showcase three individual samples under "single samples," highlighting the variability in reconstruction quality—some being closer to the ground truth than others. Additionally, we display the averaged reconstructions using 5, 10, and 20 latent samples. While averaging improves numerical PSNR scores, it often leads to oversmoothed results that fail to capture the full diversity of plausible 3D structures.

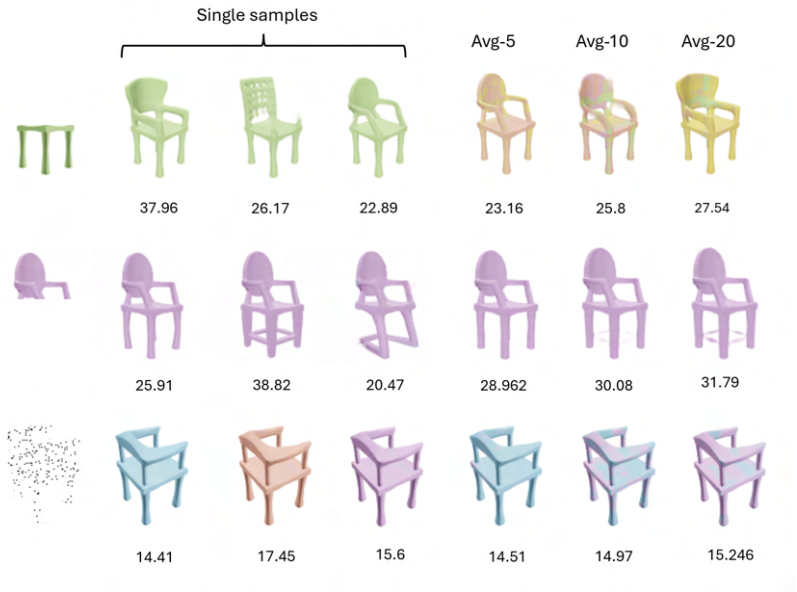


Figure 1-8: Averaging latent samples results in higher PSNR scores but fails to capture the full posterior distribution. In the first row, the bottom half of an image is used as guidance, while in the second row, the top half is used. Since most generated samples closely match the ground truth, averaging leads to a higher PSNR but collapses the distribution into a single reconstruction, limiting the diversity of plausible outcomes. In the third row, guidance is provided using only a few pixels from a depth image. The generated samples vary in color, as color information is not available from the depth input. Averaging across multiple samples produces an intermediate color that achieves a higher PSNR but fails to reflect the full posterior, which contains diverse color variations.

Chapter 2

Tokenized Neural Fields: Structured Representations of Continuous Signals

Abstract

We introduce Tokenized Neural Fields (TNF), a unified framework for representing continuous signals through a compact set of learnable tokens. The tokens are optimized against a shared coordinate-conditioned decoder. Unlike encoder-based pipelines or global latent codes, TNF provides a structured tokenization in which individual tokens specialize to distinct regions of a signal and interact with coordinate queries via cross-attention. By learning tokens directly through gradient-based optimization rather than from fixed grids or patches, TNF naturally handles the sparse, scattered, non-uniform observation regimes that are characteristic of scientific data. We validate TNF across 1D Gaussian-process samples, 2D physical fields (Navier-Stokes fluid dynamics and simulated black-hole imaging), ERA5 weather reanalysis, 2D natural images, and 3D scenes. We show that tokenized representations achieve superior fidelity with fewer parameters than encoder-based baselines, and substantially outperform grid-based tokenization in sparse-observation regimes. TNF tokens further exhibit emergent semantic specialization without supervision, support interpolation between signals, and provide an effective representation space for downstream generative modeling. Together these results position learned tokenization as a flexible primitive for representing and reasoning about continuous scientific signals.

2.1 Introduction

Learning compact and generalizable representations of continuous signals remains a central challenge in machine learning. This capability is particularly important in various scientific domains, where a fundamental challenge is reconstructing underlying physical fields and quantifying their uncertainty from sparse, scattered measurements. Classical approaches to representation learning typically map data into latent spaces using modality-specific encoders. While modern foundation models have demonstrated the power of tokenized representations to capture structure and reason about it, adapting this approach to continuous signals has typically required handcrafted discretizations, such as image patches or voxel grids. These grid-based tokenizers face significant limitations in scientific applications, where they struggle to handle non-uniform or sparse observations because they rely on a fixed discretization grid.

Implicit Neural Representations (INRs), also called Neural Fields (NFs), offer an alternative by modeling signals as continuous functions, mapping coordinates to values. Examples include SIREN [Sitzmann et al., 2020b] for high-frequency image

reconstruction, NeRF [Mildenhall et al., 2020] for 3D scene rendering, and applications to scientific imaging such as CryoDRGN [Zhong et al., 2021] for heterogeneous cryo-EM and NeRP [Shen et al., 2024] for sparse medical reconstruction. While most INRs are limited in scalability due to their per-instance training, some conditional variants such as Functa [Dupont et al., 2022] improve efficiency by sharing a decoder across instances. However, these models often compress each signal into a *global latent*, which can discard localized structure and harm performance [Bauer et al., 2023]. This motivates our central question: can continuous signals be tokenized into compact, learnable sets of tokens that provide structured representations for reconstruction, generation, and probabilistic inference without relying on grid-based discretizations?

We propose **Tokenized Neural Fields (TNF)**, a unified framework that represents each signal through a small set of learnable tokens coupled with a shared coordinate-conditioned decoder. Rather than relying on an imposed feed-forward encoder or predefined patch grids to extract features, TNF's tokens are directly optimized per instance. This optimization-driven approach yields a structured tokenization where individual tokens naturally specialize to distinct aspects of the signal, interacting with coordinate queries solely via cross-attention. By abandoning rigid grid assumptions in favor of direct interaction with continuous coordinates, TNF naturally handles sparse data regimes where grid-based tokenizers typically fail. This approach yields scalable training, efficient adaptation and structured token spaces that support both high-quality reconstruction and probabilistic generative modeling.

To demonstrate the efficacy of TNF for complex signal representation, we validate our approach across a range of continuous domains:

- **1D functions**: We show that TNF's tokens can accurately recover smooth continuous functions with dynamic structure.
- **2D Physical Fields**: We demonstrate the capacity to reconstruct 2D Navier-Stokes fluid dynamics and simulated black hole imaging from highly sparse observations, significantly outperforming the grid-based approach.
- **Weather Fields**: We evaluate TNF on ERA5 weather reanalysis data, demon-

strating accurate reconstruction of temperature, pressure, and wind fields from sparse observations while outperforming grid-based tokenization approaches.

- **2D Natural Images**: We benchmark our method on natural images against state-of-the-art vision architectures, outperforming encoder-based baselines while using highly compact representations and enabling downstream generative modeling via diffusion transformers.

- **3D scenes**: We provide evidence for TNF's capacity to model complex, high-dimensional functions, where tokenized representations enable geometry-free, consistent novel view synthesis and 3D generative modeling.

Together, these results highlight TNF's tokenization as a powerful approach for bridging neural fields with the structured reasoning required for effective representation and inference in continuous domains.

2.2 Related Work

Below we review the closest threads of prior work relevant to TNF's design and its scientific scope.

Probabilistic models of continuous signals. A classical tool for inferring continuous fields from scattered observations is the Gaussian Process (GP), which provides a calibrated posterior over function values but scales poorly with the number of observations. Neural-network counterparts of this idea aim to retain the probabilistic perspective while learning the underlying prior from data. Conditional Neural Processes (CNPs) [Garnelo et al., 2018a] and Neural Processes (NPs) [Garnelo et al., 2018b] learn distributions over functions conditioned on context observations, blending GP-style inference with deep learning. Attentive variants such as attentive-NP [Kim et al., 2019a] and Transformer-NP [Nguyen and Grover, 2022] use attention to better capture long-range dependencies between context and target points, improving generalization to data regimes typical in scientific applications, such as non-uniform

sampling, partial observability, and varying observation density. These methods operate directly on (x, y) pairs of coordinates and values and emphasize uncertainty in the predictions. Our approach is complementary: instead of producing a posterior over function values at queried locations, we produce *structured, tokenized summaries* of an entire underlying field, which can subsequently be used for downstream tasks, transferred across instances, and (as we discuss) support probabilistic reasoning at the level of the representation itself.

Per-instance NF training. The simplest use of Neural Fields (NFs) is to fit one network per signal (e.g. an image, a weather snapshot, a single flow field or a single 3D scene). Architectures such as SIREN [Sitzmann et al., 2020b], which uses sinusoidal activations to faithfully represent high-frequency content, and NeRF [Mildenhall et al., 2020], which reconstructs volumetric scenes from images via differentiable rendering, achieve high reconstruction fidelity in this regime. A common limitation in this approach for scientific applications, is that each new observation is treated as an isolated optimization problem and structure is never amortized across the dataset, so regularities present in a collection of related physical fields are not exploited. Our framework instead shares a decoder across all instances and represents each one through a compact set of per-instance tokens, so that the shared component captures the geometry of the field family while the tokens specialize to each individual realization.

Learned initialization for NFs. A partial response to the cost of per-instance optimization is to meta-learn a good initialization that converges quickly on new signals. Tancik et al. [2020a] is a representative example, where the weights of each NF per scene are meta-learned using the MAML [Finn et al., 2017] approach. While this reduces wall-clock time per fit, the optimized weights themselves remain the representation of each signal, and the resulting weight vectors are high-dimensional and lack a structure that one can transport across datasets or use as input to further models. This motivates representations that are explicitly compact and structured by design, which is what we pursue here.

Conditional NFs and auto-decoding. A more powerful approach is to learn a shared decoder conditioned on per-instance latent vectors, so that the decoder captures common structure across the dataset and the latents act as compact representations. Functa [Dupont et al., 2022] applies this idea to general continuous signals, and several works pursue it for 3D scene generation [Bautista et al., 2022, Chen et al., 2023, Yang et al., 2023, Haider and Rosenbaum, 2025]. The underlying mechanism, jointly optimizing per-instance latents with a shared decoder, is known as *auto-decoding* [Bojanowski et al., 2019, Park et al., 2019]. In the scientific setting this is appealing because the decoder serves as a learned prior over fields and the latents act as a low-dimensional summary of each measurement or simulation. However, when the latent is a single global vector, it tends to compress out local structure, which is a real problem when the field of interest is spatially heterogeneous, as is typical in weather, fluid, or imaging data. Spatial-Functa [Bauer et al., 2023] addresses this by imposing a spatially-structured latent. Our work takes a further step: rather than fix the structure of the representation in advance, we let it emerge from optimization, so that tokens specialize to distinct regions or modes of variation of the field without an a-priori spatial template.

Attention-based and token-based representations. A separate thread of work uses attention to make NFs more reusable across instances. TransINR [Chen and Wang, 2022] treats Transformers as meta-learners for INRs. IPC [Kim et al., 2023] introduces compositional instance-pattern codes to encourage reusability of representations across signals. ANR [Zhang et al., 2024b] uses attention within a heavy encoder–decoder hypernetwork to generate signal-specific tokens. In 3D, token-based scene representations have been explored by OSRT [Sajjadi et al., 2022], 3DShape2VecSet [Zhang et al., 2023], and equivariant neural fields [Wessels et al., 2025], the latter directly leveraging geometric structure of the underlying domain. These methods establish that attention and tokenization are productive primitives for structured signal representation. They share, however, a reliance on handcrafted or heavy auxiliary encoder networks to produce tokens from raw inputs, a design that is most natural for image or 3D volume

inputs and harder to port to the irregular, noisy, partially-observed measurement modalities ubiquitous in scientific applications. We instead obtain tokens directly via gradient-based optimization against the observed measurements, which removes the encoder design choice and applies uniformly across modalities, including sparse and scattered observation regimes that grid- or patch-based tokenizers cannot natively handle.

Neural operators and scientific machine learning. A complementary line of work targets not the representation of a single field but the learning of *maps between function spaces*, for example from an initial condition to a PDE solution. The Fourier Neural Operator [Li et al., 2021] and DeepONet [Lu et al., 2021] are two influential instances; the broader neural-operator framework is reviewed by Kovachki et al. [2023]. PDEBench [Takamoto et al., 2022] provides standardized benchmarks for this line of work, including Navier–Stokes settings related to the fluid-dynamics evaluation we report. Neural operators amortize a *forward solution map* across initial conditions or parameters, whereas TNF amortizes the *representation* of individual fields across a dataset of observations. The two goals are complementary, and one could in principle compose them by using TNF tokens as the input or output of an operator that maps between scientific-field instances.

Foundation models for atmospheric science. Several recent works train large neural models on ERA5 reanalysis and related atmospheric science data for short- and medium-range forecasting. FourCastNet [Pathak et al., 2022] uses adaptive Fourier neural operators for global forecasting at high spatial resolution. GraphCast [Lam et al., 2023] uses a graph neural network on a hierarchical icosahedral mesh. AtmoRep [Lessig et al., 2023] trains a stochastic Transformer on multi-variable atmospheric data. ClimaX [Nguyen et al., 2023] adapts the Vision Transformer architecture to multi-variable weather and climate data and treats spatial patches as tokens. These approaches inherit grid-based tokenization (patches over the latitude–longitude grid); TNF instead learns tokens directly via gradient-based optimization, avoiding

the fixed-grid assumption that becomes a limitation as observation density varies.

Scientific imaging reconstruction. Implicit neural representations have been applied to a range of scientific imaging tasks where the underlying signal is continuous and observations are noisy or sparse. CryoDRGN [Zhong et al., 2021] represents heterogeneous protein conformations as a coordinate-conditioned neural field optimized against cryo-EM micrographs; NeRP [Shen et al., 2024] applies INRs with prior embedding to sparsely sampled medical-image reconstruction. In radio interferometric imaging, regularized maximum-likelihood reconstruction methods such as CHIRP [Bouman et al., 2016] laid the algorithmic groundwork later used in the first Event Horizon Telescope reconstruction of the M87 supermassive black hole [The Event Horizon Telescope Collaboration, 2019], a setting that, like the black-hole imaging benchmark in our experiments, requires reconstruction from extremely sparse, scattered observations. These applications motivate frameworks like TNF that aim to make continuous-signal representations compact, reusable, and well-suited to data regimes where grid-based methods struggle.

2.3 Tokenized Neural Fields (TNF)

We propose a unified framework for learning compact and flexible signal representations using a small set of learnable tokens and a shared coordinate-conditioned decoder. Each instance, whether a function, image, or any other continuous signal, is represented by a fixed set of latent tokens. These tokens are optimized to reconstruct the underlying signal via a shared decoder network, enabling instance-specific adaptation while maintaining global generalization. This formulation removes the need for modality-specific encoders or per-instance networks and supports scalable, transferable representations across domains.

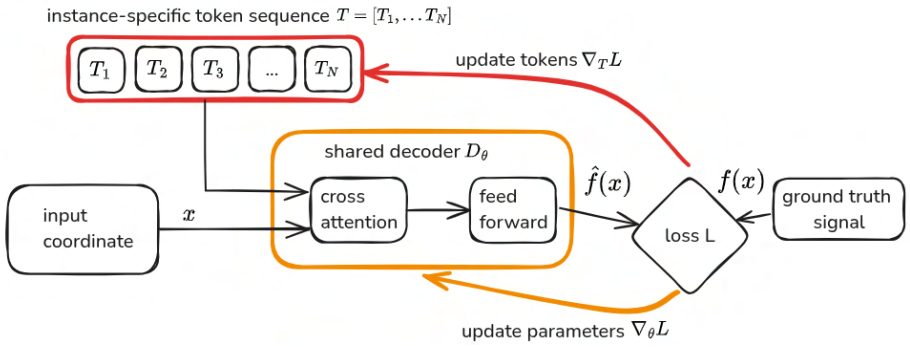


Figure 2-1: TNF: An instance-specific token sequence T is queried by and input coordinate x via a shared decoder D_θ using cross-attention and feed-forward layers. The resulting prediction $\hat{f}(x)$ is compared to the ground truth $f(x)$ to compute a loss L, which is then used to update both the tokens and the shared decoder parameters.

2.3.1 Overview

Let $f : \mathbb{R}^d \to \mathbb{R}^c$ denote a continuous signal (e.g., a 1D curve, a 2D image or a physical field). Rather than using an encoder on patches of points in f, or fitting a dedicated neural field for each f, we associate it with a learnable token set:

$$T = \{T_n\}_{n=1}^{N}, \quad T_n \in \mathbb{R}^{d_t}$$

A shared decoder D_θ maps coordinates and tokens to signal outputs:

$$\hat{f}(x) = D_\theta(x, T)$$

As illustrated in Figure 2-1, the decoder is trained across a dataset of signals, while the token sets are optimized per instance. During inference, the decoder remains fixed and only the tokens are adapted to new data.

2.3.2 Coordinate-Conditioned Decoding

TNF's decoder maps a coordinate $x \in \mathbb{R}^{d_x}$ and the token set T to the signal prediction $\hat{f}(x)$. Its design is flexible and can adapt to various data modalities. A natural choice when using token-based representation is to use cross-attention between the coordinate x and the signal's token set T. Cross-attention enables dynamically extracting relevant

content from the token set into different locations in the continuous signal.

The architecture of the decoder consists of passing the inputs (together with sequential positional encoding) through linear transformations followed by a sine activation function. Then, cross attention is performed between the coordinate and the tokens, followed by several linear layers coupled with sine activations, except for a sigmoid activation in the last layer. We follow the standard transformer and use layer normalization on the tokens before the cross attention, a residual connection between the query and the attention output and layer normalization after the residual connection. We use this approach throughout our experiments, where our decoders mainly differ in the number of layers, their dimensions and the dimensions of the input x and output $\hat{f}(x)$. More details are specified in Appendix 2.6.1 and Figure 2-15.

2.3.3 Training and Inference

A core design principle of TNF is to separate instance-specific and instance-agnostic components: the token set T^i is unique to each instance i, while the decoder D_θ is shared across all instances.

Training is done end-to-end across a dataset of M signals by minimizing a reconstruction loss:

$$\arg \min_{\theta,\ \{T^i\}_{i=1}^M} \frac{1}{M} \sum_{i=1}^{M} \frac{1}{K} \sum_{j=1}^{K} \left\| f_i(x_j) - D_\theta(x_j, T^i) \right\|^2 \tag{2.1}$$

where $\{x_j\}_{j=1}^K$ are randomly sampled coordinates.

We consider two main approaches for training:

Standard Auto-Decoding: Following this approach [Bojanowski et al., 2019, Park et al., 2019], each signal instance f_i, is coupled with a persistent token set $T^i = \{T_n^i\}_{n=1}^N$ which is jointly optimized with the shared decoder D_θ, both using the same reconstruction loss (illustrated in Figure 2-1). Training is performed by stochastic gradient descent, where each iteration updates the shared decoder parameters and the token set corresponding to the signals used in the batch. At inference time, the decoder is kept fixed and new tokens are adapted to unseen data using the same

gradient-based optimization as used in training.

Meta-Learning for Accelerated Inference: To improve the efficiency of token adaptation at inference time, we optionally employ a meta-learning optimization strategy based on MAML [Finn et al., 2017] (as used also in Functa [Dupont et al., 2022]) in some of our experiments. This approach incorporates the gradient-based adaptation process as part of the forward pass during training. For each sample in a batch, the token set undergoes an inner loop of a few gradient descent steps (typically three) to reconstruct the signal. The shared decoder is then updated in the outer loop by backpropagating through these unrolled inner optimization steps. This encourages the decoder to learn a parameters space where novel signals can be accurately tokenized with only a minimal number of gradient updates, drastically accelerating the adaptation process for downstream tasks.

2.4 Experiments

We evaluate Tokenized Neural Fields (TNF) across 1D function regression, 2D physical field reconstruction, weather data modeling, 2D natural images, and 3D scenes. Across these domains, our goal is to demonstrate the generality, compactness, and flexibility of learning per-instance tokens paired with a shared decoder.

Beyond standard dense reconstruction, we introduce a variable-density evaluation for our scientific datasets to highlight a key advantage of TNF: the ability to handle non-uniform, sparse observations without relying on fixed discretization grids. We compare TNF against a grid-based baseline where both models reconstruct the full signal from randomly subsampled points. While the baseline requires interpolating these sparse points onto a regular grid, TNF directly optimizes its tokens on the scattered coordinates.

Finally, to establish the effectiveness of our approach in a domain where many specialized methods have been developed, we benchmark TNF against state-of-the-art encoder-based models on natural images. We then turn to two natural-signal domains

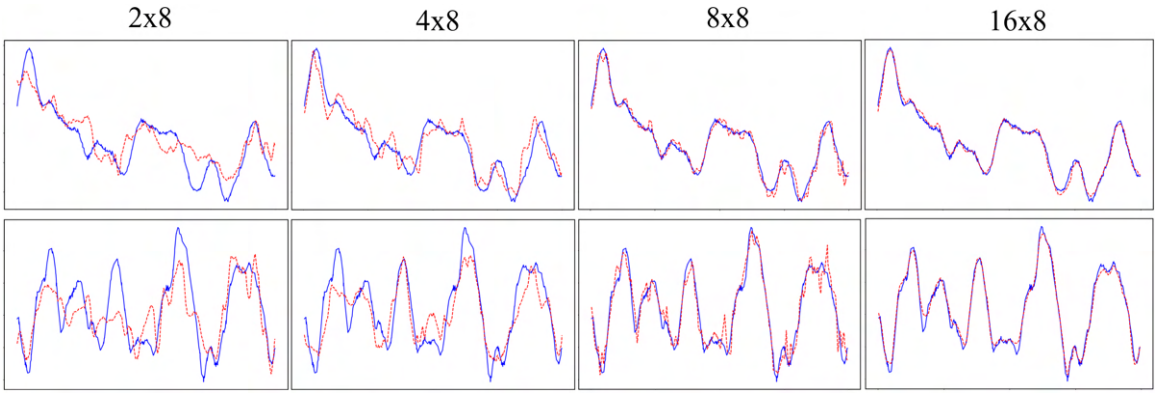


Figure 2-2: 1D function reconstruction using TNF. Ground truth functions (blue) are sampled from Gaussian Processes with RBF kernel (top row) and Matérn kernel (bottom row). TNF reconstructions are shown in red. Columns correspond to representation with increasing token counts (2×8, 4×8, 8×8, and 16×8), demonstrating improved fidelity with more tokens.

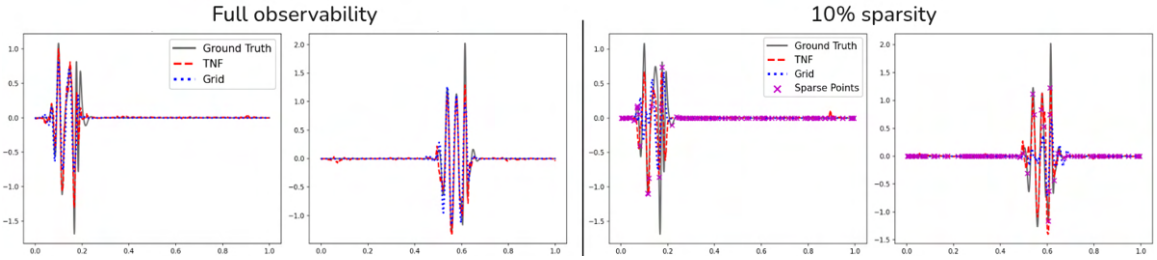


Figure 2-3: 1D wave function reconstruction under variable density. Left: Reconstruction with 100% observability. Right: Reconstruction from 10% sparse observations (scattered points). TNF avoids the smoothing artifacts inherent to the grid baseline's interpolation step, accurately recovering the localized, high-frequency structures of the target signal even in the sparse regime.

where qualitative evaluation is possible, 2D natural images and 3D scenes. On those domains we demonstrate that TNF tokens exhibit emergent semantic specialization and provide a highly effective representation space for downstream generative modeling.

2.4.1 1D Functions

We sample 1K one-dimensional functions from Gaussian processes using RBF and Matérn kernels. Each function is represented by a compact set of learnable tokens optimized jointly with a shared coordinate-conditioned network. Detailed architectural and optimization settings are provided in Appendix 2.6.3.

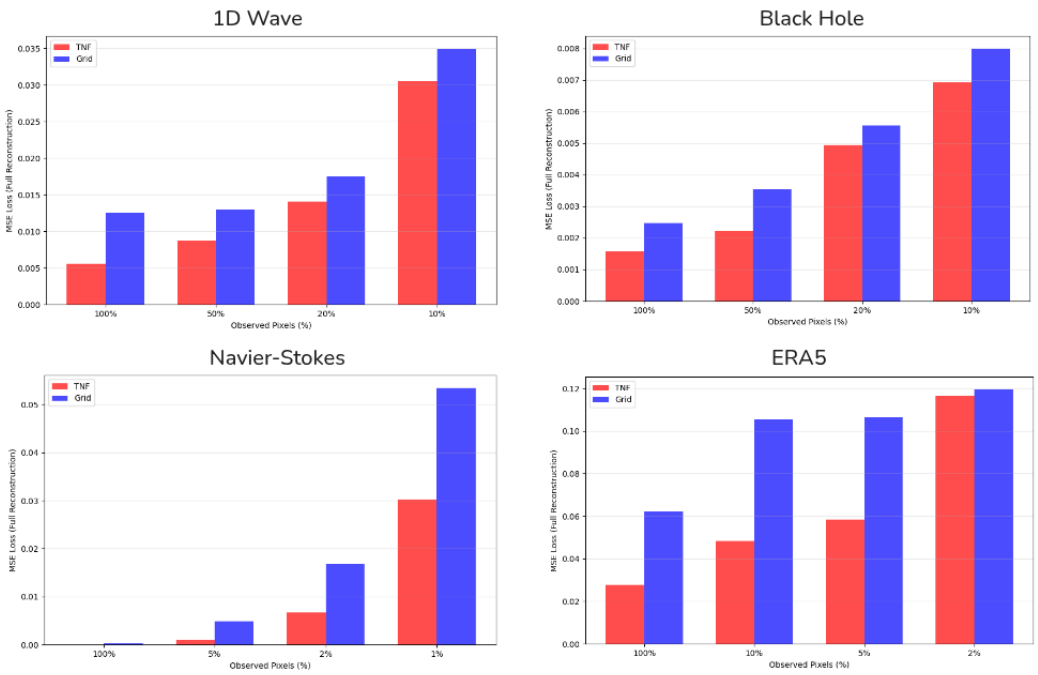


Figure 2-4: Reconstruction error (MSE) across varying observation densities. Comparing TNF (red) against a grid-based baseline (blue) on 1D Wave, Black Hole, Navier-Stokes, and ERA5 datasets. TNF consistently outperforms the grid baseline at all sparsity levels, demonstrating superior robustness when reconstructing complete signals from highly sparse, non-uniform observations.

As illustrated in Figure 2-2, increasing the token capacity (2×8, 4×8, 8×8, and 16×8) consistently improves reconstruction fidelity, demonstrating that TNF provides a controllable tradeoff between compactness and representation accuracy for continuous 1D signals.

Next, we evaluate TNF under variable-density sparse observation conditions, comparing its performance against a grid-based baseline. The target signals are compositions of sinusoidal functions modulated by Gaussian envelopes with random amplitudes, frequencies, and centers, producing localized high-frequency structure over 1000 uniformly spaced points.

Both TNF and the grid baseline use matched token capacities, while differing in how sparse observations are processed. TNF directly optimizes its tokens on scattered observations, whereas the grid baseline first interpolates the sparse subset onto a regular grid before encoding. We employ the same meta-learning framework with 3 inner-loop adaptation steps for both methods. Additional implementation details are

provided in Appendix 2.6.3.

During training, each batch receives a random subset of observed points sampled uniformly from $n_{\text{sub}} \in [50, 1000]$. Both methods are then evaluated on reconstruction of the full underlying signal.

Figure 2-3 presents qualitative comparisons under full and sparse observability, while Figure 2-4 reports quantitative reconstruction error across varying observation densities. Across all sparsity levels, TNF consistently achieves lower reconstruction error than the grid-based baseline.

2.4.2 2D Physical Fields

We next evaluate TNF on two-dimensional physical fields from computational physics simulations: simulated black hole accretion disk images and Navier–Stokes fluid dynamics snapshots. These domains are representative of scientific settings where observations are naturally sparse and scattered. For both experiments, we use the same meta-learning framework with 3 inner-loop adaptation steps. Detailed architectural and optimization settings are provided in Appendix 2.6.3.

Black Hole Imaging

The dataset consists of 30K simulated images of black hole accretion disks [Karras et al., 2025], resized to 28×28 single-channel images. Each image is represented as a continuous signal $f : [-1, 1]^2 \to \mathbb{R}$, where coordinates are generated from a regular meshgrid.

We compare TNF against a grid-based baseline with matched token capacity. For sparse reconstruction, TNF directly optimizes its tokens on scattered observations, while the grid baseline first maps the sparse observations onto the regular image grid using nearest-neighbor interpolation.

Results are presented in Figure 2-4. TNF significantly reduces reconstruction error across observation densities. Figure 2-5 shows qualitative comparisons for full and 10% observation densities. At full observability, TNF recovers more details than the

grid-based baseline. At 10% density ($\sim$ 78 observed pixels), TNF better maintains the characteristic ring structure of the accretion disk.

Navier–Stokes Fluid Dynamics

The dataset consists of 30K vorticity snapshots from Navier–Stokes simulations [Shi et al., 2025a], resized to 28×28 single-channel images. We use the same experimental setup as in the black hole experiment and evaluate reconstruction under more extreme sparsity levels, down to 1% observation density (approximately 8 observed points), in order to probe the limits of reconstruction from minimal observations.

Figure 2-4 and Figure 2-6 present the quantitative and qualitative results. TNF achieves substantially lower reconstruction error across all sparsity levels. At full density, TNF achieves near-perfect reconstruction (MSE $= 1.4 \times 10^{-5}$) compared to the grid baseline (MSE $= 2.9 \times 10^{-4}$). At 1% density, corresponding to only 7 scattered observations, TNF reduces the reconstruction error of the grid baseline by nearly half.

At this extreme sparsity level, the grid baseline must reconstruct the full 28×28 field from a nearest-neighbor interpolation of only a few observations, resulting in blocky and piecewise-constant artifacts. In contrast, TNF leverages its learned continuous prior through token optimization, producing smooth and physically plausible reconstructions even from highly sparse measurements.

2.4.3 ERA5 Weather Reanalysis

To evaluate TNF on real-world scientific data, we use the ERA5 reanalysis dataset [Hersbach et al., 2020] produced by the European Centre for Medium-Range Weather Forecasts (ECMWF). The dataset consists of 34,560 weather snapshots represented on a 41×41 latitude-longitude grid with four meteorological variables corresponding to surface pressure, 2-meter temperature, and the eastward and northward components of 10-meter wind velocity.

Coordinates are centered and scaled, and outputs are normalized separately for

scalar quantities (pressure and temperature) and vector wind components. Wind vectors are normalized by the mean wind speed in order to preserve rotational structure.

We use the same sparse reconstruction framework as in the previous scientific experiments, employing meta-learning with 3 inner-loop adaptation steps. TNF directly optimizes its token representations on sparse scattered observations, while the grid baseline first interpolates sparse measurements onto a regular grid before encoding. Detailed architectural and optimization settings are provided in Appendix 2.6.3.

Figure 2-4 presents the quantitative results. TNF consistently outperforms the grid baseline, reducing reconstruction error by 45–55% at observation densities of 5% and above, with smaller improvements at the extreme 2% sparsity regime.

Figure 2-7 shows qualitative reconstructions of temperature and wind fields. At full observation density, TNF accurately reproduces fine spatial gradients and flow structure, while the grid baseline exhibits visible smoothing artifacts. At 2% density, both methods recover the large-scale spatial patterns, though fine-scale details become increasingly ambiguous.

2.4.4 2D Image Modeling

We evaluate TNF on 2D image reconstruction using the CelebA [Liu et al., 2015] and ImageNette [Howard, 2019] datasets, and compare against recent encoder-based approaches including TransINR, IPC, and ANR, as well as the MAML-based neural field approach of [Tancik et al., 2020a]. In contrast to these methods, TNF directly optimizes compact per-instance token representations while sharing a single coordinate-conditioned network across all images.

For both CelebA and ImageNette, we train on 10K images of resolution 178×178 using the standard auto-decoding setup and evaluate on 100 held-out test images. Detailed architectural and optimization settings are provided in Appendix 2.6.3.

As shown in Figure 2-8, TNF achieves higher PSNR than all baselines while using $4\times$ to $10\times$ fewer representation parameters. Beyond the quantitative gains, TNF reconstructions exhibit strong perceptual quality and sharpness. Even with compact token representations, the model successfully captures high-frequency structures such

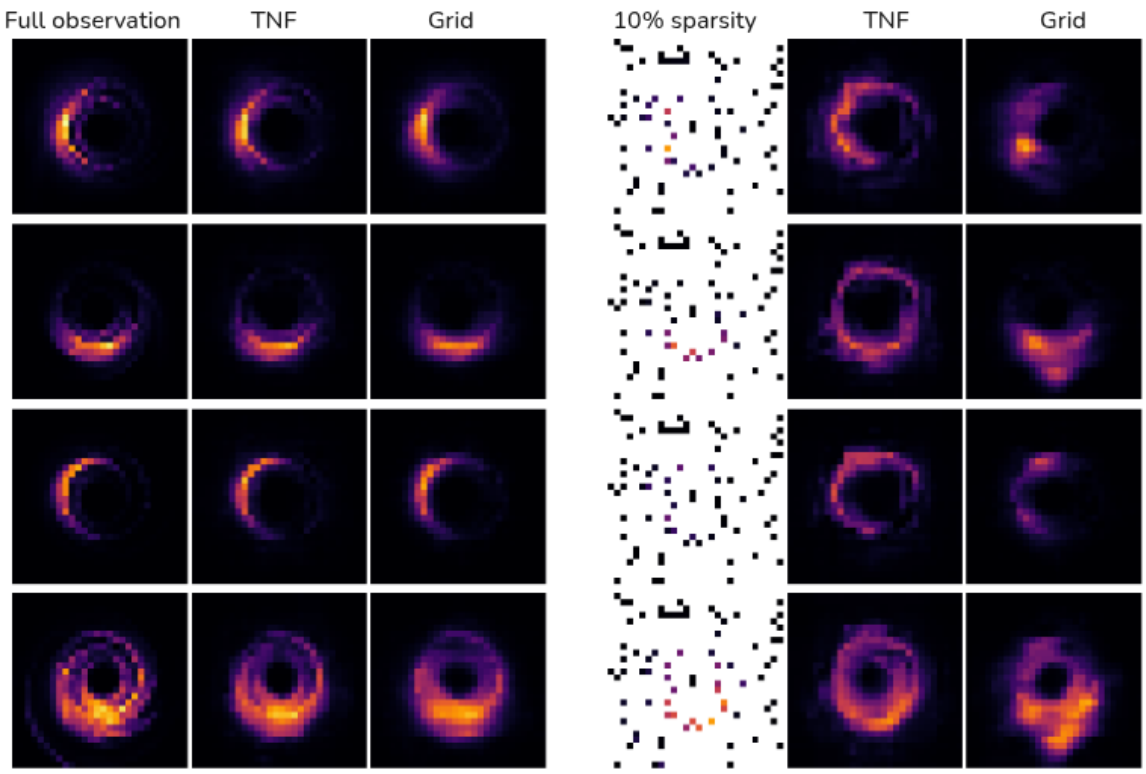


Figure 2-5: Black hole reconstructions from 28×28 simulated images using 16 tokens at different observation densities. Comparison of TNF against a grid-based baseline under full observation and 10% sparse observation (∼78 pixels).

as facial contours, texture, and edges, outperforming significantly larger encoder-based pipelines.

Token specialization and generative modeling. Beyond reconstruction quality, TNF tokens exhibit both structured specialization and strong generative potential. Figure 2-9 visualizes cross-attention heatmaps from a CelebA model trained with 16 tokens. Despite the absence of spatial supervision, individual tokens consistently attend to coherent semantic regions such as hair, background, and facial structure, indicating the emergence of interpretable token specialization. Additional attention visualizations across all tokens and held-out images are provided in Appendix 2.6.3.

To evaluate the expressiveness of the learned token space, we train a Diffusion Transformer (DiT) [Peebles and Xie, 2023a] directly on compact token representations (64×32). Figure 2-10 shows generated samples obtained by decoding the generated

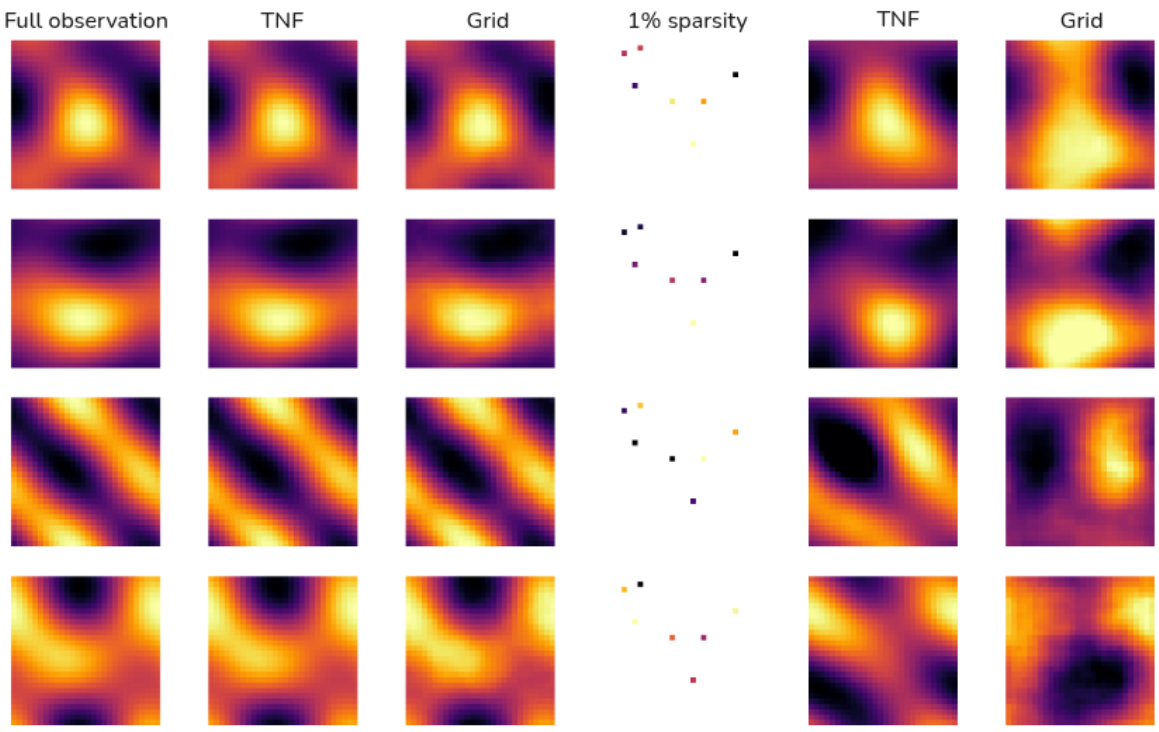


Figure 2-6: Navier–Stokes vorticity reconstructions using 16 tokens at different observation densities. TNF and a grid-based baseline reconstruct 28×28 vorticity snapshots from full and 1% sparse observations (∼8 scattered points). At 1% density the grid baseline produces blocky, piecewise-constant artifacts from nearest-neighbor interpolation, while TNF leverages its learned continuous prior to produce smooth, physically plausible reconstructions.

tokens through the shared TNF network. The resulting images are diverse and coherent, demonstrating that compact token representations form an effective latent space for generative modeling.

To assess memorization, Appendix 2.6.3 compares generated samples with their nearest neighbors in the training set. The clear visual differences indicate that the model synthesizes novel outputs rather than reproducing memorized training examples.

2.4.5 3D Scene Reconstruction

We evaluate TNF on neural scene reconstruction using the car category from the ShapeNet SRN dataset [Sitzmann et al., 2020c]. The dataset contains approximately 3K car scenes, each observed from 250 viewpoints. Each scene is represented by a compact set of learned tokens and modeled through a light field representation [Sitzmann

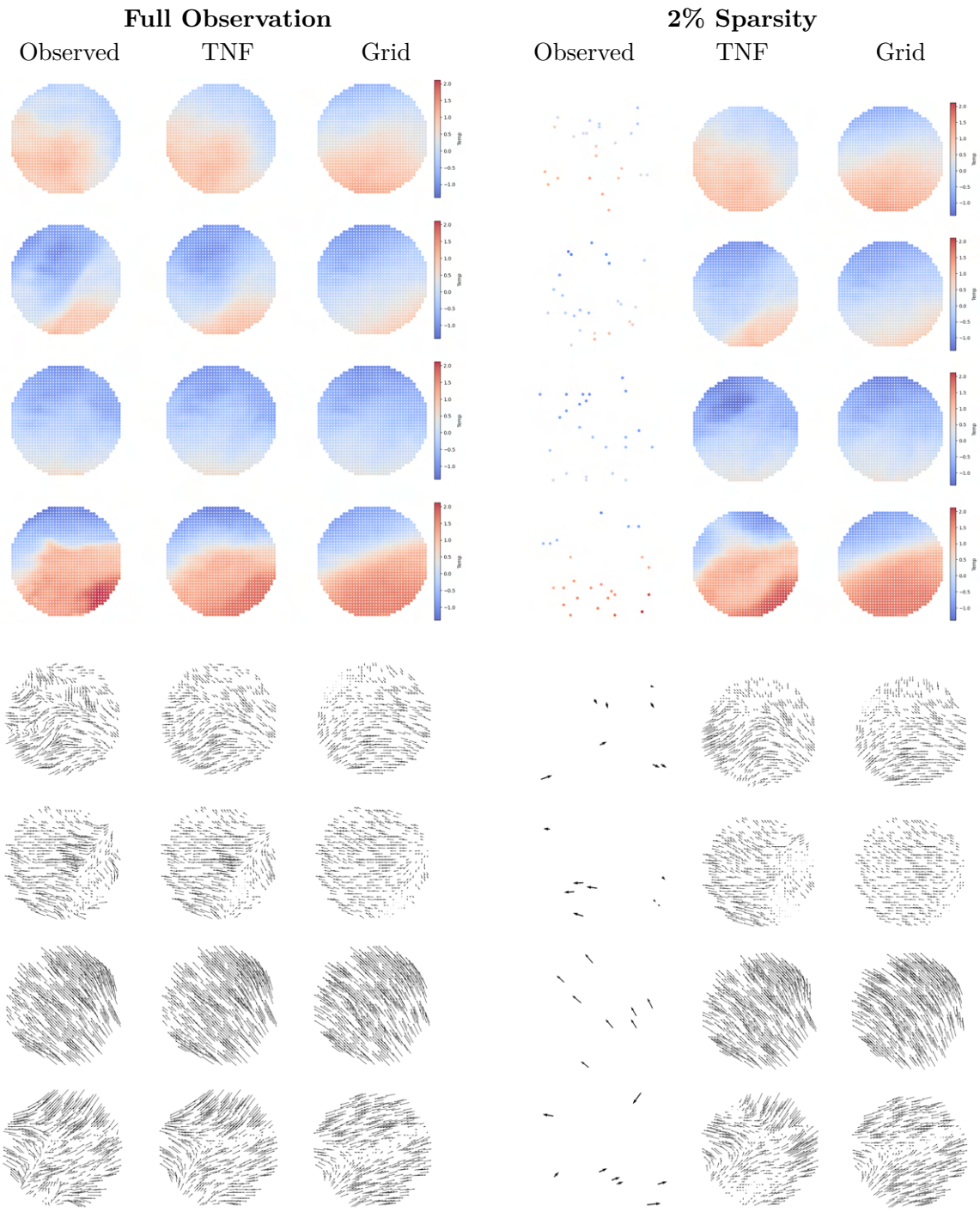


Figure 2-7: ERA5 temperature (top) and wind (bottom) reconstructions on a 41×41 latitude–longitude grid using 20 tokens. Each row represents a different sample, showing the observed input, the TNF reconstruction, and the grid-baseline reconstruction under full observation (left block) and 2% sparse observation (right block). At full density TNF reproduces fine spatial gradients and flow structure while the grid baseline shows visible smoothing; at 2% density both methods recover large-scale patterns and fine-scale details become ambiguous.

Dataset	Method	PSNR	$N \times d_t$	Rep (K)	Model (M)
CelebA	Learned Init	30.37	–	199	0.2
	TransINR	33.33	256×259	66	43
	IPC	35.96	256×256	66	43
	ANR	35.91	512×256	131	54
	TNF	31.5	64×32	2	2
	TNF	**37.02**	512×16	**8**	2
	TNF	**42.14**	512×32	**16**	2
ImageNette	Learned Init	27.07	–	199	0.2
	TransINR	29.77	256×259	66	43
	IPC	38.46	256×256	66	43
	ANR	40.30	512×256	131	54
	TNF	**44.40**	1024×32	**32**	3
	TNF	**47.98**	1024×64	**66**	7



Figure 2-8: Comparison of reconstruction performance. (Left) Quantitative results on CelebA and ImageNette. TNF achieves higher PSNR with much smaller representation and model size compared to prior methods. (Right) Qualitative reconstructions with 512×32 tokens: Left column: CelebA; right column: ImageNette. Image pairs show ground truth (left) vs. TNF output (right).



Figure 2-9: Cross-attention heatmaps for two representative tokens from a 16-token CelebA model. Each row tracks a single token across six different held-out images (columns). Without spatial supervision, individual tokens consistently attend to coherent semantic regions such as the area beneath the eyes and the mouth. Full per-token visualizations across all tokens and more held-out images are provided in Appendix 2.6.3.



Figure 2-10: Unconditional image samples from a Diffusion Transformer (DiT) trained directly on compact (64×32) TNF token sets, decoded back to images through the shared decoder. The resulting samples are diverse and coherent, indicating that the learned token space forms an effective latent for downstream generative modeling.

et al., 2021], where pixels across viewpoints are addressed by their corresponding rays $x = (\mathbf{o}, \mathbf{d}) \in \mathbb{R}^6$, consisting of camera origin and viewing direction.

This formulation represents a 3D scene as a continuous function over ray space. As in the previous domains, tokens summarize the scene while continuous coordinate queries extract signal values through cross-attention. Unlike NeRF-style pipelines, this approach does not require volumetric rendering, explicit geometry, spatial grids, or scene-specific encoders.

The dataset provides RGB images from multiple viewpoints around each object. During training, we randomly sample batches of rays and supervise each query with its corresponding ground-truth pixel color. Detailed architectural and optimization settings are provided in Appendix 2.6.3.

Once optimized, the learned tokens serve as a compact reusable representation of the scene and can be used to reconstruct training views or synthesize novel viewpoints.

Qualitative Results. Figure 2-11 shows both training-view reconstruction and novel-view synthesis results. Despite the compact token representation, TNF produces coherent and detailed renderings across viewpoints. Importantly, all scenes share the same coordinate-conditioned network while differing only in their token representations. These results demonstrate that TNF can model complex 3D appearance and geometry without explicit 3D supervision, mesh representations, or volumetric rendering pipelines.

Token Interpolation and Generative Modeling.

To further analyze the structure of the learned token space, we evaluate both interpolation and generative modeling. Figure 2-12 shows linear interpolation between token sets corresponding to two different scenes. The resulting renderings exhibit smooth transitions in both geometry and appearance, suggesting that the learned token space forms a continuous and semantically meaningful representation manifold.

To assess generative capabilities, we train a Denoising Diffusion Probabilistic Model (DDPM) [Ho et al., 2020] directly on compact scene token representations (64×32). Figure 2-13 presents generated scenes obtained by decoding the generated tokens through the shared TNF network. The resulting samples exhibit coherent multi-view

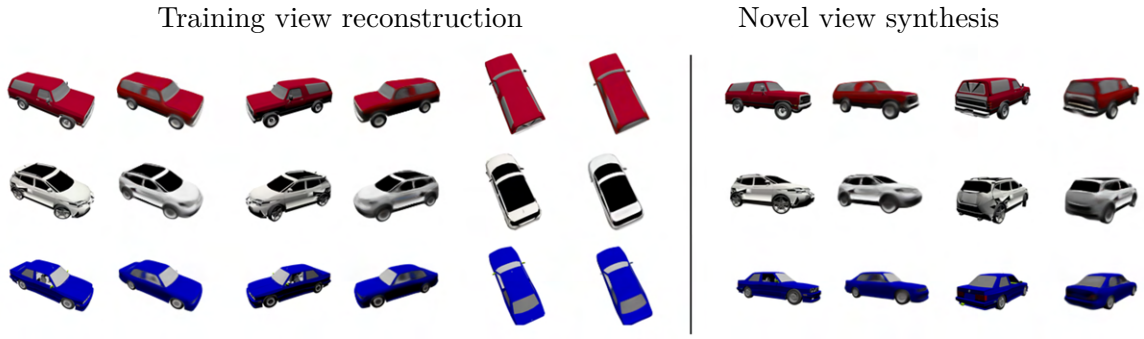


Figure 2-11: 3D reconstruction on the ShapeNet SRN car category. Left block: reconstruction of training viewpoints; right block: novel-view synthesis from held-out viewpoints. In each pair the ground-truth view is on the left and the TNF rendering on the right. All scenes share a single coordinate-conditioned network and differ only in their per-scene tokens, with no volumetric rendering, explicit geometry, or scene-specific encoders.



Figure 2-12: Linear interpolation in token space between two scenes. Shown are two examples (rows) from different viewpoints; in each, the scene on the left smoothly transitions through 10 interpolation steps into a different scene on the right, with gradual changes in geometry and appearance.

structure together with diverse geometry and appearance, demonstrating that TNF tokens provide an effective latent space for generative 3D scene modeling.

Token specialization. Figure 2-14 illustrates that even in 3D, individual tokens tend to focus on distinct semantic parts of the scene. Each row corresponds to a specific token, and columns show different car scenes from the SRN dataset. Notably, despite being trained without supervision, tokens specialize in modeling particular regions (e.g., roof, wheels, or headlights), demonstrating the emergence of structured and interpretable token-space behavior in volumetric domains.

2.5 Conclusion and outlook

We introduced **Tokenized Neural Fields (TNF)**, a unified framework for representing continuous signals through compact sets of learnable tokens optimized against



Figure 2-13: Unconditional 3D scene samples from a Diffusion Transformer (DiT) trained on compact (64×32) TNF scene tokens, decoded through the shared decoder. The samples exhibit coherent multi-view structure together with diverse geometry and appearance, demonstrating that the token space supports generative modeling of 3D scenes.

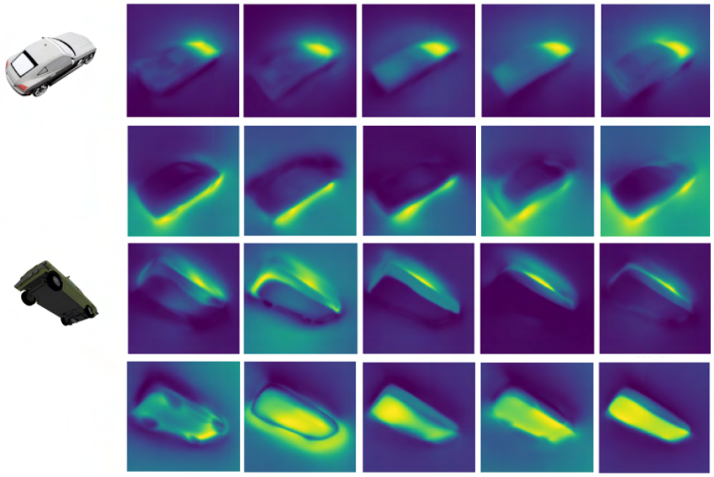


Figure 2-14: Visualization of token specialization in 3D. Each row corresponds to a different token, highlighting the regions of the car it attends to across multiple scenes (columns). The colored images on the left indicate the viewing angles of the corresponding scenes.

a shared coordinate-conditioned decoder. Across 1D functions, 2D physical fields (Navier-Stokes fluid dynamics and simulated black-hole imaging), ERA5 weather reanalysis, 2D natural images, and 3D scenes, TNF achieves high-fidelity reconstruction with substantially fewer parameters than encoder-based or global-latent baselines, and outperforms grid-based tokenization in the sparse-observation regimes that are characteristic of scientific data.

Our experiments highlight three properties: (i) *parameter efficiency*, with strong reconstructions at significantly fewer representation and model parameters than encoder-based pipelines (Figure 2-8). (ii) *architectural decoupling*, where a single shared coordinate-conditioned decoder serves arbitrarily many instances through

compact per-instance tokens. The decoupling separates dataset-level prior information, absorbed by the shared decoder, from per-instance information carried by the tokens. This lets TNF reconstruct signals robustly from highly sparse observations where grid-based methods that have a more constrained prior degrade (Figure 2-4). (iii) *emergent structure*, as tokens specialize to coherent regions of a signal without explicit supervision (Figures 2-9, 2-14), yielding a token space that also serves as an effective latent for downstream generative modeling (Figures 2-10, 2-13).

The framework also inherits limitations: per-instance adaptation requires gradient-based optimization, which introduces inference-time latency relative to encoder-based one-shot inference. While the proposed meta-learning approach offers faster inference, it may still benefit from stronger inductive biases within the gradient-based optimization process through amortized inference strategies (e.g. Marino et al. [2018]).

An important potential direction for scientific applications is *probabilistic inference over the token space*. In many scientific inverse problems the observation alone does not pin down the underlying signal: a sparse set of weather stations is consistent with multiple plausible high-resolution fields, a partially observed fluid state with multiple coherent extrapolations, and a small set of viewpoints with multiple consistent 3D scenes. Treating reconstruction as posterior inference over tokens, by combining the TNF decoder with a learned prior over the token space, would enable posterior sampling, uncertainty quantification, and plausible imputation in under-constrained regimes. Amortized inference or conditional diffusion models trained directly in token space are natural mechanisms for this, and our generative-modeling experiments on natural images and 3D scenes provide first evidence that the token space is well-behaved for such priors. This combination of compact, structured representations with token-space probabilistic inference offers a path towards uncertainty-aware reconstruction and analysis in the sparse and scattered observation regimes that characterize scientific imaging and modeling.

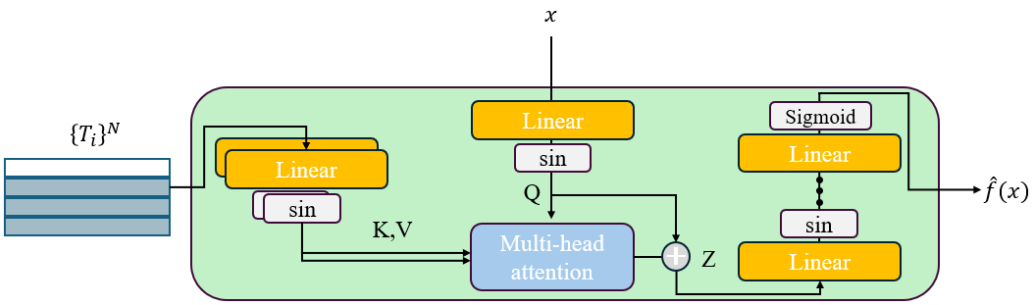


Figure 2-15: TNF: Our token-based coordinate-attention decoder. Each function f is represented by a different set of tokens $\{T_i\}^N$, and can be evaluated at point x using the decoder.

2.6 Appendix

2.6.1 Decoder Implementation

The decoder implementation is depicted in Fig. 2-15.

1D signals. For scalar functions $f : \mathbb{R} \to \mathbb{R}$, the coordinate x is used as a scalar query in a cross-attention module over the token set. The resulting embedding is then decoded through several layers, and projected back to a 1D scalar output. Essentially, the decoder is a mapping between a sequence representation to a continuous signal.

2D signals: fields and images. In order to represent 2D signals such as physical fields or natural images (which can be seen as 2D fields of color), each coordinate $x = (u, v) \in [0, 1]^2$ is used as a query into a cross-attention layer over the signal-specific token set. The resulting attended feature is decoded by an MLP to predict the values. This structure allows the model to generalize across instances while preserving spatial precision via coordinate querying and instance adaptation via tokens.

High dimensional signals: light fields. Light fields [Sitzmann et al., 2021] is a method to represent multiple views of 3D scenes. In this representation, pixels from the different views are represented by a ray $x = (\mathbf{o}, \mathbf{d}) \in \mathbb{R}^6$ (camera origin and direction). Essentially this is a continuous signal over a 6-dimensional space, mapping the ray coordinate to a 3D color value. In our decoder, this ray representation is used to query the token set via cross-attention. To reconstruct an image of a 3D scene from a certain view, we query the model with the light field representation of all the pixels

in the image. For each query, the output of the cross-attention is decoded into a pixel color. This enables reconstructing images of arbitrary views of a 3D scene without using geometry, encoders, or 3D grids. This method aligns closely with the 1D/2D cases, showing that TNF supports fully flexible geometry-free rendering from tokens.

2.6.2 Cross-Attention Querying

Cross-attention enables dynamically extracting relevant content from the token set into different locations in the continuous signal. Given a query coordinate $x \in \mathbb{R}^{d_x}$, we compute:

$$Q = \phi_q(x), \quad Q \in \mathbb{R}^{1\times d_h}$$

$$K = \phi_k(T), \quad K \in \mathbb{R}^{N\times d_h}$$

$$V = \phi_v(T), \quad V \in \mathbb{R}^{N\times d_h}$$

Here, $T \in \mathbb{R}^{N\times d_t}$ is the set of learnable tokens for an instance, d_t is the token dimensionality, d_h is the attention hidden dimension, and N is the number of tokens.

We then compute the scaled dot-product attention:

$$A = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_h}}\right), \quad A \in \mathbb{R}^{1\times N}$$

$$Z = AV + Q, \quad Z \in \mathbb{R}^{1\times d_h}$$

The attention output Z is passed through a small final MLP to produce the final prediction (e.g., RGB value or scalar signal). This querying formulation follows the original Transformer attention mechanism Vaswani et al. [2017a], adapted to continuous-coordinate domains through instance-specific token sets.

2.6.3 Experiment Details and More Results

This appendix provides implementation and optimization details for all experiments presented in the paper. Across all domains, TNF represents each instance using a

Table 2.1: Architecture and optimization details across domains. N_t denotes the number of tokens and d_t the token dimension. CA denotes cross-attention blocks. Unless otherwise stated, all final MLPs use sinusoidal activations. The 3D SRN experiment uses ReLU activations.

Domain	**Tokens**	**Att. Dim**	**Heads**	**CA**	**Token LR**	**Net LR**	**Final MLP**
1D GP	2×8	32	8	1	10^{-3}	10^{-4}	6 layers, hid. 64
1D Sparse Wave	4×8	64	4	2	10^{3}	10^{-4}	2 layers, hid. 64
Black Hole	16×8	64	4	2	10^{3}	10^{-4}	2 layers, hid. 64
Navier–Stokes	16×8	64	4	2	10^{3}	10^{-4}	2 layers, hid. 64
ERA5 Weather	20×8	64	4	2	10^{3}	10^{-4}	2 layers, hid. 64
CelebA	512×16, 512×32	256	8	2	2×10^{-5}	8×10^{-4}	6 layers, hid. 512
ImageNette	1024×32, 1024×64	256	8	2	2×10^{-5}	8×10^{-4}	6 layers, hid. 512
3D SRN Cars	512×32	512	8	8	10^{-3}	10^{-3}	8 layers, hid. 512

compact set of learnable tokens together with a shared coordinate-conditioned decoder. The decoder parameters are shared across all instances, while only the tokens are instance-specific.

Unless otherwise stated, coordinates are projected into an attention space and queried against the token set through cross-attention layers. The attended features are then decoded using multilayer perceptrons with sinusoidal activations. All models are trained using Adam. Table 2.1 summarizes the architectural configurations and optimization settings used across all experimental domains.

1D Gaussian Process Functions

We sample (N=1000) one-dimensional functions from Gaussian processes using either RBF or Matérn kernels. Each function is observed at (M=400) coordinates sampled over the interval ([-2,2]). We use an 80/20 train-test split.

During training, each function is represented by a learnable token set optimized

jointly with the shared decoder. At every iteration, we randomly subsample 100 of the 400 coordinates for supervision. Training is performed for 300 epochs with batch size 4. At test time, the decoder parameters are frozen and only the tokens of held-out functions are optimized for 150 steps.

Sparse Reconstruction of Continuous fields

For the sparse reconstruction experiments (1D Wave, Black Hole, Navier–Stokes, and ERA5), we employ the same meta-learning framework with 3 inner-loop adaptation steps. TNF directly optimizes the token representations on sparse scattered observations, while the grid baseline first interpolates the sparse measurements onto a regular grid before encoding. Detailed per-domain configurations are summarized in Table 2.1.

For all sparse experiments, the decoder consists of coordinate-conditioned cross-attention layers with shared parameters across instances. Observation densities are sampled dynamically during training.

2D Image Modeling

CelebA and ImageNette images are center-cropped and resized to (178×178) RGB images. Detailed architectural and optimization settings are summarized in Table 2.1. Both optimizers use exponential decay with factor 0.999.

DiT for 2D Token Generation

In our 2D generative modeling experiment, we train a DiT model [Peebles and Xie, 2023a] with 28 transformer layers, a hidden size of 1152, and 16 attention heads. The model is optimized using AdamW with learning rate (10^{-4}). The DiT operates directly in the token space, learning to generate compact token representations which are then decoded into images using the shared TNF decoder.

As shown in Figure 2-16, the generated samples (right) and their nearest reconstructions from the training set (left) exhibit clear visual differences, indicating that the model generates novel and diverse outputs rather than memorizing training examples.



Figure 2-16: Generated token samples (right) and their nearest training-set reconstructions (left). The visual differences indicate that the model produces novel and diverse samples rather than memorizing the training data.

Token specialization

To complement the results shown in the main paper, we visualize the full set of token attention maps. Figure 2-17 displays the activation heatmaps of 16 tokens (rows) across 12 test scenes (columns), highlighting how each token consistently attends to specific regions. This reinforces the observation that token specialization emerges naturally even without spatial supervision.

3D scene representation modeling

The ShapeNet SRN car dataset provides 3k scenes of cars, each has 250 images from different camera positions split evenly to train and test. The scenes contain RGB images of size 128×128. During training, we randomly sample a batch of $B = 4096$ rays from all training images per step. Optimization and architectural settings are summarized in Table 2.1. Both optimizers use exponential decay with factor 0.999.

DiT for 3D Token Generation.

For 3D generative modeling, we use the same DiT framework as in 2D but with 12 transformer layers, a hidden size of 768, and 12 attention heads. The model is trained directly in the token space to generate compact 3D scene tokens, which are decoded by our shared decoder. Figure 2-18 compares generated samples with their nearest neighbors in the training set, measured in token space. While neighbors share coarse similarities, clear geometric and appearance differences remain, indicating that

the model synthesizes novel and diverse scenes rather than overfitting to training examples.



Figure 2-17: Full attention heatmaps for 16 tokens (rows) across 12 test scenes (columns). Each token consistently focuses on similar regions across different images, demonstrating spatial specialization consistency.



Figure 2-18: Generated token samples (right) and their nearest training-set neighbors (left). The visual differences indicate the model generates novel and diverse samples rather than memorizing the training data.

Chapter 3

Flow Neural Processes: Efficient Probabilistic Inference via Amortized Probability Flows

Abstract

Neural Processes (NPs) provide a flexible framework for learning distributions over functions from sparse observations. Existing approaches often face a tradeoff between expressive uncertainty modeling and efficient inference. We address this challenge with a unified flow-based framework for Neural Processes. First, we introduce Flow Neural Processes (FlowNP), which formulates conditional function generation through flow matching and learn conditional probability flows over arbitrary context and target sets. FlowNP enables coherent posterior sampling and likelihood estimation while achieving state-of-the-art performance across standard NP benchmarks. However, sample generation requires iterative ODE integration, resulting in increased inference costs. To address this limitation, we introduce Direct Flow Neural Processes (D-FlowNP), a step-amortized extension that learns large integration steps through self-consistency objectives, enabling accurate posterior sampling with substantially fewer function evaluations. We evaluate the proposed framework on synthetic Gaussian processes, image completion, weather forecasting, Navier-Stokes simulations, and black hole imaging. Across domains, FlowNP improves predictive performance and uncertainty modeling, while D-FlowNP preserves sample quality under aggressive reductions in inference cost, enabling efficient one-step and few-step generation. Together, these results establish flow matching as an effective foundation for Neural Processes and demonstrate that inference-step amortization substantially improves their practical applicability.

3.1 Introduction

Many scientific and machine learning problems can be formulated as probabilistic inference over continuous functions. Given sparse observations, the goal is to infer function values at unobserved locations while quantifying predictive uncertainty. This setting arises in applications such as time-series forecasting, weather prediction, image completion, scientific simulation, and spatial interpolation, where practical systems require not only accurate predictions but also calibrated uncertainty estimates, diverse posterior samples, and likelihood evaluation.

Gaussian Processes (GPs) [Rasmussen and Williams, 2006] provide a principled Bayesian framework for probabilistic function inference by defining distributions directly over functions. While GPs naturally provide uncertainty estimates, their computational complexity limits scalability to large datasets. Neural Processes (NPs) [Garnelo et al., 2018c] address this limitation by learning distributions over functions

from data, combining the probabilistic formulation of GPs with the scalability of neural networks. Since their introduction, Neural Processes have become a flexible framework for probabilistic prediction across a wide range of continuous domains.

Recent advances in flow-based generative modeling [Lipman et al., 2022] have demonstrated that continuous probability flows can learn expressive probability distributions while supporting both sampling and likelihood evaluation. Motivated by these developments, we introduce *Flow Neural Processes* (FlowNP), a framework that formulates conditional function inference through context-conditioned probability flows. Rather than parameterizing predictive distributions through latent variables, FlowNP learns the velocity field governing the conditional probability flow, enabling expressive posterior sampling, uncertainty quantification, and likelihood evaluation within a unified framework.

The principal limitation of FlowNP is computational efficiency. Both posterior sampling and likelihood evaluation require numerically integrating the learned probability flow, making inference cost proportional to the number of function evaluations. This challenge is common to flow- and diffusion-based generative models and has motivated extensive research on accelerated inference through improved numerical solvers, consistency learning, and learned trajectory approximations. While these approaches have achieved impressive acceleration for image generation, they have not been explored in the context of Neural Process-based probabilistic function inference.

To address this limitation, we introduce *Direct Flow Neural Processes* (D-FlowNP), a step-amortized extension of FlowNP that learns large integration steps directly. Instead of relying solely on many small ODE steps during inference, D-FlowNP learns to approximate long-range transitions along the probability flow trajectory, substantially reducing the number of required function evaluations while preserving predictive quality. The resulting framework combines the expressive probabilistic modeling capabilities of FlowNP with the computational efficiency required for practical deployment.

We evaluate the proposed framework on synthetic Gaussian processes, image completion, weather forecasting, Navier–Stokes simulations, and black hole imag-

ing. Across standard Neural Process benchmarks, FlowNP achieves state-of-the-art predictive performance while providing expressive posterior sampling and likelihood estimation. D-FlowNP further reduces inference cost by an order of magnitude while maintaining predictive quality, enabling efficient probabilistic inference across diverse continuous prediction tasks.

The contributions of this work are summarized as follows:

- We introduce Flow Neural Processes (FlowNP), a Flow Matching formulation of Neural Processes for expressive probabilistic function inference, supporting posterior sampling, uncertainty quantification, and likelihood evaluation.
- We identify iterative ODE integration as the primary computational bottleneck in flow-based probabilistic inference and propose Direct Flow Neural Processes (D-FlowNP), a step-amortized extension that learns large integration steps to substantially reduce inference cost while preserving predictive performance.
- We provide an extensive empirical evaluation across diverse continuous prediction tasks, demonstrating favorable trade-offs among predictive accuracy, calibration, sample diversity, likelihood estimation, and computational efficiency.

3.2 Background

In this section, we briefly review Neural Processes and Flow Matching.

3.2.1 Neural Processes

Neural processes (NP) [Garnelo et al., 2018a,c], provide a framework for learning distributions over functions from a dataset of sampled functions. Given such a dataset, a model is trained to predict an arbitrary *target* set of points along a function, based on a *context* set of observed points along the same function. More formally, if the functions are defined as $\mathcal{F} = \{f : \mathcal{X} \to \mathcal{Y}\}$, models are trained to predict a conditional

distribution over the target set:

$$p_\theta\left(y^{\mathtt{tgt}}|x^{\mathtt{tgt}}, \{x^{\mathtt{ctx}}, y^{\mathtt{ctx}}\}\right) \tag{3.1}$$

where $x^{\mathtt{ctx}} \in \mathcal{X}^M$ and $x^{\mathtt{tgt}} \in \mathcal{X}^N$ are the positions of the M context points and N target points respectively, and $y^{\mathtt{ctx}} \in \mathcal{Y}^M$ and $y^{\mathtt{tgt}} \in \mathcal{Y}^N$ are the function evaluations of these points $y_i = f(x_i)$. $\mathcal{X}$ and $\mathcal{Y}$ can be of arbitrary dimensions. The context is placed in '{}' brackets for better readability.

This approach of modeling stochastic processes follows from Kolmogorov's extension theorem [Øksendal, 2003] stating that stochastic processes can be defined via a collection of joint distributions over finite sets, $p_{x_{1:n}}(y_{1:n})$, if it meets two conditions: exchangeability and consistency.

Exchangeability. This condition requires that all joint distributions over the sets are equivariant with respect to permutations:

$$p_{x_{1:n}}(y_{1:n}) = p_{\pi(x_{1:n})}(\pi(y_{1:n})) \coloneqq p_{x_{\pi(1),\ldots,\pi(n)}}(y_{\pi(1),\ldots,\pi(n)}), \tag{3.2}$$

for any given permutation π. Models that directly predict conditional distributions such as in Eq. 3.1 should also be invariant to permutations in the context set.

Consistency. This condition requires that joint distributions are consistent with marginalizations:

$$p_{x_{1:m}}(y_{1:m}) = \int p_{x_{1:n}}(y_{1:n}) dy_{m+1:n}, \quad \forall 1 \leq m \leq n. \tag{3.3}$$

For models that directly predict conditional distributions as in Eq. 3.1, this translates to the following formulation of *marginal consistency*:

$$p_\theta\left(y_{1:m}|x_{1:m}, \{x^{\mathtt{ctx}}, y^{\mathtt{ctx}}\}\right) = \int p_\theta\left(y_{1:n}|x_{1:n}, \{x^{\mathtt{ctx}}, y^{\mathtt{ctx}}\}\right) dy_{m+1:n}, \quad \forall 1 \leq m \leq n, \tag{3.4}$$

and additionally, models should hold *conditional consistency*, i.e. the chain rule:

$$p_\theta\left(y_{s^1+s^2}|x_{s^1+s^2},\{\}\right) = p_\theta\left(y_{s^1}|x_{s^1},\{x_{s^2},y_{s^2}\}\right) p_\theta\left(y_{s^2}|x_{s^2},\{\}\right), \tag{3.5}$$

for any two sets of indexes s^1 and s^2. We note that while most NP models follow the exchangeability condition, they do not fully guarantee that the consistency conditions hold (see App. **??**).

3.2.2 Flow Matching

Flow matching [Lipman et al., 2022, Liu et al., 2022, Albergo and Vanden-Eijnden, 2022, Albergo et al., 2023] is a generative modeling paradigm that has recently achieved significant traction. It is closely related to diffusion modeling and its appeal stems from being both conceptually very simple and versatile.

The fundamental object in flow matching is a probability path $p_t(x)$ defined over a continuous time parameter $t \in [0,1]$. This path defines a smooth transition between the two distributions $p_0(x)$ and $p_1(x)$. Most commonly, $p_0(x)$ corresponds to a simple tractable distribution such as a Gaussian, and $p_1(x)$ corresponds to a target distribution which is defined only through samples from the training data such as images.

In its simplest formulation, which we also adopt here, training is performed by generating samples of a conditional probability path $p_t(x_t|x_1)$ computed as an interpolation $tx_1 + (1-t)x_0$ between a data sample x_1 and a noise sample x_0. Given this sample, a model is trained to predict a conditional velocity $u_t(x_t|x_1) = x_1 - x_0$, using a squared loss:

$$\mathcal{L}(\theta) = \mathbb{E}_{t,x_1,x_t}\ \left\|u^\theta(x_t,t) - u_t(x_t|x_1)\right\|^2 \tag{3.6}$$

It can be shown that the expectation over x_1 results in the model approximating the unconditional velocity $u_t(x_t) = \frac{d}{dt}x_t$ defining an ordinary differential equation (ODE) which can be used for both sampling and likelihood estimation.

For sampling, we first generate a sample from the noise $p_0(x_0)$ and then use an

ODE solver to transform it into a sample from the data distribution $p_1(x_1)$. To compute the likelihood of a given sample x_1, we employ the change of variable formula. This involves transforming x_1 back to a corresponding noise sample from the tractable distribution $p_0(x_0)$, and then calculating the likelihood of x_0 while correcting for the flow's volume change with an estimate of the Jacobian trace.

3.3 Related Work

3.3.1 Neural Processes

Neural Processes (NPs) [Garnelo et al., 2018c] were introduced as scalable neural counterparts to Gaussian Processes, combining probabilistic function inference with amortized neural prediction. Given a set of context observations, NPs learn a conditional distribution over target function values, enabling uncertainty-aware prediction across families of related tasks. Conditional Neural Processes [Garnelo et al., 2018a] simplify the latent-variable formulation by directly predicting target values conditioned on the observed context, while Attentive Neural Processes [Kim et al., 2019b] improve context utilization through attention mechanisms. Subsequent work has further improved Neural Processes through convolutional architectures [Gordon et al., 2019], Gaussian predictive models [Bruinsma et al., 2021], autoregressive formulations [Bruinsma et al., 2023], and Transformer-based architectures [Nguyen and Grover, 2023], substantially improving predictive accuracy, scalability, and uncertainty estimation.

More recently, expressive generative models have been incorporated into the Neural Process framework. Bootstrapping Neural Processes [Lee et al., 2020] improve uncertainty estimation through resampling-based training, while Neural Diffusion Processes [Dutordoir et al., 2023] and Diffusion-Augmented Neural Processes [Bonito et al., 2023] introduce diffusion-based generative modeling for function-space prediction. These methods demonstrate the benefits of combining Neural Processes with modern generative models. Our work follows this direction, but replaces diffusion dynamics with conditional probability flows and introduces a step-amortized formulation for

efficient inference.

3.3.2 Probabilistic Modeling of Functions

Probabilistic inference over functions has traditionally been dominated by Gaussian Processes [Rasmussen and Williams, 2006], which define Bayesian distributions directly over continuous functions. Although highly expressive and well-calibrated, exact Gaussian Process inference scales poorly with the number of observations, motivating neural approaches that amortize inference across tasks. Neural Processes can be viewed as learned, data-driven analogues of Gaussian Processes that replace hand-designed kernels with flexible neural representations.

Beyond Neural Processes, several works have explored generative modeling directly in function space. These include neural fields and implicit neural representations [Dupont et al., 2021, Sitzmann et al., 2020a, Tancik et al., 2020b], as well as infinite-dimensional diffusion models [Kerrigan et al., 2022, Pidstrigach et al., 2023, Bond-Taylor and Willcocks, 2023] that extend score-based generative modeling to continuous functions. Functional Flow Matching [Kerrigan et al., 2024] similarly studies flow-based generative modeling over function spaces. Unlike these approaches, our work focuses on conditional probabilistic inference, where predictions are conditioned on arbitrary context observations rather than generated unconditionally.

3.3.3 Flow-Based Generative Modeling

Flow-based generative models learn continuous transformations that transport samples from simple reference distributions to complex data distributions. Early normalizing flows [Rezende and Mohamed, 2015, Durkan et al., 2019] achieved exact likelihood evaluation through invertible transformations. More recently, continuous-time formulations have become increasingly popular through Flow Matching [Lipman et al., 2022], Rectified Flow [Liu et al., 2022], stochastic interpolants [Albergo and Vanden-Eijnden, 2022, Albergo et al., 2023], and related probability transport formulations. These methods learn velocity fields that define probability flows while naturally supporting

both sampling and likelihood computation.

Our work builds on Flow Matching by learning context-conditioned probability flows for Neural Processes. Unlike previous flow-based generative models, which primarily focus on targets with fixed dimensions, our objective is conditional probabilistic inference of arbitrary dimensions applicable to function inference from sparse observations.

3.3.4 Efficient Generative Inference

A major limitation of both diffusion and flow-based generative models is the computational cost of inference. Generating high-quality samples typically requires solving an iterative denoising or probability-flow process involving many network evaluations. Consequently, considerable research has focused on accelerating inference.

One line of work improves numerical integration through higher-order solvers, multistep methods, and optimized sampling schedules [Lu et al., 2022, 2025, Zhao et al., 2023, Frankel et al., 2025, Stancevic et al., 2025, Williams et al., 2024, Sabour et al., 2024]. A second line modifies the training objective to enable fewer inference steps. Representative examples include Consistency Models [Song et al., 2023] and their extensions [Luo et al., 2023, Kim et al., 2024, Heek et al., 2024], as well as Rectified Flow [Liu et al., 2022], which reduces trajectory curvature to improve the accuracy of coarse numerical integration.

A complementary direction amortizes the ODE solver itself. More broadly, this follows the general machine learning principle of replacing an expensive iterative computation with a learned direct predictor, an idea that has previously appeared in sparse-coding optimization [Gregor and LeCun, 2010] and discriminatively trained image priors [Rosenbaum and Weiss, 2015]. Rather than relying on many small numerical integration steps, these methods learn transition operators that directly approximate the effect of larger integration intervals.

Such approaches are increasingly unified under the notion of *flow maps* [Boffi et al., 2024, Sabour et al., 2025], which learn direct mappings between different points along the same probability-flow trajectory rather than only the instantaneous velocity,

enabling few-step or one-step generation. Existing flow-map methods differ primarily in the objectives used to learn these transition operators. Shortcut Models [Frans et al., 2025], which our work builds upon, condition the network on a desired jump size Δt and enforce a self-consistency objective in which a single large step is trained to match two consecutive smaller steps, effectively amortizing numerical integration. In contrast, MeanFlow [Geng et al., 2025, Zhang et al., 2025] learns an average-velocity field and relates it to the instantaneous velocity, while other flow-map formulations enforce analogous consistency constraints between learned transition operators and the underlying velocity field [Boffi et al., 2024, Sabour et al., 2025].

Our proposed Direct Flow Neural Processes extend this shortcut-based perspective to context-conditioned probability flows for Neural Processes, substantially reducing the number of function evaluations required for posterior sampling while preserving predictive quality.

3.4 Flow Neural Processes

Our goal is to model conditional distributions over arbitrary context and target sets, as defined in Eq. 3.1. We achieve this by learning a conditional probability flow over the target values conditioned on the observed context.

3.4.1 The FlowNP Model

FlowNP is implemented using a Transformer [Vaswani et al., 2017b] that predicts the conditional velocity field along the probability flow. The model operates on two types of tokens: context tokens representing observed function values and target tokens representing intermediate target values along the flow (Fig. 3-1).

Target tokens. Each target token represents an intermediate target value together with its spatial location and flow time:

$$\texttt{token}^{\texttt{tgt}_i} = \texttt{embed}([x^{\texttt{tgt}_i},\, t,\, y_t^{\texttt{tgt}_i}]). \tag{3.7}$$

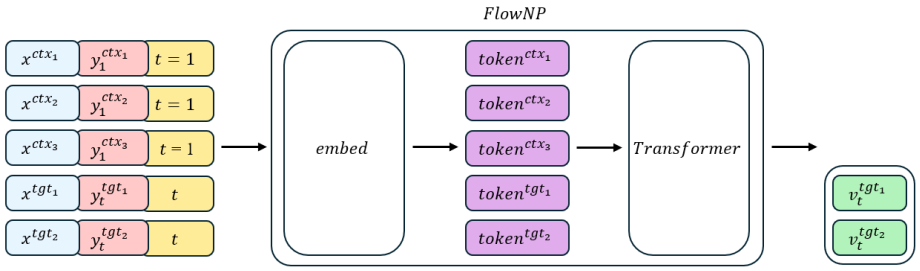


Figure 3-1: FlowNP models the probability flow over target values conditioned on observed context points. Context tokens remain fixed at the data distribution ($t = 1$), while target tokens evolve along the probability flow. At each integration step, the model predicts the target velocity field conditioned on the context.

Context tokens. Context tokens are constructed identically, except that they always contain the observed function value and correspond to the data distribution ($t = 1$):

$$\texttt{token}^{\texttt{ctx}_i} = \texttt{embed}([x^{\texttt{ctx}_i},\, 1,\, y^{\texttt{ctx}_i}]). \tag{3.8}$$

The output corresponding to each target token is projected to $\dim(\mathcal{Y})$ and interpreted as the predicted velocity field.

Training

Training follows the standard Neural Process protocol. For each function, random context and target sets are sampled, and target values are interpolated with Gaussian noise to obtain intermediate states along the probability flow:

$$y_t^{\texttt{tgt}} = t\, y_1^{\texttt{tgt}} + (1-t) y_0^{\texttt{tgt}}, \qquad y_0^{\texttt{tgt}} \sim \mathcal{N}(0, I). \tag{3.9}$$

The model is trained to predict the conditional flow velocity $v_t(y_t^{\texttt{tgt}} | y_1^{\texttt{tgt}}) = y_1^{\texttt{tgt}} - y_0^{\texttt{tgt}}$ using the objective

$$\mathcal{L}(\theta) = \mathbb{E}\left[\left\| v^\theta(y_t^{\texttt{tgt}}, t, x^{\texttt{tgt}}, x^{\texttt{ctx}}, y_1^{\texttt{ctx}}) - (y_1^{\texttt{tgt}} - y_0^{\texttt{tgt}}) \right\|^2 \right]. \tag{3.10}$$

The expectation is taken over

$$f \sim \mathcal{F},\ \{x^{\texttt{tgt}}, y_1^{\texttt{tgt}}, x^{\texttt{ctx}}, y_1^{\texttt{ctx}}\} \sim f,\ t \sim \mathcal{U},\ y_0^{\texttt{tgt}} \sim \mathcal{N}.$$

The complete training procedure is given in Alg. 2.

Inference

After training, FlowNP supports both posterior sampling and likelihood evaluation.

For sampling, target values are initialized from $y_0^{\texttt{tgt}} \sim \mathcal{N}(0, I)$ and evolved from $t = 0$ to $t = 1$ using a numerical ODE solver. At each solver step, the velocity is predicted as

$$v_t \approx v^\theta(y_t^{\texttt{tgt}}, t, x^{\texttt{tgt}}, x^{\texttt{ctx}}, y_1^{\texttt{ctx}}).$$

The sampling algorithm is provided in Alg. 3.

Likelihoods are computed by integrating the probability flow in the reverse direction, mapping observations to the base Gaussian distribution. The resulting likelihood is evaluated using the change-of-variables formula, where the Jacobian trace is estimated with the Hutchinson estimator.

Running time. Compared to TNP, FlowNP uses only $N+M$ tokens instead of $2N+M$ because it does not require causal masking. Sampling complexity is independent of the number of target points and depends only on the chosen ODE solver, whereas TNP requires N autoregressive evaluations. The main computational disadvantage of FlowNP is likelihood evaluation, which requires multiple ODE evaluations rather than a single forward pass. Wall-clock comparisons are reported in Fig. 3-2 and Fig. 3-3.

Exchangeability and Consistency

FlowNP is permutation invariant with respect to both the context and target sets, satisfying the exchangeability requirement described in Sec. 3.2.1. This follows directly from the Transformer architecture, which processes tokens as an unordered set and does not rely on sequence-dependent positional encodings.

Consistency, however, is not guaranteed by the architecture itself. Nevertheless, the Neural Process training procedure encourages both marginal and conditional consistency by repeatedly sampling context and target sets from the same underlying function, making the training objective a Monte Carlo approximation of the underlying stochastic process.

As with previous Neural Process models, this guarantee remains approximate. CNP, NP, and TNP satisfy marginal consistency (Eq. 3.4) but not conditional consistency (Eq. 3.5), whereas NDP exhibits the opposite behavior. Additional discussion is provided in Appendix 3.7.1.

Detailed implementation specifications for the shared FlowNP and D-FlowNP backbone are provided in Appendix 3.7.5.

3.4.2 Experiments

Following the standard evaluation protocol of NP models, we compare conditional sampling and conditional likelihood across three domains: synthetic 1D Gaussian processes, image completion on EMNIST Cohen et al. [2017] and CelebA Liu et al. [2018], and weather prediction on ERA5 Hersbach et al. [2020].

Unless stated otherwise, all experiments use the same FlowNP architecture, adapting only the input $\texttt{dim}(\mathcal{X})$ and output $\texttt{dim}(\mathcal{Y})$ dimensions. The model consists of a 6-layer Transformer with 128 hidden dimensions and 4 attention heads. We use sinusoidal encodings of the input coordinates x and flow time t with 10 frequencies per dimension (40 for ERA5), where the encoding depends on the coordinates rather than their order in the sequence. Sampling is performed using a 100-step Euler solver, while likelihoods are computed with the 100-step midpoint solver of Lipman et al. [2024]. Training and sampling algorithms are provided in Alg. 2 and Alg. 3 in the appendix. Our implementation is based on the Transformer architecture of Nguyen

and Grover [2023], and all experiments are performed on a single NVIDIA RTX4090 GPU.

The appendix includes additional CelebA experiments using a larger model, as well as a modified sampling procedure that injects small amounts of noise during sampling.

Baselines. We compare FlowNP with CNP [Garnelo et al., 2018a], NP [Garnelo et al., 2018c], ANP [Kim et al., 2019b], TNP [Nguyen and Grover, 2023], and NDP [Dutordoir et al., 2023], using the implementations of Nguyen and Grover [2023] and Lee et al. [2020]. To isolate differences in probabilistic modeling rather than architecture, we implement FlowNP, TNP, and NDP using the same Transformer backbone. TNP differs only in its autoregressive maximum-likelihood objective and causal masking, while NDP is trained on unconditional joint distributions using a linear variance-preserving diffusion objective that predicts denoised samples rather than flow velocities. For a fair comparison, our NDP implementation omits the bidirectional attention used in the original model and performs conditional sampling via guidance. Likelihoods for both NDP and FlowNP are computed using the same probability flow ODE solver. On the Fixed-Noisy RBF and Fixed-Noisy Matérn benchmarks, our NDP implementation achieves higher log-likelihood than reported in the original paper.

Synthetic 1D functions We first evaluate FlowNP on the standard synthetic 1D Gaussian Process (GP) benchmarks used by TNP Nguyen and Grover [2023] and NDP Dutordoir et al. [2023], following the evaluation protocols of Lee et al. [2020] and Bruinsma et al. [2021]. The first protocol consists of RBF (squared exponential), Matérn-$\frac{5}{2}$, and Periodic kernels with randomly sampled hyperparameters. Evaluation is performed on held-out functions using a random number of context and target points between 3 and 47, with at most 50 points in total. The second protocol uses fixed RBF and Matérn-$\frac{5}{2}$ kernels with Gaussian observation noise ($\sigma = 0.05$), a random context size between 1 and 10, and 50 target points. We refer to these datasets as Fixed-Noisy RBF and Fixed-Noisy Matérn-$\frac{5}{2}$. Likelihoods for NP and ANP are estimated using variational importance sampling Burda et al. [2015] with 200 samples, while NDP and

Table 3.1: Comparison of log-likelihood computed on the target set for various 1D GP datasets. We provide mean and standard deviation over five runs. FlowNP consistently achieves state-of-the-art performance, outperforming or matching all other models across all datasets.

Model	RBF	Matern-$\frac{5}{2}$	Periodic	fixed-noisy RBF	fixed-noisy Matérn-$\frac{5}{2}$
CNP	0.31 ± 0.04	0.12 ± 0.02	-0.63 ± 0.01	-1.00 ± 0.02	-1.09 ± 0.01
NP	0.31 ± 0.04	0.13 ± 0.05	-0.61 ± 0.01	-1.13 ± 0.02	-1.18 ± 0.01
ANP	1.10 ± 0.04	0.85 ± 0.04	-0.89 ± 0.04	-0.87 ± 0.03	-0.98 ± 0.02
TNP	1.65 ± 0.02	1.29 ± 0.02	-0.58 ± 0.01	0.68 ± 0.01	0.30 ± 0.01
NDP	1.20 ± 0.04	0.94 ± 0.03	-0.54 ± 0.01	0.71 ± 0.01	0.27 ± 0.02
FlowNP (ours)	1.69 ± 0.02	1.30 ± 0.02	-0.50 ± 0.01	0.71 ± 0.01	0.30 ± 0.01

Table 3.2: Target log-likelihood (mean ± standard deviation) on the EMNIST, CelebA, and ERA5 datasets. FlowNP consistently achieves the best performance.

Model	EMNIST 0-9	EMNIST 10-46	CelebA	ERA5
CNP	1.27 ± 0.04	0.73 ± 0.04	2.10 ± 0.01	4.06 ± 0.06
NP	1.17 ± 0.03	0.92 ± 0.02	2.53 ± 0.03	3.35 ± 0.08
ANP	1.35 ± 0.02	1.18 ± 0.01	3.24 ± 0.01	7.76 ± 0.08
TNP	2.08 ± 0.02	1.80 ± 0.05	3.95 ± 0.01	11.32 ± 0.09
NDP	1.58 ± 0.02	1.47 ± 0.01	4.28 ± 0.04	6.76 ± 0.07
FlowNP (ours)	2.50 ± 0.01	2.42 ± 0.01	6.37 ± 0.01	12.79 ± 0.03

FlowNP use the 100-step midpoint ODE solver.

Table 3.1 reports the mean and standard deviation over five runs with different random seeds and evaluation sets. All baselines outperform their originally reported results Nguyen and Grover [2023], Bruinsma et al. [2021] due to our hyperparameter tuning (Appendix 3.7.2). FlowNP consistently achieves the best or tied-best log-likelihood across all benchmarks.

Figure 3-2 provides qualitative comparisons on the fixed-parameter RBF and Matérn benchmarks. ANP primarily represents uncertainty through local predictive variance, whereas TNP, NDP, and FlowNP generate coherent posterior functions. Compared to TNP, FlowNP generates all target points in parallel, producing smoother samples with lower inference time. Compared to NDP, FlowNP performs conditional sampling directly without guidance. Figure 3-15 in the appendix visualizes the sampling trajectory.



Figure 3-2: Samples from models trained on RBF (left) and Matérn-$\frac{5}{2}$ (right) kernels. ANP captures uncertainty mainly through local predictive variance (shaded area), whereas TNP, NDP, and FlowNP generate coherent posterior samples that capture global uncertainty. Unlike NDP, FlowNP performs conditional sampling directly without guidance. Unlike the autoregressive TNP, FlowNP generates all target points in parallel, resulting in smoother and faster sampling.

Images. We next evaluate FlowNP on EMNIST Cohen et al. [2017] and CelebA Liu et al. [2018], where the inputs $\mathcal{X}$ are 2D pixel coordinates and the outputs $\mathcal{Y}$ are grayscale or RGB pixel values. Models are trained from randomly sampled subsets of image pixels. For EMNIST, we train on the digit classes (0–9) and evaluate both in-distribution and on the remaining character classes. Following Lienen et al. [2025], Gaussian noise with variance 0.01^2 is added to the images to avoid arbitrarily large likelihoods for continuous models.

Table 3.2 shows that FlowNP achieves the highest log-likelihood on both EMNIST and CelebA, including the out-of-distribution EMNIST evaluation. Figure 3-3 com-

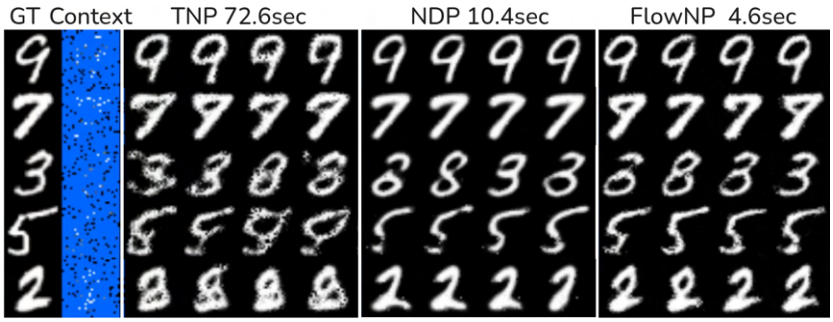


Figure 3-3: Conditional EMNIST samples generated by TNP, NDP, and FlowNP from 70 observed pixels. FlowNP produces sharp and diverse samples while requiring fewer function evaluations than TNP and no guidance for conditional generation, unlike NDP.

pares conditional samples using 70 observed pixels. FlowNP generates sharp and diverse samples while requiring approximately seven times fewer function evaluations than TNP and avoiding the guidance required by NDP, resulting in faster sampling.

Weather data. To assess the performance of the FlowNP model in real-world dynamic and multi-dimensional tasks, we use meteorological data from the ERA5 global dataset Hersbach et al. [2020]. We follow the setup in Holderrieth et al. [2021] and extract data from a circular region with a 520km radius centered around Memphis, USA. Each sample represents a weather snapshot at a single point in time during the winter months from 1980 to 2018 and consists of 1245 grid points of multiple meteorological variables: temperature, pressure, and two wind components (eastward and northward). We split the data into 34K training samples and 17.5K evaluation samples. Training is performed using the NP setup where $\mathcal{X}$ represents the 2D longitude and latitude, $\mathcal{Y}$ represents the 4D meteorological variables, and random subsets out of the 1245 points are used as the context and target sets.

The performance of FlowNP and the baselines are compared in Table 3.2, where the target set log-likelihood is reported for each model. FlowNP outperforms all other models.

For a qualitative evaluation, Figure 3-4 provides conditional samples of temperature and wind directions generated by FlowNP on the ERA5 weather data. The figure demonstrates how prediction quality scales with the size of the context set: the top

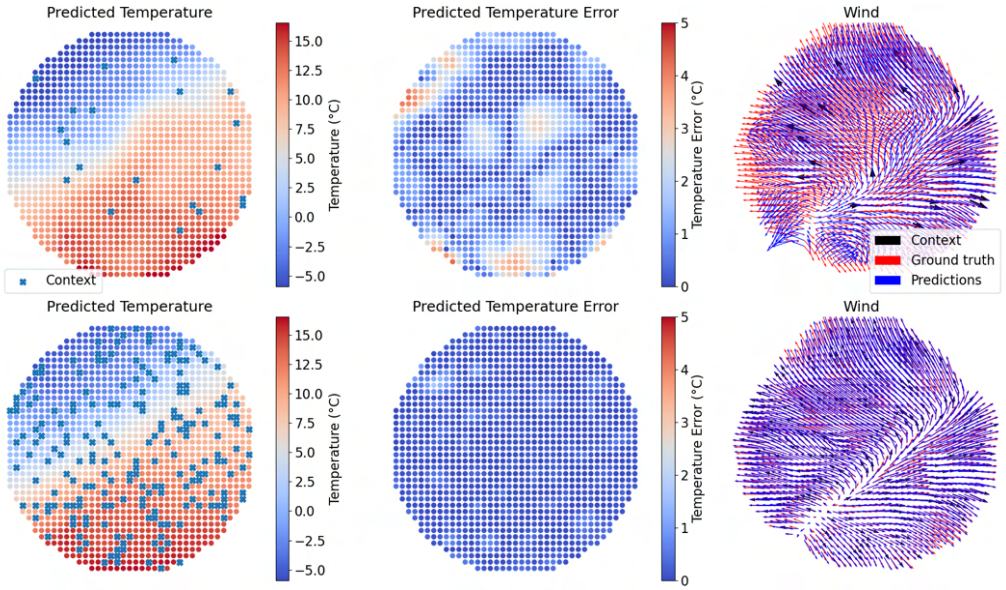


Figure 3-4: Visualization of temperature and wind prediction using conditional samples generated by FlowNP on two held-out data points from ERA5. The model generates coherent samples based on the context points. As more context points are given (bottom row) predictions become more accurate.

row, using fewer context points, shows predictions with a greater degree of error compared to the bottom row, which is conditioned on a larger context set.

3.4.3 Analysis

We provide analysis of FlowNP compared to the autoregressive approach of TNP and on different differentiating aspects compared to NDP. For more analysis on running times, different ODE solvers and the number of ODE steps for log-likelihood computation see Appendix 3.7.3.

FlowNP vs. TNP We consider the 1D step function, where the output changes from $y = 0$ to $y = 1$ at a random point of x. This function was used in Neal [1998] and Dutordoir et al. [2023] as an example that cannot be captured by Gaussian processes. Here we show that an autoregressive transformer-based model like TNP also cannot capture this function accurately. In Fig. 3-5 we show samples from TNP and FlowNP. While TNP samples are noisy in the transition area, FlowNP samples are sharp. The reason is that even though TNP can model complex distributions via

the autoregression, it is constrained to Gaussians at every step. This is evident when looking at the marginal distribution at $x = 0$, shown on the right of Fig. 3-5 for both models. Comparing the marginal distribution of a single point with samples from the entire function indicates that, even without a formal guarantee, FlowNP's training approximately preserves the consistency property (Eq. 3.4).

FlowNP vs. NDP We conduct an ablation study, summarized in Table 3.3, to investigate three key aspects differentiating FlowNP from the NDP baseline:

1. The output of the network at each step (clean target data y_1, noise y_0, or flow velocity $y_1 - y_0$)
2. The noise schedule used during training $y_t = \alpha_t y_1 + \beta_t y_0$
3. Amortizing context conditioning vs. modeling joint distributions only and computing conditional likelihoods through the joint/context ratio:

$$p\left(y^{\texttt{tgt}} | x^{\texttt{tgt}}, \{x^{\texttt{ctx}}, y^{\texttt{ctx}}\}\right) = p_\theta\left(y^{\texttt{tgt+ctx}} | x^{\texttt{tgt+ctx}}\right) / p_\theta\left(y^{\texttt{ctx}} | x^{\texttt{ctx}}\right)$$

Our results show that using the flow velocity as the network output and conditional optimal-transport noise schedule (cond-ot) yields superior performance across the models. We note that training FlowNP only on joint distributions (flow:joint) and using it to compute conditional likelihood through the joint/context ratio, achieves a slightly higher likelihood than a FlowNP model that is trained conditioned on contexts directly. This is perhaps expected as a model trained unconditionally dedicates its entire capacity to modeling the joint data distribution, a less demanding task than amortizing the conditioning on all possible context sets. However, this slightly higher likelihood comes at a cost of requiring auxiliary methods to generate conditional samples. In contrast, conditional training of FlowNP provides slightly lower maximum log-likelihood but is inherently capable of direct, auxiliary-free conditional sampling.

Towards efficient inference. The experiments above demonstrate that FlowNP provides expressive posterior modeling and strong predictive performance across a

Table 3.3: Target log-likelihood for ablations on three aspects differentiating FlowNP from NDP.

Model	network output	noise schedule	conditioning	RBF	EMNIST
NDP	clean: y_1	linear-vp: $\alpha_t = t, \beta_t = \sqrt{1-t^2}$	unconditional	1.20±0.04	1.58±0.02
diffusion:clean	clean: y_1	linear-vp: $\alpha_t = t, \beta_t = \sqrt{1-t^2}$	conditional	1.38±0.01	1.64±0.01
diffusion:noise	noise: y_0	linear-vp: $\alpha_t = t, \beta_t = \sqrt{1-t^2}$	conditional	1.33±0.04	1.56±0.01
flow:lin-vp	velocity: $y_1 - y_0$	linear-vp: $\alpha_t = t, \beta_t = \sqrt{1-t^2}$	conditional	0.41±0.02	0.48±0.01
flow:poly2	velocity: $y_1 - y_0$	polynomial-2: $\alpha_t = t^2, \beta_t = (1-t)^2$	conditional	1.08±0.02	1.60±0.02
flow:cosine	velocity: $y_1 - y_0$	cosine: $\alpha_t = \sin(\frac{1}{2}\pi t), \beta_t = \cos(\frac{1}{2}\pi t)$	conditional	1.22±0.02	1.80±0.01
flow:joint	velocity: $y_1 - y_0$	cond-ot: $\alpha_t = t, \beta_t = 1-t$	unconditional	1.73±0.04	2.54±0.03
FlowNP	velocity: $y_1 - y_0$	cond-ot: $\alpha_t = t, \beta_t = 1-t$	conditional	1.69±0.02	2.50±0.01

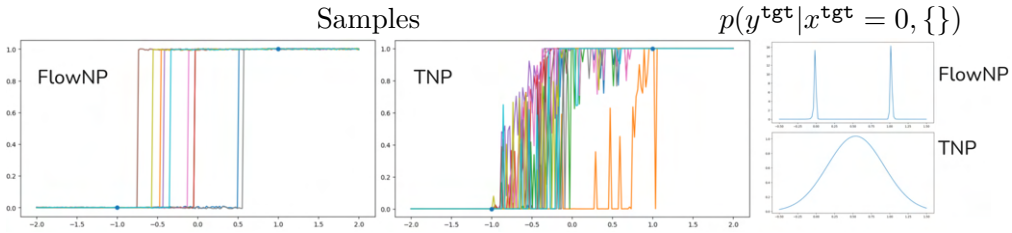


Figure 3-5: FlowNP vs. TNP for a random step function. FlowNP samples capture the sharp transition occurring in random positions, while TNP cannot model this function as each step in the autoregressive prediction is Gaussian. On the right: the marginal distribution of y predicted for a single point where $x = 0$ further demonstrates this and highlights FlowNP's capacity to capture multimodal distributions.

diverse range of Neural Process benchmarks. However, these gains come at the cost of iterative inference. Both posterior sampling and likelihood evaluation require numerically solving the probability flow ODE through multiple network evaluations, making inference substantially more expensive than direct prediction methods. While this cost is acceptable for many applications, it becomes a bottleneck in scenarios that require repeated posterior queries, such as Bayesian optimization, active learning, or uncertainty-aware decision making.

To address this limitation, we next introduce *Direct Flow Neural Processes (D-FlowNP)* in Sec. 3.5, which amortizes the integration process by learning large transitions along the probability flow trajectory. This substantially reduces the number of function evaluations required during inference while preserving the expressive posterior modeling capabilities of FlowNP.

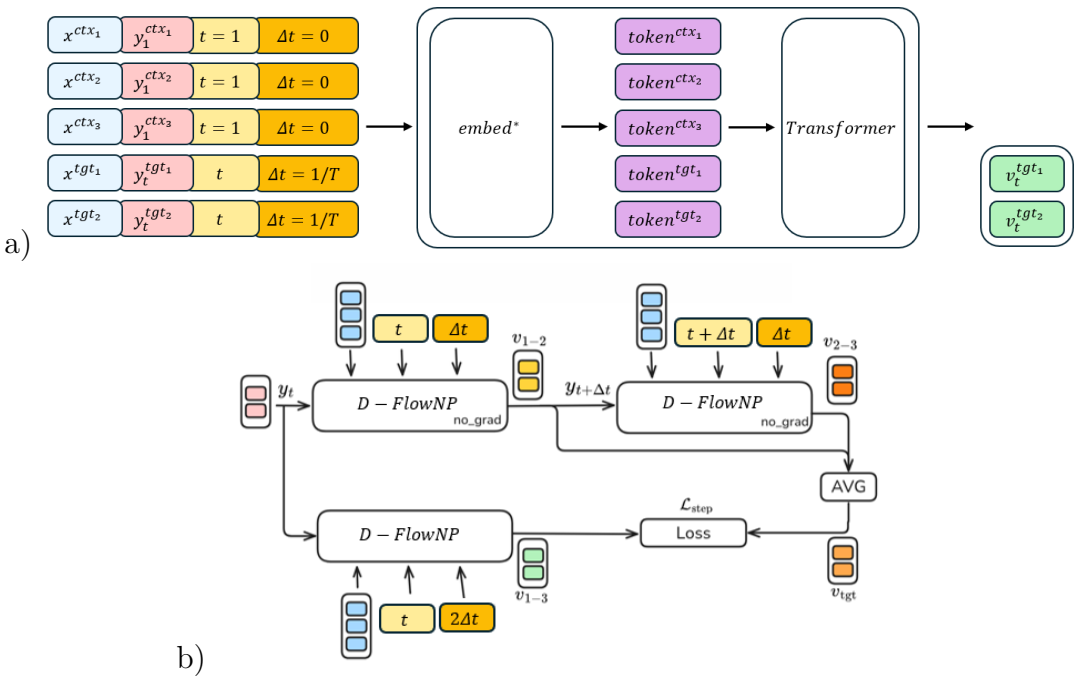


Figure 3-6: a) D-FlowNP architecture. The model extends FlowNP by replacing the original embedding module with a step-conditioned embedding (embed*), which additionally receives the desired integration step size Δt as input. The remainder of the architecture is identical to FlowNP. b) Training D-FlowNP via Step Amortization. The model is trained to minimize the discrepancy between a single coarse step of size $2\Delta t$ (student) and two fine steps of size Δt (teacher). This self-consistency objective $\mathcal{L}_{\text{step}}$ is added to the standard flow matching loss, enabling the model to learn accurate large-step transitions directly.

3.5 Direct Flow Neural Processes

3.5.1 Motivation

As shown in the previous section, FlowNP (Sec 3.4) models conditional distributions over functions through a time-dependent probability flow, enabling expressive posterior sampling and likelihood evaluation. However, inference requires numerically integrating the learned velocity field from the base distribution ($t = 0$) to the posterior distribution ($t = 1$). Consequently, generating a single posterior sample requires multiple evaluations of the network, making repeated posterior queries computationally expensive.

Our goal is to retain the expressive posterior modeling capabilities of FlowNP

while substantially reducing the computational cost of inference. Specifically, given an arbitrary context set $\mathcal{C} = (x^{\texttt{ctx}}, y^{\texttt{ctx}})$ and target locations $x^{\texttt{tgt}}$, we seek to generate accurate posterior samples using significantly fewer function evaluations than standard ODE integration.

To achieve this, we introduce *Direct Flow Neural Processes (D-FlowNP)*, which amortizes the probability flow itself by learning large transitions along the trajectory. Rather than relying on many small numerical integration steps during inference, D-FlowNP learns to approximate these larger transitions directly, enabling accurate posterior sampling under a wide range of computational budgets.

3.5.2 Method

Step-Amortized Flow

While FlowNP amortizes vector field computation, inference remains slow due to ODE integration, where using many small steps is accurate but costly, while fewer large steps are faster but degrade sample quality.

To address this, we propose to extend the principle of amortization to the numerical solver itself. Rather than relying on a generic integrator that is blind to the data distribution, we amortize the integration step by explicitly conditioning the model on the step size Δt.

Formally, we modify the network to accept the step size as an additional input:

$$v_\theta(y_t^{\texttt{tgt}}, t, \Delta t, x^{\texttt{tgt}}, \mathcal{C}) \tag{3.11}$$

This allows the model to predict a "corrected" velocity that accounts for the discretization error inherent in a step of size Δt. In effect, the model learns to approximate the transition operator $y_t \to y_{t+\Delta t}$ directly. This enables the use of significantly larger steps, requiring fewer function evaluations to cover the entire flow trajectory without the degradation in sample quality associated with standard solvers.

Training an Amortized Inference Step

In order to enable accurate inference with fewer integration steps, we adopt the approach in Frans et al. [2025] and augment the loss to minimize the discrepancy between a single coarse step of size $2\Delta t$ and two fine steps of size Δt *in velocity space*. Concretely, we form a *teacher* velocity model as an exponential moving average (EMA) of the student parameters $v_{\bar{\theta}}$, and use it to define a stable target consisting of two steps of half the size.

Given a state y_t at time t and a step size Δt, we compute:

$$
\begin{aligned}
v_{1-2} &= v_{\bar{\theta}}\Big(y_t^{\texttt{tgt}},\, t,\, \Delta t, x^{\texttt{tgt}}, \mathcal{C}\Big) && (3.12)\\
y_{t+\Delta t}^{\texttt{tgt}} &= y_t^{\texttt{tgt}} + \Delta t\, v_{1-2} && (3.13)\\
v_{2-3} &= v_{\bar{\theta}}\Big(y_{t+\Delta t}^{\texttt{tgt}},\, t+\Delta t,\, \Delta t, x^{\texttt{tgt}}, \mathcal{C}\Big) && (3.14)\\
v_{\text{tgt}} &= \tfrac{1}{2}\left(v_{1-2} + v_{2-3}\right) && (3.15)
\end{aligned}
$$

The student predicts the coarse step velocity

$$
v_{1-3} = v_\theta\Big(y_t^{\texttt{tgt}},\, t,\, 2\Delta t, x^{\texttt{tgt}}, \mathcal{C}\Big)\,, \tag{3.16}
$$

and we minimize the loss

$$
\mathcal{L}_{\text{step}}(\theta) = \mathbb{E}_{t,\Delta t}\Big\| v_{1-3} - sg(v_{tgt)}\Big\|_2^2 \tag{3.17}
$$

where sg denotes a stop-gradient.

By conditioning on Δt, the model learns to follow the flow manifold's curvature directly; as visualized in Appendix 3.7.4, this allows D-FlowNP to accurately 'shortcut' the trajectory where standard solver steps would otherwise diverge.

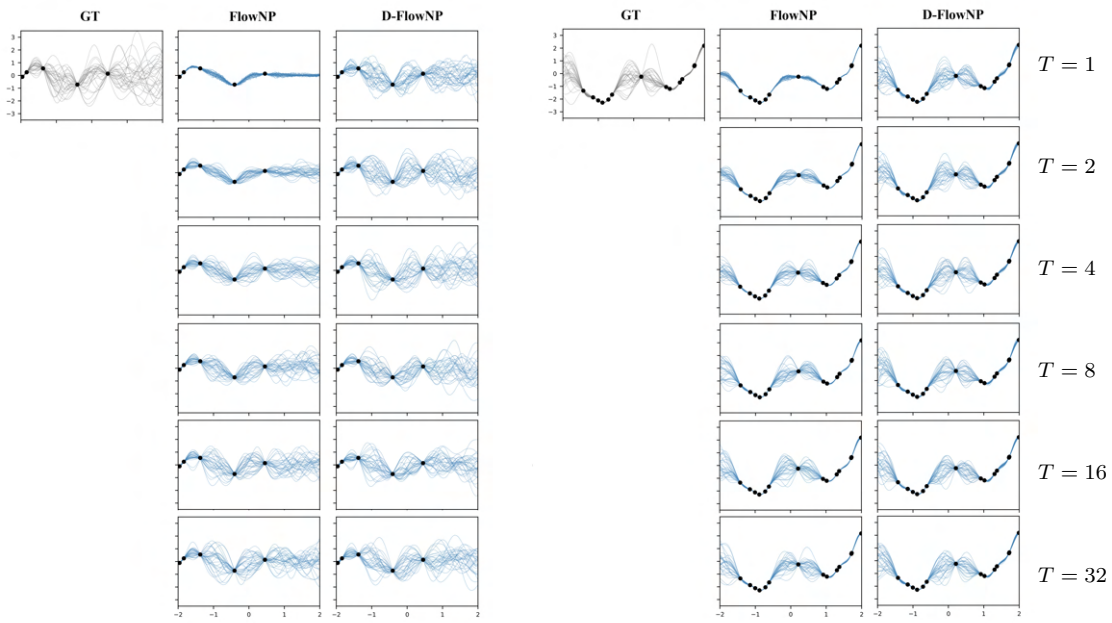


Figure 3-7: 1D Gaussian Process Regression. Posterior samples from FlowNP and D-FlowNP conditioned on the same context sets. Rows correspond to different numbers of inference steps T, equivalent to the number of function evaluations (NFE) for the Euler solver used here. While FlowNP samples degrade into noise, lose coherence and collapse to the mean at low NFE, D-FlowNP maintains higher-quality, smoother and more diverse function samples even at $T = 1$ (NFE=1).

Final objective. The step loss is combined with the standard FlowNP loss computed with $\Delta t = 0$, namely $\mathcal{L}_{\text{base}} = \mathbb{E}\left[\|v_\theta(y_t^{\texttt{tgt}}, t, 0, x^{\texttt{tgt}}, \mathcal{C}) - (y_1^{\texttt{tgt}} - y_0^{\texttt{tgt}})\|^2\right]$. Thus the combined loss in training is:

$$\mathcal{L}(\theta) \;=\; \mathcal{L}_{\text{base}}(\theta) \;+\; \mathcal{L}_{\text{step}}(\theta), \tag{3.18}$$

Figure 3-6 depicts the training process of step amortization.

Implementation

Architecture and inference cost. The network backbone remains identical to FlowNP, namely, a transformer network where each token encodes one context or target point (x, y), we only modify the input layer's projection dimension to accommodate the scalar step size Δt.

Consequently, our inference-time procedure has the same computational cost as

FlowNP for a given number of solver steps T, which corresponds to the number of function evaluations (NFE) for the Euler solver used in our main experiments.

The only additional cost introduced by D-FlowNP occurs during training, due to the extra forward passes required to construct the teacher-consistent step target in Eq. equation 3.17.

Discrete step sizes. For step amortization, we restrict the integration step sizes to a dyadic set

$$\Delta t \in \left\{1, \tfrac{1}{2}, \tfrac{1}{4}, \tfrac{1}{8}, \ldots, \tfrac{1}{128}\right\} = \left\{2^{-n}\right\}_{n=0}^{7}. \tag{3.19}$$

At each training iteration, we sample Δt uniformly from this set. Given a sampled step size Δt, we sample the flow time t on a grid that is compatible with the step,

$$t \sim \mathrm{U}\left(\{0, \Delta t, 2\Delta t, 3\Delta t, \ldots, 1 - 2\Delta t\}\right), \tag{3.20}$$

so that both t and $t + 2\Delta t$ remain valid times in $[0, 1]$. This ensures the teacher construction (two steps) is well-defined and avoids boundary effects near $t = 0$ and $t = 1$.

Training procedure. During training, each function uses randomly sampled $(t, \Delta t)$ pairs as above. We optimize the combined objective $\mathcal{L}(\theta) = \mathcal{L}_{\text{base}}(\theta) + \mathcal{L}_{\text{step}}(\theta)$, where $\mathcal{L}_{\text{base}}$ corresponds to standard FlowNP training (implemented by setting $\Delta t = 0$ in our parameterization), and $\mathcal{L}_{\text{step}}$ trains the model to perform larger jumps using an EMA teacher.

Inference procedure. At inference time, we choose the number of solver steps as $T = 2^n$ with $n \in \{0, 1, \ldots, 7\}$ and use a fixed step size $\Delta t = 1/T$. We then integrate from $t = 0$ to $t = 1$ using the same solver as FlowNP, but with the velocity field explicitly conditioned on the chosen Δt.

More implementation details are available in Appendix 3.7.5.

3.5.3 Experimental Setup

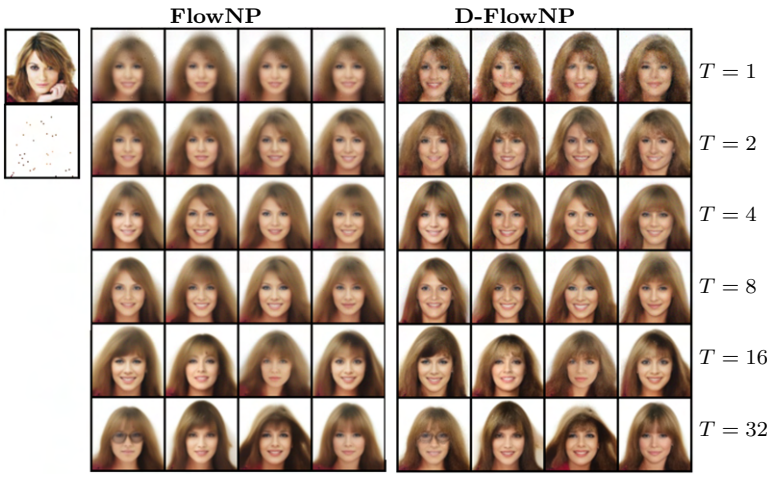


Figure 3-8: CelebA Image Completion. Target predictions conditioned on sparse context pixels. As the number of solver steps T decreases, standard FlowNP produces increasingly blurry samples and exhibits mode collapse. D-FlowNP preserves high-frequency detail and sample diversity even in the single-step setting.

We validate D-FlowNP across four diverse datasets, spanning synthetic Gaussian processes, natural images, and complex physical simulations. For each domain, we compare D-FlowNP against a baseline FlowNP with an identical backbone architecture to isolate the impact of the amortization objective. Detailed model configurations, hyperparameter settings, and the full experimental codebase are provided in Appendices 3.7.5– 3.7.10.

Gaussian Processes (GP) We first use synthetic datasets of functions sampled from GP priors Rasmussen and Williams [2006] using **RBF** and **Matérn-5/2** kernels. For every function, $N \in [3, 200]$ points are sampled from a latent mapping $f : \mathbb{R}^1 \to \mathbb{R}^1$ with randomized kernel hyperparameters. These points are partitioned into context and target sets, testing the model's capacity to quantify uncertainty and preserve structural priors (e.g., smoothness) under variable-sized observations.

CelebA Images Following standard NP benchmarks, we treat the CelebA dataset Liu et al. [2015] as a distribution of continuous functions $f : \mathbb{R}^2 \to \mathbb{R}^3$ mapping 64×64 pixel coordinates to RGB values. During training, we sample context sizes n_c

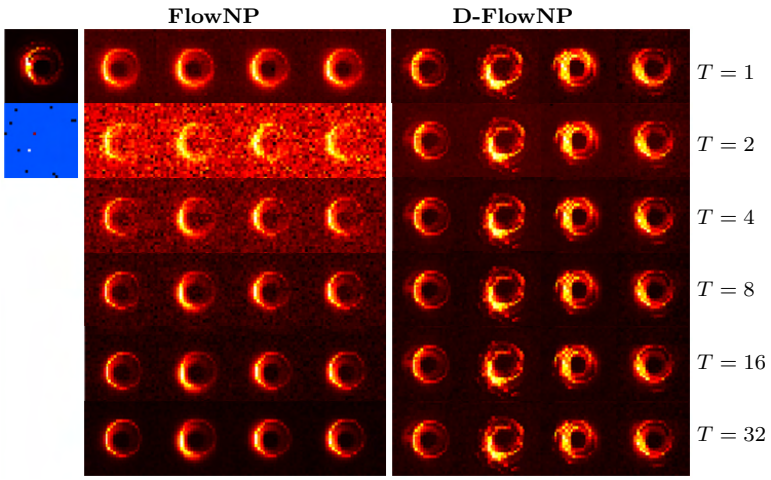


Figure 3-9: Black Hole Imaging. Reconstruction of simulated black hole disks from sparse data. D-FlowNP successfully reconstructs coherent and diverse ring structures even at $T = 1$, whereas the baseline samples degrade significantly with lower T.

uniformly from $[10, N/4]$ and target sets $n_t = N/3$ from a shared random ordering. This controlled overlap setup evaluates the model's ability to maintain high-frequency facial features and sample diversity.

Scientific Simulations We further assess the method on two 2D physical fields ($f : \mathbb{R}^2 \to \mathbb{R}^1$) represented as 28×28 grids: (1) **Navier-Stokes** fluid velocity fields Shi et al. [2025b] and (2) simulated **Black Hole** accretion disks Karras et al. [2025]. Both datasets consist of 30,000 samples. Following the GP partitioning protocol, we evaluate the model's capacity to preserve complex, non-Gaussian structures—such as fluid turbulence or ring artifacts—while maintaining the physical consistency of the sampled fields without the computational overhead of iterative ODE solvers.

Evaluation metrics We evaluate sample quality using metrics appropriate to the available ground-truth information. For the GP experiments, where samples from the true posterior are available, we measure distributional agreement using Maximum Mean Discrepancy (MMD; lower is better), together with sample Precision and Recall (higher is better), which quantify sample fidelity and posterior coverage, respectively. For the Navier–Stokes and black-hole datasets, only one ground-truth target is available for each context set. We therefore report Calibration Error, which measures agreement

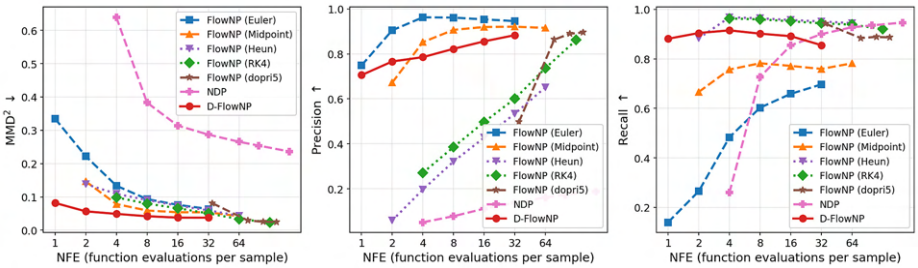


Figure 3-10: Efficiency-Quality Pareto Front (1D GP). D-FlowNP strictly dominates numerical solvers at low NFE. It maintains high Recall (diversity) even at $T = 1$, whereas baseline solvers suffer from integration drift or mode-collapse.

between predicted and empirical uncertainty levels; Coverage, defined as the minimum RMSE among the generated samples; Diversity, measured by the standard deviation across samples; and a Coverage–Diversity Score that jointly rewards accurate and diverse posterior samples. Full definitions are provided in Appendix 3.7.8.

3.5.4 Analysis

Sampling Quality

We evaluate the generative performance of D-FlowNP across varying computational budgets T. In standard FM models, reducing T introduces integration drift that manifests as noise, blurriness, or mode collapse. D-FlowNP is designed to maintain sample fidelity even at $T = 1$ by learning the manifold curvature directly.

Low-Dimensional Regression On 1D GP tasks, baseline FlowNP samples exhibit significant degradation as T decreases; integration error causes trajectories to diverge from the true posterior, resulting in noisy functions that violate the GP prior's smoothness (see Fig. 3-7). Furthermore, the sample variance artificially tightens, underestimating uncertainty. In contrast, D-FlowNP produces coherent, smooth samples with consistent predictive variance across all values of T, effectively correcting for discretization error.

This degradation persists even when employing higher-order numerical solvers; as shown in Appendix 3.7.7, while solvers such as Midpoint and RK4 improve trajectory

accuracy at higher budgets, they remain unable to produce coherent samples at $T \leq 2$, whereas D-FlowNP maintains structural integrity.

High-Dimensional and Physical Fields Across image generation on CelebA and physical domains such as Navier–Stokes and black-hole imaging, standard FlowNP exhibits increasingly mean-seeking behavior as the number of solver steps decreases, resulting in blurry samples and a loss of high-frequency structural detail. D-FlowNP substantially mitigates this degradation, producing sharper and more diverse samples even at $T = 1$. As shown in Figure 3-8, D-FlowNP preserves fine facial details and sample diversity in the low-T regime, whereas FlowNP produces increasingly blurry and over-smoothed completions. This improvement is also reflected in FID: at $T = 1$, D-FlowNP achieves 64.65 compared with 115.42 for FlowNP, and at $T = 2$ it achieves 33.70 compared with 57.28; the complete results are reported in Appendix 3.7.6. Similarly, in Figure 3-9, FlowNP tends toward sample averaging at low step counts, while D-FlowNP generates a diverse set of coherent ring configurations. Additional qualitative grids for CelebA, Navier–Stokes, and black-hole imaging are provided in Appendix 3.7.6.

Quantitative Benchmarking and Tradeoffs

To evaluate the efficiency-quality tradeoff of flow amortization, we compare D-FlowNP against a baseline FlowNP with standard numerical solvers (Euler, Midpoint, RK4) and the Neural Diffusion Process (NDP) [Dutordoir et al., 2023] baseline.

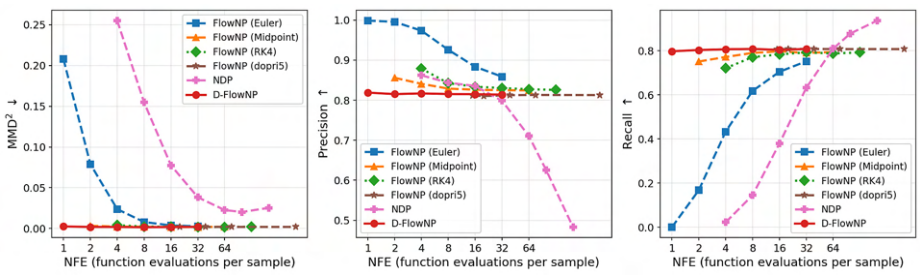
Efficiency-Quality Pareto Fronts We characterize the relationship between computational budget and generative quality using Maximum Mean Discrepancy (MMD) and Precision/Recall metrics (Fig. 3-10). In the low-NFE regime ($T \leq 4$), D-FlowNP strictly dominates both iterative numerical solvers and the diffusion-based NDP. Notably, a single step ($T = 1$) of D-FlowNP achieves lower MMD and higher Recall than four steps ($T = 4$) of an RK4 solver, representing a fourfold improvement in inference efficiency while preserving superior sample diversity.

Cross-domain performance Figure 3-11 shows that the low-NFE advantage of D-FlowNP extends beyond the 1D GP setting. On the 2D GP task, D-FlowNP maintains low MMD together with stable Precision and Recall at one and two function evaluations, whereas standard FlowNP solvers require substantially more steps to recover comparable posterior coverage. On Navier–Stokes and black-hole imaging, D-FlowNP similarly achieves substantially lower Calibration Error and Coverage–Diversity Score in the low-NFE regime. It also maintains competitive best-sample Coverage while avoiding the loss of sample diversity exhibited by the baseline solvers. These results indicate that step amortization improves not only individual reconstruction quality but also the calibration and diversity of the predicted posterior.

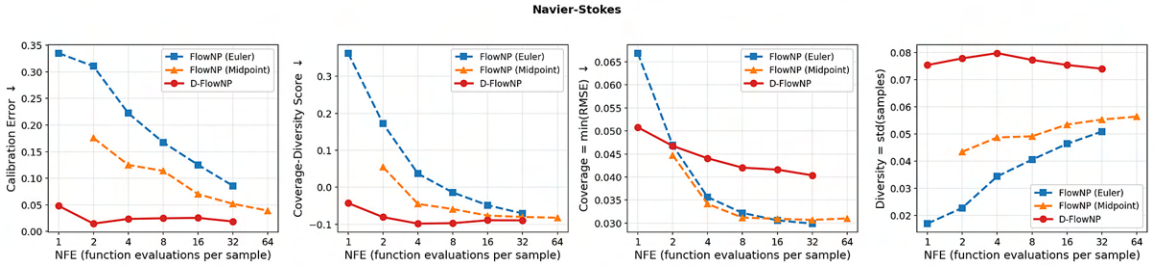
Inference Latency and Capacity Tradeoffs Beyond discretization steps, we evaluate performance against actual wall-clock inference time to ensure the overhead of step-conditioning is negligible. As shown in Appendix 3.7.9, D-FlowNP's Pareto dominance holds when measured in milliseconds, confirming its utility for latency-sensitive applications. At very high budgets ($T \geq 32$), we observe that higher-order solvers like RK4 converge to slightly higher precision. This reflects an inherent architectural tradeoff: since the backbone is identical, D-FlowNP must allocate a portion of its representational capacity to learn the large-step shortcut mappings in addition to the instantaneous velocity field. However, in the low-budget settings often required for sequential decision-making loops, D-FlowNP provides the optimal quality-to-speed ratio.

3.5.5 Application

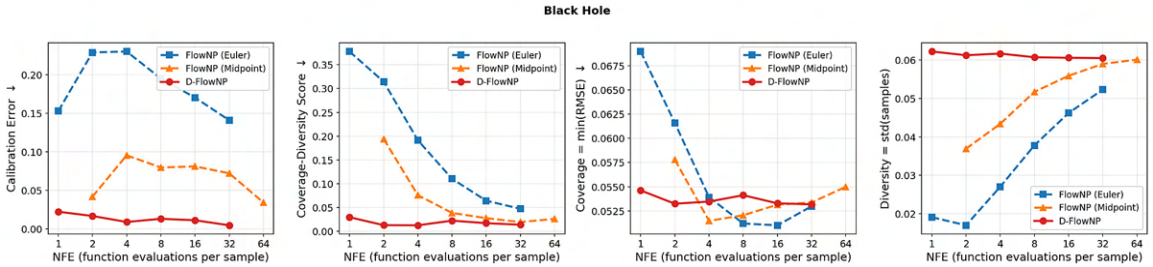
We demonstrate the practical utility of our method on Bayesian Optimization (BO), a sequential decision-making task involving inference in the loop, where the model must repeatedly query the posterior in order to select the next evaluation point and find the function's minimum Garnelo et al. [2018c]. We employ Thompson Sampling, drawing samples from the model's posterior and choosing the next evaluation point as the minimum value in the sample.



(a) 2D GP Metrics



(b) Navier-Stokes Metrics



(c) Black Hole Imaging Metrics

Figure 3-11: Cross-domain efficiency–quality tradeoffs on (a) 2D Gaussian processes, (b) Navier–Stokes reconstruction, and (c) black-hole imaging. Across domains, D-FlowNP provides improved posterior quality in the low-computational-budget regime.

We evaluate the performance using the difference between the lowest evaluated point and the true minimum (regret) and compare D-FlowNP against (1) FlowNP (2) Random search - choosing the points at random which should serve as the upper bound of regret, and (3) Transformer Neural Process (TNP) Nguyen and Grover [2023] - a Gaussian baseline sharing the same transformer architecture as our flow models but outputting a Gaussian distribution per point rather than a flow velocity. For TNP we report results using both Expected Improvement (EI) and Independent Thompson Sampling (ITS) by generating a sample for each point independently (which can also be considered a stochastic version of UCB). In appendix 3.7.10 we also compare

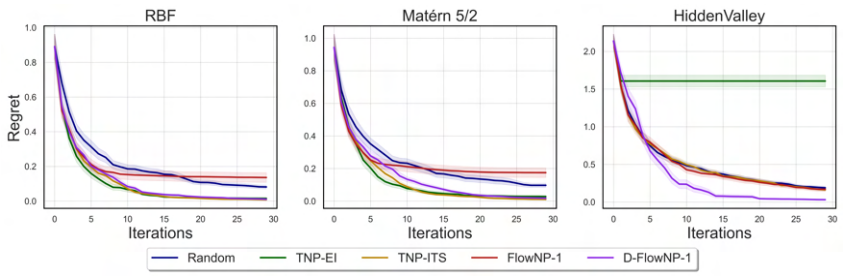


Figure 3-12: Bayesian optimization regret over 1D functions. On standard GPs, D-FlowNP with a single step ($T = 1$) performs similarly to the Gaussian baseline (TNP) and generally outperforms FlowNP ($T = 1$). On the *Hidden Valley* task, which is designed to confound Gaussian predictors with a specific non-Gaussian posterior, TNP with EI strategy fails completely, while D-FlowNP successfully identifies the valley, and significantly outperforms TNP with Thompson Sampling and FlowNP-1 which are at random level.

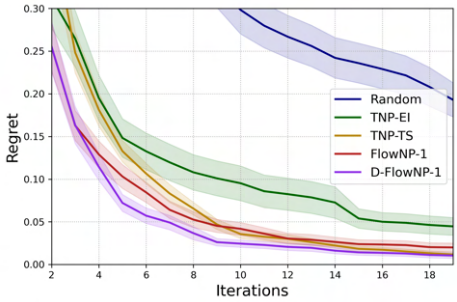


Figure 3-13: Bayesian optimization regret on Navier Stokes Flows. D-FlowNP with one sampling step ($T = 1$) consistently achieves lower regret compared to both the Gaussian baseline (TNP) and standard FlowNP with one step.

D-FlowNP with FlowNP in more detail, using various step counts $T = \{1, 2, 4, 16\}$ for both models.

On standard 1D GP regression tasks with RBF and Matérn-5/2 kernels (Fig. 3-12), D-FlowNP generally outperforms FlowNP using one sampling step (see Fig. 3-24 for more step counts). Compared to the Gaussian baseline TNP, the performance of D-FlowNP is very similar. This is expected, as the true data generation process is Gaussian, and thus the added expressivity of the flow is not strictly required.

To evaluate performance under non-Gaussian posteriors, we introduce a synthetic "Hidden Valley" task designed to confound Gaussian predictors. The function exhibits a rare, low-valued region that is difficult to identify without maintaining sufficient posterior diversity. In this setting, Gaussian models fail to explore effectively, while

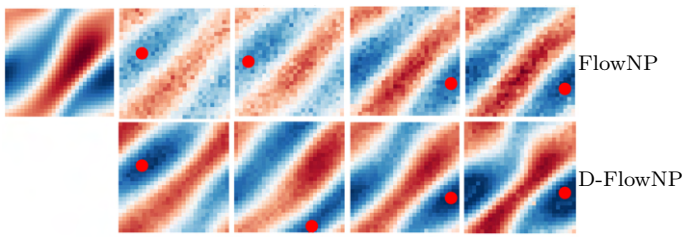


Figure 3-14: A comparison of the first four steps of the Bayesian optimization process on the Navier Stokes dataset using FlowNP and D-FlowNP with low step counts ($T = 2$). FlowNP produces noisier, less coherent field predictions, causing Thompson Sampling to suggest poor candidate points. In contrast, D-FlowNP generates coherent predictions that allow the optimization loop to effectively target the minimum.

D-FlowNP successfully captures the underlying multimodal structure and consistently identifies the valley (Fig. 3-12). Full details of the task construction are provided in Appendix 3.7.10.

Finally, we evaluate performance on the Navier Stokes dataset (Fig. 3-13). D-FlowNP with one sampling step consistently achieves lower regret than both TNP and FlowNP with one step (see Fig. 3-25 for more step counts). To visualize this difference, we examine the first 4 steps of the optimization process (Fig. 3-14). At low step counts ($T = 2$), FlowNP produces noisier and less coherent field predictions, leading Thompson Sampling to suggest poor candidate points. D-FlowNP with $T = 2$ generates coherent predictions that allow the optimization loop to effectively target the minimum, demonstrating that amortized inference can unlock complex downstream tasks previously infeasible for flow-based models.

3.6 Conclusion

We presented a unified framework for probabilistic inference over continuous functions based on conditional probability flows. First, we introduced **Flow Neural Processes (FlowNP)**, which formulates Neural Processes through Flow Matching to learn expressive conditional distributions over arbitrary context and target sets. Unlike previous Neural Process formulations based on latent-variable or autoregressive objectives, FlowNP enables direct posterior sampling and likelihood estimation

within a single coherent probabilistic framework. Across Gaussian process regression, image completion, weather forecasting, and scientific simulation datasets, FlowNP consistently achieves strong predictive performance while producing coherent posterior samples and well-calibrated uncertainty estimates.

We then addressed the principal limitation of flow-based inference: the computational cost of iterative ODE integration. To this end, we proposed **Direct Flow Neural Processes (D-FlowNP)**, which amortizes the integration process by learning large transitions along the probability flow trajectory through self-consistency objectives. This substantially reduces the number of function evaluations required during inference while preserving the quality and diversity of posterior samples. Our experiments demonstrate that D-FlowNP maintains competitive predictive performance even in the extreme low-NFE regime, enabling accurate one-step and few-step posterior generation across diverse continuous domains.

Together, these results establish conditional probability flows as an effective foundation for Neural Processes and demonstrate that inference-step amortization substantially improves their practical applicability. We hope this work encourages further research on efficient probabilistic inference for Neural Processes and other continuous generative models.

3.7 Appendix

The implementation in https://github.com/danrsm/flowNP contains the configurations of all the models.

3.7.1 Consistency of NP Models

We discuss the consistency property of different models in more detail. The original CNP and NP models use architectures that ensure that given a context, all target points are predicted independently. This implies that consistency over marginalization (Eq. 3.4) holds. However, the model structure does not guarantee consistency over conditioning (Eq. 3.5) because computing the model conditioned on different contexts

cannot be guaranteed to lead to consistent results that comply with an underlying joint distribution.

TNP, which is based on a transformer performs best when it is used as an autoregressive model. In that case it does not comply neither with exchangeability nor consistency, since even when using clever masking to make the model invariant to the context order, the joint distribution of the target still depends on the ordering of the autoregression in the target points. While the authors of TNP propose different variants of the models that make it exchangeable and consistent with marginalization, they lead to a significant drop in performance, and are still not guaranteed to be consistent in terms of conditioning.

The reason that NDP is consistent over conditioning is that it only models joint distributions and therefore computes conditionals using the conditional definition:

$$p\left(y^{\mathtt{tgt}}|x^{\mathtt{tgt}}, \{x^{\mathtt{ctx}}, y^{\mathtt{ctx}}\}\right) = p_\theta\left(y^{\mathtt{tgt}+\mathtt{ctx}}|x^{\mathtt{tgt}+\mathtt{ctx}}\right) / p_\theta\left(y^{\mathtt{ctx}}|x^{\mathtt{ctx}}\right)$$

However the full self-attention architecture leads to predictions of target points that cannot be separated into independent factors and therefore cannot ensure consistency over marginalizations.

3.7.2 Training and Sampling

The algorithms for training and sampling are provided in Alg. 2 and Alg. 3.

Fig. 3-15 provides a demonstration of the sampling process for 1D GP data by visualizing the transformation from random noise to samples from the conditional distribution of $p(y^{\mathtt{tgt}}|x^{\mathtt{tgt}}, \{x^{\mathtt{ctx}}, y^{\mathtt{ctx}}\})$.

The hyperparameters of the baselines are tuned to improve their performance. Namely, we find that for CNP and TNP no bound on the output variance is required, while for the models trained by variational inference, NP and ANP, we set the output variance bound to 0.05^2 and use 8 importance samples for training.

Algorithm 2: Training

input: dataset of functions $\mathcal{F}$
repeat
  $f_i \sim \mathcal{F}$
  $M, N \sim \mathcal{U}$
  $x^{\texttt{ctx}} \sim \mathcal{X}^M,\ x^{\texttt{tgt}} \sim \mathcal{X}^N$
  $y^{\texttt{ctx}} \leftarrow f_i(x^{\texttt{ctx}}),\ y^{\texttt{tgt}} \leftarrow f_i(x^{\texttt{tgt}})$
  $t \sim \mathcal{U}[0, 1]$
  $y_0^{\texttt{tgt}} \sim \mathcal{N}(0, I)$
  $y_t^{\texttt{tgt}} \leftarrow t y^{\texttt{tgt}} + (1-t) y_0^{\texttt{tgt}}$
  $\texttt{tokens}^{\texttt{ctx}} \leftarrow \{\texttt{embed}_\theta([x^{\texttt{ctx}}, 1, y^{\texttt{ctx}}])\}$
  $\texttt{tokens}^{\texttt{tgt}} \leftarrow \{\texttt{embed}_\theta([x^{\texttt{tgt}}, t, y_t^{\texttt{tgt}}])\}$
  $\hat{u}_t \leftarrow u_\theta([\texttt{tokens}^{\texttt{ctx}}, \texttt{tokens}^{\texttt{tgt}}])$
  $\mathcal{L}(\theta) \leftarrow \|\hat{u}_t - (y^{\texttt{tgt}} - y_0^{\texttt{tgt}})\|^2$
  $\theta \leftarrow \texttt{update}(\nabla_\theta \mathcal{L}(\theta))$
until convergence
output: trained parameters θ

Algorithm 3: Sampling

input: context $x^{\texttt{ctx}}, y^{\texttt{ctx}}$, target positions $x^{\texttt{tgt}}$

$\texttt{tokens}^{\texttt{ctx}} \leftarrow \{\texttt{embed}_\theta([x^{\texttt{ctx}}, 1, y^{\texttt{ctx}}])\}$
$y^{\texttt{tgt}} \sim \mathcal{N}(0, I)$
$\delta \leftarrow 1/\texttt{n_steps}$
$t \leftarrow 0$
while $t < 1$ **do**
  $\texttt{tokens}^{\texttt{tgt}} \leftarrow \{\texttt{embed}_\theta([x^{\texttt{tgt}}, t, y^{\texttt{tgt}}])\}$
  $\hat{u}_t \leftarrow u_\theta([\texttt{tokens}^{\texttt{ctx}}, \texttt{tokens}^{\texttt{tgt}}])$
  $y^{\texttt{tgt}} \leftarrow y^{\texttt{tgt}} + \delta \hat{u}_t$
  $t \leftarrow t + \delta$
end while
output: predicted sample $y^{\texttt{tgt}}$

3.7.3 Analysis

ODE solver We compute the likelihood using several different ODE solvers for the FlowNP model on the RBF experiment. The results, presented in Table 3.4, show that except for the Euler method which overestimates the likelihood, all other solvers compute very similar values.

Number of ODE steps We run an ablation on sampling and log-likelihood evaluation for different numbers of steps in the ODE solver using the midpoint method. For GP-RBF data, the mean of 5 random seeds with different number of steps is presented in Fig. 3-16. The standard deviation is 0.014 across all settings. This shows that the evaluation plateaus after T=60 steps. We observe similar behavior for sample quality.

Running time Comparing the sampling time between FlowNP, NDP and TNP using an NVIDIA RTX4090 GPU and 100 sampling steps for FlowNP: the time to generate 1 sample in the GP experiment with 200 target points is 0.2sec for FlowNP, 0.5sec for NDP and 0.8sec for TNP. The time to generate 1 EMNIST sample with 748 target points is 4.6sec for FlowNP, 10.4sec for NDP and 72.6sec for TNP.

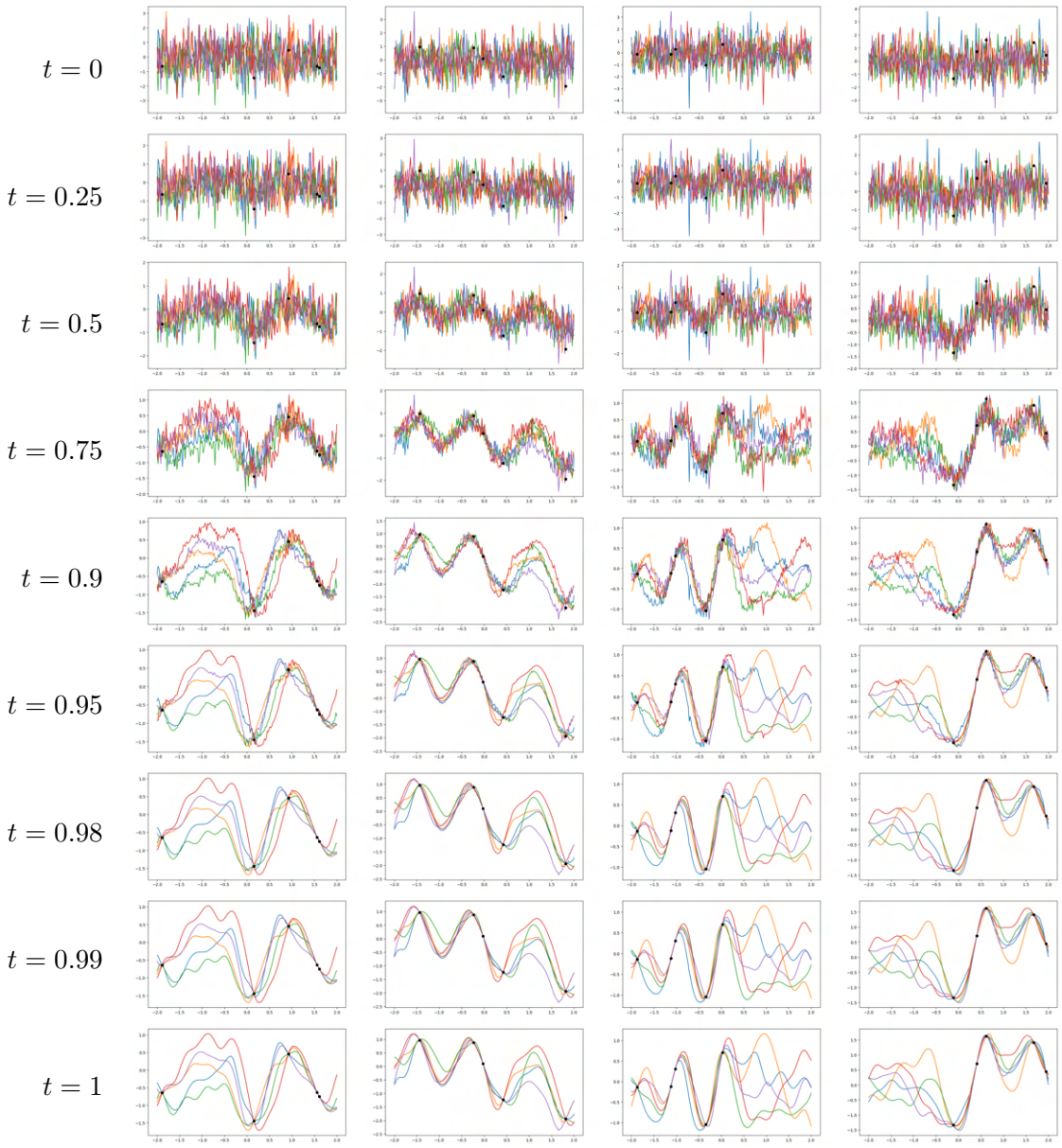


Figure 3-15: A visualization of the sampling process, showing the intermediate values $y_t^{\texttt{tgt}}$ for different time steps. Each column is conditioned on a different context and each color represents a different random sample generated by FlowNP.

ODE-solver	log-likelihood
Euler	1.7941
Midpoint	1.6890
RK4	1.6856
Dopri5	1.6857

Table 3.4: Comparing different ODE solvers on the RBF GP data.

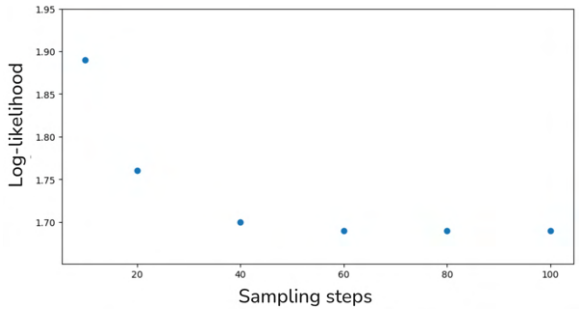


Figure 3-16: Log-likelihood as a function of number of ODE steps, computed using the midpoint ODE solver on the RBF GP data.

3.7.4 Geometric Analysis

To verify that D-FlowNP effectively amortizes the ODE integration, we compare its sampling trajectories against standard FlowNP in a 2-target Gaussian Process setting. In standard FlowNP, using a small NFE (e.g., $NFE = 1$) forces the model to use a large, locally linear Euler step. As shown in Figure 3-17, these linear steps (red) fail to account for the flow's curvature, causing the samples to diverge into low-probability regions. In contrast, D-FlowNP (blue) is trained to predict the integral over Δt directly. The visualization confirms that D-FlowNP learns a curved mapping that stays on the high-probability manifold, explaining its superior performance and diversity at low computational budgets.

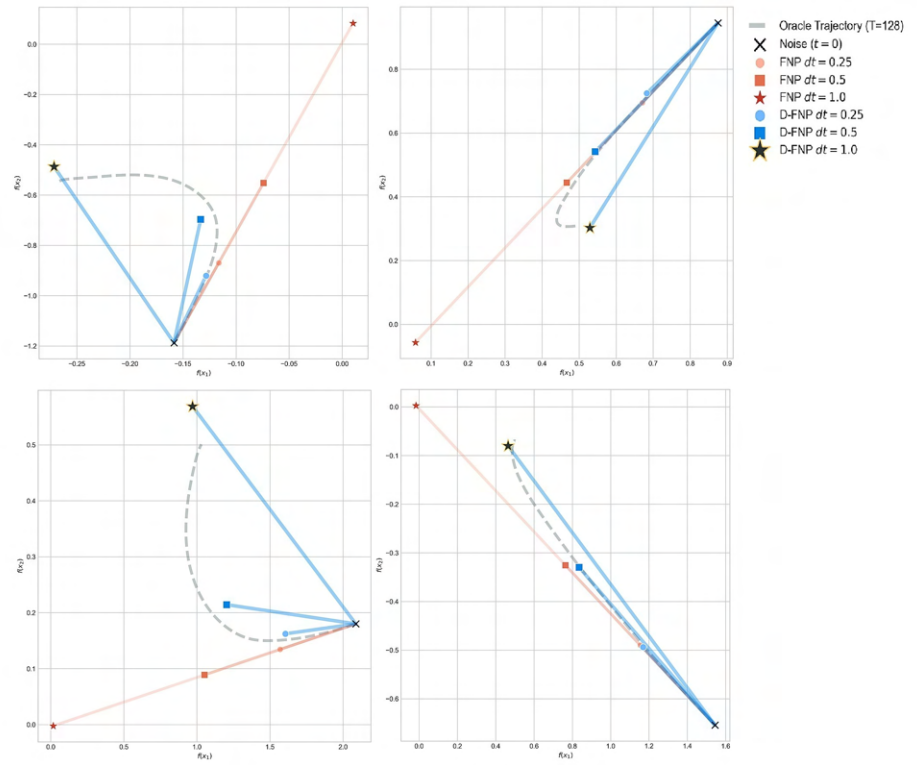


Figure 3-17: Comparison of Sampling Trajectories. We visualize the flow from the prior (noise) to the posterior for a 2-target GP. Standard FlowNP steps (red) diverge into low-probability regions when NFE is small due to the lack of curvature awareness. In contrast, D-FlowNP (blue) learns to follow the curved manifold, enabling accurate "shortcuts" in a single step.

3.7.5 Implementation Details for FlowNP and D-FlowNP

This appendix describes the implementation shared by FlowNP and D-FlowNP, followed by the modifications specific to D-FlowNP. Both models use the same token construction, attention mechanism, positional encoding, and Transformer backbone. D-FlowNP additionally conditions the model on the integration step size Δt and is trained with the step-amortization objective described in Sec. 3.5.2.

Token construction and context/target packing

We model each training instance as a set of context observations $\mathcal{C} = \{(x_i^{\text{ctx}}, y_i^{\text{ctx}})\}_{i=1}^{N_{\text{ctx}}}$ and target query locations $\mathcal{T} = \{x_j^{\text{tgt}}\}_{j=1}^{N_{\text{tgt}}}$. At training time, for each target location we additionally provide a *noised* target value $y^{\text{tgt}}(t)$ along a linear interpolation between

Gaussian noise and the ground-truth target value:

$$y^{\mathrm{tgt}}(t) \;=\; t\, y^{\mathrm{tgt}}(1) \;+\; (1-t)\, y^{\mathrm{tgt}}(0), \quad y^{\mathrm{tgt}}(0) \sim \mathcal{N}(0, I). \tag{3.21}$$

The velocity target for flow matching is $v^\star = y^{\mathrm{tgt}}(1) - y^{\mathrm{tgt}}(0)$.

We build a single token sequence by concatenating context tokens and target tokens:

$$\text{tokens} \;=\; [\, \tau_1^{\mathrm{ctx}}, \ldots, \tau_{N_{\mathrm{ctx}}}^{\mathrm{ctx}},\; \tau_1^{\mathrm{tgt}}, \ldots, \tau_{N_{\mathrm{tgt}}}^{\mathrm{tgt}} \,]. \tag{3.22}$$

Each token is an embedding of *(position, auxiliary scalars, value)*:

$$\tau_i^{\mathrm{ctx}} = \mathrm{MLP}_{\mathrm{emb}}\Big(\mathrm{PE}([x_i^{\mathrm{ctx}},\; 1,\; 0]) \,\|\, y_i^{\mathrm{ctx}}\Big), \tag{3.23}$$

$$\tau_j^{\mathrm{tgt}} = \mathrm{MLP}_{\mathrm{emb}}\Big(\mathrm{PE}([x_j^{\mathrm{tgt}},\; t,\; \Delta t]) \,\|\, y_j^{\mathrm{tgt}}(t)\Big), \tag{3.24}$$

where $\|$ denotes concatenation. We explicitly append two scalars to each coordinate: for context tokens we use $(1, 0)$ (a fixed marker), and for target tokens we use the current ODE time t and the solver step size Δt. This allows the model to condition on both the integration time and the discretization used at inference.

Attention masking. We apply a causal-style block mask so that context tokens cannot attend to target tokens, preventing leakage of target information into the context representation. Optionally, for the independent-prediction setting we additionally mask target-to-target attention (while keeping each target token self-visible) to enforce per-target conditional independence.

Positional encoding

We use a fixed Fourier positional encoding (sin/cos) applied per input coordinate dimension. Given a scalar coordinate u, we map it to:

$$\mathrm{PE}(u) \;=\; \Big[\sin(\omega_1 u), \ldots, \sin(\omega_K u),\; \cos(\omega_1 u), \ldots, \cos(\omega_K u)\Big], \tag{3.25}$$

with $K = \frac{1}{2}$ `dim_posenc` frequencies. For a vector coordinate (e.g. $[x, t, \Delta t]$), we apply the encoding to each component and concatenate. In our implementation, the frequency bank uses powers of two (scaled by 2π), yielding multi-scale features spanning low to high frequencies. The resulting encoded coordinate is concatenated with the corresponding value y before entering the token embedder MLP.

Time conditioning and adaptive LayerNorm (AdaLN)

Each transformer layer uses Adaptive Layer Normalization, where the LayerNorm affine parameters are predicted from a learned time embedding. We compute sinusoidal embeddings for both t and Δt, project them with an MLP, and use the result as the conditioning vector $e \in \mathbb{R}^{d_{\text{model}}}$:

$$e \;=\; \text{MLP}_t\Big(\text{emb}(t) \| \text{emb}(\Delta t)\Big). \tag{3.26}$$

Within each transformer block, we use two LayerNorms with `elementwise_affine=False`. A small modulation MLP produces scale/shift parameters for attention and MLP sublayers:

$$(\gamma_{\text{attn}}, \beta_{\text{attn}}, \gamma_{\text{mlp}}, \beta_{\text{mlp}}) = \text{MLP}_{\text{mod}}(e), \tag{3.27}$$

and applies them as

$$\text{AdaLN}(h; \gamma, \beta) = \text{LN}(h) \odot (1 + \gamma) + \beta. \tag{3.28}$$

We use standard residual connections around attention and the feed-forward sublayer.

Transformer and MLP sizes

The model consists of: (i) a token embedding MLP that maps the concatenated PE($\cdot$) features and y into d_{model}, (ii) L Transformer blocks with multi-head self-attention and AdaLN modulation, and (iii) a predictor MLP that maps the final target-token states to the output dimension d_y. The embedding MLP contains `emb_depth` linear layers, with a ReLU activation applied after each layer except the final one. Unless otherwise

stated, the feed-forward hidden size is `dim_feedforward`, the number of attention heads is `nhead`, the number of Transformer blocks is `num_layers`, and the dropout rate is denoted by `dropout`. The model configurations used for the GP, Navier–Stokes, and black-hole experiments are summarized in Table 3.5.

Table 3.5: Model hyperparameters used in all experiments.

	GP	Navier-Stokes	Black Hole
`dim_x`	1	2	2
`dim_y`	1	1	1
`dim_posenc`	40	20	20
`d_model`	128	64	64
`emb_depth`	4	4	4
`dim_feedforward`	256	128	128
`nhead`	8	4	4
`dropout`	0.0	0.0	0.0
`num_layers`	8	6	6

D-FlowNP training objective

Training mixes two losses:

1. **Flow matching:** for a subset of batch elements, we sample $t \sim \mathcal{U}(\{0, \dots, T-1\}/T)$ with fixed step size $\Delta t = 1/T$ and regress the true velocity $v^\star$ using MSE.

2. **Step amortization:** for the remaining batch elements, we sample a dyadic step size $\Delta t \in \{1, \frac{1}{2}, \frac{1}{4}, \dots\}$ and a valid t such that $t + 2\Delta t \leq 1$. A teacher model produces a two-step velocity estimate (at $t \to t + \Delta t \to t + 2\Delta t$), and the student is trained to match this with a single step of size $2\Delta t$.

The total loss is

$$\mathcal{L} = \mathcal{L}_{\text{base}} + \mathcal{L}_{\text{step}}, \tag{3.29}$$

We use a fraction `step_frac` of batch elements for the amortization term. `step_frac`=0.25 for all experiments.

CelebA (DiT-B backbone with Δt conditioning)

For CelebA we use a different architecture based on a DiT-B transformer backbone Peebles and Xie [2023b].

How Δt is incorporated. In addition to the usual timestep embedding of t, we embed the solver step size Δt with a second timestep embedder of the same dimensionality and inject it through the same DiT conditioning pathway:

$$c \;=\; e_t(t) \;+\; e_{\Delta t}(\Delta t), \tag{3.30}$$

where $e_t(\cdot)$ and $e_{\Delta t}(\cdot)$ are two independent DiT-style timestep embedding networks (MLPs). The resulting conditioning vector $c \in \mathbb{R}^D$ is passed to every DiT block and the final prediction layer, modulating their AdaLN parameters (scale/shift) exactly as in standard DiT, but now conditioned on both t and Δt.

Target/context token packing. We apply Fourier positional encoding to the coordinates and concatenate with the token values:

$$h = \mathrm{Linear}\Big(\big[\mathrm{PE}(x^{\mathtt{tgt}})\|y^{\mathtt{tgt}}\big] \;\|\; \big[\mathrm{PE}(x^{\mathtt{ctx}})\|y^{\mathtt{ctx}}\big]\Big), \tag{3.31}$$

then process the full sequence with DiT blocks conditioned on c. Only the first (target) segment of the output tokens is used to form the final prediction.

3.7.6 Qualitative Sampling Grids

We compare D-FlowNP with a baseline FlowNP using an identical backbone architecture, allowing us to isolate the effect of the amortization objective. In each figure, the ground truth and observed context points are shown on the left, followed by four independent samples from FlowNP and D-FlowNP at each inference budget. As the number of solver steps T decreases, FlowNP exhibits increasing blurriness, sample averaging, and signs of mode collapse. In contrast, D-FlowNP maintains sharper reconstructions, richer high-frequency detail, and greater sample diversity, including in the single-step setting.

CelebA examples are presented in Figure 3-18, with FID scores across inference budgets reported in Table 3.6. Figure 3-19 and 3-20 provide additional qualitative results for Navier–Stokes fluid dynamics and black-hole imaging, respectively.



Figure 3-18: CelebA Image Completion. Target predictions conditioned on sparse context pixels. As the number of solver steps T decreases, standard FlowNP exhibits increasing blurriness and mode collapse, while D-FlowNP preserves sharper high-frequency details and maintains greater sample diversity, even in the single-step regime.

Table 3.6: FID scores on CelebA for varying number of solver steps T (lower is better).

Model	$T = 1$	$T = 2$	$T = 4$	$T = 8$	$T = 16$	$T = 32$
FlowNP	115.42	57.28	71.49	75.44	65.81	43.72
D-FlowNP	64.65	33.70	67.06	74.18	64.64	41.49

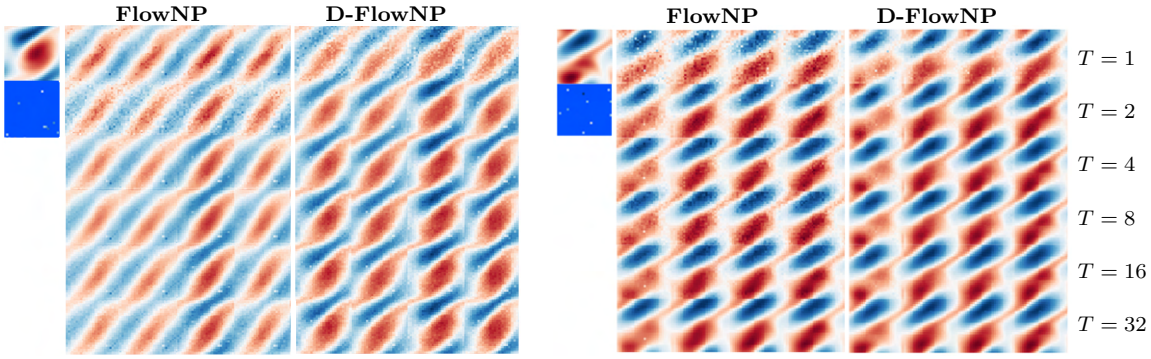


Figure 3-19: Navier-Stokes Fluid Dynamics. Posterior samples of fluid velocity fields given sparse sensor measurements. D-FlowNP produces sharper reconstructions across all values of T.

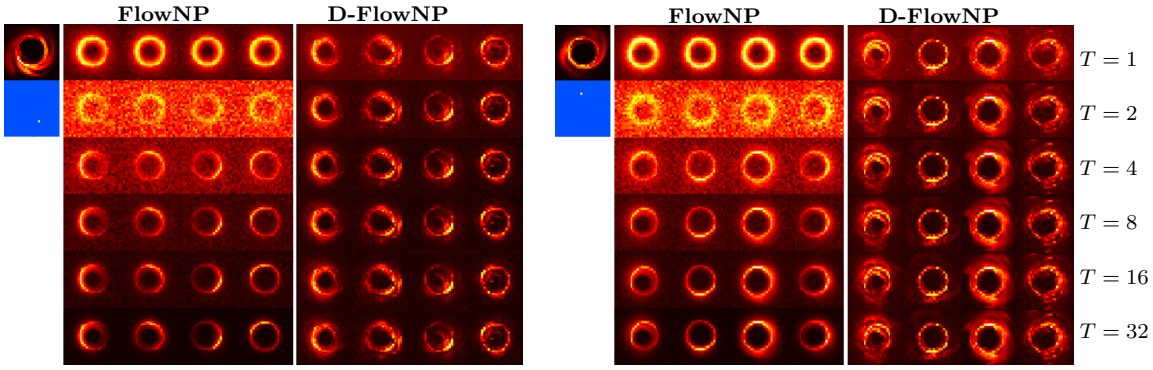


Figure 3-20: Black Hole Imaging. Reconstruction of simulated black hole disks from very sparse data. D-FlowNP reconstructs coherent and diverse ring structures even at $T = 1$, whereas the baseline samples become increasingly averaged and degraded as T decreases.

3.7.7 Qualitative Comparison with Higher-Order Solvers

D-FlowNP has the same computational cost per function evaluation as FlowNP because they use the same architecture. For the Euler updates used by FlowNP and D-FlowNP, each integration step requires one function evaluation. Midpoint and RK4 require twice and four times as many function evaluations per integration step, respectively. D-FlowNP is able to produce high-quality samples using only one or a few evaluations, whereas standard FlowNP generally requires substantially more evaluations to achieve comparable quality.

We visualize posterior samples for 1D GP tasks using Euler, Midpoint, and RK4 solvers compared against D-FlowNP.

As illustrated in Figure 3-21 and 3-22, while higher-order solvers (RK4, Midpoint) converge to the true posterior more quickly than Euler as T increases, they still exhibit significant mode collapse and noise when T is very small ($T = 1, 2$). In contrast, D-FlowNP's learned shortcuts allow it to generate high-fidelity, diverse samples even in the single-step regime where traditional numerical solvers fundamentally fail.

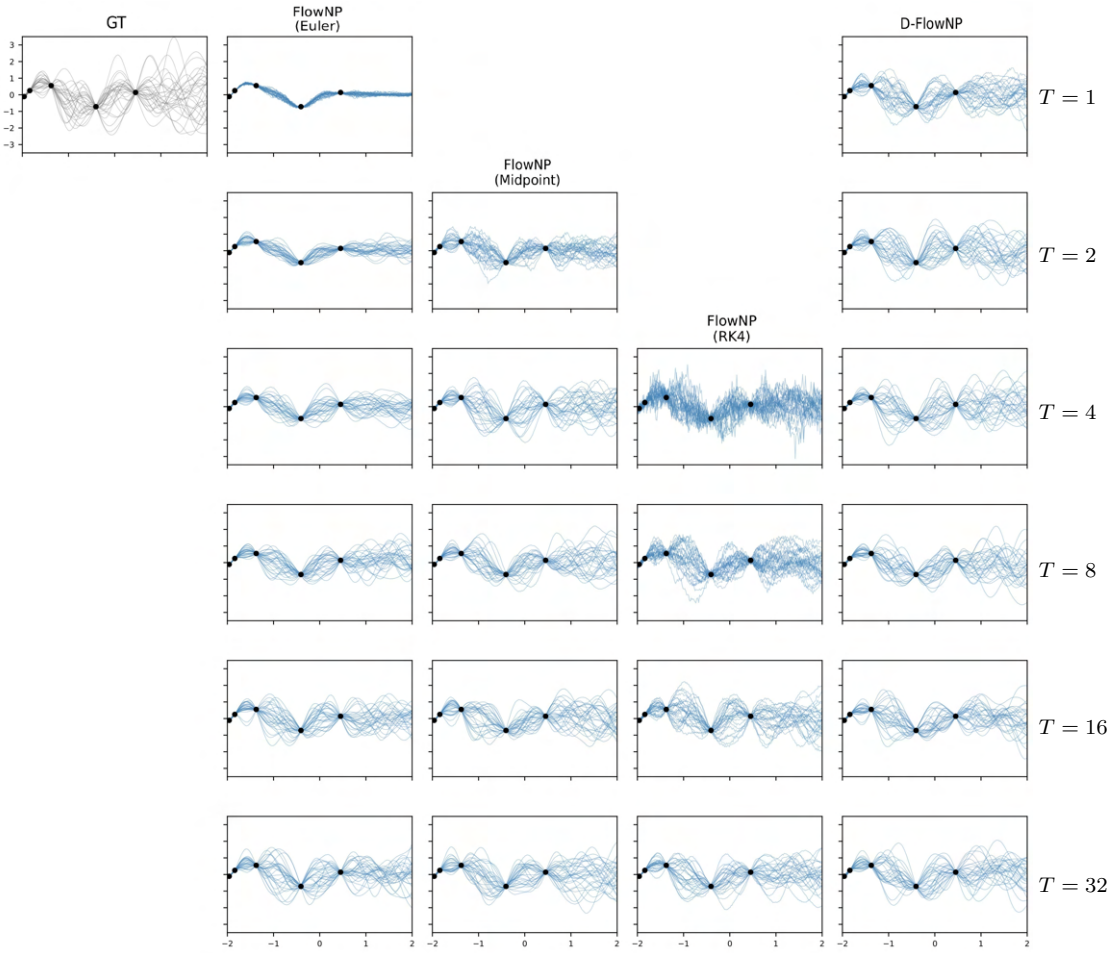


Figure 3-21: Comparison of solvers across NFE (Task A). Standard solvers (Euler, Midpoint, RK4) fail to capture the GP smoothness and diversity at low T, while D-FlowNP remains robust.

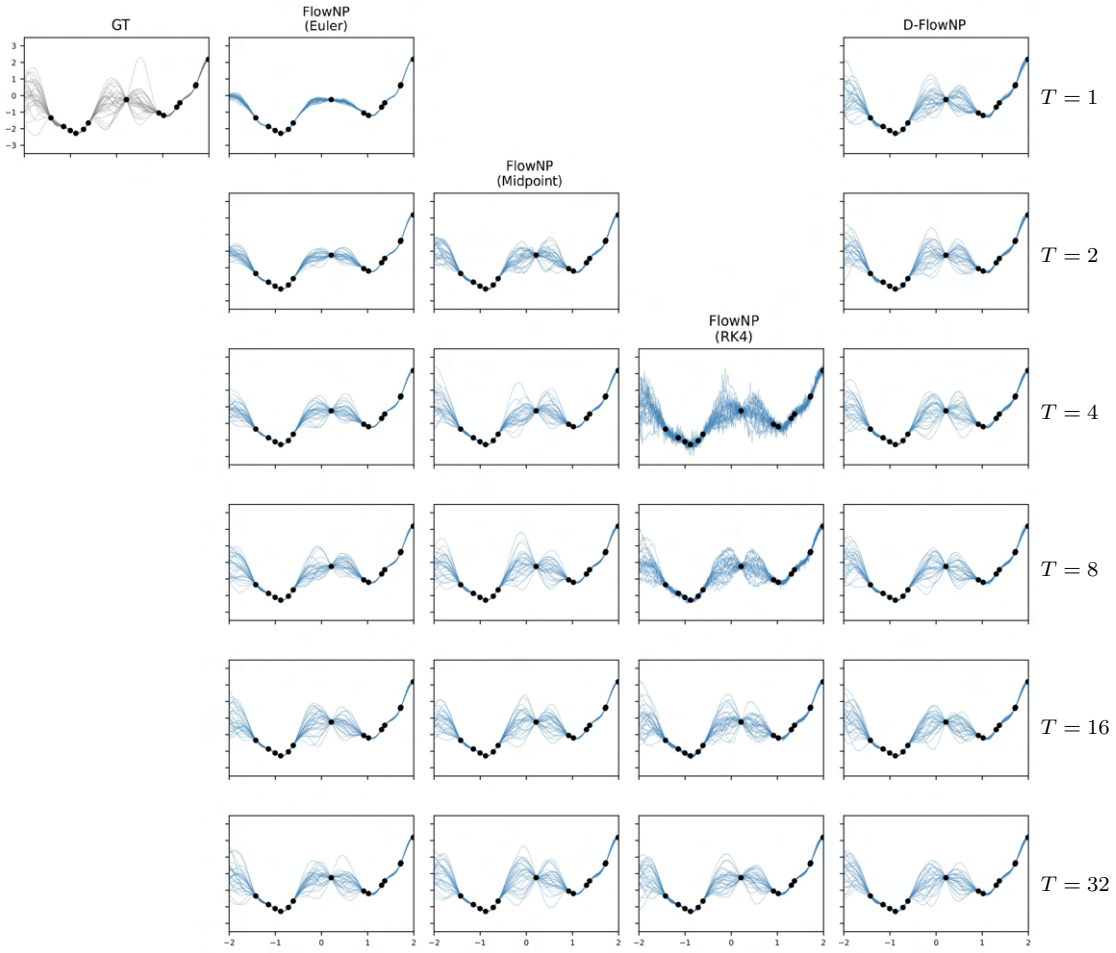


Figure 3-22: Comparison of solvers across NFE (Task B). Even higher-order methods like RK4 require multiple steps to match the sample quality D-FlowNP achieves in a single step.

3.7.8 Evaluation Metrics

We provide a detailed description of all evaluation metrics used in our experiments. We distinguish between settings where ground-truth posterior samples are available (Gaussian Process experiments) and settings with only a single ground-truth observation (scientific datasets).

Metrics with Ground-Truth Posterior Samples (GP)

MMD^2 (Maximum Mean Discrepancy) Gretton et al. [2012]. Lower is better. We measure the distance between the distribution of model samples $\{x_i\}_{i=1}^S$ and ground-truth posterior samples using the squared Maximum Mean Discrepancy (MMD) with an RBF kernel. The kernel bandwidth is selected via the median heuristic. MMD captures discrepancies in both mean and higher-order statistics, reflecting both sample accuracy and diversity.

Precision Kynkäänniemi et al. [2019] (k-NN, $k = 5$). Higher is better. Precision measures the fraction of model samples that lie within the support of the ground-truth distribution, estimated using a k-nearest neighbor (k-NN) manifold. High precision indicates that generated samples are realistic and lie on the true data manifold.

Recall Kynkäänniemi et al. [2019] (k-NN, $k = 5$). Higher is better. Recall measures the fraction of ground-truth samples that are covered by the model distribution. High recall indicates good posterior coverage, while low recall signals mode collapse.

Metrics with a Single Ground-Truth Observation (Scientific Datasets)

Calibration Error Gneiting et al. [2007]. Lower is better. We evaluate uncertainty calibration by comparing empirical coverage to nominal coverage levels. For each target point, we compute the empirical rank of the ground-truth value among S model samples and estimate the coverage of central prediction intervals. The calibration

error is defined as the average absolute deviation:

$$\frac{1}{|\mathcal{A}|} \sum_{\alpha \in \mathcal{A}} |\hat{c}(\alpha) - \alpha| \,,$$

where $\mathcal{A} = \{0.05, 0.10, \ldots, 0.95\}$ and $\hat{c}(\alpha)$ denotes empirical coverage. This metric penalizes both under-dispersion (mode collapse) and over-dispersion.

Coverage–Diversity Score (CDS). Lower is better. We define:

$$\text{CDS} = \min_i \|x_i - y\| - \frac{1}{2S^2} \sum_{i,j} \|x_i - x_j\|,$$

where y is the ground-truth target. The first term measures *coverage* (distance of the closest sample to the ground truth), while the second term measures *diversity* (average pairwise sample distance). This metric encourages models to produce at least one accurate sample while maintaining sufficient diversity. CDS is a heuristic metric designed to jointly capture accuracy and diversity in the absence of full posterior samples.

Coverage (min RMSE). Lower is better. We compute the root mean squared error (RMSE) of the best sample among S draws:

$$\min_i \text{RMSE}(x_i, y).$$

This metric reflects the best achievable prediction under the model posterior.

Diversity (Sample Standard Deviation). Descriptive. We measure the average per-target standard deviation across S samples, aggregated over all spatial locations. This metric quantifies the overall spread of the predicted posterior and is used to detect mode collapse.

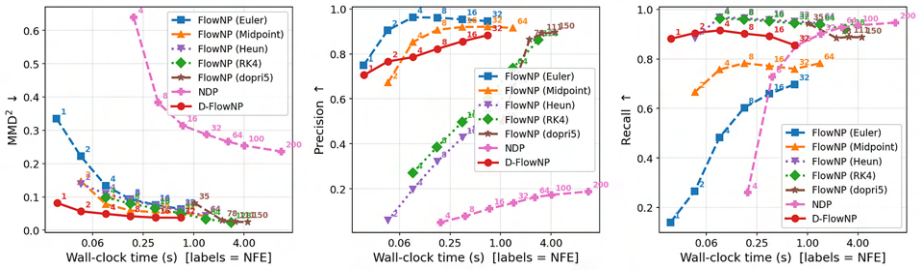
Summary. Together, these metrics disentangle distribution matching (MMD, Precision/Recall), uncertainty calibration (Calibration Error, Diversity), and best-case

accuracy (Coverage), providing a comprehensive evaluation of posterior quality.

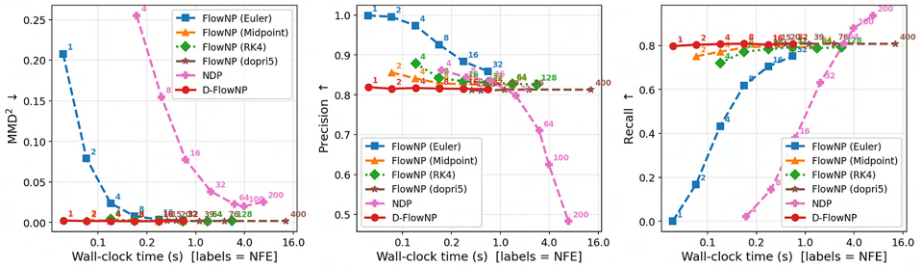
3.7.9 Wall-Clock Inference Analysis

In addition to the number of function evaluations, we evaluate the efficiency–quality tradeoff with respect to actual wall-clock inference time. Figure 3-23 reports MMD, Precision, and Recall as functions of sampling time for the 1D and 2D GP tasks.

Because FlowNP and D-FlowNP use identical backbone architectures, their cost per function evaluation is approximately equal. The advantage of D-FlowNP therefore arises from its ability to obtain high-quality samples using substantially fewer evaluations. In both settings, D-FlowNP reaches low MMD at considerably lower inference times than standard numerical solvers, confirming that its low-NFE advantage translates into a practical latency improvement.



(a) 1D GP metrics versus wall-clock inference time.



(b) 2D GP metrics versus wall-clock inference time.

Figure 3-23: Wall-clock efficiency–quality tradeoffs on the 1D and 2D GP tasks. D-FlowNP reaches low MMD while maintaining stable Precision and Recall at substantially lower inference times than standard FlowNP solvers, demonstrating that its advantage in function evaluations translates into a practical latency improvement.

3.7.10 Bayesian Optimization

Hidden Valley Task

To showcase the necessity of non-Gaussian posteriors, we introduce the "Hidden Valley" dataset. This 1D function is explicitly designed to confound Gaussian predictors. It consists of uniform noise $U[-\sqrt{3}, \sqrt{3}]$ across the domain, except for five specific locations. One of these is randomly chosen as the true valley with values in $[-2.1, -1.9]$, while the other four have higher values of $[0.4, 0.6]$. Until the valley is observed, the statistics consist of a mean of 0 and standard deviation of 1 everywhere. Consequently, TNP fails completely on this task, performing no better than random search (or even worse when using EI). In contrast, D-FlowNP successfully models the complex, multimodal posterior required to identify the valley, significantly outperforming the baselines including FlowNP with $T = 1$.

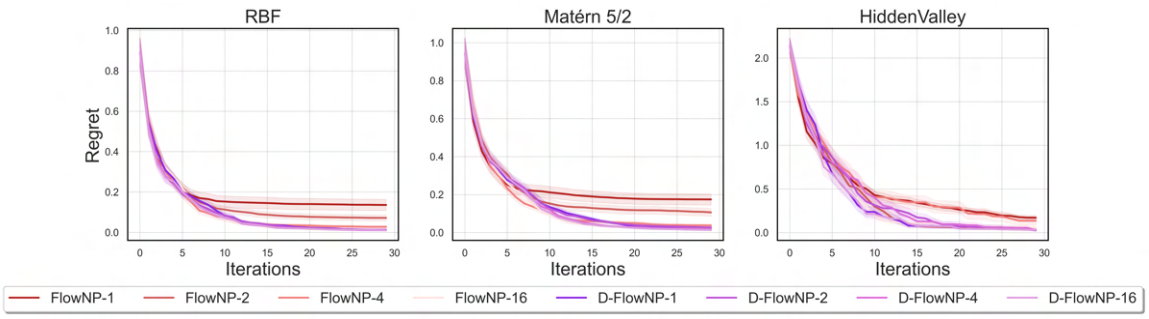


Figure 3-24: On standard 1D GP regression tasks with RBF and Matérn-5/2 kernels and on the HiddenValley function, D-FlowNP generally outperforms FlowNP at low step counts. While FlowNP with 4 steps achieves competitive results, reducing the number of steps to 2 or 1 yields significantly higher regret.

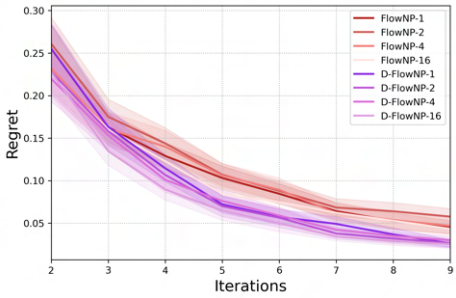


Figure 3-25: Regret on the Navier Stokes task comparing FlowNP and D-FlowNP across varying computational budgets ($T \in \{1, 2, 4, 16\}$). The plots generally show that D-FlowNP maintains lower regret at lower step counts compared to the baseline FlowNP, validating that step amortization allows for efficient and effective sequential decision-making.

Bibliography

Michael S Albergo and Eric Vanden-Eijnden. Building normalizing flows with stochastic interpolants. *arXiv preprint arXiv:2209.15571*, 2022.

Michael S Albergo, Nicholas M Boffi, and Eric Vanden-Eijnden. Stochastic interpolants: A unifying framework for flows and diffusions. *arXiv preprint arXiv:2303.08797*, 2023.

Matthias Bauer, Emilien Dupont, Andy Brock, Dan Rosenbaum, Jonathan Richard Schwarz, and Hyunjik Kim. Spatial functa: Scaling functa to imagenet classification and generation, 2023. URL https://arxiv.org/abs/2302.03130.

Miguel Angel Bautista, Pengsheng Guo, Samira Abnar, Walter Talbott, Alexander Toshev, Zhuoyuan Chen, Laurent Dinh, Shuangfei Zhai, Hanlin Goh, Daniel Ulbricht, Afshin Dehghan, and Josh Susskind. Gaudi: A neural architect for immersive 3d scene generation, 2022. URL https://arxiv.org/abs/2207.13751.

Nicholas M. Boffi, Michael S. Albergo, and Eric Vanden-Eijnden. Flow map matching with stochastic interpolants: A mathematical framework for consistency models, 2024. URL https://arxiv.org/abs/2406.07507.

Piotr Bojanowski, Armand Joulin, David Lopez-Paz, and Arthur Szlam. Optimizing the Latent Space of Generative Networks, May 2019. URL http://arxiv.org/abs/1707.05776.

Sam Bond-Taylor and Chris G Willcocks. ∞-diff: Infinite resolution diffusion with subsampled mollified states. *arXiv preprint arXiv:2303.18242*, 2023.

Lorenzo Bonito, James Requeima, Aliaksandra Shysheya, and Richard E. Turner. Diffusion-augmented neural processes, 2023. URL https://arxiv.org/abs/2311.09848.

Katherine L. Bouman, Michael D. Johnson, Daniel Zoran, Vincent L. Fish, Sheperd S. Doeleman, and William T. Freeman. Computational imaging for vlbi image reconstruction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

Wessel P Bruinsma, James Requeima, Andrew YK Foong, Jonathan Gordon, and Richard E Turner. The gaussian neural process. *arXiv preprint arXiv:2101.03606*, 2021.

Wessel P. Bruinsma, Stratis Markou, James Requiema, Andrew Y. K. Foong, Tom R. Andersson, Anna Vaughan, Anthony Buonomo, J. Scott Hosking, and Richard E. Turner. Autoregressive conditional neural processes, 2023. URL https://arxiv.org/abs/2303.14468.

Yuri Burda, Roger Grosse, and Ruslan Salakhutdinov. Importance weighted autoencoders. *arXiv preprint arXiv:1509.00519*, 2015.

Eric R Chan, Connor Z Lin, Matthew A Chan, Koki Nagano, Boxiao Pan, Shalini De Mello, Orazio Gallo, Leonidas Guibas, Jonathan Tremblay, Sameh Khamis, et al. Efficient Geometry-aware 3D Generative Adversarial Networks. *arXiv preprint arXiv:2112.07945*, 2021.

Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. Tensorf: Tensorial radiance fields. In *European Conference on Computer Vision (ECCV)*, 2022.

Hansheng Chen, Jiatao Gu, Anpei Chen, Wei Tian, Zhuowen Tu, Lingjie Liu, and Hao Su. Single-stage diffusion nerf: A unified approach to 3d generation and reconstruction, 2023. URL https://arxiv.org/abs/2304.06714.

Yinbo Chen and Xiaolong Wang. Transformers as meta-learners for implicit neural representations, 2022. URL https://arxiv.org/abs/2208.02801.

Jooyoung Choi, Sungwon Kim, Yonghyun Jeong, Youngjune Gwon, and Sungroh Yoon. Ilvr: Conditioning method for denoising diffusion probabilistic models. *arXiv preprint arXiv:2108.02938*, 2021.

Hyungjin Chung, Jeongsol Kim, Michael Thompson Mccann, Marc Louis Klasky, and Jong Chul Ye. Diffusion posterior sampling for general noisy inverse problems. In *Int. Conf. Learn. Represent.*, 2023. URL https://openreview.net/forum?id=OnD9zGAGT0k.

Gregory Cohen, Saeed Afshar, Jonathan Tapson, and Andre Van Schaik. Emnist: Extending mnist to handwritten letters. In *2017 international joint conference on neural networks (IJCNN)*, pages 2921–2926. IEEE, 2017.

Matt Deitke, Dustin Schwenk, Jordi Salvador, Luca Weihs, Oscar Michel, Eli VanderBilt, Ludwig Schmidt, Kiana Ehsani, Aniruddha Kembhavi, and Ali Farhadi. Objaverse: A universe of annotated 3d objects. *arXiv preprint arXiv:2212.08051*, 2022.

Emilien Dupont, Yee Whye Teh, and Arnaud Doucet. Generative models as distributions of functions. *arXiv preprint arXiv:2102.04776*, 2021.

Emilien Dupont, Hyunjik Kim, S. M. Ali Eslami, Danilo J. Rezende, and Dan Rosenbaum. From data to functa: Your data point is a function and you should treat it like one. *CoRR*, abs/2201.12204, 2022. URL https://arxiv.org/abs/2201.12204.

Conor Durkan, Artur Bekasov, Iain Murray, and George Papamakarios. Neural spline flows. *Advances in Neural Information Processing Systems*, 32:7511–7522, 2019.

Vincent Dutordoir, Alan Saul, Zoubin Ghahramani, and Fergus Simpson. Neural diffusion processes. In *International Conference on Machine Learning*, pages 8990–9012. PMLR, 2023.

Ziya Erkoç, Fangchang Ma, Qi Shan, Matthias Nießner, and Angela Dai. Hyperdiffusion: Generating implicit neural fields with weight-space diffusion, 2023. URL https://arxiv.org/abs/2303.17015.

S. Eslami, Danilo Jimenez Rezende, Frederic Besse, Fabio Viola, Ari Morcos, Marta Garnelo, Avraham Ruderman, Andrei Rusu, Ivo Danihelka, Karol Gregor, David Reichert, Lars Buesing, Theophane Weber, Oriol Vinyals, Dan Rosenbaum, Neil Rabinowitz, Helen King, Chloe Hillier, Matt Botvinick, and Demis Hassabis. Neural scene representation and rendering. *Science*, 360:1204–1210, 06 2018. doi: 10.1126/science.aar6170.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks, 2017. URL https://arxiv.org/abs/1703.03400.

Eric Frankel, Sitan Chen, Jerry Li, Pang Wei Koh, Lillian J. Ratliff, and Sewoong Oh. S4s: Solving for a diffusion model solver, 2025. URL https://arxiv.org/abs/2502.17423.

Kevin Frans, Danijar Hafner, Sergey Levine, and Pieter Abbeel. One step diffusion via shortcut models, 2025. URL https://arxiv.org/abs/2410.12557.

Marta Garnelo, Dan Rosenbaum, Chris Maddison, Tiago Ramalho, David Saxton, Murray Shanahan, Yee Whye Teh, Danilo Rezende, and S. M. Ali Eslami. Conditional neural processes. *ICML*, 2018a.

Marta Garnelo, Jonathan Schwarz, Dan Rosenbaum, Fabio Viola, Danilo J. Rezende, S. M. Ali Eslami, and Yee Whye Teh. Neural processes, 2018b. URL https://arxiv.org/abs/1807.01622.

Marta Garnelo, Jonathan Schwarz, Dan Rosenbaum, Fabio Viola, Danilo J Rezende, SM Eslami, and Yee Whye Teh. Neural processes. *arXiv preprint arXiv:1807.01622*, 2018c.

Zhengyang Geng, Mingyang Deng, Xingjian Bai, J. Zico Kolter, and Kaiming He.

Mean flows for one-step generative modeling, 2025. URL https://arxiv.org/abs/2505.13447.

Tilmann Gneiting, Fadoua Balabdaoui, and Adrian E. Raftery. Probabilistic forecasts, calibration and sharpness. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 69(2):243–268, 2007.

Lily Goli, Cody Reading, Silvia Sellán, Alec Jacobson, and Andrea Tagliasacchi. Bayes' rays: Uncertainty quantification for neural radiance fields, 2023. URL https://arxiv.org/abs/2309.03185.

Jonathan Gordon, Wessel P Bruinsma, Andrew YK Foong, James Requeima, Yann Dubois, and Richard E Turner. Convolutional conditional neural processes. In *International Conference on Learning Representations*, 2019.

Alexandros Graikos, Nikolay Malkin, Nebojsa Jojic, and Dimitris Samaras. Diffusion models as plug-and-play priors. In *Thirty-Sixth Conference on Neural Information Processing Systems*, 2022. URL https://arxiv.org/pdf/2206.09012.pdf.

Karol Gregor and Yann LeCun. Learning fast approximations of sparse coding. In *Proceedings of the 27th International Conference on Machine Learning (ICML)*, pages 399–406, 2010.

Arthur Gretton, Karsten M. Borgwardt, Malte J. Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *Journal of Machine Learning Research*, 13(25):723–773, 2012. URL http://jmlr.org/papers/v13/gretton12a.html.

Azmi A. Haider and Dan Rosenbaum. Predicting 3d structure by latent posterior sampling. In *Frontiers in Probabilistic Inference: Learning meets Sampling, ICLR*, 2025. URL https://arxiv.org/abs/2605.10830.

Jonathan Heek, Emiel Hoogeboom, and Tim Salimans. Multistep consistency models, 2024. URL https://arxiv.org/abs/2403.06807.

Hans Hersbach, Bill Bell, Paul Berrisford, Shoji Hirahara, András Horányi, Joaquín Muñoz-Sabater, Julien Nicolas, Carole Peubey, Raluca Radu, Dinand Schepers, et al. The era5 global reanalysis. *Quarterly Journal of the Royal Meteorological Society*, 146(730):1999–2049, 2020.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models, 2020. URL https://arxiv.org/abs/2006.11239.

Peter Holderrieth, Michael J Hutchinson, and Yee Whye Teh. Equivariant learning of stochastic fields: Gaussian processes and steerable conditional neural processes. In *International Conference on Machine Learning*, pages 4297–4307. PMLR, 2021.

Jeremy Howard. Imagenette: A smaller subset of imagenet. https://github.com/fastai/imagenette, 2019. Accessed: 2025-08-25.

Johanna S. Karras, He Sun, and Katherine L. Bouman. Deep neural networks for black hole imaging. GitHub repository, 2025. URL https://github.com/johannakarras/Deep-Neural-Networks-for-Black-Hole-Imaging. Accessed: 2026-01-27.

Bahjat Kawar, Michael Elad, Stefano Ermon, and Jiaming Song. Denoising diffusion restoration models. *Adv. Neural Inform. Process. Syst.*, 35:23593–23606, 2022.

Gavin Kerrigan, Justin Ley, and Padhraic Smyth. Diffusion generative models in infinite dimensions. *arXiv preprint arXiv:2212.00886*, 2022.

Gavin Kerrigan, Giosue Migliorini, and Padhraic Smyth. Functional flow matching. In *The 27th International Conference on AI and Statistics (AISTATS)*, 2024.

Chiheon Kim, Doyup Lee, Saehoon Kim, Minsu Cho, and Wook-Shin Han. Generalizable implicit neural representations via instance pattern composers, 2023. URL https://arxiv.org/abs/2211.13223.

Dongjun Kim, Chieh-Hsin Lai, Wei-Hsiang Liao, Naoki Murata, Yuhta Takida, Toshimitsu Uesaka, Yutong He, Yuki Mitsufuji, and Stefano Ermon. Consistency trajec-

tory models: Learning probability flow ode trajectory of diffusion, 2024. URL https://arxiv.org/abs/2310.02279.

Hyunjik Kim, Andriy Mnih, Jonathan Schwarz, Marta Garnelo, Ali Eslami, Dan Rosenbaum, Oriol Vinyals, and Yee Whye Teh. Attentive neural processes, 2019a. URL https://arxiv.org/abs/1901.05761.

Hyunjik Kim, Andriy Mnih, Jonathan Schwarz, Marta Garnelo, S. M. Ali Eslami, Dan Rosenbaum, and Yee Whye Teh. Attentive neural processes. In *ICML*, 2019b.

Adam R. Kosiorek, Heiko Strathmann, Daniel Zoran, Pol Moreno, Rosalia Schneider, Soňa Mokrá, and Danilo J. Rezende. Nerf-vae: A geometry aware 3d scene generative model, 2021a.

Adam R. Kosiorek, Heiko Strathmann, Daniel Zoran, Pol Moreno, Rosalia Schneider, Soňa Mokrá, and Danilo J. Rezende. Nerf-vae: A geometry aware 3d scene generative model, 2021b. URL https://arxiv.org/abs/2104.00587.

Nikola Kovachki, Zongyi Li, Burigede Liu, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural operator: Learning maps between function spaces with applications to pdes. *Journal of Machine Learning Research*, 24(89):1–97, 2023. URL http://jmlr.org/papers/v24/21-1524.html.

Tuomas Kynkäänniemi, Tero Karras, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Improved precision and recall metric for assessing generative models, 2019. URL https://arxiv.org/abs/1904.06991.

Remi Lam, Alvaro Sanchez-Gonzalez, Matthew Willson, Peter Wirnsberger, Meire Fortunato, Ferran Alet, Suman Ravuri, Timo Ewalds, Zach Eaton-Rosen, Weihua Hu, Alexander Merose, Stephan Hoyer, George Holland, Oriol Vinyals, Jacklynn Stott, Alexander Pritzel, Shakir Mohamed, and Peter Battaglia. Learning skillful medium-range global weather forecasting. *Science*, 382(6677):1416–1421, December 2023. ISSN 1095-9203. doi: 10.1126/science.adi2336. URL http://dx.doi.org/10.1126/science.adi2336.

Tuan Anh Le, Pavel Sountsov, Matthew D. Hoffman, Ben Lee, Brian Patton, and Rif A. Saurous. Robust inverse graphics via probabilistic inference, 2024. URL https://arxiv.org/abs/2402.01915.

Juho Lee, Yoonho Lee, Jungtaek Kim, Eunho Yang, Sung Ju Hwang, and Yee Whye Teh. Bootstrapping neural processes. *Advances in neural information processing systems*, 33:6606–6615, 2020.

Christian Lessig, Ilaria Luise, Bing Gong, Michael Langguth, Scarlet Stadtler, and Martin G. Schultz. Atmorep: A stochastic model of atmosphere dynamics using large scale representation learning. *CoRR*, abs/2308.13280, 2023. doi: 10.48550/ARXIV.2308.13280. URL https://doi.org/10.48550/arXiv.2308.13280.

Zongyi Li, Nikola Borislavov Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew M. Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=c8P9NQVtmnO.

Marten Lienen, Marcel Kollovieh, and Stephan Günnemann. Generative modeling with bayesian sample inference. 2025. URL https://arxiv.org/abs/2502.07580.

Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

Yaron Lipman, Marton Havasi, Peter Holderrieth, Neta Shaul, Matt Le, Brian Karrer, Ricky T. Q. Chen, David Lopez-Paz, Heli Ben-Hamu, and Itai Gat. Flow matching guide and code, 2024. URL https://arxiv.org/abs/2412.06264.

Ruoshi Liu, Rundi Wu, Basile Van Hoorick, Pavel Tokmakov, Sergey Zakharov, and Carl Vondrick. Zero-1-to-3: Zero-shot one image to 3d object, 2023.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. *arXiv preprint arXiv:2209.03003*, 2022.

Xinhang Liu, Jiaben Chen, Shiu hong Kao, Yu-Wing Tai, and Chi-Keung Tang. Deceptive-nerf/3dgs: Diffusion-generated pseudo-observations for high-quality sparse-view reconstruction, 2024. URL https://arxiv.org/abs/2305.15171.

Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Large-scale celebfaces attributes (celeba) dataset. *Retrieved August*, 15(2018):11, 2018.

Cheng Lu, Tao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, Jun Zhu, and Weinan Zhang. Dpm-solver: A fast ode solver for diffusion probabilistic model sampling in around 10 steps. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver++: Fast solver for guided sampling of diffusion probabilistic models. *Machine Intelligence Research*, 22(4):730–751, June 2025. ISSN 2731-5398. doi: 10.1007/s11633-025-1562-4. URL http://dx.doi.org/10.1007/s11633-025-1562-4.

Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis. Learning nonlinear operators via deeponet based on the universal approximation theorem of operators. *Nat. Mach. Intell.*, 3(3):218–229, 2021. doi: 10.1038/S42256-021-00302-5. URL https://doi.org/10.1038/s42256-021-00302-5.

Simian Luo, Yiqin Tan, Longbo Huang, Jian Li, and Hang Zhao. Latent consistency models: Synthesizing high-resolution images with few-step inference, 2023. URL https://arxiv.org/abs/2310.04378.

Joe Marino, Yisong Yue, and Stephan Mandt. Iterative amortized inference. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning*

Research, pages 3403–3412. PMLR, 10–15 Jul 2018. URL https://proceedings.mlr.press/v80/marino18a.html.

Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020.

Radford M Neal. Regression and classification using gaussian process priors. *Bayesian statistics*, 6:475, 1998.

Tung Nguyen and Aditya Grover. Transformer neural processes: Uncertainty-aware meta learning via sequence modeling. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 16569–16594. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/nguyen22b.html.

Tung Nguyen and Aditya Grover. Transformer neural processes: Uncertainty-aware meta learning via sequence modeling, 2023. URL https://arxiv.org/abs/2207.04179.

Tung Nguyen, Johannes Brandstetter, Ashish Kapoor, Jayesh K Gupta, and Aditya Grover. ClimaX: A foundation model for weather and climate. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 25904–25938. PMLR, 23–29 Jul 2023.

Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation, 2019. URL https://arxiv.org/abs/1901.05103.

Jaideep Pathak, Shashank Subramanian, Peter Harrington, Sanjeev Raja, Ashesh Chattopadhyay, Morteza Mardani, Thorsten Kurth, David Hall, Zongyi Li, Kam-

yar Azizzadenesheli, Pedram Hassanzadeh, Karthik Kashinath, and Animashree Anandkumar. Fourcastnet: A global data-driven high-resolution weather model using adaptive fourier neural operators. *CoRR*, abs/2202.11214, 2022. URL https://arxiv.org/abs/2202.11214.

William Peebles and Saining Xie. Scalable diffusion models with transformers, 2023a. URL https://arxiv.org/abs/2212.09748.

William Peebles and Saining Xie. Scalable diffusion models with transformers, 2023b. URL https://arxiv.org/abs/2212.09748.

Jakiw Pidstrigach, Youssef Marzouk, Sebastian Reich, and Sven Wang. Infinite-dimensional diffusion models. *arXiv preprint arXiv:2302.10130*, 2023.

Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion, 2022.

Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2006.

Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *International conference on machine learning*, pages 1530–1538. PMLR, 2015.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, pages 234–241, 2015.

Dan Rosenbaum and Yair Weiss. The return of the gating network: Combining generative models and discriminative training in natural image priors. In *Advances in Neural Information Processing Systems*, volume 28, 2015.

Amirmojtaba Sabour, Sanja Fidler, and Karsten Kreis. Align your steps: Optimizing sampling schedules in diffusion models, 2024. URL https://arxiv.org/abs/2404.14507.

Amirmojtaba Sabour, Sanja Fidler, and Karsten Kreis. Align your flow: Scaling continuous-time flow map distillation, 2025. URL https://arxiv.org/abs/2506.14603.

Mehdi S. M. Sajjadi, Daniel Duckworth, Aravindh Mahendran, Sjoerd van Steenkiste, Filip Pavetić, Mario Lučić, Leonidas J. Guibas, Klaus Greff, and Thomas Kipf. Object scene representation transformer, 2022. URL https://arxiv.org/abs/2206.06922.

Jianxiong Shen, Antonio Agudo, Francesc Moreno-Noguer, and Adria Ruiz. Conditional-flow nerf: Accurate 3d modelling with reliable uncertainty quantification, 2022. URL https://arxiv.org/abs/2203.10192.

Liyue Shen, John M. Pauly, and Lei Xing. Nerp: Implicit neural representation learning with prior embedding for sparsely sampled image reconstruction. *IEEE Trans. Neural Networks Learn. Syst.*, 35(1):770–782, 2024. doi: 10.1109/TNNLS.2022.3177134. URL https://doi.org/10.1109/TNNLS.2022.3177134.

Yaozhong Shi, Zachary E. Ross, Domniki Asimaki, and Kamyar Azizzadenesheli. Mesh-informed neural operator : A transformer generative approach, 2025a. URL https://arxiv.org/abs/2506.16656.

Yaozhong Shi, Zachary E. Ross, Domniki Asimaki, and Kamyar Azizzadenesheli. Mesh-informed neural operator : A transformer generative approach, 2025b. URL https://arxiv.org/abs/2506.16656.

J. Ryan Shue, Eric Ryan Chan, Ryan Po, Zachary Ankner, Jiajun Wu, and Gordon Wetzstein. 3d neural field generation using triplane diffusion, 2022. URL https://arxiv.org/abs/2211.16677.

Vincent Sitzmann, Michael Zollhöfer, and Gordon Wetzstein. Scene representation networks: Continuous 3d-structure-aware neural scene representations. In *Advances in Neural Information Processing Systems*, 2019.

Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. *Advances in Neural Information Processing Systems*, 33, 2020a.

Vincent Sitzmann, Julien N. P. Martel, Alexander W. Bergman, David B. Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. *CoRR*, abs/2006.09661, 2020b. URL https://arxiv.org/abs/2006.09661.

Vincent Sitzmann, Michael Zollhöfer, and Gordon Wetzstein. Scene representation networks: Continuous 3d-structure-aware neural scene representations, 2020c. URL https://arxiv.org/abs/1906.01618.

Vincent Sitzmann, Semon Rezchikov, William T. Freeman, Joshua B. Tenenbaum, and Frédo Durand. Light field networks: Neural scene representations with single-evaluation rendering. *CoRR*, abs/2106.02634, 2021. URL https://arxiv.org/abs/2106.02634.

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *Int. conf. machine learning*, pages 2256–2265, 2015.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020.

Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models, 2023. URL https://arxiv.org/abs/2303.01469.

Dejan Stancevic, Florian Handke, and Luca Ambrogioni. Entropic time schedulers for generative diffusion models, 2025. URL https://arxiv.org/abs/2504.13612.

Niko Sünderhauf, Jad Abou-Chakra, and Dimity Miller. Density-aware nerf ensembles: Quantifying predictive uncertainty in neural radiance fields, 2022. URL https://arxiv.org/abs/2209.08718.

Makoto Takamoto, Timothy Praditia, Raphael Leiteritz, Daniel MacKinlay, Francesco Alesiani, Dirk Pflüger, and Mathias Niepert. Pdebench: An extensive benchmark for scientific machine learning. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022. URL http://papers.nips.cc/paper_files/paper/2022/hash/0a9747136d411fb83f0cf81820d44afb-Abstract-Datasets_and_Benchmarks.html.

Matthew Tancik, Ben Mildenhall, Terrance Wang, Divi Schmidt, Pratul P. Srinivasan, Jonathan T. Barron, and Ren Ng. Learned initializations for optimizing coordinate-based neural representations. *CoRR*, abs/2012.02189, 2020a. URL https://arxiv.org/abs/2012.02189.

Matthew Tancik, Pratul P Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. In *Advances in Neural Information Processing Systems*, 2020b.

The Event Horizon Telescope Collaboration. First m87 event horizon telescope results. i. the shadow of the supermassive black hole. *The Astrophysical Journal Letters*, 875(1):L1, April 2019. ISSN 2041-8213. doi: 10.3847/2041-8213/ab0ec7. URL http://dx.doi.org/10.3847/2041-8213/ab0ec7.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017a. URL http://arxiv.org/abs/1706.03762.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017b.

Qianqian Wang, Zhicheng Wang, Kyle Genova, Pratul Srinivasan, Howard Zhou, Jonathan T. Barron, Ricardo Martin-Brualla, Noah Snavely, and Thomas Funkhouser. Ibrnet: Learning multi-view image-based rendering, 2021.

Daniel Watson, William Chan, Ricardo Martin-Brualla, Jonathan Ho, Andrea Tagliasacchi, and Mohammad Norouzi. Novel view synthesis with diffusion models, 2022. URL https://arxiv.org/abs/2210.04628.

David R Wessels, David M Knigge, Samuele Papa, Riccardo Valperga, Sharvaree Vadgama, Efstratios Gavves, and Erik J Bekkers. Grounding continuous representations in geometry: Equivariant neural fields, 2025. URL https://arxiv.org/abs/2406.05753.

Christopher Williams, Andrew Campbell, Arnaud Doucet, and Saifuddin Syed. Score-optimal diffusion schedules, 2024. URL https://arxiv.org/abs/2412.07877.

Guandao Yang, Abhijit Kundu, Leonidas J. Guibas, Jonathan T. Barron, and Ben Poole. Learning a diffusion prior for nerfs, 2023. URL https://arxiv.org/abs/2304.14473.

Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelnerf: Neural radiance fields from one or few images, 2021.

Biao Zhang, Jiapeng Tang, Matthias Niessner, and Peter Wonka. 3dshape2vecset: A 3d shape representation for neural fields and generative diffusion models, 2023. URL https://arxiv.org/abs/2301.11445.

Bowen Zhang, Yiji Cheng, Jiaolong Yang, Chunyu Wang, Feng Zhao, Yansong Tang, Dong Chen, and Baining Guo. Gaussiancube: A structured and explicit radiance representation for 3d generative modeling, 2024a. URL https://arxiv.org/abs/2403.19655.

Huijie Zhang, Aliaksandr Siarohin, Willi Menapace, Michael Vasilkovsky, Sergey Tulyakov, Qing Qu, and Ivan Skorokhodov. Alphaflow: Understanding and improving meanflow models, 2025. URL https://arxiv.org/abs/2510.20771.

Shuyi Zhang, Ke Liu, Jingjun Gu, Xiaoxu Cai, Zhihua Wang, Jiajun Bu, and Haishuai Wang. Attention beats linear for fast implicit neural representation generation, 2024b. URL https://arxiv.org/abs/2407.15355.

Wenliang Zhao, Lujia Bai, Yongming Rao, Jie Zhou, and Jiwen Lu. Unipc: A unified predictor-corrector framework for fast sampling of diffusion models, 2023. URL https://arxiv.org/abs/2302.04867.

Ellen D. Zhong, Tristan Bepler, Bonnie Berger, and Joseph H. Davis. Cryodrgn: reconstruction of heterogeneous cryo-em structures using neural networks. *Nature Methods*, 18(2):176–185, February 2021. ISSN 1548-7105. doi: 10.1038/s41592-020-01049-4. URL http://dx.doi.org/10.1038/s41592-020-01049-4.

Bernt Øksendal. Stochastic Differential Equations, 2003.