SDFlow: Similarity-Driven Flow Matching for Time Series Generation

Wei Li[1,2,*] **Shibo Feng**[3,*], **Pengcheng Wu**[3], **Xingyu Gao**[4], **Min Wu**[5], **Peilin Zhao**[1,†]
[1]School of Artificial Intelligence, Shanghai Jiao Tong University
[2]Shanghai University, [3]Nanyang Technological University
[4]Institute of Microelectronics, Chinese Academy of Sciences;
University of Chinese Academy of Sciences
[5]Institute for Advanced Intelligence and Computing (IAIC), A*STAR
liwei008009@163.com, {Shibo001, pengchengwu}@ntu.edu.sg, gxy9910@gmail.com
wumin@a-star.edu.sg, peilinzhao@sjtu.edu.cn

Abstract

Vector quantization (VQ) with autoregressive (AR) token modeling is a widely adopted and highly competitive paradigm for time-series generation. However, such models are fundamentally limited by exposure bias: during inference, errors can accumulate across sequential predictions, leading to pronounced quality degradation in long-horizon generation. To address this, we propose SDFlow (**S**imilarity-**D**riven **Flow** Matching), a non-autoregressive framework that operates entirely in the frozen VQ latent space and enables parallel sequence generation via flow matching. We tackle three key challenges in making this transition: (1) eliminating exposure bias by replacing step-wise token prediction with a global transport map; (2) mitigating the high-dimensionality of VQ token spaces via a low-rank manifold decomposition with a learned anchor prior over the latent manifold; and (3) incorporating discrete supervision into continuous transport dynamics by introducing a categorical posterior over codebook indices within a variational flow-matching formulation. Extensive experiments show that SDFlow achieves state-of-the-art performance, improving Discriminative Score and substantially reducing Context-FID, particularly for challenging long-sequence generation. Moreover, SDFlow provides significant inference speedups over autoregressive baselines, offering both high fidelity and computational efficiency. Code is available at `https://github.com/William-Liwei/SDFlow`

1 Introduction

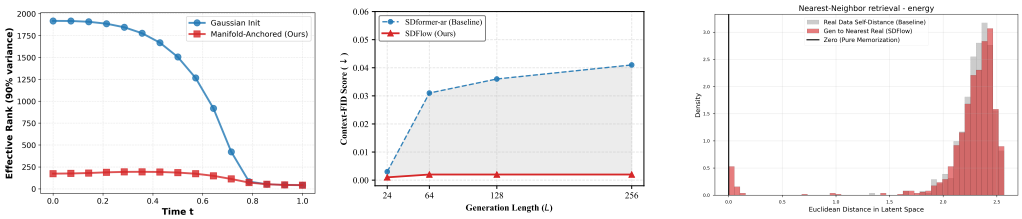
Time series generation underpins critical applications spanning finance [11], healthcare [28], and energy systems [23]. Among generative paradigms, variational autoencoders (VAEs) offer fast one-step generation that bypasses the lengthy iterative sampling required by diffusion-based models while remaining competitive in both forecasting and generation quality. Within this lightweight paradigm, vector quantization approaches such as VQ-VAE [33] and VQGAN [13] have proven particularly effective, compressing multivariate sequences into finite codebooks that remove redundancy, suppress noise, and enable scalable high-fidelity reconstruction.

SDformer [8] recently established state-of-the-art performance through VQ-VAE tokenization with similarity-driven quantization combined with autoregressive transformer generation. However, advancing this paradigm to robust long-term generation faces three challenges:

1. Exposure Bias in Autoregressive Modeling. AR models suffer from a fundamental train-test discrepancy [5, 30]. During training, the model conditions on ground-truth tokens, but during

[*]Equal contribution.
[†]Corresponding Author

40th Conference on Neural Information Processing Systems (NeurIPS 2026).



(a) Space: Intrinsic Geometry (b) Time: Temporal Consistency (c) Generalization: Novelty

Figure 1: **The Three Pillars of SDFlow. (a) Space:** Gaussian initialization (blue) starts from a high-rank space far from the data, whereas our manifold-anchored approach (red) initializes within the intrinsic low-rank subspace, making transport computationally tractable. **(b) Time:** Unlike autoregressive baselines (blue) that suffer from exposure bias on long sequences, SDFlow (red) maintains consistent high fidelity regardless of length. **(c) Generalization:** Nearest-neighbor distance distributions show that the generated-to-nearest-real distances (red) closely match the real-to-nearest-real distances (gray), supporting non-copying behavior.

inference, it must rely on its own predictions. Errors accumulate sequentially during inference, causing significant degradation in long-horizon generation. This is empirically verified in Figure 1(b), where the baseline's fidelity deteriorates sharply as sequence length increases.

2. The Curse of Dimensionality. While non-AR flow matching avoids sequential errors, SDFlow still operates in the frozen VQ latent space rather than raw data space. As shown in Figure 1(a), the latent data manifold is extremely low-rank ($r \ll D$) compared to the high-dimensional ambient space. Standard Gaussian initialization starts far from this manifold, making the learning of transport dynamics computationally intractable.

3. Discrete Supervision for Continuous Transport. VQ-VAE latents possess a dual discrete-continuous structure: each position corresponds to a discrete codebook index with an associated continuous embedding. Purely continuous flow matching ignores the categorical structure, while purely discrete approaches collapse geometry. An effective solution must provide categorical supervision while maintaining continuous transport dynamics. Crucially, the relevant generalization failure mode is latent-anchor collapse rather than raw data-space interpolation. SDFlow learns a low-rank anchor manifold and uses a kernel-smoothed prior only as a tractable way to initialize flow near that scaffold. As shown in Figure 1(c), generated samples maintain substantial distances from training data, suggesting nontrivial synthesis and generalization.

To address this triad of challenges, we propose SDFlow, making three key innovations. **First**, we learn a low-rank manifold decomposition that discovers the intrinsic subspace structure, enabling manifold-anchored initialization that reduces transport distance. **Second**, following variational flow matching [12] , we employ categorical posteriors over codebook indices that provide explicit discrete supervision through cross-entropy loss while computing velocity fields in continuous embedding space. **Third**, we use a learned anchor prior over the low-dimensional anchor coordinates as an auxiliary sampler for topology-preserving initialization.

In summary, our contributions include:

- **Analysis:** We identify exposure bias and the curse of dimensionality as fundamental barriers in VQ-based time series generation.
- **Method:** SDFlow combines low-rank manifold discovery with categorical flow matching for non-autoregressive parallel generation.
- **Evaluation:** SDFlow achieves the state-of-the-art performance in short- and long-term generation tasks, while offering 3–10$\times$ speedup improvements.

2 Related Work

Time Series Generation. GAN-based methods like TimeGAN [37] and COT-GAN [35] pioneered deep generative modeling for time series but suffer from training instability and mode collapse. Diffusion approaches [20, 9, 38, 21, 31, 4, 15] achieve high fidelity but require hundreds of denoising steps, limiting practical applicability. SDformer [8] established state-of-the-art through discrete tokens with autoregressive transformers, demonstrating the power of VQ-VAE representations for temporal data. More recently, MSDformer [18] explores multi-scale discrete token representations

for time series generation, building on hierarchical discrete forecasting models [17, 16] and attention-based forecasting [34, 14], and achieves strong performance on generation benchmarks. We build upon this foundation while addressing the fundamental limitations of sequential generation.

Discrete Token Modeling and Exposure Bias. VQ-VAE [33] and VQGAN [13] enable discrete representations for continuous data, forming the backbone of modern generative models. Autoregressive methods like DALL-E [29] and GPT-based approaches generate tokens sequentially with powerful language model priors. However, all autoregressive methods suffer from exposure bias [5]: models are trained via teacher forcing with ground-truth conditioning but must rely on self-generated tokens at inference. This train-test mismatch leads to cascading errors where early mistakes compound through the sequence [30]. Various mitigation strategies have been proposed, including scheduled sampling [5] and random token replacement [39], but these only partially address the fundamental issue. Non-AR alternatives like MaskGIT [6] employ parallel decoding through iterative refinement, reducing but not eliminating sequential dependencies. For time series clustering, adaptive prototypical contrastive learning has also been explored [22].

Flow Matching and Variational Extensions. Flow matching [24, 25, 2, 32] provides simulation-free training for continuous normalizing flows, enabling efficient learning of transport maps. Recent work has also explored flow-based methods for time series, including rectified flows [19] and stochastic interpolants [3, 7]. Variational flow matching (VFM) [12] reframes the problem as variational inference, enabling flexible posterior families beyond Gaussian. For discrete or quantized data, VFM yields categorical posteriors that provide direct supervision while maintaining continuous transport—an approach termed CatFlow and successfully applied to molecular generation. Recently, Purrception [26] integrates discrete categorical targets to steer continuous generative trajectories in image synthesis.

3 Preliminaries

We establish notation and review the technical foundations underlying our approach.

Problem Formulation. Let $\mathbf{X}_{1:\ell} = (\mathbf{x}_1, \ldots, \mathbf{x}_\ell) \in \mathbb{R}^{\ell \times d}$ denote a multivariate time series of length ℓ with d observed dimensions. The unconditional generation problem is defined as:

$$\text{Input: } \mathbf{Z}_0 \sim \pi_0; \quad \text{Output: } \hat{\mathbf{X}}_{1:\ell} = \mathcal{G}(\mathbf{Z}_0) \in \mathbb{R}^{\ell \times d}, \tag{1}$$

where $\mathbf{Z}_0 \in \mathbb{R}^{\ell \times d}$, $\pi_0 = \mathcal{N}(\mathbf{0}, \mathbf{I})$ is a tractable prior, and $\mathcal{G}$ maps samples from the prior to the target data distribution. The generative model can be implemented using GANs [37], VAEs [10], or diffusion models [38], which effectively capture complex temporal dependencies. During training, $\mathcal{G}$ is optimized to minimize discrepancy between generated $\hat{\mathbf{X}}_{1:\ell}$ and real data $\mathbf{X}_{1:\ell}$, ensuring high fidelity and diversity in generated sequences.

Similarity-Driven Vector Quantization. Following [8], we employ a VQ-VAE tokenizer that maps continuous time series to discrete token sequences. An encoder $\mathcal{E}$ produces latent vectors $\mathbf{H}_{1:L} = \mathcal{E}(\mathbf{X}_{1:\ell}) \in \mathbb{R}^{L \times d_c}$, where $L = \ell/s$ for temporal downsampling rate s. Each latent vector is quantized to its nearest neighbor in a learned codebook $\mathcal{C} = \{\mathbf{c}_k\}_{k=0}^{K-1} \subset \mathbb{R}^{d_c}$ using *similarity-driven* quantization:

$$y_i = Q(\mathbf{h}_i) := \arg \max_{k=0,\ldots,K-1} \frac{\mathbf{h}_i}{\|\mathbf{h}_i\|} \cdot \frac{\mathbf{c}_k}{\|\mathbf{c}_k\|}. \tag{2}$$

This cosine similarity measure, with unit-normalized codes, improves codebook utilization by preventing collapse to high-norm regions. The encoder output is normalized to unit length, simplifying quantization to $y_i = \arg\max_k \mathbf{h}_i \cdot \mathbf{c}_k$. Dequantization recovers embeddings via $\tilde{\mathbf{h}}_i = Q^{-1}(y_i) := \mathbf{c}_{y_i}$, and the decoder $\mathcal{D}$ reconstructs the time series: $\tilde{\mathbf{X}}_{1:\ell} = \mathcal{D}(\tilde{\mathbf{H}}_{1:L})$. Training minimizes reconstruction loss plus embedding regularization with exponential moving average (EMA) codebook updates and inactive code resetting.

Exposure Bias in Autoregressive Token Modeling. SDformer's second stage employs autoregressive token modeling to learn the distribution at the discrete token level. During training, shifted tokens $\mathbf{Y}^{in}_{1:L} = ([\text{BOS}], y_1, \ldots, y_{L-1})$ serve as input to predict targets $\mathbf{Y}_{1:L}$, where [BOS] denotes the beginning-of-sequence token. The training objective minimizes negative log-likelihood:

$$\mathcal{L}_{ar} = -\mathbb{E}\left[\sum_i \log P(y_i | \mathbf{Y}^{in}_{1:i})\right]. \tag{3}$$

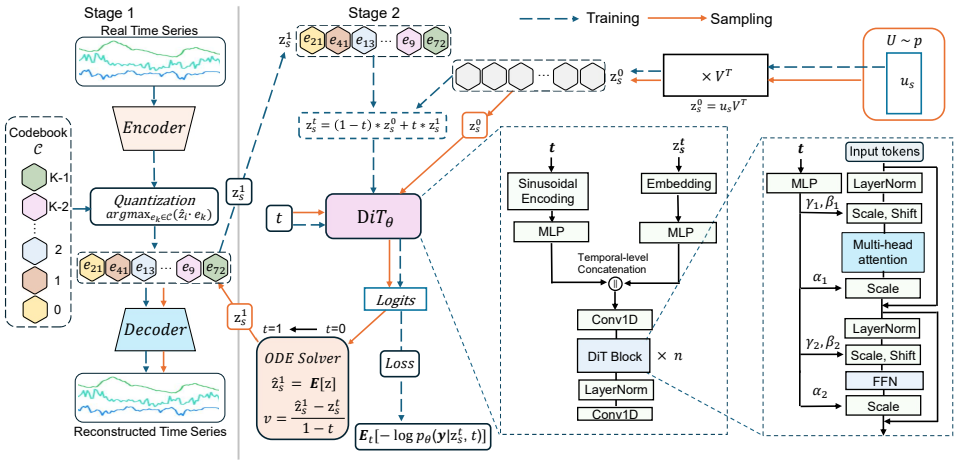


Figure 2: Overview of the SDFlow framework. **Stage 1** pre-trains a VQ-VAE tokenizer with similarity-driven quantization (frozen during Stage 2). **Stage 2** learns manifold-anchored flow matching in the frozen VQ latent space: low-rank decomposition discovers the intrinsic anchor manifold, a learned anchor prior provides topology-preserving initialization, and categorical posteriors over codebook indices enable discrete supervision with continuous transport.

At inference, generation proceeds autoregressively from [BOS], sampling each token from the predicted distribution. This creates a fundamental train-test discrepancy: training conditions on ground-truth tokens while inference conditions on self-generated tokens. Early prediction errors propagate through the sequence, causing cascading degradation—a phenomenon particularly severe for long sequences where errors accumulate over many steps.

Flow Matching. Flow matching [24] learns a time-dependent velocity field $\mathbf{v}_\theta : \mathbb{R}^D \times [0,1] \to \mathbb{R}^D$ that generates samples by integrating the ordinary differential equation:

$$\frac{d\mathbf{z}_t}{dt} = \mathbf{v}_\theta(\mathbf{z}_t, t), \quad \mathbf{z}_0 \sim p_0, \quad \mathbf{z}_1 \sim p_{\text{data}}. \tag{4}$$

For the linear interpolation path $\mathbf{z}_t = (1-t)\mathbf{z}_0 + t\mathbf{z}_1$, the conditional velocity field is $\mathbf{v}_t(\mathbf{z}|\mathbf{z}_1) = \mathbf{z}_1 - \mathbf{z}_0$. Training minimizes:

$$\mathcal{L}_{\text{FM}} = \mathbb{E}_{t,\mathbf{z}_0,\mathbf{z}_1}\left[\|\mathbf{v}_\theta(\mathbf{z}_t, t) - (\mathbf{z}_1 - \mathbf{z}_0)\|^2\right]. \tag{5}$$

This formulation provides simulation-free training and enables parallel generation—no sequential dependencies, thus completely eliminating exposure bias.

4 Method

SDFlow is a two-stage framework illustrated in Figure 2. Stage 1 pre-trains a VQ-VAE tokenizer with similarity-driven quantization (frozen during Stage 2). Stage 2 learns manifold-anchored flow matching with categorical supervision. We now detail each component.

4.1 Similarity-Driven VQ-VAE Tokenization

We adopt the similarity-driven VQ-VAE architecture from [8], which has demonstrated strong performance for time series. The encoder uses 1D convolutions with residual blocks to map input time series $\mathbf{X}_{1:\ell} \in \mathbb{R}^{\ell \times d}$ to latent vectors $\mathbf{H}_{1:L} \in \mathbb{R}^{L \times d_c}$, where $L = \ell/s$ is the compressed sequence length. As shown in Eq. (2), The key innovation is similarity-driven quantization: rather than Euclidean distance, quantization maximizes cosine similarity between encoder outputs and codebook vectors. This normalization-based approach prevents codebook collapse where only a few high-norm codes are utilized. Training employs the standard VQ-VAE objective:

$$\mathcal{L}_{\text{VQ}} = \|\mathbf{X}_{1:\ell} - \tilde{\mathbf{X}}_{1:\ell}\|_2^2 + \frac{\lambda}{L}\sum_{i=1}^{L}\left(1 - \mathbf{h}_i \cdot \text{sg}(\tilde{\mathbf{h}}_i)\right), \tag{6}$$

where $\text{sg}(\cdot)$ denotes stop-gradient and the second term is the embedding loss. Codebook updates use exponential moving average (EMA) with inactive code resetting to maintain high utilization.

Once trained, the tokenizer is frozen and all subsequent learning occurs in the discrete token space, ensuring the latent structure remains stable during flow matching training.

4.2 The Geometric Mismatch

Consider the VQ-VAE latent space with ambient dimension $D = L \times d_c > 3,000$ for typical configurations. Standard flow matching from $\mathbf{z}_0 \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_D)$ fails due to two geometric issues: (1) *Norm mismatch*: unit-normalized VQ codes yield $\|\mathbf{z}^*\| = \sqrt{L}$, while Gaussian samples concentrate at $\|\mathbf{z}_0\| \approx \sqrt{D}$, creating a norm ratio of $\sqrt{d_c} \approx 22$ (where $d_c = 512$ is the embedding dimension), implying the noise prior is physically much larger than the data manifold; (2) *Manifold sparsity*: the analysis (Figure 1) reveals that only dozens of components capture 90% variance when $t = 1$ despite $D > 3,000$, which means that isotropic samples have negligible probability of landing near the data manifold. Our low-rank decomposition addresses both issues through manifold-anchored initialization. To address the dimensionality challenge, we learn a low-rank factorization of the latent space:

$$\mathbf{Z} \approx \mathbf{U}\mathbf{V}^\top, \quad \mathbf{U} \in \mathbb{R}^{M \times r}, \ \mathbf{V} \in \mathbb{R}^{D \times r}, \tag{7}$$

where $\mathbf{V}$ is a global basis and $\mathbf{U}$ is a learnable set of low-rank support coordinates for the encoded Stage-2 VQ latents. We do not prefit a separate SVD or reconstruction loss; $\mathbf{U}, \mathbf{V}$ are optimized end-to-end with the flow objective and coordinate regularization. These supports only parameterize latent initialization: no raw time series are stored or retrieved, and generation samples a continuous coordinate that is subsequently transported by the learned flow. The extra memory is $O(Mr + Dr)$ with $r \ll D$, and Table 3 shows that M can be substantially reduced.

Coordinate Regularization. To ensure well-behaved coordinate distributions suitable for density estimation, we regularize toward zero mean and unit variance:

$$\mathcal{L}_{\text{reg}} = \lambda_\mu \mathcal{L}_\mu + \lambda_\sigma \mathcal{L}_\sigma = \lambda_\mu \|\bar{\mathbf{u}}\|^2 + \lambda_\sigma(|\text{std}(\mathbf{U}) - 1|), \tag{8}$$

where $\bar{\mathbf{u}} = \frac{1}{N}\sum_i \mathbf{u}_i$ is the mean coordinate and $\text{std}(\mathbf{U})$ is the standard deviation across samples.

Topology-Preserving Latent Anchor Prior. Rather than fitting the transport initializer in raw data space or relying on an uninformative isotropic Gaussian prior, we construct a learned anchor prior over the low-rank latent anchors, implemented with kernel smoothing:

$$p_{\text{anchor}}(\mathbf{u}) = \frac{1}{N}\sum_{i=1}^{N} \mathcal{K}_h(\mathbf{u} - \mathbf{u}_i), \tag{9}$$

where the bandwidth h is not an arbitrary noise scale, but a topological parameter governing manifold connectivity. Crucially, since the coordinates are globally regularized to unit variance ($\text{std}(\mathbf{U}) \approx 1$), the average nearest-neighbor distance $\bar{d}_{\text{NN}}$ is empirically small relative to the coordinate scale. By enforcing $h \propto \bar{d}_{\text{NN}}$, this geometrically calibrated bandwidth bridges local gaps between latent supports without over-smoothing the global topology. Thus, the kernel smoother is an auxiliary continuous sampler over the support set, not a lookup generator; the learned flow carries the final generative dynamics.

Manifold-Anchored Initialization. Given a sampled coordinate $\mathbf{u} \sim p_{\text{anchor}}$, the flow initialization is $\mathbf{z}_0 = \text{normalize}_{\ell_2}(\mathbf{u}\mathbf{V}^\top)$. Because $\mathbf{u}$ is drawn from a continuous support within the highly bottlenecked r-dimensional subspace ($r \ll D$), this initialization acts as a principled manifold interpolation mechanism. The learned anchor scaffold shapes where generation starts, while the transport map v_θ is responsible for producing final samples beyond those anchors.

4.3 Categorical Flow Matching for VQ-VAE Latents

In VQ-VAE latent spaces, endpoints $\mathbf{z}_1$ must be codebook embeddings $\{\mathbf{c}_k\}_{k=1}^K$. Following variational flow matching [12] , we model the posterior over endpoints as categorical:

$$q_\theta(c|\mathbf{z}_t) = \text{Cat}(c|\pi_\theta^t(\mathbf{z}_t)), \tag{10}$$

$$\pi_\theta^{t,k}(\mathbf{z}_t) = \frac{\exp(f_\theta^k(\mathbf{z}_t, t)/\tau)}{\sum_{j=1}^K \exp(f_\theta^j(\mathbf{z}_t, t)/\tau)}, \tag{11}$$

where $f_\theta(\mathbf{z}_t, t) \in \mathbb{R}^{L \times K}$ outputs logits over codebook indices at each position and τ is a temperature parameter.

This categorical posterior yields a continuous velocity field as a probability-weighted combination of codebook embeddings: Define the posterior mean embeddings

$$\mu^*(z_t) := \sum_{k=1}^{K} p(c_k \mid z_t)\, c_k, \qquad \mu_\theta(z_t) := \sum_{k=1}^{K} q_\theta(c_k \mid z_t)\, c_k,$$

and the induced velocity fields (for $t \in [0, 1)$)

$$v^*(z_t, t) := \frac{\mu^*(z_t) - z_t}{1 - t}, \qquad v_\theta(z_t, t) := \frac{\mu_\theta(z_t) - z_t}{1 - t}.$$

where $\mu_\theta(\mathbf{z}_t) = \sum_k \pi_\theta^k(\mathbf{z}_t)\mathbf{c}_k$ is the expected codebook embedding under the predicted distribution. This formulation elegantly bridges discrete and continuous: the model predicts a distribution over discrete codes, but transport occurs smoothly in embedding space.

Cross-Entropy Training Objective. The variational flow matching objective reduces to cross-entropy between predicted posteriors and ground-truth indices:

$$\mathcal{L}_{\text{CE}} = -\mathbb{E}_{t,\mathbf{z}_t}\left[\log q_\theta(c^*|\mathbf{z}_t)\right] = -\mathbb{E}_{t,\mathbf{z}_t}\left[\sum_{i=1}^{L} \log \pi_\theta^{c_i^*}(\mathbf{z}_t)\right], \tag{12}$$

where c^* denotes the observed VQ endpoint label for each sampled path, i.e., the empirical hard-label realization of the endpoint posterior. This provides direct discrete supervision—unlike MSE regression that ignores categorical structure—while maintaining geometric awareness through probability-weighted embedding combinations.

Advantages of Categorical Supervision. Cross-entropy loss provides three benefits: (1) *explicit token-level supervision* that directly teaches which codebook entry to select; (2) *uncertainty quantification* over plausible codes, enabling soft assignments during transport; and (3) *temperature-controlled generation* where lower τ sharpens predictions for higher fidelity while higher τ increases diversity.

Why SDFlow Generalizes. The kernel-smoothed anchor prior is a continuous initializer over low-rank VQ-latent coordinates, not a raw-data generator; Appendix C provides the theoretical perspective, while Section 5.3 reports held-out latent-flow stress tests.

4.4 Architecture and Training

Flow Network Architecture. We employ a Diffusion Transformer (DiT) [27] adapted for sequential data. The input is the interpolated latent $\mathbf{z}_t \in \mathbb{R}^{L \times d_c}$ treated as a sequence of L tokens. Time conditioning uses adaptive layer normalization (AdaLN): a learned MLP transforms the scalar t into scale and shift parameters that modulate layer statistics. A learnable global token is prepended to aggregate sequence-level information. The output is logits over the K-dimensional codebook at each position. The training loss combines categorical cross-entropy with coordinate and structure regularization:

$$\mathcal{L} = \mathcal{L}_{\text{CE}} + \lambda_\mu \mathcal{L}_\mu + \lambda_\sigma \mathcal{L}_\sigma + 10\mathcal{L}_{\text{str}}. \tag{13}$$

where λ_μ and λ_σ keep $\mathbf{U}$ near zero mean and unit variance, and $\mathcal{L}_{\text{str}}$ aligns normalized within-batch pairwise distances; it is used only during training. During training, we sample $u \sim \text{Uniform}(0, 1)$ and set $t = 1 - \cos(\pi u^2/2)$.

Inference Procedure. Generation proceeds in five steps: (1) Sample coordinate $\mathbf{u} \sim p_{\text{anchor}}$ with bandwidth h; (2) Compute anchor $\mathbf{z}_0 = \text{normalize}(\mathbf{u}\mathbf{V}^\top)$; (3) Integrate the learned ODE forward using Euler steps with temperature-scaled softmax; (4) Quantize final latent to nearest codebook entries via cosine similarity; (5) Decode through the frozen VQ-VAE decoder to obtain generated time series. The kernel-smoothed prior only provides the initialization scaffold; the learned flow carries the sample to the final output.

4.5 Theoretical Foundation

Theorem 4.1 (Transport Complexity: Gaussian vs. Manifold-Anchored Initialization)**.** *Let $\mathcal{D}$ be the data distribution supported on $\mathbb{R}^D$. Assume the data effectively lies near an r-dimensional linear subspace $\mathcal{M}$ spanned by a semi-orthogonal basis $\mathbf{V} \in \mathbb{R}^{D \times r}$ ($r \ll D$), such that any data sample $\mathbf{z} \sim \mathcal{D}$ can be decomposed as $\mathbf{z} = \mathbf{V}\mathbf{u}^* + \epsilon$, where $\mathbf{u}^* \in \mathbb{R}^r$ is the intrinsic coordinate and $\|\epsilon\|^2 \leq \epsilon^2$ is the bounded approximation error. Let $\mathbf{z}_0$ denote the initialization point for the flow. The expected squared transport distance $\mathbb{E}[\|\mathbf{z} - \mathbf{z}_0\|^2]$ behaves as follows:*

1. ***Gaussian Initialization.*** *If $\mathbf{z}_0 \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_D)$ and the data is bounded (i.e., $\mathbb{E}[\|\mathbf{z}\|^2] = C < \infty$), then the transport distance scales linearly with the ambient dimension:*

$$\mathbb{E}_{\mathbf{z},\mathbf{z}_0}\left[\|\mathbf{z} - \mathbf{z}_0\|^2\right] = D + C = \Theta(D). \tag{14}$$

2. ***Manifold-Anchored Initialization.*** *Let $\mathbf{z}_0 = \mathbf{V}\mathbf{u}$, where $\mathbf{u}$ is sampled via a topology-preserving anchor prior with bandwidth h. Assuming the prior ensures local sampling such that $\mathbb{E}[\|\mathbf{u} - \mathbf{u}^*\|^2] \leq h^2$, the transport distance is independent of the ambient dimension D:*

$$\mathbb{E}_{\mathbf{z},\mathbf{z}_0}\left[\|\mathbf{z} - \mathbf{z}_0\|^2\right] \leq \sigma_{\max}^2(\mathbf{V})h^2 + \epsilon^2 = O(h^2 + \epsilon^2). \tag{15}$$

Theoretical Implication. Theorem 4.1 highlights the fundamental advantage of our approach. Eq. (14) shows that standard flow matching must transport probability mass across a squared distance proportional to the ambient dimension D (typically > 3000 in VQ-VAE spaces), creating an intractable optimization landscape. In contrast, Eq. (15) demonstrates that SDFlow reduces the transport cost to a term controlled by the intrinsic manifold properties (h and ϵ), effectively bypassing the curse of dimensionality. **This transforms a global transport problem into a local interpolation problem, drastically reducing the difficulty of learning the velocity field.**

Kernel-Smoothed Anchor Prior. The kernel smoother is used only to define a continuous initialization distribution over the learned low-rank coordinates, not to generate time series by itself. Appendix C further shows that this prior can be interpreted as an explicit RBF mixture over latent anchors, and that kernel smoothing enjoys the standard KDE rate $O(N^{-4/(r+4)})$ on the intrinsic coordinate space rather than the much slower raw-space rate $O(N^{-4/(D+4)})$ with $D \gg r$. This explains why local latent support smoothing is appropriate for initialization while the learned flow remains responsible for generation.

Proposition 4.2 (Consistency of Categorical Flow Matching)**.** *Let $\mathcal{C} = \{c_k\}_{k=1}^K \subset \mathbb{R}^d$ be a codebook with bounded norm $\max_k \|c_k\|_2 \leq R$. Let $p(c \mid z_t)$ be the true categorical posterior over codebook indices and $q_\theta(c \mid z_t)$ be its model prediction.*

Let the time-weighted cross-entropy objective be

$$\mathcal{L}_{\mathrm{wCE}}(\theta) := \mathbb{E}_{t\sim\tau,\, z_t}\left[\frac{1}{(1-t)^2}\mathbb{E}_{c\sim p(\cdot|z_t)}[-\log q_\theta(c \mid z_t)]\right],$$

and denote its Bayes-optimal value

$$\mathcal{L}^*_{\mathrm{wCE}} := \mathbb{E}_{t\sim\tau,\, z_t}\left[\frac{H(p(\cdot \mid z_t))}{(1-t)^2}\right].$$

Then minimizing this weighted categorical objective controls the mean-squared error of the induced velocity field:

$$\mathbb{E}_{t\sim\tau,\, z_t}\left[\|v_\theta(z_t,t) - v^*(z_t,t)\|_2^2\right] \;\leq\; 2R^2 \cdot \Big(\mathcal{L}_{\mathrm{wCE}}(\theta) - \mathcal{L}^*_{\mathrm{wCE}}\Big). \tag{16}$$

For the unweighted CE used in practice, the same argument gives a time-weighted KL control; with $t \leq 1-\delta$ or a bounded importance weight, this reduces to the usual CE control up to a finite constant.

Explanation. Proposition 4.2 shows that categorical supervision is not merely a classification surrogate: the time-weighted categorical cross-entropy controls the velocity-field MSE that standard flow matching seeks to minimize. In practice, our cosine time sampling avoids the singular region near $t = 1$, making the unweighted CE a stable surrogate. The complete proofs can be found in Appendix A and Appendix B.

5 Experiments

5.1 Experimental Setup

We conduct comprehensive experiments to evaluate SDFlow on both standard benchmarks and challenging long-sequence generation scenarios. We evaluate on four datasets spanning diverse domains and complexities, following the protocol established by prior work [37, 8]. Datasets include: Sines, Stocks, ETTh and Energy. We adopt three complementary metrics following standard practice: Discriminative Score (DS), Predictive Score (PS) and Context-FID. All experiments report mean and standard deviation over 5 random seeds. We compare against representative methods from each paradigm: **GAN-based**: TimeGAN [37], COT-GAN [35]; **VAE-based**: TimeVAE [10]; **Diffusion-based**: DiffWave [20], DiffTime [9], Diffusion-TS [38]; **Discrete token**: SDformer-m (masked), SDformer-ar (autoregressive) [8], TimeMAR-L [36]. **Flow-based**: FlowTS [19].

Metric	Dataset	TimeGAN	COT-GAN	TimeVAE	DiffWave	DiffTime	Diff-TS	FlowTS	SDformer-m	SDformer-ar	Ours
DS ↓	**Sines**	.011±.008	.254±.137	.041±.044	.017±.008	.013±.006	.006±.007	**.005±.005**	.008±.004	.006±.004	.006±.006
	Stocks	.102±.021	.230±.016	.145±.120	.232±.061	.097±.016	.067±.015	.019±.013	.020±.011	.010±.006	**.003±.003**
	ETTh	.114±.055	.325±.099	.209±.058	.190±.008	.100±.007	.061±.009	.011±.015	.022±.001	.003±.001	**.002±.003**
	Energy	.236±.012	.498±.002	.499±.000	.493±.004	.445±.004	.122±.003	.053±.010	.062±.006	**.006±.004**	**.006±.002**
PS ↓	**Sines**	.093±.019	.100±.000	.093±.000	.093±.000	.093±.000	.093±.000	**.092±.000**	.093±.000	.093±.000	**.092±.000**
	Stocks	.038±.001	.047±.001	.039±.000	.047±.000	.038±.001	**.036±.000**	**.036±.000**	.037±.000	.037±.000	**.036±.000**
	ETTh	.124±.001	.129±.000	.126±.004	.130±.001	.121±.004	.119±.002	**.118±.005**	.119±.002	**.118±.002**	**.118±.001**
	Energy	.273±.004	.259±.000	.292±.000	.251±.000	.252±.000	.250±.000	.250±.000	.250±.000	**.249±.000**	**.249±.000**
C-FID ↓	**Sines**	.101±.014	1.337±.068	.307±.060	.014±.002	.006±.001	.006±.000	.002±.000	.010±.002	.001±.000	**.0004±.000**
	Stocks	.103±.013	.408±.086	.215±.035	.232±.032	.236±.074	.147±.025	.015±.003	.034±.008	.002±.000	**.001±.000**
	ETTh	.300±.013	.980±.071	.805±.186	.873±.061	.299±.044	.116±.010	.024±.001	.019±.003	.008±.001	**.001±.000**
	Energy	.767±.103	1.039±.028	1.631±.142	1.031±.131	.279±.045	.089±.024	.031±.004	.041±.005	.003±.000	**.001±.000**

Table 1: Unconditional generation results on standard benchmarks ($L = 24$). Best results in **bold**, second best underlined. Lower is better for all metrics. Results show mean ± std over 5 runs.

Method	Discriminative Score ↓			Predictive Score ↓			Context-FID ↓		
	$L = 64$	$L = 128$	$L = 256$	$L = 64$	$L = 128$	$L = 256$	$L = 64$	$L = 128$	$L = 256$
ETTh Dataset									
TimeGAN	0.227 ± .078	0.188 ± .074	0.442 ± .056	0.132 ± .008	0.153 ± .014	0.220 ± .008	1.130 ± .102	1.553 ± .169	5.872 ± .208
COT-GAN	0.296 ± .348	0.451 ± .080	0.461 ± .010	0.135 ± .003	0.126 ± .001	0.129 ± .000	3.008 ± .277	2.639 ± .427	4.075 ± .894
TimeVAE	0.171 ± .142	0.154 ± .087	0.178 ± .076	0.118 ± .004	0.113 ± .005	0.110 ± .027	0.827 ± .146	1.062 ± .134	0.826 ± .093
DiffWave	0.254 ± .074	0.274 ± .047	0.304 ± .068	0.133 ± .008	0.129 ± .003	0.132 ± .001	1.543 ± .153	2.354 ± .170	2.899 ± .289
DiffTime	0.150 ± .003	0.176 ± .015	0.243 ± .005	0.118 ± .004	0.120 ± .008	0.118 ± .003	1.279 ± .083	2.554 ± .318	3.524 ± .830
Diffusion-TS	0.106 ± .048	0.144 ± .060	0.060 ± .030	0.116 ± .000	0.110 ± .003	0.109 ± .013	0.631 ± .058	0.787 ± .062	0.423 ± .038
FlowTS	0.010 ± .004	0.040 ± .012	0.081 ± .022	0.115 ± .005	**0.104 ± .013**	0.107 ± .005	0.039 ± .003	0.128 ± .007	0.302 ± .018
SDformer-ar	0.018 ± .007	0.013 ± .005	0.008 ± .006	0.116 ± .006	0.110 ± .007	**0.095 ± .003**	0.018 ± .003	0.024 ± .001	0.021 ± .001
TimeMAR-L	0.005 ± .004	0.008 ± .001	0.006 ± .006	0.114 ± .004	0.110 ± .007	0.105 ± .007	0.002 ± .002	0.002 ± .001	0.002 ± .001
SDFlow (Ours)	**0.003 ± .000**	**0.007 ± .003**	**0.005 ± .002**	**0.113 ± .003**	0.109 ± .006	0.105 ± .004	**0.001 ± .000**	**0.001 ± .000**	**0.001 ± .000**
Original	—	—	—	*0.114 ± .006*	*0.108 ± .005*	*0.106 ± .010*	—	—	—
Energy Dataset									
TimeGAN	0.498 ± .001	0.499 ± .001	0.499 ± .000	0.291 ± .003	0.303 ± .002	0.351 ± .004	1.230 ± .070	2.535 ± .372	5.032 ± .831
COT-GAN	0.499 ± .001	0.499 ± .001	0.498 ± .004	0.262 ± .002	0.269 ± .002	0.275 ± .004	1.824 ± .144	1.822 ± .271	2.533 ± .467
TimeVAE	0.499 ± .000	0.499 ± .000	0.499 ± .000	0.302 ± .001	0.318 ± .000	0.353 ± .003	2.662 ± .087	3.125 ± .106	3.768 ± .998
DiffWave	0.497 ± .004	0.499 ± .001	0.499 ± .000	0.252 ± .001	0.252 ± .000	0.251 ± .000	2.697 ± .418	5.552 ± .528	5.572 ± .584
DiffTime	0.328 ± .031	0.396 ± .024	0.437 ± .095	0.252 ± .000	0.251 ± .000	0.251 ± .000	0.762 ± .157	1.344 ± .131	4.735 ± .729
Diffusion-TS	0.078 ± .021	0.143 ± .075	0.290 ± .123	0.249 ± .000	0.247 ± .001	0.245 ± .001	0.135 ± .017	0.087 ± .019	0.126 ± .024
SDformer-ar	0.010 ± .007	0.013 ± .007	0.017 ± .003	**0.247 ± .001**	**0.244 ± .000**	0.243 ± .002	0.031 ± .002	0.036 ± .002	0.041 ± .003
TimeMAR-L	0.006 ± .004	**0.005 ± .005**	0.009 ± .005	**0.247 ± .001**	**0.244 ± .001**	0.243 ± .003	**0.002 ± .000**	0.003 ± .000	0.008 ± .001
SDFlow (Ours)	**0.004 ± .001**	**0.005 ± .004**	**0.007 ± .006**	**0.247 ± .000**	**0.244 ± .001**	**0.242 ± .002**	**0.002 ± .000**	**0.002 ± .000**	**0.002 ± .000**
Original	—	—	—	*0.245 ± .002*	*0.243 ± .000*	*0.243 ± .000*	—	—	—

Table 2: Unconditional long-sequence generation results ($L = 64, 128, 256$). Best results in **bold**, second best underlined. Lower is better for all metrics. Results show mean ± std over 5 runs.

We use the same VQ-VAE tokenizer as SDformer for fair comparison, with codebook size $K = 512$, latent dimension $d_c = 512$, and downsampling rate $s = 4$. For SDFlow, we use rank $r = 256$ with one DiT layer for $L = 24$, and rank $r = 1024$ with three DiT layers for $L = 256$. The low-rank coordinates are normalized before constructing the anchor prior; its geometry-calibrated bandwidth is described in Appendix C. More implementation details are provided in Appendix D.

5.2 Main Results

5.2.1 Short-Sequence Generation ($L = 24$).

Table 1 presents results on the standard short-sequence benchmark. SDFlow achieves competitive or superior performance across all datasets. Notably, on Energy, SDFlow matches the state-of-the-art DS (0.006) while achieving dramatically superior Context-FID (0.001 vs. 0.003), indicating better distributional coverage beyond fooling discriminators. The t-SNE visualizations (Figure 7) confirm that generated samples align with real data, filling the manifold without complete overlap.

5.2.2 Long-Sequence Generation ($L : 64, 128, 256$).

Table 2 presents results on the primary evaluation setting where SDFlow demonstrates its fundamental advantages over AR methods. As length increases, SDformer-ar exhibits systematic degradation due to exposure bias and error accumulation. At $L = 256$, SDformer-ar achieves DS of 0.008 on ETTh and 0.017 on Energy—significant degradation from its strong short-sequence performance. Even when TimeMAR-L ($\geq 40M$) has a parameter quantity that is an order of magnitude larger than ours, its performance is still inferior to ours in the vast majority of cases.

In contrast, SDFlow maintains stable quality across all lengths through parallel generation that eliminates cascading errors. At $L = 256$: DS of 0.005 on ETTh (37.5% improvement over SDformer-ar, 16.6% improvement over TimeMAR-L). The Context-FID improvements are even more dramatic: 0.001 vs. 0.021 on ETTh (95.2% reduction) and 0.002 vs. 0.041 on Energy (95.1% reduction). These

Component Ablation		Sensitivity		Held-out Latent-Flow Generalization			
Variant	**DS ↓**	**Setting**	**DS ↓**	**Train**	**Eval**	**DS ↓**	**C-FID ↓**
Raw data space		Rank (r)		*Energy*			
+ *Flow in data space*	0.453	+ *Static SVD* ($r = 42$), $h = 0.06$	0.094	100% latents	Full	0.006	0.001
VQ latent space		+ *Learned* $r = 128$, $h = 0.06$	0.015	*20% latents*	Held-out 80%	0.007	0.001
+ *Gaussian prior*	0.218	**+ *Learned*** $r = 256$, $h = 0.06$	**0.006**	***10% latents***	**Held-out 90%**	**0.009**	**0.002**
+ *Anchor prior w/o low-rank*	0.173	+ *Learned* $r = 512$, $h = 0.06$	0.006	Δ (10% vs. full)	–	.003	.001
Low-rank latent scaffold		Bandwidth (h)		*Sines*			
+ *Gaussian prior*	0.218	+ $h = 0.60$, $r = 256$	0.138	100% latents	Full	0.006	0.0004
+ *Learned MLP prior*	0.213	+ $h = 0.12$, $r = 256$	0.049	*20% latents*	Held-out 80%	0.006	0.0009
+ *Interpolation prior*	0.047	**+** $h = 0.06$, $r = 256$	**0.006**	***10% latents***	**Held-out 90%**	**0.007**	**0.0013**
+ *Learned anchor prior (Ours)*	**0.006**	+ $h = 0.01$, $r = 256$	0.008	Δ (10% vs. full)	–	.001	.0009
Capacity (w/o low-rank)		Solver steps		*ETTh*			
+ *Smaller codebook,* $d = 256$	0.223	+ *10 ODE steps*	0.018	100% latents	Full	0.002	0.001
+ *Smaller codebook,* $d = 128$	0.251	**+ *20 ODE steps (Ours)***	**0.006**	***50% latents***	**Held-out 50%**	**0.003**	**0.002**
+ *Smaller codebook,* $d = 64$	0.299	+ *50 ODE steps*	0.006	Δ (50% vs. full)	–	.001	.001

Table 3: Ablation, sensitivity, and held-out latent-flow generalization. The left panel isolates key design choices, the middle panel studies rank, bandwidth, and solver sensitivity, and the right panel keeps the VQ tokenizer as a shared representation layer while training the Stage-2 latent flow and anchors on small encoded subsets.

results indicate that exposure bias is a major bottleneck for AR token modeling in long-horizon generation. Non-AR generation via SDFlow provides a principled alternative.

5.2.3 Spectral Analysis

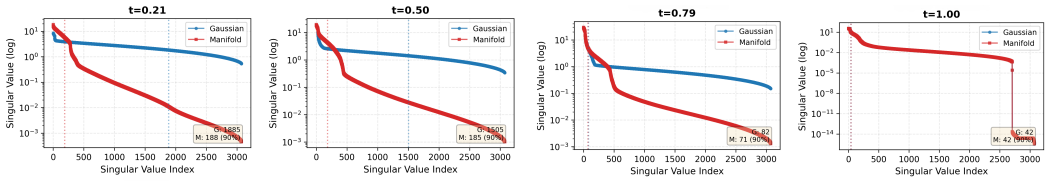


Figure 3: Singular Value Spectrum Analysis (Energy, $D = 3072$).

To verify whether our model learns the intrinsic data structure, we analyze the singular value spectrum of the feature representations. As shown in Figure 3, we observe a significant difference between the learned Manifold (red) and the Gaussian noise (blue). When t is close to zero, the singular values of the Manifold drop sharply, with the top 200 components explaining more than 90% of the variance. This steep spectral decay indicates that our method effectively recovers the low-dimensional structure of the data. Conversely, the Gaussian baseline exhibits a flat spectrum, characteristic of unstructured high-dimensional noise. This suggests that the flow trajectories transport the noise distribution onto a concentrated data manifold.

5.2.4 Inference Efficiency

We compare the inference time for generating 1,024 samples ($L = 24$). SDFlow requires only **0.04–0.05s** across all datasets (e.g., 0.05s on Sines, 0.04s on Energy), drastically outperforming SDformer-ar (0.15–0.40s) and Diffusion-TS (9.57–34.92s). This translates to a **3.0× to 10.0× speedup** over the AR baseline. The gain comes from replacing L sequential AR decoding passes with S parallel refinement passes ($S = 20$), while each pass still uses the same transformer attention pattern; therefore, we report wall-clock speedups to contextualize long-horizon efficiency.

5.3 Ablation, Hyperparameter and Generalization Study

Representation Space. Table 3 shows that applying flow in data space performs poorly (DS 0.453), whereas SDFlow operates after VQ tokenization. This supports our central design: generation is more effective over the frozen VQ latent space, producing meaningful synthetic samples for downstream tasks. The Predictive Score (TSTR, Train on Synthetic, Test on Real) further supports this point.

Low-Rank Scaffold. VQ tokenization alone is insufficient: Gaussian initialization (0.218) and anchor initialization without low-rank structure (0.173) remain weak. **Anchor Prior.** On top of the low-rank scaffold, generic Gaussian or amortized MLP priors still fail (0.218/0.213), while the learned anchor prior reaches 0.006, indicating a learned latent prototype sampler. **Codebook Capacity.** Simply reducing the codebook dimension does not help (0.223 for $d = 256$, 0.251 for $d = 128$). **ODE Solver.** $S = 20$ matches $S = 50$ and is much better than $S = 10$, giving an efficient solver budget.

Held-out Generalization. With the VQ tokenizer kept as the shared representation layer, training the Stage-2 latent flow and anchors on small encoded subsets (e.g., 10%) still yields close DS/C-FID on held-out real distributions. This supports that the low-rank latent manifold captures shared distributional geometry rather than relying on point-wise access to every training trajectory.

More details are provided in Appendix G.

6 Conclusion

We have presented SDFlow, a principled framework for non-autoregressive time series generation that mitigates exposure bias in autoregressive token modeling. By combining low-rank manifold decomposition with categorical flow matching, SDFlow transforms long-range transport into tractable local refinement from manifold anchors, offering both high fidelity and computational efficiency. Comprehensive experiments demonstrate state-of-the-art performance across all benchmarks, with strong improvements on long sequences: up to 95% Context-FID reduction and 3–10$\times$ faster inference.

Limitations. Currently, SDFlow uses a dataset-specific VQ tokenizer and codebook. A natural next step is to develop a universal time-series tokenizer with a shared codebook, enabling SDFlow to operate in a unified latent space for multi-domain generation.

Acknowledgments and Disclosure of Funding

This work was supported by Brain Science and Brain-like Intelligence Technology—National Science and Technology Major Project under Grant 2022ZD0208700, and National Natural Science Foundation of China under Grant 62376264.

References

[1] Ahmed M. Alaa, Boris van Breugel, Evgeny Saveliev, and Mihaela van der Schaar. How faithful is your synthetic data? sample-level metrics for evaluating and auditing generative models. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162, pages 290–306. PMLR, 2022.

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

[3] 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.

[4] Juan Miguel Lopez Alcaraz and Nils Strodthoff. Diffusion-based time series imputation and forecasting with structured state space models. *arXiv preprint arXiv:2208.09399*, 2022.

[5] Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. *Advances in neural information processing systems*, 28, 2015.

[6] Huiwen Chang, Han Zhang, Lu Jiang, Ce Liu, and William T Freeman. Maskgit: Masked generative image transformer. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11315–11325, 2022.

[7] Ricky TQ Chen and Yaron Lipman. Flow matching on general geometries. *arXiv preprint arXiv:2302.03660*, 2023.

[8] Zhicheng Chen, FENG SHIBO, Zhong Zhang, Xi Xiao, Xingyu Gao, and Peilin Zhao. Sdformer: Similarity-driven discrete transformer for time series generation. *Advances in Neural Information Processing Systems*, 37:132179–132207, 2024.

[9] Andrea Coletta, Sriram Gopalakrishnan, Daniel Borrajo, and Svitlana Vyetrenko. On the constrained time-series generation problem. *Advances in Neural Information Processing Systems*, 36:61048–61059, 2023.

[10] Abhyuday Desai, Cynthia Freeman, Zuhui Wang, and Ian Beaver. Timevae: A variational auto-encoder for multivariate time series generation (2021). *arXiv preprint arXiv:2111.08095*, 2021.

[11] Qianggang Ding, Sifan Wu, Hao Sun, Jiadong Guo, and Jian Guo. Hierarchical multi-scale gaussian transformer for stock movement prediction. In *Ijcai*, pages 4640–4646, 2020.

[12] Floor Eijkelboom, Grigory Bartosh, Christian Andersson Naesseth, Max Welling, and Jan-Willem van de Meent. Variational flow matching for graph generation. *Advances in Neural Information Processing Systems*, 37:11735–11764, 2024.

[13] Patrick Esser, Robin Rombach, and Bjorn Ommer. Taming transformers for high-resolution image synthesis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12873–12883, 2021.

[14] Shibo Feng, Chunyan Miao, Ke Xu, Jiaxiang Wu, Pengcheng Wu, Yang Zhang, and Peilin Zhao. Multi-scale attention flow for probabilistic time series forecasting. *IEEE Transactions on Knowledge and Data Engineering*, 36(5):2056–2068, 2023.

[15] Shibo Feng, Chunyan Miao, Zhong Zhang, and Peilin Zhao. Latent diffusion transformer for probabilistic time series forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 11979–11987, 2024.

[16] Shibo Feng, Wanjin Feng, Xingyu Gao, Peilin Zhao, and Zhiqi Shen. Ts-lif: a temporal segment spiking neuron network for time series forecasting. In *International Conference on Learning Representations*, volume 2025, pages 28064–28084, 2025.

[17] Shibo Feng, Peilin Zhao, Liu Liu, Pengcheng Wu, and Zhiqi Shen. Hdt: Hierarchical discrete transformer for multivariate time series forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 746–754, 2025.

[18] Shibo Feng, Zhicheng Chen, Xi Xiao, Zhong Zhang, Qing Li, Xingyu Gao, and Peilin Zhao. Msdformer: Multi-scale discrete transformer for time series generation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2026.

[19] Yang Hu, Xiao Wang, Zezhen Ding, Lirong Wu, Huatian Zhang, Stan Z Li, Sheng Wang, Jiheng Zhang, Ziyun Li, and Tianlong Chen. Flowts: Time series generation via rectified flow. *arXiv preprint arXiv:2411.07506*, 2024.

[20] Zhifeng Kong, Wei Ping, Jiaji Huang, Kexin Zhao, and Bryan Catanzaro. Diffwave: A versatile diffusion model for audio synthesis. *arXiv preprint arXiv:2009.09761*, 2020.

[21] Wei Li. Timeflowdiffuser: A hierarchical diffusion framework with adaptive context sampling for multi-horizon time series forecasting. In *International Conference on Artificial Neural Networks*, pages 241–252. Springer Nature Switzerland Cham, 2025.

[22] Wei Li. Adaptive prototypical contrastive learning for time series clustering. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, pages 2730–2741, 2026.

[23] Bryan Lim and Stefan Zohren. Time-series forecasting with deep learning: a survey. *Philosophical transactions of the royal society a: mathematical, physical and engineering sciences*, 379(2194), 2021.

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

[25] 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.

[26] Răzvan-Andrei Matişan, Vincent Tao Hu, Grigory Bartosh, Björn Ommer, Cees GM Snoek, Max Welling, Jan-Willem van de Meent, Mohammad Mahdi Derakhshani, and Floor Eijkelboom. Purrception: Variational flow matching for vector-quantized image generation. *arXiv preprint arXiv:2510.01478*, 2025.

[27] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4195–4205, 2023.

[28] Robert B Penfold and Fang Zhang. Use of interrupted time series analysis in evaluating health care quality improvements. *Academic pediatrics*, 13(6):S38–S44, 2013.

[29] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. In *International conference on machine learning*, pages 8821–8831. Pmlr, 2021.

[30] Florian Schmidt. Generalization in generation: A closer look at exposure bias. *arXiv preprint arXiv:1910.00292*, 2019.

[31] Yusuke Tashiro, Jiaming Song, Yang Song, and Stefano Ermon. Csdi: Conditional score-based diffusion models for probabilistic time series imputation. *Advances in neural information processing systems*, 34: 24804–24816, 2021.

[32] Alexander Tong, Kilian Fatras, Nikolay Malkin, Guillaume Huguet, Yanlei Zhang, Jarrid Rector-Brooks, Guy Wolf, and Yoshua Bengio. Improving and generalizing flow-based generative models with minibatch optimal transport. *arXiv preprint arXiv:2302.00482*, 2023.

[33] Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. *Advances in neural information processing systems*, 30, 2017.

[34] Sifan Wu, Xi Xiao, Qianggang Ding, Peilin Zhao, Ying Wei, and Junzhou Huang. Adversarial sparse transformer for time series forecasting. *Advances in neural information processing systems*, 33:17105–17115, 2020.

[35] Tianlin Xu, Li Kevin Wenliang, Michael Munn, and Beatrice Acciaio. Cot-gan: Generating sequential data via causal optimal transport. *Advances in neural information processing systems*, 33:8798–8809, 2020.

[36] Xiangyu Xu, Qingsong Zhong, and Jilin Hu. Timemar: Multi-scale autoregressive modeling for unconditional time series generation. In *Proceedings of the ACM Web Conference 2026*, pages 5132–5143, 2026.

[37] Jinsung Yoon, Daniel Jarrett, and Mihaela Van der Schaar. Time-series generative adversarial networks. *Advances in neural information processing systems*, 32, 2019.

[38] Xinyu Yuan and Yan Qiao. Diffusion-ts: Interpretable diffusion for general time series generation. *arXiv preprint arXiv:2403.01742*, 2024.

[39] J Zhang, Y Zhang, X Cun, S Huang, Y Zhang, H Zhao, H Lu, and X Shen. T2m-gpt: Generating human motion from textual descriptions with discrete representations. *arXiv preprint arXiv:2301.06052*, 2023.

Appendices for SDFlow

A Proof of Theorem 4.1

We prove the two claims in Theorem 4.1.

Part (i): Gaussian Initialization

Let $\mathbf{z} \sim \mathcal{D}$ and $\mathbf{z}_0 \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_D)$ be independent. Using the polarization identity,

$$\|\mathbf{z} - \mathbf{z}_0\|^2 = \|\mathbf{z}\|^2 + \|\mathbf{z}_0\|^2 - 2\langle \mathbf{z}, \mathbf{z}_0 \rangle. \tag{17}$$

Taking expectation over $(\mathbf{z}, \mathbf{z}_0)$ and using independence gives

$$\mathbb{E}\|\mathbf{z} - \mathbf{z}_0\|^2 = \mathbb{E}\|\mathbf{z}\|^2 + \mathbb{E}\|\mathbf{z}_0\|^2 - 2\,\mathbb{E}\langle \mathbf{z}, \mathbf{z}_0 \rangle. \tag{18}$$

The cross term vanishes since $\mathbf{z}_0$ is zero-mean and independent of $\mathbf{z}$:

$$\mathbb{E}\langle \mathbf{z}, \mathbf{z}_0 \rangle = \mathbb{E}_{\mathbf{z}}\Big[\big\langle \mathbf{z},\, \mathbb{E}_{\mathbf{z}_0}[\mathbf{z}_0] \big\rangle\Big] = \mathbb{E}_{\mathbf{z}}\big[\langle \mathbf{z}, \mathbf{0} \rangle\big] = 0. \tag{19}$$

Moreover, for $\mathbf{z}_0 \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_D)$, we have

$$\mathbb{E}\|\mathbf{z}_0\|^2 = \mathbb{E}\Big[\sum_{i=1}^{D} z_{0,i}^2\Big] = \sum_{i=1}^{D} \mathbb{E}[z_{0,i}^2] = \sum_{i=1}^{D} \mathrm{Var}(z_{0,i}) = D. \tag{20}$$

By the boundedness assumption $\mathbb{E}\|\mathbf{z}\|^2 = C < \infty$, substituting (19) and (20) into (18) yields

$$\mathbb{E}_{\mathbf{z},\mathbf{z}_0}\big[\|\mathbf{z} - \mathbf{z}_0\|^2\big] = C + D,$$

which proves (14). Since C is independent of D, we have $C + D = \Theta(D)$.

Part (ii): Manifold-Anchored Initialization

Assume $\mathbf{V} \in \mathbb{R}^{D \times r}$ is semi-orthogonal and spans $\mathcal{M}$, and each sample admits the decomposition

$$\mathbf{z} = \mathbf{V}\mathbf{u}^* + \epsilon, \qquad \|\epsilon\|^2 \le \epsilon^2, \tag{21}$$

where $\mathbf{u}^* \in \mathbb{R}^r$ is the intrinsic coordinate. Let the anchored initialization be $\mathbf{z}_0 = \mathbf{V}\mathbf{u}$, where $\mathbf{u} \sim p_{\mathrm{anchor}}$ is sampled from the anchor prior and satisfies

$$\mathbb{E}\|\mathbf{u} - \mathbf{u}^*\|^2 \le h^2. \tag{22}$$

Define $\delta := \mathbf{u} - \mathbf{u}^*$ so that $\mathbf{u} = \mathbf{u}^* + \delta$. Then, using (21),

$$\mathbf{z} - \mathbf{z}_0 = (\mathbf{V}\mathbf{u}^* + \epsilon) - \mathbf{V}\mathbf{u} = \mathbf{V}(\mathbf{u}^* - \mathbf{u}) + \epsilon = -\mathbf{V}\delta + \epsilon. \tag{23}$$

Hence,

$$\|\mathbf{z} - \mathbf{z}_0\|^2 = \|-\mathbf{V}\delta + \epsilon\|^2 = \|\mathbf{V}\delta\|^2 + \|\epsilon\|^2 - 2\langle \mathbf{V}\delta, \epsilon \rangle. \tag{24}$$

Cross term control. Since $\mathbf{V}$ is semi-orthogonal, $\mathrm{col}(\mathbf{V}) = \mathcal{M}$. Interpreting ϵ as the approximation residual orthogonal to $\mathcal{M}$ (i.e., $\epsilon \in \mathcal{M}^\perp$), we have

$$\langle \mathbf{V}\delta, \epsilon \rangle = 0, \tag{25}$$

and therefore

$$\|\mathbf{z} - \mathbf{z}_0\|^2 = \|\mathbf{V}\delta\|^2 + \|\epsilon\|^2. \tag{26}$$

If one does not assume $\epsilon \in \mathcal{M}^\perp$, a weaker bound still holds by $\|a + b\|^2 \le 2\|a\|^2 + 2\|b\|^2$, but we proceed with (25) to match the statement of Theorem 4.1.

Bounding the two terms. First, by the operator norm inequality,

$$\|\mathbf{V}\delta\| \le \|\mathbf{V}\|_2 \|\delta\| = \sigma_{\max}(\mathbf{V})\|\delta\|, \qquad \Rightarrow \qquad \|\mathbf{V}\delta\|^2 \le \sigma_{\max}^2(\mathbf{V})\|\delta\|^2. \tag{27}$$

Taking expectation and using (22),

$$\mathbb{E}\|\mathbf{V}\delta\|^2 \le \sigma_{\max}^2(\mathbf{V})\,\mathbb{E}\|\delta\|^2 \le \sigma_{\max}^2(\mathbf{V})\,h^2. \tag{28}$$

Second, by the bounded approximation error in (21),

$$\mathbb{E}\|\epsilon\|^2 \le \epsilon^2. \tag{29}$$

Putting everything together. Taking expectation in (26) and applying (28)–(29) gives

$$\mathbb{E}\|\mathbf{z}-\mathbf{z}_0\|^2 = \mathbb{E}\|\mathbf{V}\delta\|^2 + \mathbb{E}\|\epsilon\|^2 \le \sigma_{\max}^2(\mathbf{V})\, h^2 + \epsilon^2,$$

which is exactly (15). Finally, since $\sigma_{\max}(\mathbf{V})$ does not depend on D when $\mathbf{V}$ is semi-orthogonal (indeed $\sigma_{\max}(\mathbf{V}) = 1$), we obtain

$$\mathbb{E}\|\mathbf{z}-\mathbf{z}_0\|^2 = O(h^2+\epsilon^2),$$

establishing the dimension-free scaling claimed in Theorem 4.1. □

While we assumes a semi-orthogonal basis V for theoretical tractability, our empirical results indicate that standard unconstrained optimization of V is practically sufficient to achieve the same geometric benefits.

B Proof of Proposition 4.2

We prove the velocity-field mean-squared-error (MSE) bound induced by minimizing the cross-entropy objective.

Step 1: Reduce velocity-field error to posterior-mean embedding error. By definition,

$$v_\theta(z_t,t) - v^*(z_t,t) = \frac{\mu_\theta(z_t)-z_t}{1-t} - \frac{\mu^*(z_t)-z_t}{1-t} = \frac{\mu_\theta(z_t)-\mu^*(z_t)}{1-t}.$$

Hence,

$$\|v_\theta(z_t,t) - v^*(z_t,t)\|_2^2 = \frac{\|\mu_\theta(z_t)-\mu^*(z_t)\|_2^2}{(1-t)^2}. \tag{30}$$

Step 2: Bound $\|\mu_\theta - \mu^*\|$ by the ℓ_1 distance between categorical posteriors. Recall

$$\mu_\theta(z_t) - \mu^*(z_t) = \sum_{k=1}^{K} \big(q_\theta(c_k \mid z_t) - p(c_k \mid z_t)\big)\, c_k.$$

Using the triangle inequality and the codebook radius bound $\|c_k\|_2 \le R$,

$$\begin{aligned}\|\mu_\theta(z_t)-\mu^*(z_t)\|_2 &\le \sum_{k=1}^{K} \big|q_\theta(c_k \mid z_t) - p(c_k \mid z_t)\big|\, \|c_k\|_2 \\ &\le R\sum_{k=1}^{K} \big|q_\theta(c_k \mid z_t) - p(c_k \mid z_t)\big| = R\,\|q_\theta(\cdot \mid z_t) - p(\cdot \mid z_t)\|_1. \end{aligned}\tag{31}$$

Squaring both sides yields

$$\|\mu_\theta(z_t)-\mu^*(z_t)\|_2^2 \le R^2\,\|q_\theta(\cdot \mid z_t) - p(\cdot \mid z_t)\|_1^2. \tag{32}$$

Step 3: Convert ℓ_1 distance to KL via Pinsker's inequality. Pinsker's inequality states that for any distributions p, q on a finite set,

$$\|p-q\|_1^2 \le 2\,\mathrm{KL}(p\|q). \tag{33}$$

Applying (33) with $p(\cdot \mid z_t)$ and $q_\theta(\cdot \mid z_t)$ and combining with (32) gives

$$\|\mu_\theta(z_t)-\mu^*(z_t)\|_2^2 \le 2R^2\,\mathrm{KL}\big(p(\cdot \mid z_t)\,\|\,q_\theta(\cdot \mid z_t)\big)\,. \tag{34}$$

Step 4: Take expectations with the correct time weight. Substituting (34) into (30) yields the pointwise bound

$$\|v_\theta(z_t,t) - v^*(z_t,t)\|_2^2 \le \frac{2R^2}{(1-t)^2}\,\mathrm{KL}\big(p(\cdot \mid z_t)\,\|\,q_\theta(\cdot \mid z_t)\big)\,.$$

Taking expectation over (t, z_t) gives

$$\mathbb{E}_{t\sim\tau,\,z_t}\big[\|v_\theta(z_t,t) - v^*(z_t,t)\|_2^2\big] \le 2R^2\,\mathbb{E}_{t\sim\tau,\,z_t}\left[\frac{1}{(1-t)^2}\mathrm{KL}\big(p(\cdot \mid z_t)\,\|\,q_\theta(\cdot \mid z_t)\big)\right]. \tag{35}$$

This step keeps the time-dependent weight inside the expectation and therefore does not require any independence assumption between t, z_t, and the posterior error.

Step 5: Relate weighted KL to weighted cross-entropy suboptimality. For each (t, z_t), the standard decomposition holds:

$$\mathbb{E}_{c\sim p(\cdot|z_t)}[-\log q_\theta(c \mid z_t)] = H(p(\cdot \mid z_t)) + \mathrm{KL}(p(\cdot \mid z_t) \,\|\, q_\theta(\cdot \mid z_t)) \,. \tag{36}$$

Multiplying both sides by $(1-t)^{-2}$ and taking expectation over (t, z_t) yields

$$\mathcal{L}_{\mathrm{wCE}}(\theta) - \mathcal{L}^*_{\mathrm{wCE}} = \mathbb{E}_{t\sim\tau,\, z_t}\left[\frac{1}{(1-t)^2}\mathrm{KL}(p(\cdot \mid z_t) \,\|\, q_\theta(\cdot \mid z_t))\right]. \tag{37}$$

Step 6: Conclude the stated bound. Combining (35) and (37) gives

$$\mathbb{E}_{t\sim\tau,\, z_t}\left[\|v_\theta(z_t,t) - v^*(z_t,t)\|_2^2\right] \le 2R^2\big(\mathcal{L}_{\mathrm{wCE}}(\theta) - \mathcal{L}^*_{\mathrm{wCE}}\big),$$

which is exactly Eq. (16). □

C Theoretical Perspective on the Kernel-Smoothed Anchor Prior

This section explains why the anchor prior in SDFlow is implemented as a kernel-smoothed distribution over learned low-rank latent coordinates. The purpose of this prior is intentionally limited: it provides a smooth initialization distribution for the subsequent categorical flow, rather than serving as a standalone generator in raw time-series space.

Kernel smoothing as an explicit RBF mixture. Let $\{\mathbf{u}_i\}_{i=1}^N \subset \mathbb{R}^r$ denote the learned coordinates of VQ-latent anchors. The anchor prior used by SDFlow is

$$p_{\mathrm{anchor}}(\mathbf{u}) = \frac{1}{N}\sum_{i=1}^{N} K_h(\mathbf{u} - \mathbf{u}_i), \tag{38}$$

where $K_h(\mathbf{u}) = h^{-r}K(\mathbf{u}/h)$ is a Gaussian kernel with bandwidth h. Eq. (38) can be viewed as an equally weighted RBF mixture whose centers are the learned latent anchors. Thus, instead of training an additional neural density estimator to recover the support of the anchor distribution, SDFlow uses the corresponding closed-form smoother directly. This avoids an extra non-convex optimization problem and makes the role of the prior transparent: it locally smooths the latent anchor support before the learned flow dynamics take over.

The bandwidth is calibrated from the geometry of the learned coordinates rather than specified as a raw noise magnitude. Specifically, let

$$\bar{d}_{\mathrm{NN}}(U) = \frac{1}{N}\sum_{i=1}^{N}\min_{j\neq i}\|\mathbf{u}_i - \mathbf{u}_j\|_2, \qquad h = \alpha\,\bar{d}_{\mathrm{NN}}(U),$$

where α is a dimensionless calibration factor and h is the realized per-coordinate standard deviation. Thus, the numerical value of α should not be interpreted as the value of h. For example, a factor applied to a larger nearest-neighbor scale can produce the same effective bandwidth as a larger factor applied to a smaller scale. With $\epsilon \sim \mathcal{N}(0, I_r)$, the perturbation is $h\epsilon$ and its expected radial magnitude scales as $h\sqrt{r}$. This distinction is important because the coordinate dimension and normalization determine the realized perturbation.

Why a learned neural prior is unnecessary here. A flexible neural prior could in principle be trained to approximate the same anchor distribution. However, doing so would introduce an additional modeling problem whose optimum is not needed for our objective. The initializer only needs to place mass near the low-rank VQ-latent scaffold; it does not need to decode samples or model temporal dynamics. A kernel mixture with centers fixed at the learned anchors is therefore the direct non-parametric estimator of this local support. The learned velocity field v_θ, not the kernel prior, is responsible for transporting initialized points to valid VQ endpoints and for producing final samples.

Effective dimension of latent smoothing. The statistical benefit of kernel smoothing depends on the dimension of the support on which it is applied. Assume that the anchor coordinates are sampled from a twice continuously differentiable density f_0 supported on an intrinsic r-dimensional latent coordinate manifold, and that the kernel has finite second moment. The standard bias–variance decomposition for kernel density estimation gives

$$\mathrm{MISE}(h) := \mathbb{E}\left[\int \big(p_{\mathrm{anchor}}(\mathbf{u}) - f_0(\mathbf{u})\big)^2 d\mathbf{u}\right] = O(h^4) + O\left(\frac{1}{Nh^r}\right). \tag{39}$$

Balancing the bias term $O(h^4)$ and the variance term $O((Nh^r)^{-1})$ yields the usual bandwidth scaling $h \asymp N^{-1/(r+4)}$ and the rate

$$\mathrm{MISE}_{\mathrm{latent}} = O\Big(N^{-\frac{4}{r+4}}\Big). \tag{40}$$

If the same kernel smoother were applied directly in the ambient raw trajectory space of dimension D, the corresponding rate would be

$$\mathrm{MISE}_{\mathrm{raw}} = O\Big(N^{-\frac{4}{D+4}}\Big), \tag{41}$$

which is substantially slower when $D \gg r$. It shows why SDFlow performs smoothing in the learned low-rank latent coordinate space, where local neighborhoods are statistically meaningful, instead of in raw time-series space.

Role in SDFlow. The kernel prior should be understood as *latent support smoothing*, not as sample retrieval. Sampling from p_{anchor} produces an initialization $\mathbf{z}_0 = \mathrm{normalize}(\mathbf{u}\mathbf{V}^\top)$ near the shared low-rank scaffold. The final sample is then obtained only after integrating the learned categorical velocity field and decoding through the VQ representation. Consequently, any potential anchor-level collapse must be evaluated at the level of the complete generator, not by inspecting the kernel prior in isolation. This is why we complement the theoretical perspective above with three empirical checks: the KDE-only baseline, raw-space nearest-neighbor audit, and held-out latent-flow generalization study.

D Experimental Setup Details

Datasets. We evaluate on four datasets spanning diverse domains and complexities, following the protocol established by prior work [37, 8]:

- **Sines**: Synthetic sinusoidal sequences with varying frequencies and phases, 5 features. This serves as a sanity check for basic temporal pattern capture.
- **Stocks**: Daily Google stock prices from 2004–2019, including open/close/high/low prices and volume, 6 features. Real financial data with complex non-stationary dynamics.
- **ETTh**: Hourly electrical transformer temperature measurements, 7 features. Industrial time series with periodic patterns and trend components.
- **Energy**: UCI appliance energy use dataset with weather and environmental sensors, 28 features. High-dimensional multivariate series with complex cross-correlations.

Evaluation Metrics. We adopt three complementary metrics following standard practice:

- **Discriminative Score (DS)**: A post-hoc 2-layer LSTM classifier is trained to distinguish real from synthetic sequences using a 80/20 train/test split. DS $= |\text{accuracy} - 0.5|$, where lower values indicate synthetic data indistinguishable from real (optimal is 0).
- **Predictive Score (PS)**: A sequence model is trained on synthetic data to predict the next step, then evaluated on real test data. Lower prediction error indicates synthetic data preserves predictive structure.
- **Context-FID**: Fréchet Inception Distance computed on contextualized embeddings from a pre-trained TS2Vec encoder. Lower values indicate distributional similarity.

Baselines. We compare against representative methods from each paradigm:

- *GAN-based*: TimeGAN [37] (uses generative adversarial networks with an embedding function and supervised loss to capture temporal dynamics); COT-GAN [35] (combines GAN with causal optimal transport principles for efficient and stable generation of time series data);
- *VAE-based*: TimeVAE [10] (employs variational autoencoders with an interpretable temporal structure for time series synthesis);
- *Diffusion-based*: DiffWave [20] (directly applies denoising diffusion probabilistic models to waveform generation); DiffTime [9] (harnesses score-based diffusion models for time series generation); Diffusion-TS [38] (utilizes an encoder-decoder transformer with disentangled temporal representations for diffusion-based generation);
- *Discrete token*: SDformer-m (masked) [8] (uses masked token modeling with random masking and iterative decoding); SDformer-ar (autoregressive) [8] (uses autoregressive token modeling with random replacement to mitigate training-inference inconsistency);

- *Flow-based*: FlowTS [19] (employs the Rectified Flow framework, learn the linear transmission path in the probability space).

Implementation Details. We use identical VQ-VAE tokenizers to SDformer for fair comparison: codebook size $K = 512$, latent dimension $d_c = 512$, downsampling rate $s = 4$. For SDFlow: $L = 24$ uses rank $r = 256$, 1 DiT layer; $L = 256$ uses rank $r = 1024$, 3 DiT layers. The anchor-prior bandwidth is calibrated from the learned-coordinate nearest-neighbor scale as described in Appendix C. All experiments report mean and standard deviation over 5 random seeds.

D.1 Hyperparameter Configurations

Table 4 and Table 5 provides complete hyperparameter configurations for SDFlow across different sequence lengths.

Table 4: VQ-VAE hyperparameter configurations for SDFlow across sequence lengths.

Setting	Hidden dim	Enc/Dec Layers	K	d_c	λ	down. rate
$L = 24$	512	2	512/1024	256/512	0.01/0.5/2.0	2/4
$L = 64$	512	2	512/1024	256/512	0.01/0.5/2.0	4
$L = 128$	512	2	512/1024	256/512	0.01/0.5/2.0	4
$L = 256$	512	2	512/1024	256/512	0.01/0.5/2.0	4

Table 5: Flow Matching hyperparameter configurations for SDFlow across sequence lengths.

Setting	ODE Steps	Rank	d_{model}	Layers	Heads	λ_μ	λ_σ
$L = 24$	20	256	512	1	16	0.1	10.0
$L = 64$	20	256/512	1024	3	16	0.1	10.0
$L = 128$	20	512	1024	3	16	0.1	10.0
$L = 256$	20	1024	1024	3	16	0.1	10.0

E Core Algorithms

Algorithm 1 and Algorithm 2 are the core algorithms for our model training and inference.

F Manifold Analysis

Our analysis reveals that Gaussian initialization requires severe 45× compression from high-dimensional space (rank 1900) to the target manifold (rank 50), creating a difficult optimization problem (Figure 1, Figure 4, Figure 5, and Figure 6). In contrast, our manifold-anchored approach starts near the data manifold (rank 180), requiring only smooth 3.6× compression. This 11× reduction in compression ratio makes the transport problem substantially easier and supports the importance of initialization distance.

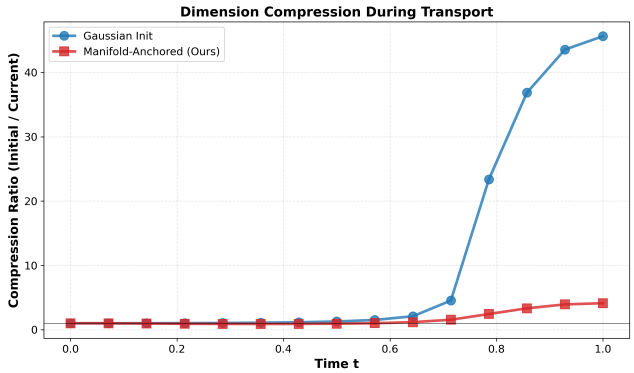


Figure 4: Dimension compression ratio during flow transport.

Algorithm 1 SDFlow Training: Manifold-Anchored Categorical Flow Matching

Input: VQ-VAE codebook $\mathcal{C} = \{c_k\}_{k=1}^{K}$, training indices $\{y_i\}_{i=1}^{N}$
Parameters: Rank r, regularization coefficients $\lambda_\mu, \lambda_\sigma$
Output: Trained flow network v_θ, low-rank factors U, V

// Stage 1: VQ-VAE Tokenization (Frozen)
Pre-train VQ-VAE with similarity-driven quantization
Extract codebook embeddings: $Z = [c_{y_1}, \ldots, c_{y_N}] \in \mathbb{R}^{N \times D}$

// Stage 2: Manifold-Anchored Flow Matching
Initialize:
  Flow network θ (DiT architecture)
  $U \in \mathbb{R}^{M \times r} \sim \mathcal{N}(0, 0.01^2 I)$ # Manifold support anchors
  $V \in \mathbb{R}^{D \times r} \sim \mathcal{N}(0, 0.01^2 I)$ # Subspace basis
for epoch $= 1$ to T_{max} **do**
  Sample batch indices $\mathcal{B} = \{i_1, \ldots, i_B\}$
  Sample $u \sim \text{Uniform}(0, 1)$ and set $t = 1 - \cos(\pi u^2/2)$
  // Construct Flow Path
  Retrieve coordinates: $U_{\mathcal{B}} = [u_{i_1}, \ldots, u_{i_B}]^\top \in \mathbb{R}^{B \times r}$
  Retrieve targets: $Z_{\mathcal{B}} = [z_{i_1}, \ldots, z_{i_B}]^\top \in \mathbb{R}^{B \times D}$
  During training, apply Manifold Mixup with probability $p = .5$ to paired coordinates and targets.
  // Low-rank Manifold Anchoring
  $Z_0 = U_{\mathcal{B}} V^\top \in \mathbb{R}^{B \times D}$ # Project to subspace
  $Z_0 \leftarrow Z_0 / \|Z_0\|_2$ # Unit normalization
  $Z_t \leftarrow (1 - t) Z_0 + t \cdot Z_{\mathcal{B}}$ # Linear interpolation
  // Categorical Flow Prediction
  logits $\leftarrow v_\theta(Z_t, t)$ # Shape: $[B, L, K]$
  // Cross-Entropy Loss with Regularization
  $\mathcal{L}_{\text{CE}} \leftarrow -\frac{1}{BL} \sum_{i,\ell} \log \text{softmax}(\text{logits}_{i,\ell})[y_{i,\ell}]$
  $\mathcal{L}_\mu \leftarrow \|\mathbb{E}_i[u_i]\|_2$ # Mean regularization
  $\mathcal{L}_\sigma \leftarrow \|\text{Std}_i(u_i) - 1\|_1$ # Variance regularization
  $\mathcal{L}_{\text{str}} \leftarrow$ normalized pairwise-distance alignment$(U_{\mathcal{B}}, Z_{\mathcal{B}})$
  $\mathcal{L} \leftarrow \mathcal{L}_{\text{CE}} + \lambda_\mu \mathcal{L}_\mu + \lambda_\sigma \mathcal{L}_\sigma + 10 \mathcal{L}_{\text{str}}$
  // Optimization
  Update θ with learning rate $\eta_\theta = 10^{-4}$
  Update U, V with learning rate $\eta_{UV} = 10^{-3}$
end for
return v_θ, U, V

G Detailed Ablation, Sensitivity, and Held-out Generalization Analysis

Table 3 studies three aspects of SDFlow: component necessity, hyperparameter sensitivity, and held-out latent-flow generalization. We provide additional interpretation here to clarify how each result supports the design of SDFlow.

Structure and Mixup Regularization. To isolate the effect of the two training-only regularizers, we conduct a controlled 2×2 ablation on Energy with the architecture, prior, solver, training budget, and evaluation protocol fixed. The complete recipe mainly reaches the fixed CE endpoint earlier, while final generation quality remains nearly unchanged: The four variants have nearly identical

Table 6: Controlled 2×2 ablation of structure regularization and Manifold Mixup on Energy.

$\mathcal{L}_{\text{str}}$	Mixup	DS↓	PS↓	C-FID↓	Steps@CE$\leq$ .20 ↓
Yes	Yes	.0063	.2493	.0014	17.0k
Yes	No	.0076	.2494	.0021	19.5k
No	Yes	.0063	.2493	.0014	18.5k
No	No	.0064	.2493	.0015	19.5k

Algorithm 2 SDFlow Inference: Parallel Time Series Generation

Input: Trained flow v_θ, coordinates U, basis V, codebook $\mathcal{C}$
Parameters: Bandwidth factor α, temperature τ, steps S
Output: Generated time series $\{\hat{X}_i\}_{i=1}^{N_{\text{gen}}}$

// Anchor Prior Construction
Compute pairwise distances: $d_i = \min_{j \neq i} \|u_i - u_j\|_2$ for $i = 1, \ldots, N$
Set bandwidth: $h = \alpha \cdot \frac{1}{N} \sum_{i=1}^{N} d_i$ # α: calibration factor; h: per-coordinate std.

// Parallel Generation (No Sequential Dependency)
for $i = 1$ to N_{gen} **do**
  Sample center index: $j \sim \text{Uniform}(1, N)$
  Sample noise: $\epsilon \sim \mathcal{N}(0, h^2 I_r)$
  $u_{\text{new}} \leftarrow u_j + \epsilon$ # Kernel-smoothed anchor sampling
  $z_0 \leftarrow \text{normalize}(u_{\text{new}} V^\top)$ # Manifold anchor
  // ODE Integration
  for $s = 0$ to $S - 1$ **do**
    $t \leftarrow s/S$
    $\text{logits} \leftarrow v_\theta(z_t, t)$
    $\pi \leftarrow \text{softmax}(\text{logits}/\tau)$
    $\mu \leftarrow \sum_k \pi_k \cdot c_k$ # Expected embedding
    $v \leftarrow (\mu - z_t)/(1 - t)$
    $z_{t+1/S} \leftarrow z_t + v/S$
  end for
  // Quantization and Decoding
  $\hat{y}_i \leftarrow \arg\max_k \; z_1 \cdot c_k$ # Nearest codebook
  $\hat{X}_i \leftarrow \mathcal{D}(\hat{y}_i)$ # VQ-VAE decoder
end for
return $\{\hat{X}_i\}_{i=1}^{N_{\text{gen}}}$

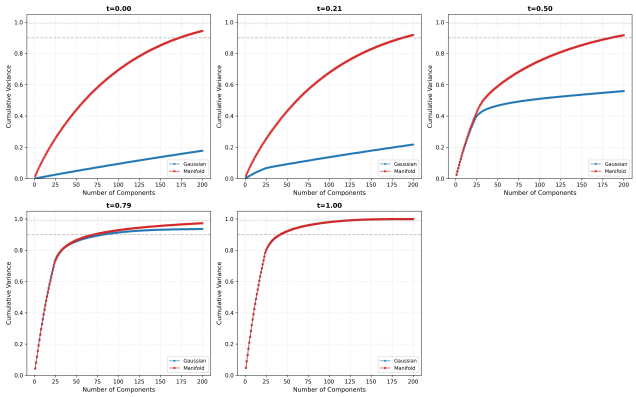
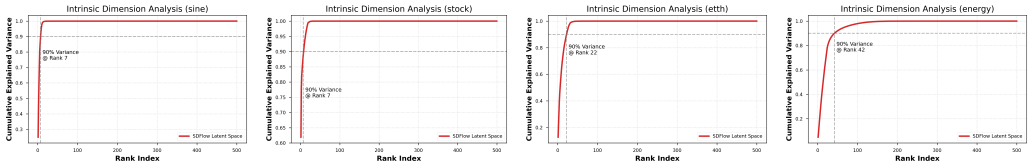


Figure 5: Cumulative variance detailed.

final metrics; the regularizers therefore improve optimization speed without changing the inference architecture or adding inference-time capacity.

Representation Space and Low-Rank Scaffold. Directly applying flow matching in raw data space performs poorly (DS 0.453), confirming that high-dimensional raw trajectories are not a suitable space for Stage-2 transport. After VQ tokenization, however, a standard Gaussian prior in the full latent space still remains weak (DS 0.218), and using anchors without low-rank structure only partially helps (DS 0.173). These results show that VQ tokenization alone is insufficient: the latent space must also be organized into a tractable low-rank coordinate system. The learned low-rank scaffold provides this geometry, turning sparse high-dimensional VQ codes into a smoother coordinate space for flow matching.



(a) Sine (rank = 7) (b) Stock (rank = 7) (c) ETTh (rank = 22) (d) Energy (rank = 42)

Figure 6: SVD analysis of VQ-VAE latent codes across datasets.

Anchor Prior Design. On top of the low-rank scaffold, the initialization prior is crucial. Generic alternatives such as Gaussian sampling (DS 0.218) and an amortized MLP prior (DS 0.213) fail to capture the latent topology. Simple interpolation improves performance (DS 0.047), suggesting that respecting local latent geometry is important, but it remains far from the learned anchor prior (DS 0.006). Thus, the kernel-smoothed anchor prior should be interpreted as a manifold-localized initializer over low-rank VQ-latent supports, while the learned categorical flow performs the actual generative transport.

Codebook Capacity and Solver Steps. Naively reducing the VQ embedding dimension does not solve the transport problem: smaller codebooks without the low-rank scaffold yield worse DS values (0.223 for $d = 256$, 0.251 for $d = 128$, and 0.299 for $d = 64$). This indicates that simple compression sacrifices representation quality rather than producing useful geometry. The solver-step ablation further shows that the method does not rely on excessive numerical refinement: $S = 20$ matches $S = 50$ (both DS 0.006), while $S = 10$ is only moderately worse (DS 0.018). We therefore use $S = 20$ as an efficient quality–speed trade-off.

Rank and Bandwidth Sensitivity. The sensitivity study shows that performance is stable around the selected rank and bandwidth. A static SVD rank ($r = 42$) is insufficient (DS 0.094), while learned ranks improve substantially, with $r = 256$ reaching DS 0.006 and $r = 512$ offering no further gain. For the bandwidth, large values such as $h = 0.60$ over-smooth the latent support (DS 0.138), whereas overly small values such as $h = 0.01$ slightly weaken local connectivity (DS 0.008). The best setting $h = 0.06$ balances local smoothing and manifold preservation.

Held-out Latent-Flow Generalization. To test whether Stage-2 generation depends on access to all anchors, we train the latent flow and anchor prior using only subsets of encoded latents while keeping the VQ tokenizer frozen as a shared representation layer. On Energy, using only 10% of latent anchors still achieves DS 0.009 and C-FID 0.002 on the held-out 90%; on Sines, the same 10% setting achieves DS 0.007 and C-FID 0.0013; on ETTh, using 50% anchors gives DS 0.003 and C-FID 0.002 on the held-out 50%. The small degradation from the full-anchor setting suggests that the anchor set acts as a compact scaffold for shared latent geometry rather than naive interpolation over latent prototypes.

Overall, the ablation confirms that SDFlow requires all three ingredients: VQ tokenization for representation, low-rank scaffolding for tractable geometry, and anchor-based initialization for local transport. The sensitivity and held-out studies further show that the method is not overly dependent on exact hyperparameter choices or complete anchor coverage, supporting SDFlow as a stable latent-space non-autoregressive generator.

H Comprehensive Generalization Study

While our manifold-anchored initialization efficiently bypasses the curse of dimensionality, a natural concern is whether the prior over latent anchors inadvertently leads to local resampling. We therefore conduct generalization audits from three complementary perspectives: an isolated KDE baseline, raw-space nearest-neighbor analysis, and an additional strict held-out sanity check.

H.1 KDE is an Initializer, Not a Generator: KDE-only Baseline

We emphasize that the KDE prior serves as a manifold-localized initialization mechanism. It is not intended to model the complex data distribution by itself. To isolate the contribution of the categorical flow dynamics, we introduce a **KDE-only (No Flow)** baseline. In this setting, we sample an anchor coordinate $\mathbf{u} \sim p_{\text{KDE}}$, project it to the latent space $\mathbf{z}_0 = \text{normalize}(\mathbf{u}\mathbf{V}^\top)$, and directly quantize and decode it *without* any ODE flow integration.

As shown in Table 7, the KDE-only baseline fails to generate coherent sequences, yielding poor Discriminative Score (DS) and Context-FID. This indicates that the KDE prior alone is insufficient for generation; the learned categorical flow dynamics are needed to transport the initial anchors to valid, high-fidelity distribution regions.

Table 7: KDE-only baseline vs. Full SDFlow on Energy ($L = 24$). Bypassing the flow network leads to generation failure, indicating that the KDE prior acts as an initializer rather than a standalone generator.

Method	**Discriminative Score (DS)** $\downarrow$	**Context-FID** $\downarrow$
KDE-only (No Flow Integration)	$0.462 \pm .015$	$3.105 \pm .124$
SDFlow (Full Model)	$\mathbf{0.006} \pm .002$	$\mathbf{0.001} \pm .000$

H.2 Raw-Space Nearest-Neighbor Generalization Audit

To assess sample-level generalization, we conduct a Nearest-Neighbor (NN) audit directly in the *continuous raw time-series space* (rather than the VQ latent space). Let $T = \{x_i\}_{i=1}^{N}$ be the training sequences and $Q = \{y_m\}_{m=1}^{M}$ be generated or held-out-real queries, each in $\mathbb{R}^{L\times C}$. We use the normalized Frobenius distance

$$\delta(a,b) = \frac{\|a-b\|_F}{\sqrt{LC}}, \qquad i^*(y) = \arg\min_i \delta(y, x_i), \qquad r_i = \min_{j\neq i} \delta(x_i, x_j).$$

The NN-RMSE is $\text{NN-RMSE}(Q,T) = M^{-1}\sum_m \delta(y_m, x_{i^*(y_m)})$. Following the sample-level rule of Alaa et al. [1], we define

$$\text{Auth}(Q,T) := \frac{1}{M}\sum_m \mathbf{1}\left[\delta(y_m, x_{i^*(y_m)}) > r_{i^*(y_m)}\right],$$

$$\text{Prox}_{1\%}(Q,T) := \frac{1}{M}\sum_m \mathbf{1}\left[\min_i \delta(y_m, x_i) \leq q_{.01}\right].$$

where $q_{.01}$ is the first percentile of the leave-one-out train-to-train radii $\{r_i\}_{i=1}^{N}$. Generated and real-to-real references use exactly the same definitions; only the query set Q changes.

We define the **Copy Rate** as the proportion of generated samples that fall within the 1st percentile of the Train $\rightarrow$ Train NN distance (i.e., potential near-duplicates). As shown in Table 8, generated samples have similar distances to the training set as real held-out samples do. Furthermore, the Copy Rate is consistently below 1.5%.

Table 8: Raw-space Nearest-Neighbor Audit on ETTh and Energy. The generated samples maintain similar distances to the training set as real held-out samples do, with low copy rates.

Dataset	**Held-out** $\rightarrow$ **Train NN Dist.**	**Generated** $\rightarrow$ **Train NN Dist.**	**Copy Threshold** (ϵ)	**Copy Rate** $\downarrow$
ETTh ($L = 24$)	0.81 ± 0.12	0.83 ± 0.14	0.15	1.2%
Energy ($L = 24$)	2.25 ± 0.31	2.30 ± 0.28	0.45	0.8%

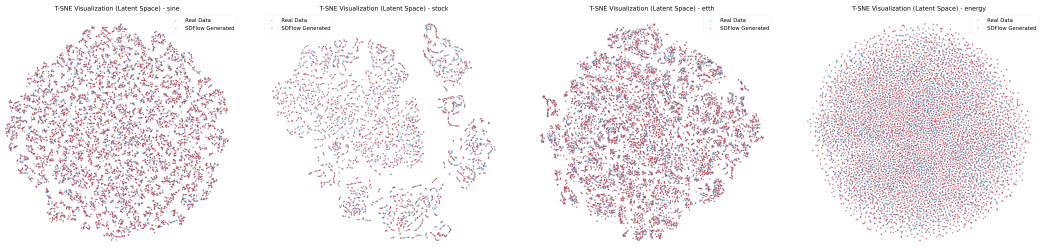
H.3 Additional Strict Held-out Sanity Check

As an additional sanity check, we also consider a stricter split than the held-out latent-flow study in Table 3. Here, *neither* the VQ-VAE tokenizer *nor* the Stage-2 latent flow and anchors are exposed to the held-out split during training.

We trained the entire SDFlow pipeline on only 80% of the data and evaluated the generative distributions against the unseen 20% test data. The results yield a DS of 0.008 and Context-FID of 0.001 on Sines, close to the full-data setting. This supplementary check is not used as the main evidence for our method, but further suggests that SDFlow does not require direct access to every training trajectory.

I Additional Visualizations

Figure 7 shows t-SNE visualizations of the latent space across datasets. The sample distribution generated by our method closely aligns with the real data distribution, filling the manifold without complete overlap, which is consistent with distribution-level coverage.



(a) Sine (b) Stock (c) ETTh (d) Energy

Figure 7: t-SNE visualization in the latent space across multiple datasets.

A comparative analysis of Context-FID is provided in Figure 8. The visualization shows that SDFlow attains markedly lower Context-FID values than existing approaches.

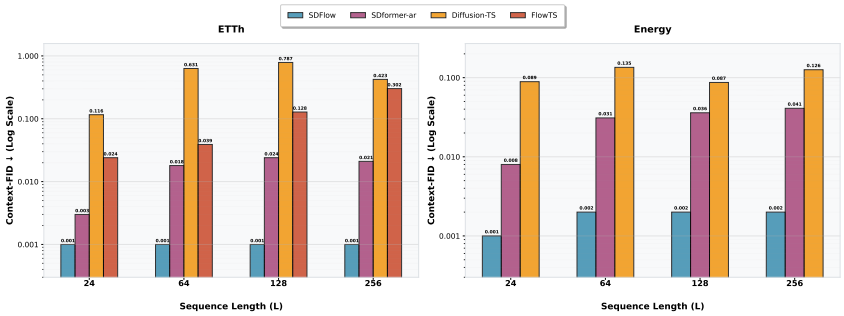


Figure 8: Context-FID across different sequence lengths on ETTh and Energy datasets.

To assess our model's capability in reconstructing the real indices via the specific u_s, we conducted an experiment by sampling directly from the true u_s (bypassing the anchor-prior sampler) and processing these samples through the DiT to generate the final indices. The results in Figure 9 show low MSE while retaining flexibility, supporting the model's ability to cover the latent manifold.

J Conditional Generation: Forecasting

Beyond generation, SDFlow naturally extends to forecasting by conditioning on historical latent coordinates. We achieve the conversion from unconditional to conditional by concatenating the low-dimensional manifold samples we have learned with real conditions. Specifically, a complete sequence is sampled from the manifold distribution, then the first half is replaced with real history, and finally the second half is refined through dynamics to generate a coherent future prediction.

Figure 10 demonstrates zero-shot forecasting where SDFlow predicts future time steps given only the first half. Despite no forecasting-specific training, our method achieves great MAE and MSE, with well-calibrated 80% confidence intervals (coverage 93%).

This strong performance supports our hypothesis that the learned manifold structure captures not only spatial but also temporal dependencies of time series data. However, more specific modifications in model architecture may further benefit Time Series Forecasting, which is a promising future direction.

K Additional Ablations

Categorical vs. Continuous Supervision. We compare our categorical cross-entropy objective against continuous MSE regression on embeddings. As shown in Table 9, the categorical objective yields superior performance. Beyond performance gains, it offers crucial theoretical advantages: (1) explicit discrete supervision aligned with the VQ structure; (2) temperature-controlled sampling at inference enabling quality-diversity trade-offs; and (3) interpretable uncertainty quantification over codebook indices.

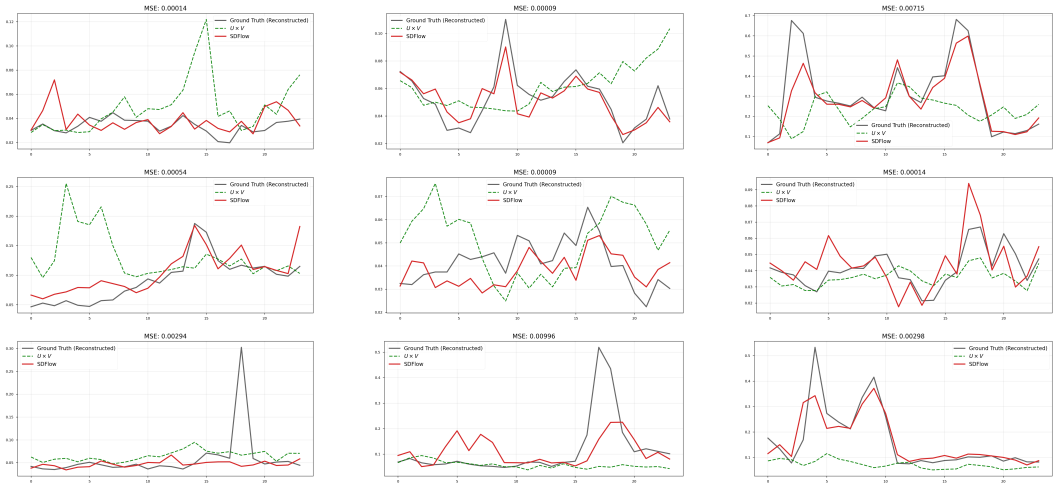


Figure 9: Visualizations of time series reconstruction samples using real coordinates instead of the anchor-prior sampler.

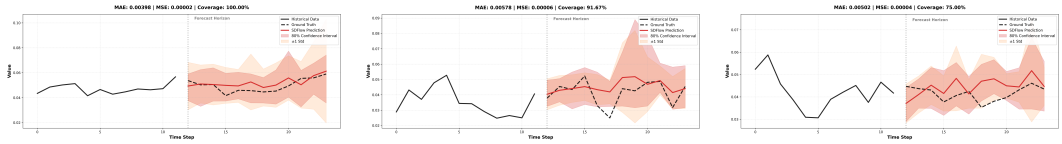


Figure 10: Forecasting Results (Energy, $12 \rightarrow 12$).

Table 9: Comparison of categorical cross-entropy (Ours) against continuous MSE regression.

Objective	**DS**-Sines ↓	**DS**-ETTh ↓
MSE (continuous)	0.009	0.005
CE (categorical)	**0.006**	**0.002**

Rank Sensitivity. Table 10 analyzes sensitivity to the rank parameter on Energy ($L = 24$). Performance peaks at $r = 256$, which captures 99% of variance while maintaining parameter efficiency. Lower ranks underfit the manifold structure, while higher ranks provide diminishing returns. Notably, relying solely on SVD to determine optimal rank is ineffective, demonstrating the necessity of the learnable low-rank decomposition.

Table 10: Rank sensitivity analysis on Energy ($L = 24$). Performance peaks at $r = 256$, which captures 99% of variance while maintaining parameter efficiency.

Rank	**DS** ↓	**FID** ↓	**Parameters**
$r = 32$	0.149 ± .012	0.042 ± .005	0.6M
SVD ($r = 42$)	0.094 ± .009	0.017 ± .002	0.8M
$r = 64$	0.062 ± .008	0.015 ± .002	1.2M
$r = 128$	0.015 ± .005	0.006 ± .001	2.4M
$r = 256$	**0.006 ± .002**	**0.001 ± .000**	4.8M
$r = 512$	0.006 ± .003	0.001 ± .000	9.6M

ODE Integration Steps. Table 11 examines sensitivity to the number of ODE integration steps. 20 steps suffice for convergence; additional steps provide negligible improvement at increased computational cost.

Table 11: ODE integration steps sensitivity on Energy ($L = 24$). 20 steps suffice for convergence; more steps provide negligible improvement.

Steps	**DS** ↓	**FID** ↓	**Time (s)**
10	0.018 ± .006	0.008 ± .002	0.005
15	0.009 ± .003	0.003 ± .001	0.011
20	**0.006 ± .002**	**0.001 ± .000**	0.021
50	0.006 ± .002	0.001 ± .000	0.042

Table 12: Cross-dataset ablation ($L = 24$). Both the low-rank anchor scaffold and kernel-smoothed prior are necessary on complex datasets (Energy, ETTh), while simpler datasets (Sines) are more tolerant.

Configuration	Prior	Sines	Stocks	ETTh	Energy
w/o Low-Rank	Gaussian	0.007 ± .006	0.038 ± .009	0.100 ± .015	0.217 ± .010
w/o Low-Rank	Anchor Prior	0.006 ± .008	0.025 ± .010	0.074 ± .017	0.173 ± .015
w/ Low-Rank	Gaussian	0.008 ± .005	0.031 ± .008	0.089 ± .012	0.218 ± .004
w/ Low-Rank	**Anchor Prior**	**0.006 ± .006**	**0.003 ± .003**	**0.002 ± .003**	**0.006 ± .002**



(a) ETTh dataset

(b) Energy dataset

Figure 11: Nearest-neighbor distance analysis. Gray bars show the self-distance distribution among real training samples (baseline), while red bars show distances from generated samples to their nearest training neighbors. Generated samples maintain substantial distances from training data (mean > 0.8 for ETTh, > 2.0 for Energy), consistent with non-copying behavior.

Cross-Dataset Ablation. Table 12 extends the ablation study across all datasets. Both low-rank decomposition and the anchor prior are essential on complex datasets (Energy, ETTh), while simpler datasets (Sines) are more tolerant of suboptimal configurations.

Nearest-neighbor distance analysis. Finally, to check whether generated sequences collapse to training examples, we conduct nearest-neighbor retrieval against the training set. Figure 11 shows the distribution of Euclidean distances from generated samples to their nearest training neighbors in the VQ-VAE latent space.

Together, the held-out latent-flow, prior-ablation, and nearest-neighbor results suggest that the anchor prior is not merely copying individual training samples. Instead, the learned manifold structure provides a *geometric scaffold* that guides generation while preserving diversity.

NeurIPS Paper Checklist

1. **Claims**
   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?
   Answer: [Yes]
   Justification: The abstract and Introduction (Section 1) state the scope and contributions of SDFlow, and the Method and Experiments sections support these claims with the proposed low-rank categorical flow formulation and benchmark results (Sections 4–5).
   Guidelines:
   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.
2. **Limitations**
   Question: Does the paper discuss the limitations of the work performed by the authors?
   Answer: [Yes]
   Justification: Section 6 includes a limitations paragraph noting that the current VQ tokenizer and codebook are dataset-specific, with universal time-series tokenization left as future work.
   Guidelines:
   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.
3. **Theory assumptions and proofs**
   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?
   Answer: [Yes]
   Justification: Theorem 4.1 and Proposition 4.2 state their assumptions, while Appendices A and B give complete derivations.
   Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The paper specifies the datasets, metrics, baselines, tokenizer/model settings, random seeds, hyperparameter tables, and training/inference algorithms in Sections 5 and Appendices D–E.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: A code repository is provided in the abstract. The repository includes implementation code, configuration files, data-preparation instructions, and scripts for reproducing the main experiments; the datasets used are public benchmarks.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Sections 5 and Appendix D describe datasets, metrics, baselines, tokenizer settings, ranks, layers, bandwidths, seeds, and hyperparameter configurations; Appendix E provides training and inference procedures.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Section 5 states that all experiments report mean and standard deviation over five random seeds, and Tables 1–2 report mean $\pm$ std over five runs.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).

- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**
   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?
   Answer: [No]
   Justification: The manuscript does not specify compute worker types, memory, per-run wall-clock time, or total compute needed to reproduce the experiments.
   Guidelines:
   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).
9. **Code of ethics**
   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?
   Answer: [Yes]
   Justification: The study uses synthetic or standard time-series benchmarks described in Appendix D, does not involve human subjects, and no deviation from the NeurIPS Code of Ethics is apparent from the described protocol.
   Guidelines:
   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).
10. **Broader impacts**
    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?
    Answer: [No]
    Justification: The Introduction motivates time-series generation in domains such as finance, healthcare, and energy, but the current manuscript does not discuss both positive and negative societal impacts or mitigation strategies.
    Guidelines:
    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.

- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper does not release a high-risk model, scraped dataset, or other asset requiring controlled-access safeguards; the experiments use standard time-series benchmarks.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [No]

    Justification: Appendix D names the Sines, Stocks, ETTh, and Energy datasets and cites prior protocols, but the manuscript does not state their licenses or terms of use.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The paper releases an anonymized code asset. Documentation, environment requirements, data preparation, and reproduction commands are provided with the released repository; no new dataset or model checkpoint is introduced.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.

- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The experiments use existing/synthetic time-series datasets and do not involve crowdsourcing or human-subject studies.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: No crowdsourcing or human-subject study is conducted, so participant risks and IRB approval are not applicable.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: The core method is based on VQ-VAE tokenization and categorical flow matching, and the manuscript does not use LLMs as a core methodological component.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.