FROM RARE EPISODES TO COMPLETE TRAJECTORIES: PRIOR-PRESERVING DIFFUSION FOR LONG-TAILED TIME-SERIES GENERATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Long-tailed multivariate time-series generation is important for simulation, data sharing, and the development of reliable analysis models, yet infrequent tail states are often poorly represented in synthetic data. We identify a fundamental difficulty in window-based training, termed *temporal rarity dilution*: as the window length increases, the same finite tail episode appears in more highly overlapping windows without providing proportionally more independent tail information, while normal observations still dominate the learning objective. Existing time-series generators therefore struggle to preserve both sparse tail states and their temporal and multivariate context, particularly over long windows. To address this problem, we propose the Episode-to-Trajectory Diffusion Transformer (E2T-DiT), which separates when tail states occur from how their corresponding trajectories are realized. First, a duration-aware autoregressive planner models the occurrence, order, and persistence of lower-tail, normal, and upper-tail states through a compact state-duration representation. The sampled state segments are then expanded into time-aligned conditions for an Axial Diffusion Transformer, which generates complete trajectories by modeling temporal evolution and cross-variable interaction along their respective axes. Second, a prior-preserving training objective combines global denoising, state-balanced supervision, and boundary consistency, strengthening the realization of sparse tail states without altering their learned occurrence distribution. Experiments on four long-tailed time-series datasets with window lengths of 32, 64, and 128 show that E2T-DiT increases the average Correlational, Predictive, and Discriminative Scores by 3.1%, 10.2%, and 8.0%, respectively, compared with the strongest baseline in each setting. It achieves the best in most comparisons, and preserves its advantage as the temporal context grows.

1 INTRODUCTION

Multivariate time-series generators support simulation, data sharing, and model development, but useful synthetic data must represent infrequent as well as ordinary behavior. In many applications, domain knowledge specifies a normal interval $[L, U]$ for one monitored variable (Hu et al., 2025); observations outside it form the lower or upper tail. Such values may arise either through smooth crossings driven by continuous variation or through abrupt changes, and both are part of the target distribution. Our goal is to reproduce them at their empirical rate together with their temporal and multivariate context.

Existing GAN-, VAE-, and diffusion-based generators optimize losses over complete windows (Yoon et al., 2019; Desai et al., 2021; Yuan & Qiao, 2024). This hides a basic mismatch between rare points and window-level supervision. If a point is in the tail with probability p, an independence approximation gives

$$P_{\text{tail-win}}(W) = 1 - (1 - p)^W.$$

As W grows, the same finite tail episode is covered by more sliding windows, so a fixed raw sequence can yield substantially more tail-containing windows and a higher tail-window fraction.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

These highly overlapping windows, however, repeatedly expose the same episode rather than provide a commensurate amount of new tail information. Moreover, most positions within each window usually remain normal, so a window-averaged objective is still dominated by normal observations. The apparent growth in tail-window count therefore does not resolve the scarcity of effective tail information, creating a bottleneck for existing time-series generators learning long-tailed distributions. We call this phenomenon *temporal rarity dilution*: increasing the window length places a tail event in more training samples without a corresponding increase in its independent information or relative contribution to optimization, and learning tail patterns becomes increasingly difficult.

Tail behavior also has temporal structure. A boundary crossing may occur through a smooth evolution with only small changes between adjacent values, or it may be caused by an abrupt change; conversely, a large local change may remain entirely within the normal range. The normal-range boundaries tell us whether each observation is normal or belongs to the lower or upper tail, but they do not describe how a tail episode develops over time. A generator must also learn how the sequence crosses the boundary, how long it remains in the tail, whether the episode continues to the edge of the window, and how other variables respond at the same or later time.

We therefore decompose long-tailed time-series generation into two coupled tasks: modeling the temporal process of lower-tail, normal, and upper-tail states, and generating a complete multivariate trajectory conditioned on this state process:

$$p(x) = \sum_{s} p_\phi(s) p_\theta(x \mid s).$$

Here, $x \in \mathbb{R}^{W \times C}$ and $s \in \{-1, 0, +1\}^W$ represents lower-tail, normal, and upper-tail states. The state prior p_ϕ is fitted to unrebalanced data and controls occurrence, order, and duration. The trajectory model p_θ receives state-aware supervision, while separate optimization preserves the distribution learned by p_ϕ.

The model we proposed, **Episode-to-Trajectory Diffusion Transformer (E2T-DiT)**, first run-length encodes s into state-duration segments and learns their distribution with an autoregressive Transformer. Explicit duration distinguishes isolated crossings from sustained episodes and compactly represents long state runs. The state-duration segments are then expanded into a time-aligned condition describing state, duration, within-segment position, and contact with window boundaries. Conditioned on this representation, an observation-space Diffusion Transformer learns to denoise the complete $W \times C$ data, thereby modeling the joint relations among state episodes, their preceding and subsequent context, and multivariate responses. Axial attention (Ho et al., 2019) captures temporal evolution and cross-variable interaction along their respective axes, while the time-aligned segment condition is injected into each DiT block to provide state-process information throughout denoising.

The trajectory model combines global denoising with a state-balanced loss and a soft boundary-consistency penalty. The former gives sparse states a direct contribution to the objective; the latter enforces agreement with $[L, U]$, while the denoising loss learns tail amplitude and local shape. Because these terms optimize $p_\theta(x \mid s)$ and $p_\phi(s)$ is trained separately by likelihood, the model strengthens tail realization while preserving its learned state prior.

Our contributions are threefold:

1. We identify temporal rarity dilution, explaining why tail-containing windows may be abundant even when their informative positions remain rare.
2. We formulate long-tailed generation as a duration-aware state prior coupled to an episode-conditioned whole-trajectory distribution.
3. We introduce a segment-conditioned Axial DiT and a prior-preserving objective that strengthens lower- and upper-tail realization while retaining global modeling.

2 Related work

2.1 Time-series generation

Time-series generators must reproduce marginal distributions, temporal dependence, and cross-variable relationships. GAN-based methods introduced recurrent adversarial generation (Mogren,

2016) and later combined adversarial, embedding, and supervised temporal objectives (Yoon et al., 2019). VAEs offer a likelihood-based alternative; for example, TimeVAE explicitly represents trend and seasonal structure (Desai et al., 2021). Both families can underrepresent low-probability behavior: adversarial training may omit modes, while a learned reconstruction bottleneck can smooth small but semantically important boundary crossings. This concern also motivates our use of observation-space rather than latent diffusion (Rombach et al., 2022).

Diffusion models provide stable distribution learning through denoising (Ho et al., 2020; Song et al., 2020b). Beyond forecasting and imputation (Rasul et al., 2021; Tashiro et al., 2021), recent generative models use Transformer denoisers and temporal decomposition (Yuan & Qiao, 2024) or population-level objectives (Li et al., 2025). These advances improve overall temporal fidelity, but the usual loss still averages errors across a complete window. Consequently, the contribution of a state remains proportional to the number of positions carrying it. E2T-DiT instead separates the occurrence prior $p_\phi(s)$ from the conditional realization $p_\theta(x \mid s)$ and applies tail-aware optimization only to the latter.

2.2 Rare temporal states and structured diffusion

Long-tailed learning typically assumes one label per sample and compensates for frequency using oversampling, class-balanced losses, margin adjustment, or decoupled training (Chawla et al., 2002; Cui et al., 2019; Cao et al., 2019; Kang et al., 2019; Zhang et al., 2023; Menon et al., 2020). A windowed time series violates this unit of analysis: one window can contain normal, lower-tail, and upper-tail positions simultaneously, and an overlapping window may be labeled *tail* because of a single point. Weighting that window also weights its normal region, while stronger resampling can alter the learned state prevalence. Our state-balanced objective therefore operates at temporal positions, and a separately trained state model estimates their occurrence process.

The state process has duration structure: identical point-level tail rates may arise from a few long episodes or many short ones. Explicit-duration models such as hidden semi-Markov models capture this distinction (Yu, 2010). E2T-DiT retains the duration-explicit representation but uses observed domain states and a history-dependent Transformer, rather than latent clusters or fixed transition assumptions. For continuous realization, DiTs inject conditions through adaptive normalization (Peebles & Xie, 2023). We further use axial attention (Ho et al., 2019) so time indexes ordered observations and the variable axis indexes physical channels. Applying attention separately along these axes reduces its complexity from $\mathcal{O}(W^2C^2)$ for full attention over the WC time-variable tokens to $\mathcal{O}(CW^2 + WC^2)$.

3 Methodology

3.1 Problem formulation and overview

Let $\mathcal{D} = \{x_i\}_{i=1}^N$ contain N fixed-length multivariate windows, where $x_i \in \mathbb{R}^{W\times C}$, W is the window length, and C is the number of variables. The indices i, $\tau \in \{1, \dots, W\}$, and $c \in \{1, \dots, C\}$ refer to the window, temporal position, and variable, respectively. A designated variable $c^\star$ has a domain-defined normal interval $[L, U]$. Before normalization, we derive the pointwise state

$$s_{i,\tau} = \begin{cases} -1, & x_{i,\tau,c^\star} < L, \\ 0, & L \le x_{i,\tau,c^\star} \le U, \\ +1, & x_{i,\tau,c^\star} > U, \end{cases} \qquad s_i \in \{-1, 0, +1\}^W. \tag{1}$$

The resulting sequence s_i records the temporal evolution among lower-tail, normal, and upper-tail regions.

Let x denote a generic trajectory and s its state sequence. With planner parameters ϕ and trajectory-model parameters θ, E2T-DiT expresses the marginal trajectory distribution by summing over the state sequences:

$$p(x) = \sum_s p_\phi(s) p_\theta(x \mid s). \tag{2}$$

The duration-aware planner p_ϕ learns state occurrence, order, and persistence from the empirical data. The trajectory realizer p_θ learns the complete multivariate trajectory associated with the sam-

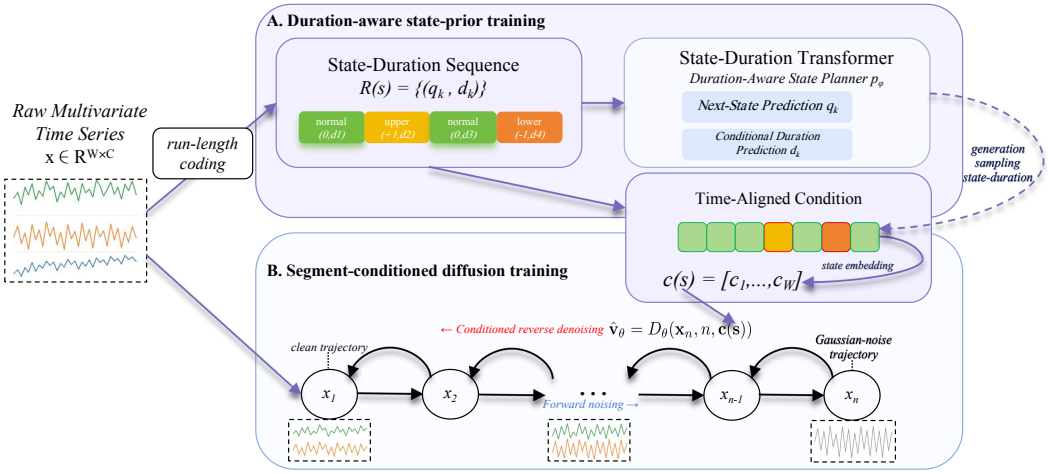

Figure 1: Overall architecture of E2T-DiT. Run-length coding converts the pointwise state sequence into state-duration segments, from which the duration-aware Transformer learns the state prior. The segments are expanded into a time-aligned condition for diffusion training; during generation, segments sampled from the learned prior guide reverse denoising from Gaussian noise to a complete multivariate trajectory.

pled process. Separate optimization assigns occurrence modeling to p_ϕ and state-aware trajectory learning to p_θ.

3.2 Duration-aware state prior

As shown in Figure 1, adjacent identical states are run-length encoded as

$$\mathcal{R}(s) = \{(q_k, d_k)\}_{k=1}^{K}, \quad q_k \in \{-1, 0, +1\}, \quad q_k \neq q_{k+1}, \quad \sum_{k=1}^{K} d_k = W. \tag{3}$$

Here, q_k and d_k are the state and duration of the k-th among K segments. This lossless representation distinguishes one-step crossings from sustained episodes while compactly representing long runs. Let $z_k = (q_k, d_k)$, let $z_{<k}$ denote the preceding segments, and let $r_k = W - \sum_{j<k} d_j$ be the remaining length. A causal Transformer models

$$p_\phi(\mathcal{R}) = \prod_{k=1}^{K} p_\phi(q_k \mid z_{<k}, r_k)\, p_\phi(d_k \mid q_k, z_{<k}, r_k). \tag{4}$$

Its input embeds the previous state-duration pair, segment index, and remaining length; two heads predict the next state and then its duration. Validity masks enforce $q_k \neq q_{k-1}$ and $1 \leq d_k \leq r_k$. Sampling terminates when $d_k = r_k$, guaranteeing exactly W positions. Teacher-forced negative log-likelihood is averaged over empirical windows with their original frequencies.

The planner produces segment-level states, whereas the diffusion model requires a condition at every temporal position. Suppose segment k spans $[a_k, b_k]$. For each τ in the segment, we construct a compact structural vector u_τ and combine it with the state embedding:

$$u_\tau = \left[\frac{d_k}{W}, \frac{\tau - a_k + 1/2}{d_k}, \mathbb{I}[a_k = 1], \mathbb{I}[b_k = W]\right], \qquad c_\tau = W_c[e_q(q_k) \| f_m(u_\tau)] + b_c. \tag{5}$$

The entries of u_τ describe relative duration, normalized progress within the segment, and contact with the left and right window boundaries. Here, $\mathbb{I}$ is the indicator function, e_q is a state embedding, f_m is a multilayer perceptron, W_c, b_c are condition-projection parameters, and $\|$ denotes concatenation. This expansion produces the time-aligned condition sequence $c(s) = \{c_\tau\}_{\tau=1}^{W}$.

3.3 SEGMENT-CONDITIONED AXIAL DIT

We diffuse complete multivariate trajectories directly in observation space. Before training, each channel is standardized using its dataset statistics, and L, U are mapped to L', U' with the same transform used for the designated channel. For a standardized clean trajectory x_0, the forward process and v-prediction target are (Ho et al., 2020; Nichol & Dhariwal, 2021; Salimans & Ho, 2022)

$$x_n = \alpha_n x_0 + \sigma_n \epsilon, \qquad v_n = \alpha_n \epsilon - \sigma_n x_0, \qquad \epsilon \sim \mathcal{N}(0, I), \tag{6}$$

where n indexes the diffusion step, ϵ is Gaussian noise, and α_n and σ_n are cosine-schedule coefficients satisfying $\alpha_n^2 + \sigma_n^2 = 1$. The noisy trajectory x_n is the continuous DiT input, and v_n is its training target.

We retain the time and physical-variable axes, assigning a token

$$h_{\tau,c}^{(0)} = W_x x_{n,\tau,c} + e_{\mathrm{var}}(c) \tag{7}$$

to each scalar. Here, W_x projects each scalar observation into the hidden space, while the learned variable embedding $e_{\mathrm{var}}(c)$ identifies its source channel. This distinction is necessary because the same numerical value can have different meanings and dynamics across physical variables. Rotary position embeddings Su et al. (2024) are incorporated before temporal attention to preserve the ordering of observations.

The tokens remain arranged as a $W \times C$ grid. Each Axial DiT block first applies temporal attention along each variable to model state entry, persistence, and exit, and then applies variable attention across channels at each time to model the multivariate response. Alternating these operations allows historical information to propagate along time and subsequently transfer between variables. The diffusion-step embedding and the time-aligned condition c_τ from Eq. (5) are injected into each block through the standard adaptive-normalization mechanism used by DiT (Peebles & Xie, 2023). All variables at time τ share the same c_τ, while their variable embeddings retain channel-specific information.

After the Axial DiT blocks, an output projection predicts $\hat{v}_\theta = D_\theta(x_n, n, c(s))$ for every time-variable token. The clean trajectory estimate is recovered as $\hat{x}_0 = \alpha_n x_n - \sigma_n \hat{v}_\theta$. The resulting denoising process jointly learns the state episode, its preceding and subsequent temporal context, and the associated cross-variable response.

3.4 PRIOR-PRESERVING TRAINING AND GENERATION

Starting from the global mean squared error (MSE) denoising loss, we introduce a state-balanced loss so that normal, lower-tail, and upper-tail states each contribute directly to trajectory learning. We then add a boundary-consistency penalty that aligns the predicted designated variable with its state condition. The resulting trajectory objective therefore retains denoising over the complete multivariate distribution, strengthens supervision for sparse tail states, and ensures that the state condition is reflected in the continuous trajectory.

For a minibatch of B windows, ordinary denoising averages over all positions, causing sparse states to contribute little. With $e_{i,\tau,c} = (\hat{v}_{i,\tau,c} - v_{i,\tau,c})^2$, we retain the global objective and additionally normalize within each state:

$$\mathcal{L}_{\mathrm{global}} = \frac{1}{BWC} \sum_{i,\tau,c} e_{i,\tau,c}, \tag{8}$$

$$\mathcal{L}_q = \frac{\sum_{i,\tau,c} \mathbb{I}[s_{i,\tau} = q] e_{i,\tau,c}}{C \sum_{i,\tau} \mathbb{I}[s_{i,\tau} = q] + \varepsilon}, \quad q \in \{-1, 0, +1\}. \tag{9}$$

Let $\mathcal{Q}_B$ contain states present in the batch. The combined loss is

$$\mathcal{L}_{\mathrm{denoise}} = (1 - \rho)\mathcal{L}_{\mathrm{global}} + \frac{\rho}{|\mathcal{Q}_B|} \sum_{q \in \mathcal{Q}_B} \mathcal{L}_q. \tag{10}$$

Here, $\rho \in [0, 1]$ controls the balance between the two terms. A tail position contributes errors from all C channels, encouraging a coherent system state. Position-level normalization assigns normal context and tail positions to their respective state terms.

We also penalize disagreement between the predicted clean designated value $\hat{y}_{i,\tau} = \hat{x}_{0,i,\tau,c^\star}$ and its condition:

$$g_q(\hat{y}) = \begin{cases} [L' - \hat{y}]_+ + [\hat{y} - U']_+, & q = 0, \\ [U' - \hat{y}]_+, & q = +1, \\ [\hat{y} - L']_+, & q = -1. \end{cases} \tag{11}$$

where $[a]_+ = \max(a, 0)$. The penalty only enforces that $\hat{y}$ lies in the value range associated with state q. Once this condition is satisfied, the penalty becomes zero and imposes no constraint on how far a tail value extends beyond the boundary or how the surrounding sequence evolves. Weighting by the signal level makes the constraint weaker for highly corrupted inputs:

$$\mathcal{L}_{\text{trajectory}} = \mathcal{L}_{\text{denoise}} + \lambda_{\text{bnd}} \frac{1}{BW} \sum_{i,\tau} \alpha_{n_i}^2 \, g_{s_{i,\tau}}(\hat{y}_{i,\tau}). \tag{12}$$

Here, λ_{bnd} controls the strength of boundary consistency, and $\alpha_{n_i}^2$ is the signal power at the diffusion step sampled for window i.

Training has two stages. We first minimize the planner likelihood on ground-truth $\mathcal{R}(s_i)$, then train the DiT with ground-truth $c(s_i)$ and the objective in Eq. (12). The two models have disjoint parameters and their respective objectives assign empirical occurrence modeling to $p_\phi(s)$ and state-aware realization to $p_\theta(x \mid s)$.

As shown in Figure 1, for generation, the planner samples valid state-duration pairs until their durations sum to W and expands them into the time-aligned condition $\hat{c}$. Starting from a Gaussian-noise trajectory, the diffusion model uses $\hat{c}$ throughout the reverse process and progressively removes noise with a DDPM or DDIM sampler (Ho et al., 2020; Song et al., 2020a), producing a complete multivariate trajectory in standardized space. Finally, each channel is transformed back to its original physical scale using the statistics stored during training. The overall generation distribution is

$$\hat{x} \sim \sum_s p_{\phi^\star}(s) p_{\theta^\star}(x \mid s), \tag{13}$$

where $\phi^\star$ and $\theta^\star$ denote the trained parameters.

4 Experiments

4.1 Datasets and baselines

We conduct experiments on four real-world multivariate time-series datasets. Boiler (Hu et al., 2025) contains multivariate measurements from an industrial boiler and specifies a normal operating range for steam temperature; tail points outside this range account for approximately 8% of all temporal observations. The Secure Water Treatment (SWaT) dataset (Mathur & Tippenhauer, 2016) records sensor and actuator measurements under normal operation and attacks; we regard temporally attack-labeled observations as tail points, which account for approximately 12% of the data. ETTh1 and ETTm1 (Zhou et al., 2021) are collected from electricity transformers; observations with oil temperature below 4 are defined as tail points and account for approximately 9% in each dataset. We compare E2T-DiT with representative time-series generators: TimeVAE (Desai et al., 2021) and TimeGAN (Yoon et al., 2019) are classical VAE- and GAN-based methods, respectively; DiffWave (Kong et al., 2020) is a classical diffusion model for audio synthesis; Diffusion-TS (Yuan & Qiao, 2024) targets high-quality general time-series generation; and DLTTS (Hu et al., 2026) is designed for long-tailed time-series generation.

4.2 Evaluation metrics

Lower values indicate better performance for all metrics. **(1) Discriminative Score:** We train an LSTM binary classifier to distinguish real from generated windows, compute its test accuracy acc, and report $|\mathrm{acc} - 0.5|$; a value closer to zero indicates that the two sources are harder to distinguish; **(2) Predictive Score:** We train an LSTM to predict the next time step from preceding observations and report the mean absolute error (MAE); **(3) Correlational Score:** We compute cross-correlation matrices for the real and generated data and report their discrepancy, where a lower score indicates better preservation of temporal and inter-variable correlation structure.

Table 1: Tail generation quality across datasets and window lengths. Lower values indicate better performance, and "-" denotes that the program requires an impractically long runtime.

Dataset		S	TimeVAE	TimeGAN	Diffusion-TS	DiffWave	DLTTS	E2T-DiT
SWaT	Correlational Score	32	**19.57**	20.48	27.16	30.19	25.21	20.33
		64	48.31	54.75	27.00	46.85	25.35	**22.62**
		128	-	45.95	24.29	37.42	22.57	**21.98**
	Predictive Score	32	0.0005	**0.00004**	0.001	0.006	0.002	0.0001
		64	0.0002	0.0001	0.004	0.003	0.008	**0.00008**
		128	-	0.0001	0.001	0.003	**0.00005**	**0.00005**
	Discriminative Score	32	0.50	0.49	0.50	0.50	**0.48**	**0.48**
		64	0.50	0.49	0.50	0.50	0.45	**0.42**
		128	-	0.49	0.50	0.50	0.47	**0.43**
ETTh1	Correlational Score	32	0.28	0.38	0.22	0.38	0.19	**0.15**
		64	0.25	0.50	0.20	0.30	0.20	**0.19**
		128	0.22	0.75	0.27	0.41	0.20	**0.18**
	Predictive Score	32	0.15	0.18	0.21	**0.10**	0.18	**0.10**
		64	0.16	0.11	0.12	0.12	0.11	**0.10**
		128	0.18	0.15	0.18	0.16	0.14	**0.12**
	Discriminative Score	32	0.47	0.17	0.42	0.49	0.48	**0.16**
		64	0.44	**0.23**	0.47	0.47	0.45	**0.23**
		128	0.44	0.46	0.44	0.46	0.43	**0.37**
ETTm1	Correlational Score	32	0.28	0.52	0.26	0.35	0.25	**0.20**
		64	0.29	0.37	0.26	0.32	**0.25**	**0.25**
		128	0.23	0.59	0.22	0.40	**0.20**	**0.20**
	Predictive Score	32	0.19	0.19	0.20	0.30	0.13	**0.11**
		64	0.16	0.17	0.17	0.69	0.17	**0.15**
		128	0.19	0.18	0.19	0.69	0.18	**0.15**
	Discriminative Score	32	0.46	0.26	0.44	0.47	0.41	**0.20**
		64	0.46	**0.23**	0.43	0.48	0.40	0.25
		128	0.43	0.49	0.45	0.48	0.38	**0.28**
Boiler	Correlational Score	32	4.24	7.71	3.47	4.54	3.07	**2.99**
		64	4.38	6.78	3.84	5.24	3.15	**3.14**
		128	4.71	9.91	3.61	6.70	**2.75**	3.08
	Predictive Score	32	0.12	0.19	0.13	0.14	0.10	**0.08**
		64	0.10	0.15	**0.09**	0.16	**0.09**	**0.09**
		128	0.08	0.13	0.08	0.10	**0.07**	**0.07**
	Discriminative Score	32	0.50	0.43	0.50	0.50	0.50	**0.31**
		64	0.50	0.43	0.50	0.50	0.39	**0.36**
		128	0.50	**0.23**	0.50	0.50	0.50	0.33

4.3 TAIL GENERATION QUALITY

We compare generation results over different window lengths on four long-tailed time-series datasets. As shown in Table 1, E2T-DiT achieves the best or tied-best result in 31 of the 36 settings and ranks within the top two in every setting. Compared with the strongest baseline selected separately for each dataset, window length, and metric, E2T-DiT increases the average Correlational, Predictive, and Discriminative Scores by 3.1%, 10.2%, and 8.0%, respectively. Its advantage is particularly evident for longer windows. Some competing methods perform well on short sequences but degrade substantially as the window length increases: for example, the Correlational Score of TimeVAE on SWaT rises from 19.57 at $S = 32$ to 48.31 at $S = 64$; similarly, the Discriminative Score of TimeGAN on ETTh1 rises from 0.17 to 0.46 as S increases from 32 to 128. In contrast, E2T-DiT remains among the top two methods in all 12 settings at $S = 128$ and obtains the best or tied-best result in 10 of them. These results indicate that its trajectory-level modeling better preserves tail-window structure as the temporal context grows.

4.4 ABLATION STUDY

We construct four variants to evaluate the main components of E2T-DiT: **w/o state loss** removes the state-balanced term while retaining global denoising and boundary consistency; **w/o boundary loss** removes the boundary-consistency term; **w/o run-length coding** replaces the duration-aware planner with a pointwise state planner while keeping the trajectory model unchanged; **w/o segment-aware condition** replaces the full segment descriptor with a state-only condition.

Table 2: Component ablation results across datasets and window lengths. Lower values indicate better performance. Best results are shown in bold and second-best results are underlined.

Dataset	Metric	S	E2T-DiT	w/o state loss	w/o boundary loss	w/o run-length coding	w/o segment-aware condition
SWaT	Correlational Score	32	**20.33**	23.64	21.14	21.96	23.43
		64	**22.62**	27.23	23.75	25.79	27.46
		128	**21.98**	27.58	23.52	26.82	28.25
	Predictive Score	32	**0.00010**	0.00011	**0.00010**	0.00011	0.00011
		64	**0.00008**	0.00011	0.00010	0.00012	0.00012
		128	**0.00005**	0.00012	0.00011	0.00013	0.00013
	Discriminative Score	32	**0.48**	0.50	0.50	0.50	0.50
		64	**0.42**	0.47	0.46	0.50	0.47
		128	**0.43**	0.50	0.48	0.50	0.50
ETTh1	Correlational Score	32	**0.15**	0.17	0.16	0.16	0.17
		64	**0.19**	0.22	0.20	0.22	0.23
		128	**0.18**	0.22	0.19	0.22	0.23
	Predictive Score	32	**0.10**	0.14	0.13	0.15	0.15
		64	**0.10**	0.11	**0.10**	0.12	0.12
		128	**0.12**	0.14	0.13	0.16	0.16
	Discriminative Score	32	**0.16**	0.17	0.17	0.18	0.17
		64	**0.23**	0.26	0.25	0.28	0.26
		128	**0.37**	0.44	0.41	0.49	0.44
ETTm1	Correlational Score	32	**0.20**	0.23	0.21	0.22	0.23
		64	**0.25**	0.30	0.26	0.28	0.30
		128	**0.20**	0.28	0.25	0.28	0.30
	Predictive Score	32	**0.11**	0.12	**0.11**	0.12	0.13
		64	**0.15**	0.17	0.16	0.18	0.19
		128	**0.15**	0.18	0.16	0.20	0.20
	Discriminative Score	32	**0.20**	0.22	0.21	0.22	0.22
		64	**0.25**	0.28	0.27	0.30	0.28
		128	**0.28**	0.33	0.31	0.36	0.33
Boiler	Correlational Score	32	**2.99**	3.41	3.11	3.23	3.38
		64	**3.14**	3.71	3.30	3.58	3.74
		128	**3.08**	3.79	3.30	3.76	3.88
	Predictive Score	32	**0.08**	0.09	**0.08**	0.09	0.09
		64	**0.09**	0.10	**0.09**	0.11	0.11
		128	**0.07**	0.08	**0.07**	0.09	0.09
	Discriminative Score	32	**0.31**	0.34	0.33	0.34	0.33
		64	**0.36**	0.41	0.39	0.42	0.41
		128	**0.33**	0.40	0.37	0.43	0.39

Table 2 reports the results. The average degradations of **w/o state loss**, **w/o boundary loss**, **w/o run-length coding**, and **w/o segment-aware condition** are 20.4%, 10.6%, 24.0%, and 24.6%, respectively. E2T-DiT achieves the best or tied-best result in all 36 settings and strictly outperforms **w/o state loss**, **w/o run-length coding**, and **w/o segment-aware condition** in every setting. Removing the state-balanced loss consistently degrades all three metrics, supporting its role in preventing sparse states from being diluted by the dominant normal positions. Run-length coding and segment-aware conditioning provide the largest overall gains, with particularly strong improvements in Predictive Score, showing that explicit episode duration and within-segment temporal context are important for trajectory realization. Their benefits also grow with the window length: from $S = 32$ to $S = 128$, the median degradation of **w/o run-length coding** increases from 9.8% to 29.4%, and that of **w/o segment-aware condition** increases from 12.8% to 28.2%. This trend shows that structured episode modeling becomes increasingly valuable as the temporal context expands.

In addition to the three general-purpose generation metrics, we evaluate tail-rate consistency to isolate the effect of the boundary-consistency loss. Let r_{real} denote the proportion of lower- and upper-tail points in the real data, as determined by the normal interval $[L, U]$. Let r_{plan} denote the proportion of lower- and upper-tail states in the state sequences sampled by the planner. Finally, let r_{gen} denote the proportion of generated values of the designated variable that fall outside $[L, U]$. We use the same sampled state sequences when comparing trajectory-model variants so that differences in r_{gen} reflect how faithfully each model realizes the planner output.

The *realization gap* measures the discrepancy between the tail rate requested by the planner and the tail rate realized in the generated trajectories:

$$E_{\mathrm{realize}} = \left| r_{\mathrm{gen}} - r_{\mathrm{plan}} \right|. \tag{14}$$

Table 3: Effect of boundary consistency on tail-rate realization. "Full" denotes E2T-DiT and "w/o" removes the boundary-consistency loss. Lower values are better, and the best result in each pair is shown in bold.

Metric	S	SWaT		ETTh1		ETTm1		Boiler	
		Full	w/o	Full	w/o	Full	w/o	Full	w/o
E_{realize}	32	**0.028**	0.060	**0.015**	0.034	**0.022**	0.045	**0.033**	0.069
	64	**0.032**	0.071	**0.017**	0.039	**0.022**	0.049	**0.037**	0.079
	128	**0.025**	0.058	**0.015**	0.036	**0.022**	0.047	**0.034**	0.074
E_{tail}	32	**0.040**	0.056	**0.020**	0.030	**0.020**	0.031	**0.039**	0.057
	64	**0.041**	0.061	**0.020**	0.032	**0.023**	0.035	**0.040**	0.061
	128	**0.038**	0.057	**0.018**	0.029	**0.021**	0.033	**0.040**	0.060

The *tail-rate error* measures the end-to-end discrepancy between the generated and empirical tail rates:

$$E_{\text{tail}} = |r_{\text{gen}} - r_{\text{real}}| . \tag{15}$$

Lower values are better for both metrics. E_{realize} primarily evaluates whether the trajectory model follows the sampled state process, whereas E_{tail} evaluates the tail prevalence of the complete generation pipeline.

Although removing the boundary loss causes a smaller degradation in the three general-purpose metrics than the other ablations, the targeted metrics in Table 3 reveal its distinct role. E2T-DiT obtains lower realization gap and tail-rate error across all four datasets and all three window lengths. Averaged over the 12 settings, removing boundary consistency rises E_{realize} from 0.0252 to 0.0551 and E_{tail} from 0.0300 to 0.0452. Computing the relative increase for each setting and then averaging shows that removing boundary consistency rises E_{realize} and E_{tail} by 120.4% and 51.9%, respectively. The improvement is consistent across window lengths: compared with the ablated model, E2T-DiT improves E_{realize} by 52.9%–55.3% and E_{tail} by 31.6%–34.6% as S varies from 32 to 128. The larger change in E_{realize} supports the intended role of the boundary term: it primarily aligns the continuous trajectory with the sampled state process, while the change in E_{tail} reflects the resulting improvement in end-to-end tail prevalence. Since both variants use the same planner samples, these differences arise from trajectory realization rather than a change in the learned state prior.

5 Conclusions

We presented E2T-DiT for generating multivariate time series with sparsely represented tail behavior. Motivated by temporal rarity dilution, the method separates a duration-aware state process from the realization of the corresponding continuous trajectory. A state-duration Transformer models the occurrence and persistence of normal, lower-tail, and upper-tail episodes, while a segment-conditioned Axial DiT generates their complete multivariate context. State-balanced denoising strengthens learning at rare temporal positions, and boundary consistency aligns generated values with the sampled state process while preserving the planner's empirical training distribution. Experiments across four real-world datasets and multiple window lengths show improvements in overall generation quality and tail-rate realization, with ablations confirming the complementary roles of position-level supervision, explicit duration modeling, segment-aware conditioning, and boundary consistency. However, the method currently relies on a reliable domain-defined normal interval, and planner errors can propagate to the trajectory model. Its diffusion-based realizer also requires iterative sampling. Future work will study uncertain or learned state boundaries, faster diffusion solvers, and tighter planner–realizer integration while retaining control of the learned state prevalence.

AI use statement

This paper utilized GPT-5.6 Sol for language-related assistance, including translation and grammar checking.

Reproducibility Statement

We provide the details of all hyperparameters and experimental settings in the appendix to facilitate the reproduction of our results. The code implementation will be released in a public repository after the acceptance of this paper.

References

Kaidi Cao, Colin Wei, Adrien Gaidon, Nikos Arechiga, and Tengyu Ma. Learning imbalanced datasets with label-distribution-aware margin loss. *Advances in neural information processing systems*, 32, 2019.

Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.

Yin Cui, Menglin Jia, Tsung-Yi Lin, Yang Song, and Serge Belongie. Class-balanced loss based on effective number of samples. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 9268–9277, 2019.

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

Jonathan Ho, Nal Kalchbrenner, Dirk Weissenborn, and Tim Salimans. Axial attention in multidimensional transformers. *arXiv preprint arXiv:1912.12180*, 2019.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

Weijun Hu, Anze Jiang, Kaihong Chen, Jie Zheng, Wenli Shang, and Zhong Cao. A long-tailed distribution time-series dataset in boiler equipment. *Scientific Data*, 12(1):742, 2025.

Weijun Hu, Haifeng Zhang, Wenbin Chen, Kaihong Chen, Jinhao Long, Wenli Shang, and Zhong Cao. Dltts: Diffusion model for long-tailed time series generation in industrial scenarios. *IEEE Transactions on Knowledge and Data Engineering*, 2026.

Bingyi Kang, Saining Xie, Marcus Rohrbach, Zhicheng Yan, Albert Gordo, Jiashi Feng, and Yannis Kalantidis. Decoupling representation and classifier for long-tailed recognition. *arXiv preprint arXiv:1910.09217*, 2019.

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.

Yang Li, Han Meng, Zhenyu Bi, Ingolv T Urnes, and Haipeng Chen. Population aware diffusion for time series generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 18520–18529, 2025.

Aditya P Mathur and Nils Ole Tippenhauer. Swat: A water treatment testbed for research and training on ics security. In *2016 international workshop on cyber-physical systems for smart water networks (CySWater)*, pp. 31–36. IEEE, 2016.

Aditya Krishna Menon, Sadeep Jayasumana, Ankit Singh Rawat, Himanshu Jain, Andreas Veit, and Sanjiv Kumar. Long-tail learning via logit adjustment. *arXiv preprint arXiv:2007.07314*, 2020.

Olof Mogren. C-rnn-gan: Continuous recurrent neural networks with adversarial training. *arXiv preprint arXiv:1611.09904*, 2016.

Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *International conference on machine learning*, pp. 8162–8171. PMLR, 2021.

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

Kashif Rasul, Calvin Seward, Ingmar Schuster, and Roland Vollgraf. Autoregressive denoising diffusion models for multivariate probabilistic time series forecasting. In *International conference on machine learning*, pp. 8857–8868. PMLR, 2021.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10684–10695, 2022.

Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. *arXiv preprint arXiv:2202.00512*, 2022.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020a.

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

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

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.

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

Shun-Zheng Yu. Hidden semi-markov models. *Artificial intelligence*, 174(2):215–243, 2010.

Xinyu Yuan and Yan Qiao. Diffusion-ts: Interpretable diffusion for general time series generation. In *International Conference on Learning Representations*, volume 2024, pp. 41582–41610, 2024.

Yifan Zhang, Bingyi Kang, Bryan Hooi, Shuicheng Yan, and Jiashi Feng. Deep long-tailed learning: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 45(9):10795–10816, 2023.

Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pp. 11106–11115, 2021.

A HYPERPARAMETER SETTINGS

Table 4 lists the hyperparameters of E2T-DiT for each dataset.

Table 4: Hyperparameter settings of E2T-DiT on the four datasets.

Parameter	**SWaT**	**ETTh1**	**ETTm1**	**Boiler**
training configuration				
batch size	64	64	64	64
timesteps	500	500	500	1000
optimizer	AdamW	AdamW	AdamW	AdamW
training steps	12000	18000	18000	25000
prediction	v	v	v	v
noise scheduler	Cosine	Cosine	Cosine	Cosine
sample steps(DDIM)	50	50	50	50
Duration-aware state planner				
attention heads	4	4	4	4
event_lr	3e-4	3e-4	3e-4	3e-4
Segment-conditioned Axial DiT				
dit.hidden_dim	192	192	192	192
dit.depth	8	8	8	8
dit.heads_time	6	6	6	6
dit.heads_channel	6	6	6	6
trajectory_lr	2e-4	2e-4	2e-4	2e-4
trajectory_batch_size	16	16	16	16
Prior-preserving objective				
state_balance	0.5	0.5	0.5	0.5
boundary	0.1	0.1	0.1	0.1

B SEGMENT CONDITION AND TIME-ALIGNED EXPANSION

Each state segment k spans the positions $[a_k, b_k]$ with $b_k = a_k + d_k - 1$. It is expanded into a per-position structural descriptor $u_\tau \in \mathbb{R}^5$, whose entries are listed in Table 5; here $\tau - a_k \in \{0, \dots, d_k - 1\}$ is the offset of position τ inside the segment.

Table 5: Entries of the structural descriptor u_τ used to build the time-aligned condition.

Index	**Expression**	**Meaning**
1	d_k/W	relative segment duration
2	$(\tau - a_k + 1/2)/d_k$	normalized progress from the segment start
3	$(b_k - \tau + 1/2)/d_k$	normalized progress from the segment end
4	$\mathbb{I}[a_k = 1]$	contact with the left window boundary
5	$\mathbb{I}[b_k = W]$	contact with the right window boundary

Entries 2 and 3 are complementary: since $b_k - a_k + 1 = d_k$, they sum to one at every position of the segment, so the pair encodes both the absolute progress and the remaining distance to the segment end, and a single-point segment receives $1/2$ for both. The condition sequence is

$$c_\tau = W_c \left[e_q(q_k) \,\|\, f_m(u_\tau) \right] + b_c, \tag{16}$$

where e_q is a learned embedding of the segment state with dimension $s = \max(16, \lfloor H/4 \rfloor) = 48$ for $H = 192$, f_m is a two-layer perceptron $5 \to s \to s$ with SiLU activation, and W_c, b_c project the concatenated $2s$-dimensional vector to the hidden dimension H. Every position of a segment therefore produces one condition vector, and all C variables at time τ share the same c_τ.

C DURATION-AWARE STATE PLANNER

Inputs. At step k the planner reads the previous state q_{k-1}, the previous duration d_{k-1}, the remaining length r_k, and the segment index $k-1$. Table 6 lists the corresponding embeddings, which are added to form the input representation.

Table 6: Additive input embeddings of the duration-aware state planner.

Signal	Embedding	Size
previous state q_{k-1}	state embedding	4
previous duration d_{k-1}	duration embedding	$W+1$
remaining length r_k	remaining-length embedding	$W+1$
segment index $k-1$	segment-position embedding	W

The state embedding has four entries: the three states plus a begin-of-sequence symbol used at the first step, where the previous duration is set to zero.

Network. The segment sequence is processed by a pre-norm Transformer encoder with 4 layers, 4 attention heads, hidden dimension 128, feed-forward ratio 4, GELU activation, dropout 0.1, and a final layer normalization. A causal mask enforces left-to-right conditioning, and a padding mask accommodates the variable number of segments K within a batch. Two heads read the encoder output: a state head, a linear map from 128 to the three states, and a duration head that is additionally conditioned on the current state through a three-entry embedding and produces one logit for each $d \in \{1, \dots, W\}$.

Masking and objective. Two validity constraints are imposed by setting the corresponding logits to $-\infty$. The state head suppresses the previous state, enforcing $q_k \neq q_{k-1}$, and the duration head suppresses every value above the remaining length, enforcing $1 \leq d_k \leq r_k$. The training objective is the sum of the state and duration cross-entropies over valid segments, with padded positions ignored. Per-window losses are summed over segments, and the batch loss is the mean over windows, so each empirical window contributes with its original frequency. Duration targets are the zero-indexed values $d_k - 1$.

Sampling. Sampling is autoregressive. At each step the next state is drawn from its categorical distribution, the next duration is drawn from the masked categorical distribution over $d \leq r_k$, and the pair is appended to the history. Sequences that reach $r = 0$ leave the active set, and the loop terminates when all sequences have finished. Because every duration is bounded by the remaining length and the final segment consumes it exactly, each sampled sequence has adjacent segments with different states and durations that sum to W.

D SEGMENT-CONDITIONED AXIAL DIT

D.1 TOKENIZATION AND CONDITIONING

Each scalar value is projected to a token

$$h^{(0)}_{\tau,c} = W_x x_{n,\tau,c} + e_{\mathrm{var}}(c),$$

where W_x maps a single scalar to $H = 192$ and e_{var} is a learned embedding with one entry per physical variable. The diffusion-step conditioning is a sinusoidal embedding of n (Section D.3) passed through a two-layer perceptron $H \to 4H \to H$ with SiLU activation and broadcast over all temporal positions; the time-aligned condition c_τ of Eq. (16) is added to it. Temporal positions are not added as a separate embedding: they enter the model through rotary position encoding inside temporal attention.

D.2 BLOCK STRUCTURE

Each of the 8 blocks contains three modulated sublayers. From the conditioning vector, a SiLU-activated linear layer produces nine vectors of dimension H: a shift, a scale, and a gate for each

sublayer. Modulation is $\mathrm{mod}(x, s, c) = x \odot (1 + c) + s$, and the gate scales the sublayer output before the residual addition. The parameters of this modulation layer are initialized to zero, so every block starts as an identity map.

1. **Temporal attention.** The block normalizes the tokens, applies the temporal shift and scale, reshapes them to $(B \cdot C, W, H)$, and performs multi-head self-attention along the temporal axis with 6 heads. Rotary position encoding is applied to queries and keys with base 10000 and head dimension $H/6 = 32$, which requires an even head dimension. The output is gated and added residually.
2. **Variable attention.** The block normalizes the tokens, applies the variable shift and scale, reshapes them to $(B \cdot W, C, H)$, and performs multi-head self-attention across the variable axis with 6 heads and without position encoding. The output is gated and added residually.
3. **Feed-forward.** A normalized and modulated token passes through a SwiGLU layer: a linear map to $2 \times 4H$ is split into a value and a gate, and the output is $W_{\mathrm{out}}(\mathrm{value} \odot \mathrm{SiLU}(\mathrm{gate}))$, with an inner dimension of $4H$. The result is gated and added residually.

All normalization layers are LayerNorm without affine parameters and with $\varepsilon = 10^{-6}$. After the blocks, a final non-affine normalization and a zero-initialized linear map to one scalar produce the predicted v-target at every time–variable position. Gradient checkpointing is applied to each block during training.

D.3 Timestep embedding

The diffusion step n is encoded as

$$\left[\cos(nf_0), \ldots, \cos(nf_{H/2-1}), \sin(nf_0), \ldots, \sin(nf_{H/2-1})\right], \qquad f_i = \exp\left(-\frac{i}{H/2} \ln 10000\right),$$

and the result is passed through a two-layer perceptron $H \to 4H \to H$ with SiLU activation.

E Diffusion process and sampling

E.1 Cosine noise schedule

The noise schedule is defined through the accumulated signal coefficient

$$\bar{\alpha}_n = \frac{\cos^2\left(\frac{n/T + s}{1 + s} \cdot \frac{\pi}{2}\right)}{\cos^2\left(\frac{s}{1 + s} \cdot \frac{\pi}{2}\right)}, \qquad s = 0.008, \tag{17}$$

where T is the number of diffusion steps. The profile is discretized over $T+1$ equally spaced points, the denominator normalizes it so that $\bar{\alpha}_0 = 1$ before any noise is added, and the per-step coefficients are

$$\beta_n = \mathrm{clip}\left(1 - \frac{\bar{\alpha}_n}{\bar{\alpha}_{n-1}},\, 10^{-8},\, 0.999\right), \qquad \alpha_n = 1 - \beta_n,$$

so that $\bar{\alpha}_n = \prod_{j \le n} \alpha_j$. The coefficients of Eq. (6) are then

$$\alpha_n = \sqrt{\bar{\alpha}_n}, \qquad \sigma_n = \sqrt{1 - \bar{\alpha}_n},$$

which satisfies $\alpha_n^2 + \sigma_n^2 = 1$. The clean trajectory and the noise are recovered from a predicted v-target as

$$\hat{x}_0 = \alpha_n x_n - \sigma_n \hat{v}, \qquad \hat{\epsilon} = \sigma_n x_n + \alpha_n \hat{v}.$$

E.2 Sampling

All reported results use the deterministic DDIM sampler with $\eta = 0$ and 50 sampling steps. The sampling timesteps are

$$n_j = \mathrm{round}(\mathrm{linspace}(T - 1, 0, S)), \quad j = 0, \ldots, S - 1,$$

which are strictly decreasing; for $S = 1$ the single timestep is $T - 1$. Starting from $x \sim \mathcal{N}(0, I)$, each step predicts $\hat{v}$, recovers $\hat{x}_0$, and updates

$$x_{n'} = \sqrt{\bar{\alpha}_{n'}}\,\hat{x}_0 + \sqrt{1 - \bar{\alpha}_{n'} - \sigma^2}\,\hat{\epsilon} + \sigma z, \qquad \sigma = \eta\sqrt{\frac{1 - \bar{\alpha}_{n'}}{1 - \bar{\alpha}_n}\left(1 - \frac{\bar{\alpha}_n}{\bar{\alpha}_{n'}}\right)}, \tag{18}$$

where n' is the next timestep and $z \sim \mathcal{N}(0, I)$ when $\eta > 0$, with $z = 0$ otherwise. With $\eta = 0$ the update is deterministic and reduces to $x_{n'} = \sqrt{\bar{\alpha}_{n'}}\,\hat{x}_0 + \sqrt{1 - \bar{\alpha}_{n'}}\,\hat{\epsilon}$. The final update, and any step with $n = 0$, sets $x = \hat{x}_0$ directly.

A stochastic DDPM sampler is also available. It requires the number of sampling steps to equal T and updates

$$x_{n-1} = \frac{\beta_n\sqrt{\bar{\alpha}_{n-1}}}{1 - \bar{\alpha}_n}\hat{x}_0 + \frac{(1 - \bar{\alpha}_{n-1})\sqrt{\alpha_n}}{1 - \bar{\alpha}_n}x_n + \sqrt{\frac{\beta_n(1 - \bar{\alpha}_{n-1})}{1 - \bar{\alpha}_n}}\,z, \qquad z \sim \mathcal{N}(0, I). \tag{19}$$

Generation is seeded with the configured seed unless another value is supplied, and samples are transformed back to the original physical scale with the stored per-channel statistics.

F Implementation of the training objective

With the squared error $e_{i,\tau,c} = (\hat{v}_{i,\tau,c} - v_{i,\tau,c})^2$ over a minibatch of B windows, the terms of Eq. (12) are evaluated as follows.

- The global term is the mean of e over all BWC entries.
- For a state q, the term $\mathcal{L}_q$ is the mean of e over the positions with $s_{i,\tau} = q$, taken over all C channels simultaneously. This is equivalent to the normalization by $C\sum_{i,\tau}\mathbb{I}[s_{i,\tau} = q] + \varepsilon$ in Eq. (9). Only the states actually present in the batch contribute, so $|\mathcal{Q}_B|$ is the number of observed states and the state term is their unweighted mean.
- The boundary term is evaluated on the designated variable only. With $\hat{y}_{i,\tau} = \hat{x}_{0,i,\tau,c^\star}$, the hinge follows Eq. (11), and each per-position violation is multiplied by $\bar{\alpha}_{n_i}$ before averaging over the BW positions; this factor is the signal power $\alpha_{n_i}^2$ in Eq. (12).
- The total objective is $\mathcal{L}_{\text{denoise}} + \lambda_{\text{bnd}}\mathcal{L}_{\text{bnd}}$, with

$$\mathcal{L}_{\text{denoise}} = (1 - \rho)\mathcal{L}_{\text{global}} + \rho\,\frac{1}{|\mathcal{Q}_B|}\sum_{q \in \mathcal{Q}_B}\mathcal{L}_q,$$

  where $\mathcal{L}_{\text{bnd}}$ denotes the averaged weighted hinge.

All C channels therefore contribute to the denoising and state-balanced terms, whereas only the designated variable contributes to boundary consistency. The global term, the three per-state terms, and the boundary term are logged separately during training.

G Data preprocessing and tail-state extraction

G.1 Input format

The implementation consumes a single array of shape $[N, W, C]$ stored in `.npy` format and cast to single precision. The windows are pre-extracted fixed-length segments: the window length is a property of the input array, whereas the stride, the channel selection, and the train/validation/test split are fixed before training and are not part of this implementation. Arrays are memory-mapped by default. The shape must satisfy $N \geq 1$, $W \geq 2$, and $C \geq 1$, and all entries must be finite; the finiteness check is performed in chunks of 4096 samples.

G.2 Per-channel standardization

Per-channel statistics are computed in a streaming fashion over all $N \cdot W$ points of each channel, combining the mean and the sum of squared deviations of consecutive chunks. The variance is

the combined sum of squared deviations divided by $\mathrm{count} - 1$, and the standard deviation is the square root of its non-negative part. A channel whose standard deviation is below 10^{-6} is marked as constant and its standard deviation is set to one, so that its normalization reduces to a shift. Values are normalized as $(x - \mu_c)/\sigma_c$ and restored as $x\sigma_c + \mu_c$. The statistics and the constant-channel mask are stored alongside the run and reused during generation.

G.3 Bounds and tail states

The designated variable defaults to the last channel, and negative indices are supported. The state of a position is determined from raw values before normalization: it is the lower tail if $y < L$, the upper tail if $y > U$, and normal otherwise, so values equal to either bound belong to the normal state. The bounds are mapped to the standardized scale of the designated variable,

$$L' = \frac{L - \mu_{c^\star}}{\sigma_{c^\star}}, \qquad U' = \frac{U - \mu_{c^\star}}{\sigma_{c^\star}},$$

and L', U' are used in the boundary term. Run-length encoding groups adjacent equal states; by construction adjacent segments carry different states, all durations are positive, and the durations sum to W, so the encoding is lossless and decoding reproduces the state sequence exactly. A data inspection command reports the array shape and dtype, the constant channels, the counts of normal, lower-tail, and upper-tail positions, the maximum number of segments per window, and the normalized bounds. The encoded representation is cached as the state sequence, the segment states, the segment durations, the cumulative offsets, and a manifest recording the data fingerprint, the array shape, the bounds, and the designated channel.