E4GEN: Event-level Explainable Extreme-Enhanced Time-series Generation

Lin Jiang, Dahai Yu, Ximiao Li, Guang Wang*
Florida State University
{lin.jiang, dahai.yu, xl24g}@fsu.edu, guang@cs.fsu.edu

Abstract

Generating realistic time series is essential for scientific research and real-world applications. However, existing methods often emphasize overall distributional fidelity while failing to faithfully capture extreme events. To advance existing research, we propose E4GEN, an explainable diffusion framework for extreme event-aware time-series generation, which provides systematic insights into **when**, **what**, and **how** to control extreme-event generation through three key components. (i) **E-Activator** to learn the dataset-adaptive *extreme-control signal activation step* during the denoising process without interfering with regular temporal components (trend and seasonality); (ii) **E-Predictor**, which determines what control signal to enforce through *Self-Driven Semantic Prediction*, where each sample derives its own control signal by inferring the latent extreme-event information during the generation process. It also includes a novel *Data-Conditioned Training, Noise-Initiated Sampling* mechanism to address the issue of unavailable training labels, and (iii) **E-Control**, which specifies how to control extreme-event generation through a trainable *Extreme Control Network*, which transforms the semantic control signal into layer-wise signals and injects it into the denoising process. We evaluate our E4GEN with six datasets and 17 metrics, and extensive experiments show that E4GEN outperforms state-of-the-art models from different dimensions, including overall fidelity, extreme-event fidelity, and downstream utility.

1 Introduction

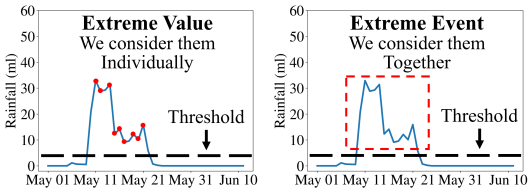
Time-series generation has emerged as an important research direction [1, 2, 3, 4, 5], because it can support a wide range of applications such as simulations [6], data augmentation [7], counterfactual analysis [8], and hypothesis testing [9]. In recent years, many methods have been proposed for this task, covering diverse paradigms, including GANs [1, 10], VAEs [11, 12], Fourier Flows [13], LLMs [5, 14], and diffusion models [4, 15, 16]. Although some works [4, 5] have achieved high overall distributional fidelity, they usually attenuate sparse extreme-event patterns [3, 17], which are also meaningful to capture because they often correspond to high-impact phenomena in the data, such as heatwaves, heavy rainfall, and sudden demand surges [18].

To achieve extreme-aware generation, some recent studies have explored long-tailed generation [19, 20] and heavy-tailed generation [3, 21, 22, 23]. Long-tailed generation improves the fidelity and diversity of tail samples by identifying *which samples are rare* without capturing *which values within each sample are extreme*, so it cannot explicitly characterize within-sample extreme patterns. Although heavy-tailed generation considers within-sample extremes, it mainly focuses on the **value level** by treating extreme values as isolated observations without modeling their correlations or temporal dependencies. Hence, neither type of method can explicitly represent event-level features, such as location, duration, and temporal shape, which is essential for characterizing time-series data and supporting diverse downstream applications.

*Corresponding author.

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

To advance existing work, we explore extreme-aware time-series generation from the **event-level** perspective. Instead of treating extreme values as isolated points, we consider extreme events as coherent temporal structures driven by underlying physical mechanisms. For example, as illustrated in Fig. 1, an extreme rainstorm typically spans a continuous temporal interval, during which rainfall intensities at neighboring timestamps are highly temporally dependent, so the event should be modeled as a coherent temporal process rather than a collection of isolated extreme points. However, there are two major challenges for event-level extreme-aware time-series generation. First, extreme events are localized and sparse in the data, with tightly constrained timing and shape, making them particularly vulnerable to distortion during generation. Second, the generation process of extreme events remains poorly understood, as existing methods provide little insight into when and how extreme events emerge during generation, making it difficult to precisely control and preserve them.



(a) Value-level view (b) Event-level view

Fig. 1: Value-Level vs. Event-Level Extremes

To address these challenges, we propose **E4GEN**, an Event-level Explainable Extreme-Enhanced diffusion framework for time-series GENeration. E4GEN provides systematic insights into when, what, and how to control extreme-event generation through three key designs. (i) **E-Activator** is designed to decide when extreme-control signals should be activated during the generation process. Unlike existing methods that generate general patterns (e.g., trend and seasonality) and extreme events in a coupled manner, E-Activator exploits the coarse-to-fine generation dynamics of diffusion models to learn a *dataset-adaptive control activation step*, where the extreme-control signal is activated only after general temporal structures have been sufficiently established. This design effectively decouples extreme-event guidance from general temporal structure formation. (ii) **E-Predictor** is designed to determine what control signals should be activated. Motivated by the observation that extreme events arise from the underlying temporal context, E-Predictor introduces a *self-driven semantic prediction* (SDSP) module, enabling each sample to derive its own semantic-based extreme-control signal from the general temporal patterns already established during the denoising process. It also introduces a novel *Data-Conditioned Training, Noise-Initiated Sampling* (DCT-NIS) mechanism to address the issue of unavailable training labels. (iii) **E-Control** is designed to answer how semantic-based control signals can be leveraged for event-level extreme-aware generation. Specifically, we design a trainable *Extreme Control Network* that transforms predicted semantics into layer-aligned residual offsets, which are used to guide extreme-event generation while preserving general temporal structures. The contributions of this paper are summarized as follows:

(i) **Conceptually**, we shift extreme-aware time-series generation from sample-level or value-level to an event-level perspective, which are essential for characterizing temporal dynamics in time-series data and supporting diverse downstream applications.

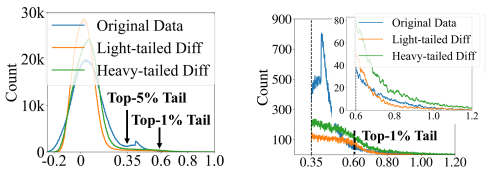
(ii) **Technically**, we propose E4GEN, an explainable diffusion framework for extreme-aware time-series generation, which provides systematic insights into the controllability of extreme-event generation through three core designs: E-Activator identifies when to activate the extreme-event control signal via a dataset-adaptive control activation step; E-Predictor defines what to control through self-driven semantic prediction combined with a DCT-NIS mechanism that addresses the training-sampling mismatch issue; and E-Control determines how to control generation by injecting semantic control signals into the denoising process through a trainable Extreme Control Network.

(iii) **Experimentally**, we evaluate E4GEN on six datasets against 9 baselines in terms of 17 metrics. Extensive experiments show that E4GEN achieves superior performance across different dimensions, such as overall generation fidelity, extreme-aware generation fidelity, and downstream utility.

2 Data-Driven Insights

In this section, we present a data-driven analysis to motivate our problem formulation and design. We summarize three key observations below.

Observation I: Value-level generation often fails to recover the true event-level structure of extremes. In Fig. 2, we compare the generated results of two representative methods: (i) a standard light-tailed diffusion model implemented with Diffusion-TS [4], and (ii) a value-level extreme-enhanced heavy-tailed diffusion model implemented with HeavyDiff [23] with the Original Data. From Fig. 2(a), we found that the generated data generally match the overall distribution, but fail to match the tail. Fig. 2(b) illustrates the top-5% tail distribution, where the original data exhibits a pronounced density surge within the range of 0.35–0.6, indicating that extreme values tend to occur within a specific and non-random value range with clear structural regularity. In contrast, both methods fail to recover this characteristic pattern. Although the heavy-tailed diffusion model generates a heavier tail distribution, it still fails to faithfully capture the underlying event-level extreme structure. Further discussion of value-level extreme-aware generation is provided in Appendix E.1.



(a) Overall Distribution (b) Tail Distribution

Fig. 2: Value-level Extreme Enhancement.

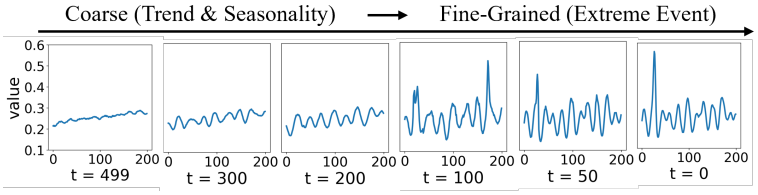


Fig. 3: A Sample Denoising Process

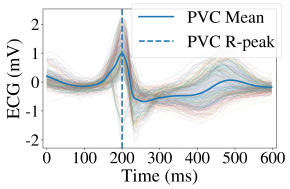


Fig. 4: Stability

Observation II: Diffusion models exhibit coarse-to-fine generation dynamics, progressively evolving from global temporal structures to localized extreme-event patterns. Fig. 3 visualizes the generation process of a time-series sample along the 500-step sampling trajectory of a diffusion model [24]. The trajectory reveals a staged generation process in which the model first establishes coarse global structures, primarily trend and seasonality, and then progressively refines localized extreme events during later sampling stages, especially within the final 100 steps. This observation motivates the design of E4GEN, which dynamically activates extreme-control signals during denoising rather than applying them uniformly throughout the entire generation trajectory. Additional visualizations are provided in Appendix E.2.

Observation III: Extreme events are typically induced by underlying physical processes, and therefore exhibit *intrinsic patterns* rather than random *extreme noise*. To validate this observation, we perform three complementary analyses including cross-sample stability, predictability, and context dependency across six datasets. Fig. 4 shows a cross-sample stability analysis, which indicates that premature ventricular contractions (PVCs) in the ECG dataset preserve clear event-level structures despite variations in amplitude and peak timing. Results for predictability and context dependency are provided in Appendix E.3. This observation can also be theoretically supported by the limited degrees of freedom (DOF) that characterize extreme-event variability [25]: although extreme events may appear irregular, their variations are often governed by a small set of latent factors, making recurring structures learnable by neural networks [26, 27].

3 Preliminaries

3.1 Basic Definitions

In this paper, we use Greek letters to indicate time-series indices and Latin letters for diffusion steps. A multivariate time series is denoted as $\mathbf{x} \in \mathbb{R}^{L \times C}$, where L is the sequence length and C is the number of feature channels. Throughout this paper, the **diffusion process** refers to the forward process that transforms a clean sample $\mathbf{x}_d(t_0)$ into pure noise $\mathbf{x}_d(t_T)$, while the **denoising process** refers to the reverse sampling process from $\mathbf{x}_n(t_T)$ back toward $\mathbf{x}_n(t_0)$. We denote the noise-initiated denoising trajectory as $\{\mathbf{x}_n(t)\}_{t=t_T}^{t_0}$ and each intermediate state denotes $\mathbf{x}_n(t) = [x_{n,\tau,\nu}(t)]_{\tau=1,\nu=1}^{L,C} \in \mathbb{R}^{L \times C}$, where τ indexes positions within the time series, ν is the feature channel, and t is the diffusion step.

3.2 $\mathbf{x}$-Prediction Network as Backbone Denoiser

In E4GEN, the **Backbone Denoiser** refers to the base denoising network that serves as the foundation of the time-series generation process, upon which the extreme-event generation modules are further built. Its primary role is to capture the global temporal patterns of the sequence, particularly long-range trend and seasonality patterns. In this work, we instantiate the Backbone Denoiser using the $\mathbf{x}$-prediction network $\hat{\mathbf{x}}_0^\theta(\cdot)$, a widely adopted and effective paradigm for time-series generation [24, 28, 4, 29], which predicts the clean sample $\mathbf{x}_n(t_0)$ from an intermediate noisy state $\mathbf{x}_n(t)$ at diffusion step t:

$$\hat{\mathbf{x}}_0(t) \triangleq \hat{\mathbf{x}}_0^\theta\big(\mathbf{x}_n(t), t\big) \;\equiv\; \hat{\mathbf{x}}_0^\theta\big(\mathbf{x}_n(t)\big) \text{ for short,} \tag{1}$$

where $\hat{\mathbf{x}}_0(t)$ denotes the predicted clean sample at step t. $\mathbf{x}$-prediction network can provide an interpretable generation trajectory with theoretical support from manifold analysis [24].

Following classical seasonal-trend decomposition [30], we parameterize the $\mathbf{x}$-prediction network as $\hat{\mathbf{x}}_0^\theta(\mathbf{x}(t), t) = \mathrm{Trend}(\mathbf{x}_n(t), \theta_{tr}) + \mathrm{Seasonality}(\mathbf{x}_n(t), \theta_{se})$. The trend and seasonality components are instantiated as [4]:

$$\mathrm{trend}(\mathbf{x}_n(t), \theta_{tr}) = \phi(\tau)^\top \big(W_{tr}\, h(\mathbf{x}_n(t); a_{tr}) + b_{tr}\big), \tag{2}$$

$$\mathrm{seasonality}(\mathbf{x}_n(t), \theta_{se}) = \mathcal{IDFT}(\mathcal{M}_K(\mathcal{DFT}(\mathbf{x}_n(t)), \theta_{se}))\,. \tag{3}$$

where $\phi(\tau) = [1, \tau, \tau^2, \ldots, \tau^p]^\top$ is a polynomial basis, and (W_{tr}, a_{tr}, b_{tr}) are the learnable trend parameters in θ_{tr}. $\mathcal{DFT}$ and $\mathcal{IDFT}$ denote the discrete Fourier transform and its inverse, respectively, while $\mathcal{M}_K(\cdot, \theta_{se})$ selects and refines the Top-K frequency components. More details of the Backbone Denoiser are provided in Appendix F.

3.3 Extreme-Event Definition and Representation

Extreme events are inherently domain-dependent, and their definitions often vary across application scenarios [31]. In E4GEN, we adopt a threshold-based definition following the threshold-exceedance perspective commonly used in extreme value analysis [32], which provides a general and flexible mechanism for identifying extreme events across diverse time-series domains. Specifically, an extreme event is defined as a maximal contiguous segment whose values exceed an extreme threshold θ for upper-tail extremes, or fall below θ for lower-tail extremes. The threshold θ is domain-specific and determined according to prior knowledge or task-specific requirements, rather than being fixed universally across datasets. Appendix G.1 further discusses how E4GEN handles cases where short non-extreme gaps appear within an extreme event.

Since extreme events cannot be directly incorporated into diffusion generation, we propose a novel extreme-event semantic representation that characterizes each event with structured attributes. This design transforms extreme events into a model-accessible and controllable semantic form, enabling explicit event-level control during generation. Specifically, for each time-series sample, we define its **extreme-event semantics** as $EES = \{\mathcal{L}, \mathcal{I}, \mathcal{S}\}$, where $\mathcal{L}$, $\mathcal{I}$, and $\mathcal{S}$ characterize the location, intensity, and temporal shape of all extreme events within it, respectively.

Location $\mathcal{L}$: The location component $\mathcal{L}$ characterizes the temporal occurrence of extreme events, including their count and spans. We define $\mathcal{L} = \{n_{\mathrm{ev}}, \{(\tau_s^m, \tau_e^m)\}_{m=1}^{n_{\mathrm{ev}}}\}$, where n_{ev} denotes the number of events, and (τ_s^m, τ_e^m) are the start and end time indices of the m-th event. Thus, $\mathcal{L}$ specifies when each event occurs and how long it persists.

Intensity $\mathcal{I}$: The intensity component $\mathcal{I}$ quantifies the strength of each event relative to its local background. We define $\mathcal{I} = \{(a_{\mathrm{peak}}^m, a_{\mathrm{prom}}^m, a_{\mathrm{eng}}^m)\}_{m=1}^{n_{\mathrm{ev}}}$. Here, $a_{\mathrm{peak}}^m = \max_{\tau \in [\tau_s^m, \tau_e^m]} |x[\tau]|$ denotes the peak magnitude within the event window; $a_{\mathrm{prom}}^m = a_{\mathrm{peak}}^m - b^m$ denotes the peak prominence above the background level b^m, estimated from nearby non-event points; and $a_{\mathrm{eng}}^m = \sum_{\tau=\tau_s^m}^{\tau_e^m} (x[\tau] - b^m)^2$ denotes the cumulative deviation energy relative to this background level.

Temporal Shape $\mathcal{S}$: The temporal-shape component $\mathcal{S}$ provides a sparse keypoint-based representation of each extreme event, summarizing its local evolution with informative points rather than all values in the event segment. For the m-th event over $[\tau_s^m, \tau_e^m]$, keypoints are selected from $x[\tau_s^m : \tau_e^m]$ to capture its temporal profile, including onset and termination, rising or falling trends, and major local variations. Formally, we define $\mathcal{S} = \{\{(p_{\mathrm{len}}^m, \mathbf{p}_\tau^m, \mathbf{p}_x^m)\}_{m=1}^{n_{\mathrm{ev}}}\}$, where each tuple encodes the keypoint-based shape of the m-th event. Here, p_{len}^m denotes the number of keypoints,

$\mathbf{p}_\tau^m = (p_{\tau,1}^m, \ldots, p_{\tau,p_{\text{len}}^m}^m)$ stores their time indices, and $\mathbf{p}_x^m = (p_{x,1}^m, \ldots, p_{x,p_{\text{len}}^m}^m)$ stores the corresponding values, with $p_{x,i}^m = x[p_{\tau,i}^m]$. In practice, we select structural points that characterize the event trajectory, such as endpoints, turning points, and local extrema. This design is motivated by the intuition that a set of informative points can effectively summarize complex temporal patterns [33].

4 Methodology

Figure 5 shows the overall framework of our proposed **E4GEN**, which integrates three key components that systematically address *when*, *what*, and *how* to control extreme-event generation during the denoising process. Section 4.1 introduces **E-Activator**, which learns to decide the dataset-adaptive control activation step t_{CA}. Section 4.2 presents **E-Predictor**, which estimates the extreme-control signal at t_{CA} with two coupled designs: Self-Driven Semantic Prediction (SDSP) and Data-Conditioned Training, Noise-Initiated Sampling (DCT-NIS) mechanism. Section 4.3 presents **E-Control**, which includes a trainable *Extreme Control Network* to inject the predicted semantic-based control signal into the Backbone Denoiser for guided extreme-event generation.

4.1 E-Activator: Learning the Control Activation Step

Motivated by the coarse-to-fine generation dynamics identified in Observation II (Section 2), E-Activator learns the optimal control activation step, denoted by t_{CA}, for extreme-event control. This activation step is dataset-dependent, since extreme patterns may emerge at different steps of the denoising process in various datasets. For instance, in a fixed 1000-step generation process, heavy-rainfall extremes may begin to appear around $t = 100$ in precipitation data, whereas traffic congestion extremes may emerge around $t = 50$ in traffic data. To avoid disrupting the generation of regular temporal patterns, we only apply the extreme-control signals starting from t_{CA} during the denoising process, which is defined based on two criteria: (1) regular patterns such as trend and seasonality should be sufficiently stabilized, so that activating the control signal will not interfere with their formation; and (2) extreme-event patterns should still be at an early stage of generation, leaving enough remaining denoising steps for the control signal to effectively guide their generation.

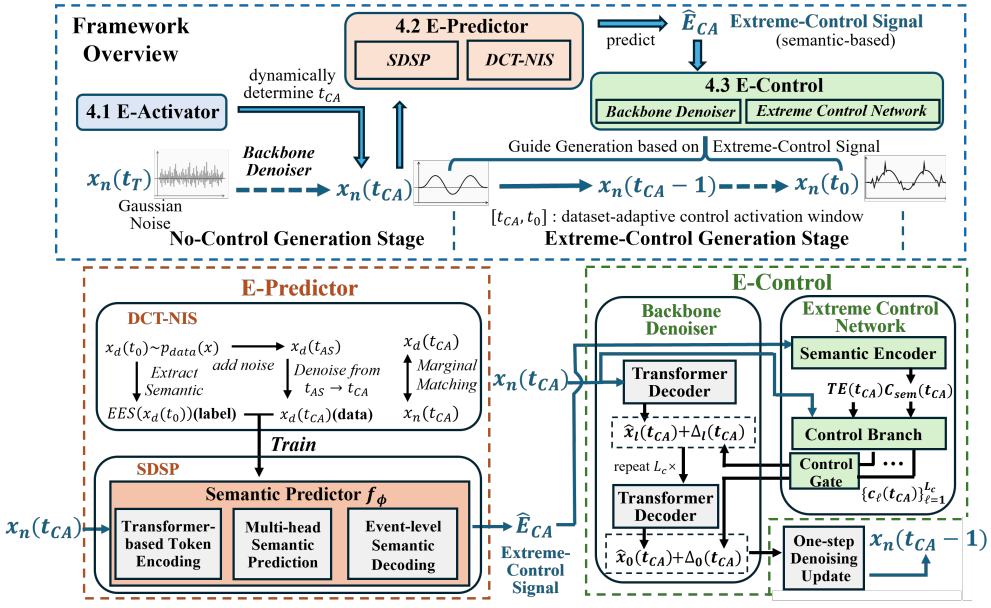


Fig. 5: The Overall Framework of E4GEN

We quantify these criteria using two complementary metrics: Backbone Drift (BD) and Outlier Emergence Score (OES), which are formulated as follows:

$$\text{BD}(t) = \mathbb{E}\left[\frac{1}{k}\sum_{j=0}^{k-1}\frac{\sqrt{\frac{1}{LC}\sum_{\tau=1}^{L}\sum_{\nu=1}^{C}\left(bb_{t-j,\tau,\nu} - bb_{t-j-1,\tau,\nu}\right)^2}}{\sqrt{\frac{1}{LC}\sum_{\tau=1}^{L}\sum_{\nu=1}^{C} bb_{t-j-1,\tau,\nu}^2}}\right] \tag{4}$$

$$\mathrm{OES}(t) = \mathbb{E}\left[\max\left(\left|\frac{\hat{x}_{0,\tau,\nu}(t) - \mathrm{med}_\tau(\hat{x}_{0,\tau,\nu}(t))}{1.4826\,\mathrm{mad}_\tau(\hat{x}_{0,\tau,\nu}(t)) + \varepsilon}\right| - \kappa,\ 0\right)\right] \tag{5}$$

BD quantifies the stability of regular temporal patterns (e.g., trend and seasonality), while OES characterizes the emergence of extreme-event patterns. Smaller values of BD and OES indicate stronger satisfaction of Criteria 1 and 2, respectively. We select t_{CA} through a two-stage procedure. First, BD and OES narrow the search space to a compact interval in which both metrics remain sufficiently low. Then, a lightweight validation grid search is performed within this interval to identify the optimal control activation step. More details are provided in Appendix H.

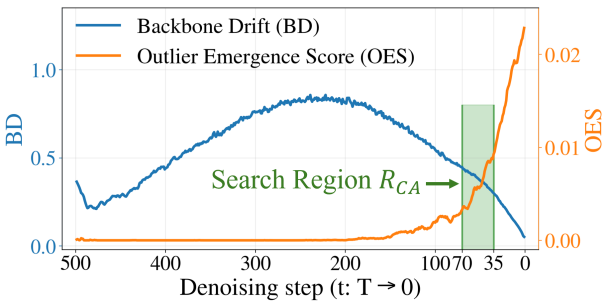


Fig. 6: Control Activation Window via BD and OES

Using Figure 6 as an illustrative example, we visualize the evolution of BD and OES on a dataset to identify a compact interval for t_{CA}. Specifically, we define the search space as $\mathcal{R}_{\mathrm{CA}} = \{t \mid \mathrm{BD}(t) \le \tau_{\mathrm{BD}} \text{ and } \mathrm{OES}(t) \le \tau_{\mathrm{OES}}\}$, where the regular temporal patterns are relatively stable while extreme-event patterns remain underdeveloped. This interval narrows the search space for t_{CA}, allowing a lightweight validation grid search to select the optimal control activation step. Once t_{CA} is selected, as shown in Figure 5, we activate the Extreme-Control Generation Stage and apply the predicted control signals throughout the control activation window $[t_{\mathrm{CA}}, t_0]$.

4.2 E-Predictor: Predicting the Extreme-Control Signal

E-Predictor predicts the extreme-control signal at step t_{CA} for the subsequent Extreme-Control Generation Stage. It consists of two key designs: Self-Driven Semantic Prediction (SDSP) and Data-Conditioned Training with Noise-Initiated Sampling (DCT-NIS). Specifically, SDSP incorporates a Semantic Predictor f_ϕ to enable a self-driven prediction process, deriving sample-specific control signals from the current intermediate denoising state. DCT-NIS further addresses the label-unavailability issue involved in training f_ϕ.

4.2.1 Self-Driven Semantic Prediction (SDSP)

Motivated by Observation III in Section 2, which shows that extreme events are learnable with context-dependent structures rather than random outliers, we design SDSP to predict extreme-control signals from the intermediate denoising state $\mathbf{x}_n(t_{\mathrm{CA}})$. As discussed in Section 4.1, the intermediate state $\mathbf{x}_n(t_{\mathrm{CA}})$ already contains stabilized trend and seasonality patterns, providing informative temporal context for predicting potential extreme events. Since extreme events are strongly associated with such context, we design a Semantic Predictor f_ϕ in SDSP, which consists of *Transformer-based token encoding*, *multi-head semantic prediction*, and *event-level semantic decoding*, to infer the most likely extreme-event semantics (Section 3.3) that will emerge in the final denoised sample $\mathbf{x}_n(t_0)$ based on $\mathbf{x}_n(t_{\mathrm{CA}})$. The predicted semantics $\hat{E}_{\mathrm{CA}}$ are then used as extreme-control signals to guide the subsequent denoising process toward realistic extreme-event patterns. In this way, each sample will receive a distinct control signal derived from its own intermediate denoising state. Detailed design will be provided in Appendix I.

4.2.2 Data-Conditioned Training, Noise-Initiated Sampling (DCT-NIS) Mechanism

We also design the DCT-NIS mechanism to address the label-unavailability issue in training the Semantic Predictor f_ϕ. Specifically, during sampling, the intermediate state $\mathbf{x}_n(t_{\mathrm{CA}})$ is generated from pure noise, while its corresponding final denoised sample $\mathbf{x}_n(t_0)$ has not yet been produced. Therefore, the ground-truth extreme-event semantics required to supervise f_ϕ are unavailable at this stage. DCT-NIS resolves this challenge by training f_ϕ on data-conditioned intermediate states paired with accessible real-data semantic labels, and then transferring the learned predictor to guide

noise-initiated states during sampling. It consists of two processes: data-conditioned training and noise-initiated sampling.

Data-conditioned training process. Given a clean sample $\mathbf{x}_d(t_0) \sim p_{\text{data}}$, we introduce an earlier alignment start step t_{AS} positioned before and close to t_{CA}. The clean sample is first diffused to t_{AS} and then denoised for a few steps from t_{AS} to t_{CA}. This allows the resulting data-conditioned state $\mathbf{x}_d(t_{\text{CA}})$ to experience the same denoising path as the noise-initiated state used during sampling:

$$\mathbf{x}_d(t-1) \sim p_\theta(\mathbf{x}_d(t-1) \mid \mathbf{x}_d(t)), \quad t = t_{\text{AS}}, \ldots, t_{\text{CA}}+1. \tag{6}$$

Here, $p_\theta(\mathbf{x}(t-1)|\mathbf{x}(t))$ denotes the one-step denoising transition introduced in Eq. 13. The Semantic Predictor f_ϕ takes the $\mathbf{x}_d(t_{\text{CA}})$ as input, with the corresponding extreme-event semantics $EES(\mathbf{x}_d(t_0))$ in final denoised sample $\mathbf{x}_d(t_0)$ serving as the supervision label:

$$\text{Semantic Predictor } f_\phi: \ \hat{\mathbf{x}}_0^\theta\big(\mathbf{x}_d(t_{\text{CA}})\big) \mapsto EES(\mathbf{x}_d(t_0)). \tag{7}$$

Noise-initiated sampling process. This stage is initialized with pure Gaussian noise $\mathbf{x}_n(t_T)$, which is denoised to obtain $\mathbf{x}_n(t_{\text{CA}})$. The trained predictor f_ϕ is then applied to $\mathbf{x}_n(t_{\text{CA}})$:

$$\hat{E}_{\text{CA}} \triangleq f_\phi\big(\hat{\mathbf{x}}_0^\theta\big(\mathbf{x}_n(t_{\text{CA}})\big)\big) = EES(\mathbf{x}_n(t_0)), \tag{8}$$

where $\hat{E}_{\text{CA}}$ denotes the predicted extreme-event semantics and serves as the extreme-control signal. Although DCT-NIS induces a training-sampling distribution shift between $\mathbf{x}_d(t_{\text{CA}})$ and $\mathbf{x}_n(t_{\text{CA}})$, we prove that the shift remains bounded under the DDPM setting [34], with two lemmas shown below and proofs are provided in Appendix J.

Lemma 1 (Semantic Consistency)**.** *Let $\mathbf{x}_d(t_0) \sim p_{\text{data}}$ and let $\mathbf{x}_d(t_{\text{AS}}) \sim q(\mathbf{x}_d(t_{\text{AS}}) \mid \mathbf{x}_d(t_0))$, where q denotes the forward diffusion process. Suppose that the denoising trajectory satisfies bounded reconstruction drift and that the event-level semantics is locally stable. If the reverse span from the alignment start step t_{AS} to the control activation step t_{CA} is sufficiently short, then the anchored semantics $EES(\mathbf{x}_d(t_0))$ remains a valid supervision signal for the predictor at $\mathbf{x}_d(t_{\text{CA}})$.*

Lemma 2 (Marginal Alignment)**.** *Under the current DDPM setting, the marginal distribution of the data-conditioned state $\mathbf{x}_d(t_{\text{CA}})$ remains close to that of the noise-initiated state $\mathbf{x}_n(t_{\text{CA}})$.*

4.3 E-Control: Semantics-Conditioned Control for Extreme-Event Generation

We then propose **E-Control**, a semantics-conditioned control module for extreme-event time-series generation. E-Control consists of two parts: a **Backbone Denoiser**, which serves as the base denoising network (Section 3.2), and a trainable **Extreme Control Network**, which is a plugin module to inject semantic control signals $\hat{E}_{\text{CA}}$ for extreme-event generation. The Extreme Control Network contains three key components: *Semantic Encoder*, *Control Branch*, and *Control Gate*.

Semantic Encoder. The Semantic Encoder maps the extreme-control signal $\hat{E}_{\text{CA}}$ into semantic features for the Control Branch. It embeds each event's location, intensity, and temporal shape into an event-aware token, and aggregates all tokens with a lightweight set encoder:

$$\mathbf{e}_m = \text{Emb}(\mathcal{L}^m, \mathcal{I}^m, \mathcal{S}^m), \quad \mathbf{C}_{\text{sem}}(t_{\text{CA}}) = \text{SetEnc}(\{\{\mathbf{e}_m\}_{m=1}^{n_{\text{ev}}}\}). \tag{9}$$

The resulting representation $\mathbf{C}_{\text{sem}}(t_{\text{CA}})$ provides event-level conditioning for layer-aligned control.

Control Branch. The Control Branch projects $\mathbf{C}_{\text{sem}}(t_{\text{CA}})$ into layer-wise control features aligned with the Backbone Denoiser. It matches the injection depth of the Backbone Denoiser, producing one control feature per injection layer. Conditioned on the denoising state $\mathbf{x}_n(t_{\text{CA}})$, the time-step embedding $\text{TE}(t_{\text{CA}})$, and the semantic $\mathbf{C}_{\text{sem}}(t_{\text{CA}})$, the Control Branch computes:

$$\mathbf{c}_\ell(t_{\text{CA}}) = \text{CB}_\ell(\mathbf{c}_{\ell-1}(t_{\text{CA}}), \text{TE}(t_{\text{CA}}), \mathbf{C}_{\text{sem}}(t_{\text{CA}})), \quad \ell = 1, \ldots, L_c, \tag{10}$$

where $\mathbf{c}_0(t_{\text{CA}})$ is initialized from $\mathbf{x}_n(t_{\text{CA}})$, and L_c equals the number of injection layers in the Backbone Denoiser. Here, $\mathbf{c}_\ell(t_{\text{CA}})$ denotes the control feature for the ℓ-th injection layer, CB_ℓ the corresponding control block, and $\text{TE}(\cdot)$ the time-step embedding function.

Control Gate. The Control Gate injects layer-aligned control features into the Backbone Denoiser as gated residual offsets. For each injection layer, it computes

$$\Delta_\ell(t_{\text{CA}}) = \sigma(\mathbf{W}_\ell^{(g)}\mathbf{c}_\ell(t_{\text{CA}})) \odot (\mathbf{W}_\ell^{(\Delta)}\mathbf{c}_\ell(t_{\text{CA}})), \quad \ell = 1, \ldots, L_c. \tag{11}$$

Here, $\sigma(\cdot)$ is the sigmoid function, $\odot$ denotes element-wise multiplication, and $\mathbf{W}_\ell^{(g)}$ and $\mathbf{W}_\ell^{(\Delta)}$ map the control feature to the gate value and residual offset. The offset $\Delta_\ell(t_{\mathrm{CA}})$ is added to the hidden representation $\hat{\mathbf{x}}_\ell(t_{\mathrm{CA}})$ or output prediction $\hat{\mathbf{x}}_0(t_{\mathrm{CA}})$ of the Backbone Denoiser, enabling adaptive extreme-event enhancement while preserving background temporal patterns.

Together, these three components transform $\hat{E}_{\mathrm{CA}}$ into layer-specific residual offsets, with each offset injected into the output representation of its corresponding layer in the Backbone Denoiser. More design details will be provided in Appendix K.

5 Evaluation

Research Questions (RQs). Our evaluation is organized around nine key research questions. RQ1: Does E4GEN faithfully generate extreme events? RQ2: Does E4GEN preserve overall generation fidelity? RQ3: Are the data generated by E4GEN useful for downstream tasks? RQ4: Can E4GEN provide interpretable insights into the denoising process? RQ5: How accurately can E4GEN predict extreme-event semantics? RQ6: How does each key module contribute to the performance of E4GEN? RQ7: How sensitive is E4GEN to key hyperparameters? RQ8: Is E4GEN computationally efficient? and RQ9: Can E4GEN controllably generate specified extreme events?

Datasets. We evaluate E4GEN on six datasets (five real and one synthetic) across four domains: climate, healthcare, energy, and transportation. The real-world datasets include WEA-TEMP and WEA-PREC, LTST-ECG, HH-POWER, and PEMS-SF. These datasets cover representative extreme events, including cold temperature, heavy rainfall, ST-segment elevation, high electricity consumption, and traffic congestion. Detailed descriptions are provided in Appendix L.1.

Metrics. We evaluate generation performance using 17 metrics from *overall generation quality* and *extreme-event generation quality* perspectives. For overall generation, we report five fidelity metrics (i.e., Wass. [35], KS [36], JS [37], MMD [38], and ACD [39]) following common practice [40], and three general downstream utility metrics (i.e., Context-FID [41] for representation learning, Pred [1] for prediction, and Recon [42] for reconstruction). For extreme-event generation, we introduce seven event-level fidelity metrics, including EM-W1, EC-Diff, ED-W1, EP-W1, EWass, EJS, and EMMD, along with two extreme-oriented downstream utility metrics, i.e., EPred for extreme-event prediction and ERecon for anomaly detection. More details are in Appendix L.2.

Baselines. We compare E4GEN with nine representative generative baselines across six paradigms: TimeGAN [1] (GAN), TimeVAE [11] and koVAE [12] (VAE), F-Flow [13] (flow), DiffWave [15] and Diffusion-TS [4] (diffusion), SDForger [5] (LLM), and two extreme-aware methods, FIDE [3] and HeavyDiff [23]. Detailed baseline descriptions are provided in Appendix L.3.

5.1 Generation Fidelity and Downstream Utility

To answer RQs 1–3, we evaluate E4GEN from three perspectives: extreme-event generation fidelity, overall generation fidelity, and downstream utility. Specifically, downstream utility is assessed under the train-on-generated, test-on-original protocol, where models are trained on generated data and evaluated on real data. Across 102 dataset–metric comparisons, E4GEN achieves state-of-the-art performance in 79.4% of cases and ranks among the top two methods in 95.1%, demonstrating strong and consistent advantages. Below, we present the results on the WEA-TEMP dataset as a representative case study, while additional experimental results are provided in Appendix M.

As shown in Tables 1 and 2, E4GEN achieves the best performance on most metrics across overall fidelity, extreme-event fidelity, and downstream utility. In the tables, ∞ indicates that the corresponding metric cannot be computed because no extreme-event time points are available. For overall fidelity, E4GEN achieves the strongest performance on the distributional discrepancy metrics (Wass., KS, JS, and MMD) as well as the temporal-dependency metric (ACD). For extreme-event fidelity, E4GEN also outperforms competing methods on most metrics, with particularly notable gains on ED-W1 and EWass., demonstrating superior modeling of extreme-event duration distributions and temporal-pattern distributions. These improvements mainly stem from the proposed extreme-event semantic design, which explicitly captures event location, intensity, and temporal shape. For downstream utility, E4GEN achieves strong Context-FID performance for representation learning and the best Pred score for prediction tasks. Notably, its leading EPred and ERecon scores further demonstrate the effectiveness of E4GEN-generated data for extreme-oriented downstream applications, such as

Table 1: Overall Generation Fidelity and General Downstream Utility

Dataset	Type	Method	Wass.	KS	JS	MMD	ACD	Context-FID	Pred	Recon
WEA-TEMP	GAN	TimeGAN (NeurIPS'19) [1]	1.1950	0.0947	0.1576	0.3650	0.8405	5.1615	0.1662	0.1165
	VAE	TimeVAE (ArXiv'21) [11]	1.8238	0.1265	0.1385	0.1781	1.0735	1.2312	0.1202	0.0326
		koVAE (ICLR'24) [12]	3.5138	0.2662	0.3984	0.3388	1.8261	2.9273	0.1851	0.0417
	Flow	F-Flow (ICLR'21) [13]	21.002	0.9955	0.8256	1.2082	10.519	32.013	1.5189	1.4458
	Diffusion	DiffWave (ICLR'21) [15]	3.9978	0.3056	0.4726	0.3808	2.0716	3.4627	0.1850	0.0303
		Diffusion-TS (ICLR'24) [4]	1.2845	0.1033	0.0916	0.1116	0.6605	0.2271	0.1116	0.0283
	LLM	SDForger (NeurIPS'25) [5]	1.5069	0.1076	0.1551	0.1317	0.8370	1.5304	0.1308	0.0539
	Extreme-Aware	FIDE (NeurIPS'24) [3]	22.225	0.7448	0.6314	0.7962	11.290	24.775	0.3035	0.1573
		HeavyDiff (ICLR'25) [23]	1.0998	0.0950	0.0862	0.0995	0.5793	0.1986	0.1185	**0.0257**
	Ours	**E4GEN**	**0.9346**	**0.0846**	**0.0851**	**0.0858**	**0.4938**	**0.1584**	**0.1084**	0.0276

Table 2: Extreme-Event Generation Fidelity and Extreme-Oriented Downstream Utility

Dataset	Type	Method	EM-W1	EC-Diff	ED-W1	EP-W1	EWass.	EJS	EMMD	EPred	ERecon
WEA-TEMP	GAN	TimeGAN [1]	3.7958	5583	2.0428	3.8054	10.961	0.6318	10.893	0.3093	0.1461
	VAE	TimeVAE [11]	1.3089	4864	1.6862	1.1608	10.600	0.5942	17.539	0.1629	0.0615
		koVAE [12]	1.8624	11425	2.3364	1.5521	∞	∞	∞	∞	∞
	Flow	F-Flow [13]	1.8883	20822	14.176	1.6071	22.109	0.8305	5.0478	0.6524	0.5295
	Diffusion	DiffWave [15]	1.7314	10654	2.2313	1.5311	∞	∞	∞	∞	∞
		Diffusion-TS [4]	0.4373	2587	0.4235	0.4267	4.7470	0.4708	8.4300	0.1163	0.0593
	LLM	SDForger [5]	0.3644	3242	1.0477	0.3736	9.1781	0.4411	**4.1654**	0.1282	0.0763
	Extreme-Aware	FIDE [3]	5.1486	15767	5.8725	12.335	64.284	0.7782	23.947	0.2452	0.6922
		HeavyDiff [23]	0.4115	3115	0.6909	0.3675	5.8522	0.4909	6.9037	0.1233	0.0579
	Ours	**E4GEN**	**0.3574**	**1090**	**0.2916**	**0.3594**	**2.2081**	**0.4216**	6.1463	**0.1158**	**0.0522**

extreme-event prediction and anomaly detection. Among the baselines, diffusion-based methods (e.g., Diffusion-TS [4] and HeavyDiff [23]) and the LLM-based SDForger [5] remain competitive, showing their capability in global temporal generation and fine-grained detail preservation.

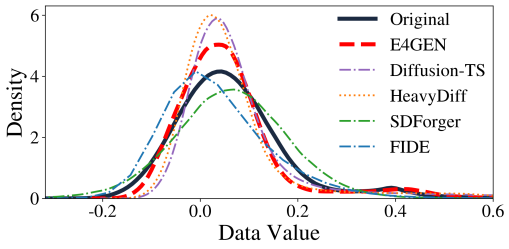


Fig. 7: Overall distribution comparison.

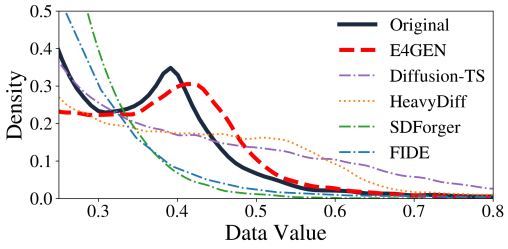


Fig. 8: Tail distribution comparison.

As shown in Fig. 7, all methods generally capture the overall distribution, although with varying levels of deviation. However, in Fig. 8, only E4GEN (red) closely aligns with the original data distribution (black) in the extreme-value region, while the other methods largely fail to reproduce this characteristic pattern. These results suggest that existing methods struggle to capture localized extreme-event structures, whereas E4GEN more faithfully recovers their distributional characteristics through explicit extreme-event semantic modeling.

5.2 Further Analysis

To answer RQ4, we visualize the denoising trajectory to analyze how extreme events emerge and are controlled in E4GEN. The results show that the generated data evolves from the initial Gaussian

distribution toward the target distribution. After control activation, the proportion of extreme values rapidly approaches the target level, whereas it remains consistently below this level without control. These findings reveal the interpretable generation dynamics of E4GEN and demonstrate how the control signal effectively guides extreme-event generation. Details are shown in Appendix L.4.

To answer RQ5, we evaluate the accuracy of E-Predictor in predicting extreme-event semantics across six datasets. The results show that E-Predictor achieves Event-F1 scores of 0.713–0.872 and Span-IoU values of 0.572–0.774, effectively identifying and localizing extreme events. Furthermore, E-Predictor accurately estimates event peaks and temporal shapes. These results demonstrate that E-Predictor provides reliable and semantically meaningful control signals to support extreme-event generation in E4GEN. Details are shown in Appendix L.5.

To answer RQ6, we conduct an ablation study to evaluate the contributions of extreme-event semantic representation, E-Activator, and E-Predictor to E4GEN. The results show that the full E4GEN achieves the best performance across all overall fidelity, extreme-event fidelity, and downstream utility metrics. Notably, removing E-Activator causes the largest performance degradation, highlighting the importance of selecting an appropriate control activation step. Removing the semantic representation or E-Predictor also degrades generation quality, confirming the effectiveness of structured semantic modeling and dynamic semantic prediction. Details are shown in Appendix L.6.

To answer RQ7, we analyze the sensitivity of E4GEN to the control activation step t_{CA} and the alignment start step t_{AS}. The results show that different t_{CA} selection strategies within the valid control window maintain competitive generation performance, with the default strategy achieving the best results, demonstrating E4GEN's robustness to control activation timing. Furthermore, selecting t_{AS} close to t_{CA} helps reduce the distributional discrepancy between training and sampling states, improving the reliability of state alignment. Details are shown in Appendix L.7.

To answer RQ8, we evaluate the training and sampling efficiency of E4GEN across six datasets. The results show that E4GEN requires only 0.09–0.12 seconds per training epoch and 0.09–0.24 seconds to generate each sample. These findings demonstrate that E4GEN maintains low computational overhead while achieving high-quality extreme-event generation, highlighting its efficiency and scalability for practical applications. Details are shown in Appendix L.8.

To answer RQ9, we evaluate the controllability of E4GEN under user-specified extreme-event semantics. The results show that E4GEN generates extreme-event patterns that closely follow the specified event locations, temporal spans, and target intensities across three different configurations. The generated extreme events occur at the specified locations and closely match the target durations and peak intensities. These findings demonstrate that E4GEN effectively translates external semantic control signals into localized extreme-event patterns, enabling controllable extreme-event generation. Details are shown in Appendix L.9.

6 Conclusion

In this paper, we propose **E4GEN**, an explainable diffusion framework for extreme-aware time-series generation. To address the limitation that existing methods often fail to faithfully capture extreme events, we formulate extreme-event generation as an interpretable control problem during denoising, centered on when, what, and how to guide the generation process. E4GEN introduces three key components: (i) *E-Activator*, which identifies when to activate extreme-control signals; (ii) *E-Predictor*, which determines what semantic-based extreme-control signals to enforce in the denoising process; and (iii) *E-Control*, which specifies how to inject these signals to enhance extreme-event generation while preserving global temporal patterns. Extensive experiments on six datasets demonstrate that E4GEN outperforms baselines across 17 metrics spanning overall generation fidelity, extreme-event generation fidelity, and downstream utility.

Acknowledgments and Disclosure of Funding

We thank the reviewers for their valuable feedback and constructive suggestions. This work was partially supported by the National Science Foundation (NSF) under Grant No. 2411152, the National Artificial Intelligence Research Resource (NAIRR) under Award No. 240332, and the FSU/AWS Computer Support Seed Fund.

References

[1] Jinsung Yoon, Daniel Jarrett, and Mihaela van der Schaar. Time-series generative adversarial networks. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

[2] Daniel Jarrett, Ioana Bica, and Mihaela van der Schaar. Time-series generation by contrastive imitation. In *Advances in Neural Information Processing Systems*, volume 34, pages 28968–28982, 2021.

[3] Asadullah Hill Galib, Pang-Ning Tan, and Lifeng Luo. FIDE: Frequency-inflated conditional diffusion model for extreme-aware time series generation. In *Advances in Neural Information Processing Systems*, volume 37, pages 114434–114457, 2024.

[4] Xinyu Yuan and Yan Qiao. Diffusion-TS: Interpretable diffusion for general time series generation. In *International Conference on Learning Representations*, 2024.

[5] Cécile Rousseau, Tobia Boschi, Giandomenico Cornacchia, Dhaval Salwala, Alessandra Pascale, and Juan Bernabe Moreno. Forging time series with language: A large language model approach to synthetic data generation. In *Advances in Neural Information Processing Systems*, volume 38, 2025.

[6] Hang Lou, Siran Li, and Hao Ni. PCF-GAN: Generating sequential data via the characteristic function of measures on the path space. In *Advances in Neural Information Processing Systems*, volume 36, pages 39755–39781, 2023.

[7] Alexander Nikitin, Letizia Iannucci, and Samuel Kaski. TSGM: A flexible framework for generative modeling of synthetic time series. In *Advances in Neural Information Processing Systems*, volume 37, pages 129042–129061, 2024.

[8] Jingquan Yan and Hao Wang. Self-interpretable time series prediction with counterfactual explanations. In *International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 39110–39125, 2023.

[9] Thomas Schreiber and Andreas Schmitz. Surrogate time series. *Physica D: Nonlinear Phenomena*, 142(3-4):346–382, 2000.

[10] Jinsung Jeon, Jeonghak Kim, Haryong Song, Seunghyeon Cho, and Noseong Park. GT-GAN: General purpose time series synthesis with generative adversarial networks. In *Advances in Neural Information Processing Systems*, volume 35, pages 36999–37010, 2022.

[11] 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.

[12] Ilan Naiman, N. Benjamin Erichson, Pu Ren, Michael W. Mahoney, and Omri Azencot. Generative modeling of regular and irregular time series data via Koopman VAEs. In *International Conference on Learning Representations*, 2024.

[13] Ahmed Alaa, Alex James Chan, and Mihaela van der Schaar. Generative time-series modeling with Fourier flows. In *International Conference on Learning Representations*, 2021.

[14] Mingtian Tan, Mike Merrill, Vinayak Gupta, Tim Althoff, and Tom Hartvigsen. Are language models actually useful for time series forecasting? In *Advances in Neural Information Processing Systems*, volume 37, pages 60162–60191, 2024.

[15] Zhifeng Kong, Wei Ping, Jiaji Huang, Kexin Zhao, and Bryan Catanzaro. DiffWave: A versatile diffusion model for audio synthesis. In *International Conference on Learning Representations*, 2021.

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

[17] Eoin Brophy, Zhengwei Wang, Qi She, and Tomás Ward. Generative adversarial networks in time series: A systematic literature review. *ACM Computing Surveys*, 55(10):1–31, 2023.

[18] Jingyi Gu, Xuan Zhang, and Guiling Wang. Beyond the norm: A survey of synthetic data generation for rare events. *arXiv preprint arXiv:2506.06380*, 2025.

[19] Tianjiao Zhang, Huangjie Zheng, Jiangchao Yao, Xiangfeng Wang, Mingyuan Zhou, Ya Zhang, and Yanfeng Wang. Long-tailed diffusion models with oriented calibration. In *International Conference on Learning Representations*, 2024.

[20] Esther Rodriguez, Monica Welfert, Samuel McDowell, Nathan Stromberg, Julian Antolin Camarena, and Lalitha Sankar. Improving generation quality of long-tailed diffusion via disentangled latent representations. In *NeurIPS Workshop on Structured Probabilistic Inference & Generative Modeling*, 2025.

[21] Priyank Jaini, Ivan Kobyzev, Yaoliang Yu, and Marcus Brubaker. Tails of Lipschitz triangular flows. In *International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 4673–4681, 2020.

[22] Juno Kim, Jaehyuk Kwon, Mincheol Cho, Hyunjong Lee, and Joong-Ho Won. t^3-variational autoencoder: Learning heavy-tailed data with Student's t and power divergence. In *International Conference on Learning Representations*, 2024.

[23] Kushagra Pandey, Jaideep Pathak, Yilun Xu, Stephan Mandt, Michael Pritchard, Arash Vahdat, and Morteza Mardani. Heavy-tailed diffusion models. In *International Conference on Learning Representations*, 2025.

[24] Tianhong Li and Kaiming He. Back to basics: Let denoising generative models denoise. *arXiv preprint arXiv:2511.13720*, 2025.

[25] Edward N. Lorenz. Deterministic nonperiodic flow. *Journal of the Atmospheric Sciences*, 20(2):130–141, 1963.

[26] Eduardo G. Altmann and Holger Kantz. Recurrence time analysis, long-term correlations, and extreme events. *Physical Review E*, 71(5):056106, 2005.

[27] Armin Bunde, Jan F. Eichner, Shlomo Havlin, and Jan W. Kantelhardt. The effect of long-term correlations on the return periods of rare events. *Physica A: Statistical Mechanics and its Applications*, 330(1-2):1–7, 2003.

[28] Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 8162–8171, 2021.

[29] Jinseong Park, Seungyun Lee, Woojin Jeong, Yujin Choi, and Jaewook Lee. TimeBridge: Better diffusion prior design with bridge models for time series generation. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.1*, pages 1135–1146, 2026.

[30] Robert B. Cleveland, William S. Cleveland, Jean E. McRae, and Irma Terpenning. STL: A seasonal-trend decomposition procedure based on Loess. *Journal of Official Statistics*, 6(1):3–73, 1990.

[31] Lauren E. McPhillips, Heejun Chang, Mikhail V. Chester, Yaella Depietri, Erin Friedman, Nancy B. Grimm, John S. Kominoski, Timon McPhearson, Pablo Méndez-Lázaro, Emma J. Rosi, et al. Defining extreme events: A cross-disciplinary review. *Earth's Future*, 6(3):441–455, 2018.

[32] Anthony C. Davison and Richard L. Smith. Models for exceedances over high thresholds. *Journal of the Royal Statistical Society: Series B (Methodological)*, 52(3):393–425, 1990.

[33] Lexiang Ye and Eamonn Keogh. Time series shapelets: A new primitive for data mining. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 947–956, 2009.

[34] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, volume 33, pages 6840–6851, 2020.

[35] Cédric Villani. *Optimal Transport: Old and New*, volume 338 of *Grundlehren der mathematischen Wissenschaften*. Springer, 2009.

[36] Nikolai V. Smirnov. On the estimation of the discrepancy between empirical curves of distribution for two independent samples. *Bulletin Mathématique de l'Université de Moscou*, 2(2):3–14, 1939.

[37] Jianhua Lin. Divergence measures based on the Shannon entropy. *IEEE Transactions on Information Theory*, 37(1):145–151, 1991.

[38] Arthur Gretton, Karsten M. Borgwardt, Malte J. Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *Journal of Machine Learning Research*, 13:723–773, 2012.

[39] Katsiaryna Mirylenka, Michele Dallachiesa, and Themis Palpanas. Data series similarity using correlation-aware measures. In *Proceedings of the 29th International Conference on Scientific and Statistical Database Management*, pages 1–12, 2017.

[40] Yihao Ang, Qiang Huang, Yifan Bao, Anthony K. H. Tung, and Zhiyong Huang. TSGBench: Time series generation benchmark. *Proceedings of the VLDB Endowment*, 17(3):305–318, 2023.

[41] Paul Jeha, Michael Bohlke-Schneider, Pedro Mercado, Shubham Kapoor, Rajbir Singh Nirwan, Valentin Flunkert, Jan Gasthaus, and Tim Januschowski. PSA-GAN: Progressive self attention GANs for synthetic time series. In *International Conference on Learning Representations*, 2022.

[42] Pankaj Malhotra, Anusha Ramakrishnan, Gaurangi Anand, Lovekesh Vig, Puneet Agarwal, and Gautam Shroff. LSTM-based encoder-decoder for multi-sensor anomaly detection. *arXiv preprint arXiv:1607.00148*, 2016.

[43] Kaiwen Xia, Li Lin, Shuai Wang, Qi Zhang, Shuai Wang, and Tian He. ProST: Prompt future snapshot on dynamic graphs for spatio-temporal prediction. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 1*, pages 1645–1656, 2025.

[44] Kaiwen Xia, Li Lin, Qi Zhang, Xinrui Zhang, Shuai Wang, Xuming Hu, and Philip S Yu. CSSG: A continuous spatio-temporal graph learning framework with scalable spatial granularity. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 1*, pages 1626–1637, 2026.

[45] Olof Mogren. C-RNN-GAN: Continuous recurrent neural networks with adversarial training. *arXiv preprint arXiv:1611.09904*, 2016.

[46] Cristóbal Esteban, Stephanie L. Hyland, and Gunnar Rätsch. Real-valued (medical) time series generation with recurrent conditional GANs. *arXiv preprint arXiv:1706.02633*, 2017.

[47] Junyoung Chung, Kyle Kastner, Laurent Dinh, Kratarth Goel, Aaron Courville, and Yoshua Bengio. A recurrent latent variable model for sequential data. In *Advances in Neural Information Processing Systems*, volume 28, 2015.

[48] 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*, volume 139 of *Proceedings of Machine Learning Research*, pages 8857–8868, 2021.

[49] Yusuke Tashiro, Jiaming Song, Yang Song, and Stefano Ermon. CSDI: Conditional score-based diffusion models for probabilistic time series imputation. In *Advances in Neural Information Processing Systems*, volume 34, pages 24804–24816, 2021.

[50] Rongchao Xu, Lin Jiang, Dahai Yu, Ximiao Li, and Guang Wang. SynHAT: A two-stage coarse-to-fine diffusion framework for synthesizing human activity traces. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 10(2):1–35, 2026.

[51] Rongchao Xu, Kunlin Cai, Lin Jiang, Dahai Yu, Zhiqing Hong, Yuan Tian, and Guang Wang. GeoGen: A two-stage coarse-to-fine framework for fine-grained synthetic location-based social network trajectory generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 1373–1381, 2026.

[52] Hao Xue and Flora D. Salim. PromptCast: A new prompt-based learning paradigm for time series forecasting. *IEEE Transactions on Knowledge and Data Engineering*, 36(11):6851–6864, 2024.

[53] Nate Gruver, Marc Finzi, Shikai Qiu, and Andrew Gordon Wilson. Large language models are zero-shot time series forecasters. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

[54] Tian Zhou, Peisong Niu, Xue Wang, Liang Sun, and Rong Jin. One fits all: Power general time series analysis by pretrained LM. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

[55] Ming Jin, Shiyu Wang, Lintao Ma, Zhixuan Chu, James Y. Zhang, Xiaoming Shi, Pin-Yu Chen, Yuxuan Liang, Yuan-Fang Li, Shirui Pan, and Qingsong Wen. Time-LLM: Time series forecasting by reprogramming large language models. In *International Conference on Learning Representations*, 2024.

[56] Abdul Fatir Ansari, Lorenzo Stella, Caner Turkmen, Xiyuan Zhang, Pedro Mercado, Huibin Shen, Oleksandr Shchur, Syama Sundar Rangapuram, Sebastian Pineda Arango, Shubham Kapoor, Jasper Zschiegner, Danielle C. Maddix, and Yuyang Wang. Chronos: Learning the language of time series. *Transactions on Machine Learning Research*, 2024.

[57] Dario Shariatian, Umut Simsekli, and Alain Durmus. Heavy-tailed diffusion with denoising Lévy probabilistic models. In *International Conference on Learning Representations*, 2025.

[58] Siddharth Bhatia, Arjit Jain, and Bryan Hooi. ExGAN: Adversarial generation of extreme samples. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 6841–6849, 2021.

[59] Lin Jiang, Rongchao Xu, and Guang Wang. Cross-view time-frequency diffusion for extreme-aware time-series generation. *arXiv preprint arXiv:2610.17407*, 2026.

[60] Lin Jiang, Dahai Yu, Ravikumar Gelli, and Guang Wang. SynEnergy: Anomaly semantic-guided diffusion for synthetic energy data generation. *arXiv preprint arXiv:2608.03087*, 2026.

[61] Todd Huster, Jeremy Cohen, Zinan Lin, Kevin Chan, Charles Kamhoua, Nandi O. Leslie, Cho-Yu Jason Chiang, and Vyas Sekar. Pareto GAN: Extending the representational power of GANs to heavy-tailed distributions. In *International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 4523–4532, 2021.

[62] Laurens de Haan and Ana Ferreira. *Extreme Value Theory: An Introduction*. Springer, 2006.

[63] Boris N. Oreshkin, Dmitri Carpov, Nicolas Chapados, and Yoshua Bengio. N-BEATS: Neural basis expansion analysis for interpretable time series forecasting. In *International Conference on Learning Representations*, 2020.

[64] James W. Cooley and John W. Tukey. An algorithm for the machine calculation of complex Fourier series. *Mathematics of Computation*, 19(90):297–301, 1965.

[65] Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3836–3847, 2023.

[66] Custom Weather. Hourly weather data [dataset], 2022. Dewey Data. `https://doi.org/10.82551/YX6H-K352`.

[67] Custom Weather. Daily weather data [dataset], 2022. Dewey Data. `https://doi.org/10.82551/VBWQ-AQ20`.

[68] Franc Jager, Alessandro Taddei, George B. Moody, Michele Emdin, Gorazd Antolič, Roman Dorn, Aleš Smrdel, Carlo Marchesi, and Roger G. Mark. Long-term ST database: A reference for the development and evaluation of automated ischaemia detectors and for the study of the dynamics of myocardial ischaemia. *Medical & Biological Engineering & Computing*, 41(2):172–183, 2003.

[69] Ary L. Goldberger, Luis A. N. Amaral, Leon Glass, Jeffrey M. Hausdorff, Plamen Ch. Ivanov, Roger G. Mark, Joseph E. Mietus, George B. Moody, Chung-Kang Peng, and H. Eugene Stanley. PhysioBank, PhysioToolkit, and PhysioNet: Components of a new research resource for complex physiologic signals. *Circulation*, 101(23):e215–e220, 2000.

[70] Georges Hebrail and Alice Berard. Individual household electric power consumption [dataset], 2006. UCI Machine Learning Repository. `https://archive.ics.uci.edu/ml/datasets/individual+household+electric+power+consumption`.

[71] Marco Cuturi. PEMS-SF [dataset], 2011. UCI Machine Learning Repository. `https://archive.ics.uci.edu/ml/datasets/pems-sf`.

[72] California Department of Transportation. Performance measurement system (PeMS), 2026. `https://dot.ca.gov/programs/traffic-operations/mpr/pems-source`.

[73] Zhihan Yue, Yujing Wang, Juanyong Duan, Tianmeng Yang, Congrui Huang, Yunhai Tong, and Bixiong Xu. TS2Vec: Towards universal representation of time series. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 8980–8987, 2022.

A Reproducibility Statement

To improve reproducibility, we release a GitHub repository at https://github.com/LinJiang18/E4GEN as part of the supplementary materials. The repository includes all public datasets in the `Data` folder, the implementations of all baseline methods in the `Baseline` folder, the parameter configuration files for all experiments in the `Config` folder, and the source files for generating all figures in the `Analysis` folder. Together with the scripts for data preparation and model training, these resources enable independent researchers to reproduce all reported results and figures.

B Related Work

B.1 Time Series Generation

B.1.1 GAN-, VAE-, and Flow-based Time-Series Generation

Time-series generation aims to synthesize realistic sequential data that preserve both marginal distributions and temporal dependencies [43, 44]. Early studies adapted general-purpose generative paradigms to sequential settings, including generative adversarial networks (GANs), variational autoencoding (VAE), and likelihood-based flow modeling. Among them, GANs learn realistic sequences through a competition between the generator and discriminator. For instance, C-RNN-GAN [45] and RCGAN [46] pioneer the combination of recurrent neural networks with adversarial training to generate continuous temporal data and condition on auxiliary information, respectively. TimeGAN [1] further introduces a supervised temporal objective in a learned embedding space, encouraging generated sequences to follow the stepwise dynamics of real data. In contrast, VAE-based methods provide a more stable probabilistic framework by learning compact latent representations for sequence reconstruction and sampling. Early approaches like VRNN [47] integrate latent random variables into RNN hidden states to model structured sequential variability. TimeVAE [11] extends VAEs to multivariate time series with sequence-aware components such as level, trend, and residual structures, while koVAE [12] replaces the conventional static latent prior with a Koopman-inspired dynamical prior to capture latent temporal evolution. Flow-based models provide another probabilistic approach to time-series generation through explicit likelihood estimation. Fourier Flows [13] transform time series into spectral representations and learn an invertible flow over Fourier-domain coefficients, enabling the model to capture global temporal and periodic patterns. While these methods provide strong general-purpose generators, their objectives are usually dominated by common temporal patterns and are not explicitly designed to preserve sparse extreme patterns.

B.1.2 Diffusion-based Time-series Generation

Diffusion models have recently become a strong paradigm for sequential generation because they synthesize samples through iterative denoising, allowing complex temporal distributions to be refined progressively. Their general formulation, established by DDPMs and subsequent improvements, provides a flexible probabilistic framework for transforming noise into structured data distributions [34, 28]. In one-dimensional sequential domains, DiffWave [15] demonstrates that diffusion backbones can synthesize high-fidelity audio waveforms from white noise, suggesting their suitability for continuous temporal signals.

Building on this denoising perspective, recent studies have adapted diffusion models to time-series tasks under different settings. TimeGrad [48] combines autoregressive networks with score-based diffusion for multivariate probabilistic forecasting, while CSDI [49] extends score-based diffusion to probabilistic time-series imputation through self-supervised masking and conditional score matching. Other works further explore controlled or structured diffusion for time-series generation. For example, constrained time-series generation [16] incorporates user-specified temporal constraints into the sampling process, while TimeBridge [29] improves diffusion prior design through bridge-based modeling. More recently, Diffusion-TS [4] combines a Transformer denoiser with trend–seasonality decomposition and directly reconstructs the clean sample during diffusion, improving both fidelity and interpretability. Diffusion models have also been extended to spatiotemporal sequence generation, including human activity traces and check-in trajectories [50, 51].

Overall, these methods establish diffusion models as effective backbones for time-series generation. However, their conditioning mechanisms are primarily designed for sample-level or value-level

guidance rather than event-level control, making it difficult to explicitly preserve rare and localized extreme-event structures.

B.1.3 LLM-based Time-series Generation

Recent studies have also explored whether the sequence modeling capacity of large language models and foundation models can be transferred to time-series forecasting and generation. Since raw time series are continuous, these methods usually convert numerical sequences into textual tokens, discrete value tokens, or patch-level embeddings before applying autoregressive modeling. Early prompt-based studies such as PromptCast [52] formulate forecasting as a text-to-text problem, while LLMTime [53] serializes numerical values as text and samples future continuations from pretrained LLMs in a zero-shot manner. Another line of work reuses pretrained language backbones through representation alignment. GPT4TS [54] segments time series into patches and projects them into the input space of a pretrained GPT-style model, enabling the language backbone to process temporal sequences with lightweight adaptation. Time-LLM [55] further introduces a reprogramming strategy that maps time-series patch representations into text-prototype-aligned token embeddings, together with prompt-based temporal context for frozen LLM adaptation. In contrast, Chronos [56] trains language-model architectures directly on discretized time-series values, turning forecasting into next-token prediction over value bins. Recent analyses also suggest that LLM-based temporal models can benefit from pretraining and sequence modeling, but their gains depend heavily on representation design, prompting, and evaluation settings [14]. However, most of these works focus on general forecasting problems instead of generation tasks, which are more challenging, especially when considering the extreme events.

For time-series generation, SDForger [5] follows a more generation-oriented paradigm by mapping time series into functional or tabular embeddings, serializing them into text, fine-tuning an autoregressive language model, and decoding the generated embeddings back into time-series samples. This representation-space generation avoids direct token-by-token synthesis over long continuous sequences while leveraging the generative capacity of LLMs. Nevertheless, compact tokens or embeddings are usually optimized to preserve global sequence information, so sparse localized extremes may still be compressed or underemphasized.

B.2 Extreme-aware Generation

Extreme-aware generation aims to improve the generation fidelity in rare, high-impact parts that are often underrepresented by average-case objectives. Existing approaches usually address this issue through long-tail sample modeling, extreme-value theory, or heavy-tailed generation dynamics. Long-tailed diffusion methods improve tail coverage by calibrating or transferring knowledge from frequent samples to rare ones [19], while heavy-tailed generative models modify distributional assumptions to better represent large-magnitude outcomes [22, 23, 57]. These designs improve tail fidelity from sample-level or value-level perspectives. However, for time-series data, extremes are not only distributional tails but also temporally organized patterns, which makes their preservation different from conventional long-tail generation.

Several studies have incorporated extreme-value modeling into time-series generators. Early explorations like ExGAN [58] combine Extreme Value Theory (EVT) with adversarial learning to generate realistic samples under user-specified tail probabilities. FIDE [3] enhances high-frequency components and conditions the generation process on block maxima sampled from a fitted GEV distribution. HeavyDiff [23] takes a complementary direction by replacing Gaussian perturbations with Student-t-based heavy-tailed noise, allowing diffusion models to better capture rare large-magnitude outcomes. More recently, CrossTF [59] exploits discrepancies between time- and frequency-view denoisers to improve extreme fidelity through late-stage cross-view fusion. SynEnergy [60] learns region-specific anomaly semantics to guide diffusion-based energy data generation and preserve anomalous consumption patterns.

These efforts demonstrate different strategies for improving extreme fidelity through tail modeling, frequency-aware generation, cross-view optimization, and anomaly-guided diffusion. However, they primarily emphasize distributional tails or extreme-pattern preservation, leaving event-level properties such as location, duration, and temporal shape less directly modeled.

C Discussion

Although E4GEN shows consistent improvements across diverse time-series datasets, several limitations remain. First, our experiments focus on six representative datasets from climate, healthcare, energy, and transportation domains. While these datasets cover different types of extreme events, broader validation on additional domains and longer time-series settings would further strengthen the generality of the conclusions. Second, the definition of extreme events relies on domain-specific thresholds or task-specific criteria. This design provides flexibility, but the generated extreme-event semantics may depend on the quality of the chosen threshold. Third, E4GEN introduces additional components for semantic prediction and control injection, which may increase training and sampling cost compared with a plain diffusion backbone. Future work can explore more efficient control modules and adaptive threshold selection strategies.

D Broader Societal Impacts

Our E4GEN advances the generation of realistic time-series data with a focus on capturing extreme events, offering positive societal impacts in domains like climate disaster preparedness, healthcare (e.g., synthesizing rare ECG anomalies without compromising patient privacy), and energy grid stress-testing. However, we also acknowledge potential negative impacts, including the malicious synthesis of fake sensor data to spoof systems or trigger false alarms, and the risk of over-reliance on synthetic extremes that may not perfectly reflect true underlying physical mechanisms, potentially leading to miscalibrated decisions in safety-critical scenarios. To mitigate these risks, E4GEN-generated data should complement rather than replace real-world observations, and future deployments could incorporate statistical watermarking to prevent the misuse of synthetic datasets.

E Data-Driven Analysis

E.1 Discussion About Value-level Extreme-aware Generation

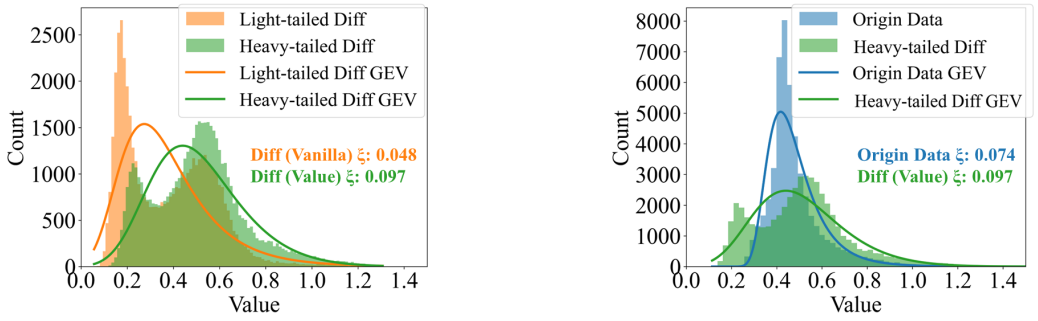


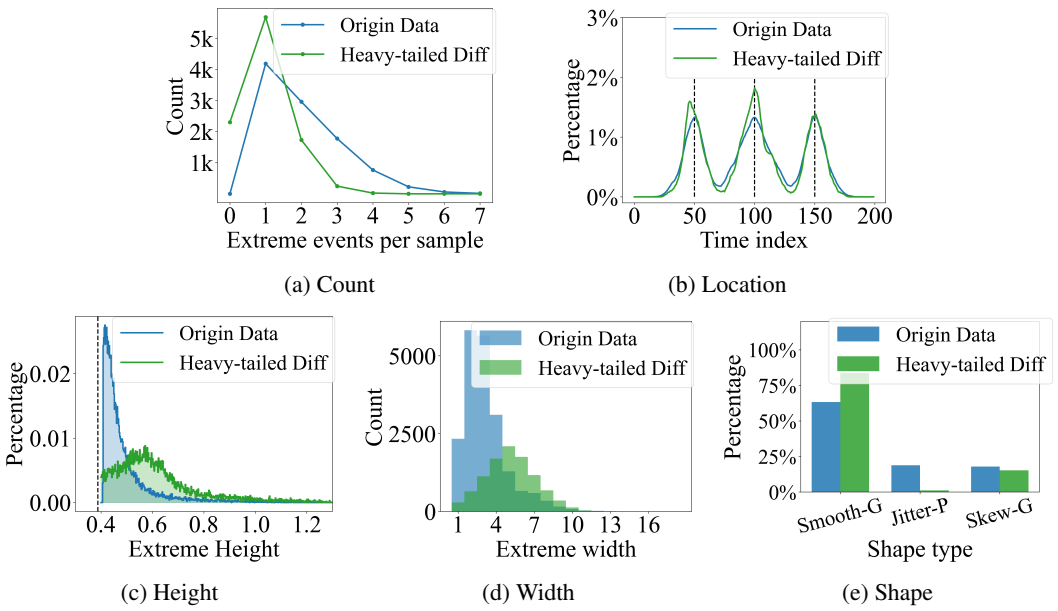
Fig. 9: GEV Distribution of Block Maxima

Value-level extreme-aware generation is a common line of research in existing studies on extreme enhancement [3, 61, 21, 22, 23], and it is often framed as **heavy-tailed generation**. Its core idea is to directly strengthen tail behavior at the value-distribution level, e.g., by replacing light-tailed Gaussian noise with heavy-tailed alternatives such as Student-t noise [23].

We investigate the advantages and limitations of value-level extreme-aware generation from the perspective of extreme value theory [62]. We use the classical block maxima framework as an example and analyze its corresponding generalized extreme value (GEV) distribution, formulated as:

$$G(z;\mu,\sigma,\xi) = \exp\left\{-\left[1+\xi\left(\frac{z-\mu}{\sigma}\right)\right]^{-1/\xi}\right\}, \tag{12}$$

where $\mu \in \mathbb{R}$ is the location parameter, $\sigma > 0$ is the scale parameter, and $\xi \in \mathbb{R}$ is the shape parameter. In general, a larger ξ indicates a heavier tail. Similar to Observation I in Section 2, we visualize two representative methods, a standard light-tailed Diffusion model implemented with Diffusion-TS and a value-level extreme-enhanced Heavy-tailed Diffusion model implemented with HeavyDiff. Figure 9



(a) Count (b) Location

(c) Height (d) Width (e) Shape

Fig. 10: Evaluation of extreme-event fidelity across five dimensions.

presents the overall distribution as the shaded area and the fitted GEV distribution as curves. The left panel compares the generated data of Heavy-tailed Diffusion with Light-tailed Diffusion, while the right panel compares the results of Heavy-tailed Diffusion with the original data. In terms of the shape parameter ξ, the generated data of Heavy-tailed Diffusion shows a heavier tail than the original data, whereas the generated data of Light-tailed Diffusion shows a lighter tail. This result indicates that value-level extreme-aware generation can strengthen tail heaviness at the statistical level. However, this raises a further question: does a heavier tail necessarily lead to higher extreme-event fidelity?

To answer this question, we design a toy example containing extreme events and evaluate it from five dimensions, including count, location, height, width, and shape. These results are shown in Figure 10. Heavy-tail methods exhibit aggregation effects across all five dimensions, namely a Matthew effect, which increases the proportion of dominant groups while weakening minor ones. This phenomenon is especially clear in count and shape. For example, in Fig. 10(e) for extreme event shape, the heavy-tailed method generates more Smooth-G events, increasing their proportion from the original 60% to 80%, but produces fewer events for the minor types, Jitter-P and Skew-G.

Therefore, the **advantage** of value-level extreme-aware generation is its ability to increase tail heaviness and produce more extreme values. However, a heavier tail does not necessarily imply higher extreme-event fidelity. Its key **limitation** is that it enhances generation mainly at the value-distribution level, without modeling the semantic structure of extreme events. As a result, it may amplify dominant patterns while weakening minor ones, thereby limiting its improvement in extreme-event generation fidelity.

E.2 Additional Analysis of Coarse-to-Fine Generation Dynamics

In Observation II of Section 2, we discuss the coarse-to-fine generation process exhibited by diffusion models in time-series generation. The model first focuses on the trend and dominant seasonality during the early sampling stage, and then gradually reconstructs fine-grained details, including extreme events, in the later stage. While Fig. 3 illustrates this process with one sample from a 500-step diffusion model, Fig. 11 provides a broader visualization using 10 randomly selected samples across different sampling steps.

E.3 Intrinsic Patterns in Extreme Events

In Observation III of Section 2, we argue that extreme events are shaped by underlying physical mechanisms rather than random noise. Therefore, understanding their intrinsic patterns is essential. To

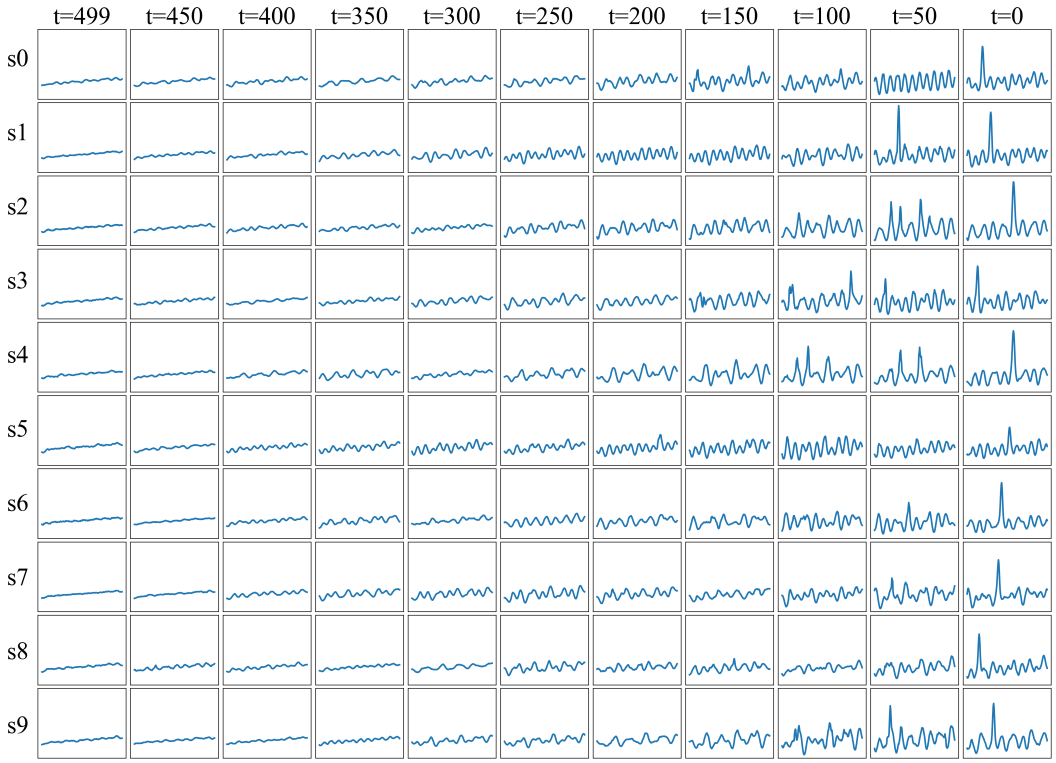
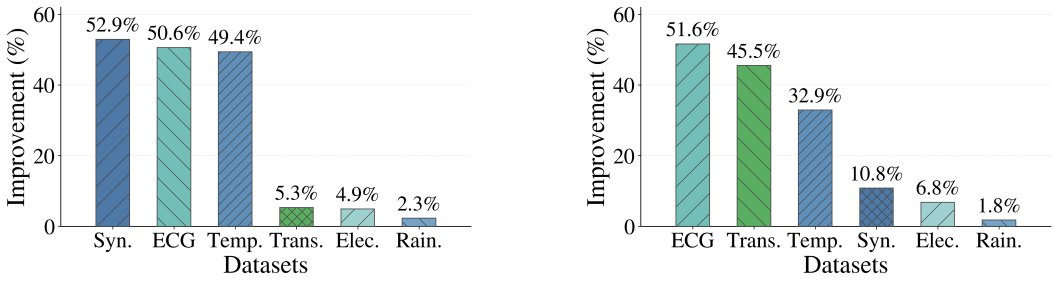


Fig. 11: Coarse-to-fine generation trajectories of 10 randomly selected samples from a 500-step diffusion model.

support this argument, we perform empirical analyses on real-world datasets from three perspectives: predictability, context dependency, and cross-sample stability.



(a) Predictability

(b) Context dependency

Fig. 12: Intrinsic patterns in extreme events.

Fig. 12 visualizes the results for predictability and context dependency across six datasets. For predictability, we examine whether extreme events can be inferred from their preceding history by comparing forecasting performance using the original history with that using shuffled history. As shown in Fig. 12(a), all six datasets show positive improvements, indicating that the temporal patterns before extreme events provide useful information for predicting their future evolution. The gains are particularly clear on the Synthetic, ECG, and Temperature datasets, reaching 52.9%, 50.6%, and 49.4%, respectively.

For context dependency, we test whether an extreme event is better aligned with its original surrounding context than with shuffled contexts from other samples. As shown in Fig. 12(b), all six datasets again show positive improvements, with especially strong gains on ECG, Transportation, and Temperature. This consistent trend indicates that extreme events are closely tied to their temporal

context. Overall, these two analyses show that extreme events exhibit both temporal predictability and contextual compatibility, providing further evidence that they are structured and non-random.

For cross-sample stability, we refer readers to Fig. 4 and the detailed discussion in Section 2.

F Additional Details on Backbone Denoiser Design

F.1 One-step Denoising Transition in the Backbone Denoiser

Given an intermediate noisy state $\mathbf{x}(t)$, the Backbone Denoiser first predicts the corresponding clean-sample estimate $\hat{\mathbf{x}}_0(t)$. This estimate parameterizes the reverse transition from diffusion step t to $t-1$, which constitutes the basic sampling operation that progressively transforms noisy states into clean time series. Under the $\mathbf{x}$-pred diffusion formulation, the one-step denoising transition is defined as:

$$p_\theta(\mathbf{x}(\mathbf{t}-\mathbf{1}) \mid \mathbf{x}(\mathbf{t})) = \frac{1}{1-\bar{\alpha}_t}\Big(\sqrt{\bar{\alpha}_{t-1}}\,\beta_t\,\hat{x}_0(t) + \sqrt{\alpha_t}\,(1-\bar{\alpha}_{t-1})\,\mathbf{x}(\mathbf{t})\Big) + \tilde{\sigma}_t\,\mathbf{z}_t, \tag{13}$$

where $\bar{\alpha}_t = \prod_{s=1}^{t}\alpha_s$, $\alpha_t = 1-\beta_t$, and $\mathbf{z}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. Here, β_t denotes the noise schedule, and $\tilde{\sigma}_t$ controls the stochasticity of the reverse update. In this way, the Backbone Denoiser provides the clean-sample estimate that guides each denoising step, while the stochastic term preserves the sampling variability of the diffusion process.

F.2 Trend-Seasonality Design in the Backbone Denoiser

This section provides additional details on the trend-seasonality design introduced in Section 3.2. This design parameterizes the $\mathbf{x}$-prediction network $\hat{\mathbf{x}}_0^\theta(\cdot)$ in the Backbone Denoiser, allowing it to model the global temporal backbone of the sequence through trend and seasonality components.

Trend Function. The trend function is designed to capture slow, long-term variations through low-frequency information. We avoid directly using the Fourier transform for trend modeling, since long-range trends may be incompletely or inaccurately represented by periodic bases. Instead, inspired by [11, 63], we adopt a polynomial regressor and define the trend component as:

$$\mathrm{trend}(\mathbf{x}(t), \theta_{tr}) = \phi(\tau)^\top\big(W_{tr}\,h(\mathbf{x}(t); a_{tr}) + b_{tr}\big), \tag{14}$$

where $\mathbf{x}(t)$ denotes the noisy time-series state at diffusion step t, τ is the time index, and $\phi(\tau) = [1,\, \tau,\, \tau^2, \ldots, \tau^p]^\top$ is a polynomial basis of degree p. The function $h(\mathbf{x}(t); a_{tr})$ is a neural feature extractor parameterized by a_{tr}, W_{tr} is a learnable projection matrix that maps the extracted features to polynomial coefficients, and b_{tr} is a bias vector. Together, W_{tr}, a_{tr}, and b_{tr} form the trend parameters θ_{tr}. In this formulation, the network term $W_{tr}\,h(\mathbf{x}(t); a_{tr}) + b_{tr}$ produces the polynomial coefficients, while the basis vector $\phi(\tau)$ encodes the temporal structure. Their inner product gives the trend value at time index τ. Since the trend mainly captures low-frequency information, p is typically set to a small value, such as 3.

Seasonality Function. The seasonality function captures dominant periodic patterns in the frequency domain. Specifically, we apply the discrete Fourier transform (DFT) to $\mathbf{x}(t)$, select the Top-K frequency components with the largest magnitudes, and reconstruct the seasonal component through the inverse discrete Fourier transform (IDFT). The formulation is given by:

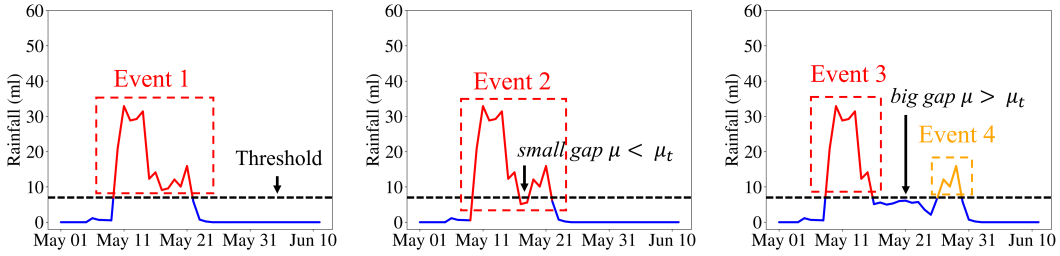
$$\mathrm{seasonality}\,(\mathbf{x}(t), \theta_{se}) = \mathcal{IDFT}\Big(\mathcal{M}_K\big(\mathcal{DFT}(\mathbf{x}(t)), \theta_{se}\big)\Big), \tag{15}$$

where $\mathcal{DFT}$ and $\mathcal{IDFT}$ denote the discrete Fourier transform and its inverse, respectively. $\mathcal{M}_K(\cdot, \theta_{se})$ is a masking and reweighting operator that preserves the Top-K frequency components with the largest magnitudes, together with their conjugate-symmetric counterparts. Here, θ_{se} denotes the learnable parameters of a Transformer network applied to the selected frequency components, which models their interactions and refines their contributions to better reconstruct the seasonal component.

G Additional Details for Extreme-Event Semantic Representation

G.1 Handling Threshold Fluctuations in Extreme-Event Definition

We define an extreme event as a localized temporal segment in which consecutive observations exceed a predefined upper threshold (or fall below a predefined lower threshold). We emphasize that an extreme event corresponds to a continuous time-series interval rather than an isolated extreme value. Here, we further discuss a common case: *when extreme values fluctuate over a period of time and briefly fall below the threshold, how should our definition handle such cases?*



(a) Fully Above-threshold Event (b) short-gap Interrupted Event (c) Long-gap Separated Event

Fig. 13: Illustration of Threshold Fluctuations in Extreme-event Definition.

As shown in Figure 13, we introduce three cases. For a fully above-threshold segment, the entire interval is treated as one extreme event. When a short below-threshold gap appears within an extreme period, we merge the adjacent above-threshold intervals if the gap length is smaller than a tolerance μ_t, thereby preserving the continuity of the same event. In contrast, when the below-threshold gap is larger than μ_t, the intervals are treated as separate events. This rule prevents over-fragmentation caused by short-term fluctuations while separating truly separate extreme events.

H Additional Details for E-Activator

In Section 4.1, we introduce two metrics for E-Activator: Backbone Drift (BD) and Outlier Emergence Score (OES), defined as follows. Both are evaluated on $\hat{\mathbf{x}}_0(t)$, the predicted final sample at step t.

$$\mathrm{BD}(t) = \mathbb{E}\left[\frac{1}{k}\sum_{j=0}^{k-1}\frac{\sqrt{\frac{1}{LC}\sum_{\tau=1}^{L}\sum_{\nu=1}^{C}\left(bb_{t-j,\tau,\nu}-bb_{t-j-1,\tau,\nu}\right)^2}}{\sqrt{\frac{1}{LC}\sum_{\tau=1}^{L}\sum_{\nu=1}^{C}bb_{t-j-1,\tau,\nu}^2}}\right] \tag{16}$$

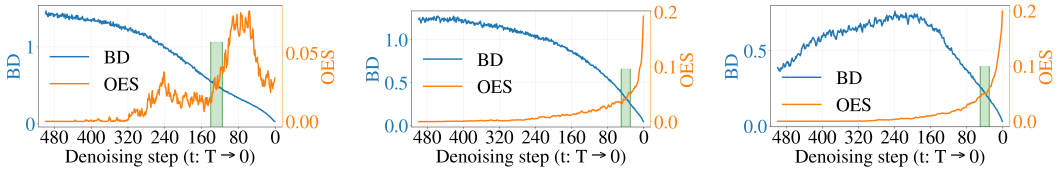
$$\mathrm{OES}(t) = \mathbb{E}\left[\max\left(\left|\frac{\hat{x}_{0,\tau,\nu}(t)-\mathrm{med}_\tau(\hat{x}_{0,\tau,\nu}(t))}{1.4826\,\mathrm{mad}_\tau(\hat{x}_{0,\tau,\nu}(t))+\varepsilon}\right|-\kappa,\,0\right)\right] \tag{17}$$

where $\mathbb{E}[\cdot]$ denotes empirical averaging. For BD, the expectation is taken over samples in a batch. For OES, the expectation is taken over batch samples, time indices, and feature dimensions. L and C denote the sequence length and the number of feature dimensions, respectively. The parameter k is the temporal smoothing window used to average backbone changes over the previous k sampling steps. The denominator in BD normalizes the backbone change by the energy of the previous low-frequency backbone, making the metric comparable across samples and diffusion steps.

Here, $bb_{t,\tau,\nu}$ denotes the low-frequency backbone of $\hat{\mathbf{x}}_0(t)$ at time index τ and feature dimension ν. It is obtained by first removing the per-series mean from $\hat{\mathbf{x}}_0(t)$, then applying rFFT along the temporal axis, retaining the first M non-DC frequency bins, and finally applying inverse rFFT to reconstruct the low-frequency component [64]. This low-pass component approximates the global trend and seasonality structure of the predicted sample.

The term $\hat{x}_{0,\tau,\nu}(t)$ denotes the (τ,ν)-th entry of $\hat{\mathbf{x}}_0(t)$, where $\tau \in [L]$ is the time index and $\nu \in [C]$ is the feature index. For OES, $\mathrm{med}_\tau(\cdot)$ and $\mathrm{mad}_\tau(\cdot)$ denote the median and median absolute deviation computed along the temporal axis for each sample and feature dimension. Following the standard robust-statistics convention, the constant 1.4826 rescales mad_τ to be comparable to the standard deviation under a Gaussian distribution. κ is the robust outlier threshold, which controls when a normalized deviation is counted as an emerging extreme structure, and ε is a small constant used for numerical stability.

We further visualize the evolution of BD and OES on several additional datasets in Fig. 14. In most cases, BD exhibits stable dynamics and gradually decreases in the later sampling stages. By contrast, OES may fluctuate on highly variable datasets, such as rainfall and temperature. For such datasets, a wider search region $\mathcal{R}_{\mathrm{CA}}$ may be needed, followed by more fine-grained tuning within the selected range.



(a) ECG Dataset (b) Electricity Dataset (c) Transportation Dataset

Fig. 14: Control Activation Window via BD and OES in More Datasets

I Additional Details for Semantic Predictor f_ϕ

In this section, we provide the detailed design of the Semantic Predictor f_ϕ in E-Predictor introduced in Section 4.2. Given the intermediate denoising state $\mathbf{x}_n(t_{\mathrm{CA}})$, f_ϕ predicts the extreme-event semantics $\hat{E}_{\mathrm{CA}}$ as the semantic-based control signal. It consists of three modules: *Transformer-based token encoding*, *multi-head semantic prediction*, and *event-level semantic decoding*.

Transformer-based Token Encoding. The first module encodes the intermediate denoising state $\mathbf{x}_n(t_{\mathrm{CA}})$ into contextualized token representations. Specifically, before token encoding, we feed $\mathbf{x}_n(t_{\mathrm{CA}})$ into the x-prediction function in Eq. 1 to predict the final clean sample $\hat{\mathbf{x}}_0(t_{\mathrm{CA}}) = \hat{\mathbf{x}}_0^\theta\big(\mathbf{x}_n(t_{\mathrm{CA}})\big)$, which is then used as the input to the Semantic Predictor. Each time step in $\hat{\mathbf{x}}_0(t_{\mathrm{CA}})$ is treated as a temporal token and projected into a 128-dimensional hidden embedding. A learnable positional embedding is added to each token to preserve temporal order. The token sequence is then fed into a Transformer encoder composed of four encoder layers. Each layer contains a multi-head self-attention module with four attention heads, followed by a position-wise feed-forward network whose hidden width is four times the token dimension. GELU activation and dropout with rate 0.1 are used in the feed-forward blocks. Through self-attention, the encoder allows each time step to aggregate information from the entire sequence, enabling the predictor to capture both local event-related variations and long-range temporal dependencies. The resulting contextualized token representations serve as the shared feature basis for subsequent semantic prediction.

Multi-head Semantic Prediction. Based on the contextualized token representations, the second module predicts token-wise semantic cues through three lightweight heads. Specifically, given the encoded token sequence, two classification heads estimate an event-region confidence map $\hat{\mathbf{c}}^{\mathrm{ev}} \in [0,1]^L$ and a keypoint confidence map $\hat{\mathbf{c}}^{\mathrm{kp}} \in [0,1]^L$, respectively, while one regression head predicts an auxiliary value sequence $\hat{\mathbf{v}} \in \mathbb{R}^L$. The event-region confidence map assigns each time step a probability of belonging to an extreme-event interval, which provides the basis for decoding the event count and event spans in the location semantic $\mathcal{L}$. The keypoint confidence map scores structurally important positions within potential event regions, such as endpoints, extrema, and turning points, which supports the construction of the temporal shape semantic $\mathcal{S}$. The auxiliary value sequence estimates the clean signal value at each time step, and is used to obtain the amplitudes of decoded keypoints as well as intensity-related quantities in $\mathcal{I}$. In this way, the multi-head design decomposes extreme-event semantic prediction into complementary token-wise cues, which can be reliably decoded into event-level semantics instead of directly regressing the complete event representation.

Event-level Semantic Decoding. The third module converts the token-wise predictions into event-level semantics $\hat{E}_{\mathrm{CA}} = \{\mathcal{L}, \mathcal{I}, \mathcal{S}\}$. We decode these semantics in three steps. First, we compute the location semantic $\mathcal{L}$, including the event count and event spans, from the event-region confidence map $\hat{\mathbf{c}}^{\mathrm{ev}}$. The confidence threshold is selected on the validation set and then fixed during decoding. Time steps whose confidence scores exceed this threshold are regarded as extreme-event positions. Consecutive selected positions are grouped into event intervals, while nearby intervals and very short segments are post-processed to reduce noisy detections. Second, we compute the intensity semantic $\mathcal{I}$, including peak-related quantities such as event peak, prominence, and energy, from the auxiliary

value sequence $\hat{\mathbf{v}}$. For each decoded event interval, we read the values of $\hat{\mathbf{v}}$ within this interval. Since $\hat{\mathbf{v}}$ estimates the signal value at each time step, the values inside the decoded interval indicate the event strength and are used to compute the intensity attributes. Third, we compute the temporal shape semantic $\mathcal{S}$, represented by a set of event-level keypoints, using the keypoint confidence map $\hat{\mathbf{c}}^{\mathrm{kp}}$ and the auxiliary value sequence $\hat{\mathbf{v}}$. Within each decoded event interval, we select representative keypoints, including the event endpoints, extrema, and high-confidence structural points. The selected time indices and their corresponding values from $\hat{\mathbf{v}}$ form the keypoint-based shape representation. This validation-calibrated decoding process converts the token-wise outputs into the event-level semantics $\hat{E}_{\mathrm{CA}}$.

J Theoretical Proof for DCT-NIS

Lemma 1 (Semantic Consistency)**.** *Let $\mathbf{x}_d(t_0) \sim p_{\mathrm{data}}$ and let $\mathbf{x}_d(t_{\mathrm{AS}}) \sim q(\mathbf{x}_d(t_{\mathrm{AS}}) \mid \mathbf{x}_d(t_0))$, where q denotes the forward diffusion process. Suppose that the denoising trajectory satisfies bounded reconstruction drift and that the event-level semantics is locally stable. If the reverse span from the alignment start step t_{AS} to the control activation step t_{CA} is sufficiently short, then the anchored semantics $EES(\mathbf{x}_d(t_0))$ remains a valid supervision signal for the predictor at $\mathbf{x}_d(t_{\mathrm{CA}})$.*

Proof of Lemma 1. Let $\hat{\mathbf{x}}_0^\theta(\mathbf{x}_d(t))$ denote the clean-sample estimate predicted from the intermediate state $\mathbf{x}_d(t)$. The goal is to show that the clean estimate associated with $\mathbf{x}_d(t_{\mathrm{CA}})$ remains close to the anchored clean sample $\mathbf{x}_d(t_0)$ when the reverse span from t_{AS} to t_{CA} is short.

We first characterize the reconstruction error at the alignment start step. Since $\mathbf{x}_d(t_{\mathrm{AS}})$ is obtained by forward noising the anchored clean sample $\mathbf{x}_d(t_0)$, the x-prediction model provides a clean estimate $\hat{\mathbf{x}}_0^\theta(\mathbf{x}_d(t_{\mathrm{AS}}))$. Let ϵ_{AS} denote an upper bound on the expected reconstruction error at t_{AS}:

$$\mathbb{E}\Big[\big\|\hat{\mathbf{x}}_0^\theta(\mathbf{x}_d(t_{\mathrm{AS}})) - \mathbf{x}_d(t_0)\big\|\Big] \le \epsilon_{\mathrm{AS}}. \tag{18}$$

Here, ϵ_{AS} measures the reconstruction error at the alignment start step. For a well-trained x-prediction diffusion model, this error remains bounded when t_{AS} is sufficiently close to the clean data endpoint.

Next, we characterize the drift caused by the short learned denoising segment from t_{AS} to t_{CA}. For each $t \in \{t_{\mathrm{CA}}+1, \dots, t_{\mathrm{AS}}\}$, let η_t denote an upper bound on the expected one-step change of the clean estimate in the reconstruction space:

$$\mathbb{E}\Big[\big\|\hat{\mathbf{x}}_0^\theta(\mathbf{x}_d(t-1)) - \hat{\mathbf{x}}_0^\theta(\mathbf{x}_d(t))\big\|\Big] \le \eta_t. \tag{19}$$

Here, η_t measures the reconstruction drift introduced by one denoising update from step t to step $t-1$. We use this quantity to characterize the local stability of the denoising trajectory in the reconstruction space; smaller η_t indicates that one denoising update causes only a mild change in the predicted clean-sample estimate.

By the triangle inequality, the cumulative reconstruction drift along the reverse span is bounded by

$$\mathbb{E}\Big[\big\|\hat{\mathbf{x}}_0^\theta(\mathbf{x}_d(t_{\mathrm{CA}})) - \hat{\mathbf{x}}_0^\theta(\mathbf{x}_d(t_{\mathrm{AS}}))\big\|\Big] \le \sum_{t=t_{\mathrm{CA}}+1}^{t_{\mathrm{AS}}} \eta_t. \tag{20}$$

Combining Eqs. (18) and (20), we obtain

$$\mathbb{E}\Big[\big\|\hat{\mathbf{x}}_0^\theta(\mathbf{x}_d(t_{\mathrm{CA}})) - \mathbf{x}_d(t_0)\big\|\Big] \le \epsilon_{\mathrm{AS}} + \sum_{t=t_{\mathrm{CA}}+1}^{t_{\mathrm{AS}}} \eta_t. \tag{21}$$

This bound shows that the clean estimate at t_{CA} remains close to the anchored clean sample when the reconstruction error at t_{AS} is controlled and the reverse span is short.

We then connect reconstruction closeness to semantic consistency. Although event-level semantics are not globally invariant to arbitrary perturbations, they are locally stable around clean samples whose extreme-event regions have sufficient margins from the decision threshold. We formalize this by assuming that there exists a constant $L_E > 0$ such that, for samples within a local neighborhood of $\mathbf{x}_d(t_0)$,

$$d_E(EES(\mathbf{x}), EES(\mathbf{x}')) \le L_E \|\mathbf{x} - \mathbf{x}'\|, \tag{22}$$

where $d_E(\cdot,\cdot)$ denotes a suitable discrepancy between event-level semantic representations. This condition means that small reconstruction perturbations around a stable clean sample only induce bounded changes in the extracted event-level semantics.

Applying Eq. (22) to $\hat{\mathbf{x}}_0^\theta(\mathbf{x}_d(t_{\mathrm{CA}}))$ and $\mathbf{x}_d(t_0)$ gives

$$\mathbb{E}\Big[d_E\big(EES\big(\hat{\mathbf{x}}_0^\theta(\mathbf{x}_d(t_{\mathrm{CA}}))\big), EES(\mathbf{x}_d(t_0))\big)\Big] \le L_E\left(\epsilon_{\mathrm{AS}} + \sum_{t=t_{\mathrm{CA}}+1}^{t_{\mathrm{AS}}} \eta_t\right). \tag{23}$$

Therefore, when t_{AS} is chosen close to t_{CA}, the cumulative drift $\sum_{t=t_{\mathrm{CA}}+1}^{t_{\mathrm{AS}}} \eta_t$ is small. As established above, ϵ_{AS} remains bounded. Therefore, the semantic discrepancy between the clean estimate at t_{CA} and the anchored clean sample remains bounded. Hence, $EES(\mathbf{x}_d(t_0))$ remains a valid supervision signal for training the semantic predictor at $\mathbf{x}_d(t_{\mathrm{CA}})$ in a bounded-error sense. □

Lemma 2 (Marginal Alignment)**.** *The marginal distribution of the data-conditioned state $\mathbf{x}_d(t_{\mathrm{CA}})$ is close to that of the noise-initiated state $\mathbf{x}_n(t_{\mathrm{CA}})$.*

Proof of Lemma 2. Let $P_d^{\mathrm{CA}} = \mathcal{L}(\mathbf{x}_d(t_{\mathrm{CA}}))$ denote the marginal distribution of the data-conditioned state, and let $P_n^{\mathrm{CA}} = \mathcal{L}(\mathbf{x}_n(t_{\mathrm{CA}}))$ denote the marginal distribution of the noise-initiated state at t_{CA}. Let q_t denote the forward-diffused marginal at step t induced by p_{data}:

$$q_t(\mathbf{x}) \triangleq \int q(\mathbf{x}(t) \mid \mathbf{x}(t_0)) p_{\mathrm{data}}(\mathbf{x}(t_0))\, d\mathbf{x}(t_0). \tag{24}$$

Thus, $q_{t_{\mathrm{CA}}}$ is the reference marginal at the control activation step under the ideal diffusion dynamics.

We compare P_d^{CA} and P_n^{CA} through this common reference marginal. By the triangle inequality of total variation distance,

$$\mathrm{TV}\big(P_d^{\mathrm{CA}}, P_n^{\mathrm{CA}}\big) \le \mathrm{TV}\big(P_d^{\mathrm{CA}}, q_{t_{\mathrm{CA}}}\big) + \mathrm{TV}\big(P_n^{\mathrm{CA}}, q_{t_{\mathrm{CA}}}\big). \tag{25}$$

We first consider the noise-initiated state. During noise-initiated sampling, the process starts from the prior distribution and applies the learned denoising transitions until step t_{CA}. Under the ideal reverse diffusion dynamics, the induced marginal at t_{CA} is exactly the forward reference marginal $q_{t_{\mathrm{CA}}}$. For the learned denoising process, we quantify its deviation from this ideal reference marginal by

$$\varepsilon_{\mathrm{nis}} \triangleq \mathrm{TV}\big(P_n^{\mathrm{CA}}, q_{t_{\mathrm{CA}}}\big). \tag{26}$$

This quantity measures the residual marginal mismatch of the practical noise-initiated sampling distribution at the control activation step. Under the bounded reverse-approximation error assumption for the Backbone Denoiser, $\varepsilon_{\mathrm{nis}}$ remains bounded.

We next consider the data-conditioned state. In data-conditioned training, we sample $\mathbf{x}_d(t_0) \sim p_{\mathrm{data}}$ and forward diffuse it to $\mathbf{x}_d(t_{\mathrm{AS}}) \sim q(\mathbf{x}(t_{\mathrm{AS}}) \mid \mathbf{x}(t_0))$. Therefore,

$$\mathcal{L}(\mathbf{x}_d(t_{\mathrm{AS}})) = q_{t_{\mathrm{AS}}}. \tag{27}$$

Starting from this forward marginal, the data-conditioned process applies the same learned denoising transitions over the span from t_{AS} to t_{CA}. Under the ideal reverse diffusion dynamics, a chain initialized from $q_{t_{\mathrm{AS}}}$ has marginal $q_{t_{\mathrm{CA}}}$ at step t_{CA}. For the learned denoising process, we quantify its deviation from this ideal reference marginal by

$$\varepsilon_{\mathrm{dct}} \triangleq \mathrm{TV}\big(P_d^{\mathrm{CA}}, q_{t_{\mathrm{CA}}}\big). \tag{28}$$

This quantity measures the residual marginal mismatch accumulated along the data-conditioned denoising segment. This mismatch is controlled by the approximation accuracy of the learned denoising transitions and by the length of the reverse span. In particular, following Lemma 1, the denoising trajectory over the span from t_{AS} to t_{CA} has bounded reconstruction drift, and this span is chosen to be sufficiently short. Therefore, only a limited number of approximate denoising updates contribute to the accumulated marginal mismatch, making $\varepsilon_{\mathrm{dct}}$ bounded and to be small.

Substituting Eqs. (26) and (28) into Eq. (25), we obtain

$$\mathrm{TV}\big(P_d^{\mathrm{CA}}, P_n^{\mathrm{CA}}\big) \le \varepsilon_{\mathrm{dct}} + \varepsilon_{\mathrm{nis}}. \tag{29}$$

Therefore, when the residual marginal mismatches $\varepsilon_{\mathrm{dct}}$ and $\varepsilon_{\mathrm{nis}}$ are small, the marginal distribution of $\mathbf{x}_d(t_{\mathrm{CA}})$ is close to that of $\mathbf{x}_n(t_{\mathrm{CA}})$. This completes the proof. □

K Additional Details for E-Control

This appendix details the implementation of E-Control in Section 4.3, which injects the predicted control signal $\hat{E}_{\mathrm{CA}}$ into the Frozen Backbone Denoiser through a trainable Extreme Control Network. Inspired by the control-based diffusion paradigm [65], E-Control is designed for event-level semantic control in time-series diffusion. The Extreme Control Network consists of three components: Semantic Encoder, Control Branch, and Control Gate.

Semantic Encoder. The Semantic Encoder takes the predicted control signal $\hat{E}_{\mathrm{CA}}$ as input and maps it into semantic features for the Control Branch. As defined in Section 3.3, $\hat{E}_{\mathrm{CA}}$ follows the event-level semantic representation $EES = \{\mathcal{L}, \mathcal{I}, \mathcal{S}\}$, where each event is described by its location, intensity, and temporal shape. For each event, we embed its semantic tuple $(\mathcal{L}^m, \mathcal{I}^m, \mathcal{S}^m)$ into an event-aware token $\mathbf{e}_m$ using a shared projection module:

$$\mathbf{e}_m = \mathrm{Emb}(\mathcal{L}^m, \mathcal{I}^m, \mathcal{S}^m), \quad m = 1, \ldots, n_{\mathrm{ev}}. \tag{30}$$

We then aggregate all event tokens with a lightweight set encoder to obtain the semantic control representation:

$$\mathbf{C}_{\mathrm{sem}}(t_{\mathrm{CA}}) = \mathrm{SetEnc}(\{\mathbf{e}_m\}_{m=1}^{n_{\mathrm{ev}}}). \tag{31}$$

Here, $\mathrm{Emb}(\cdot)$ denotes a shared embedding function for event-level semantics, and $\mathrm{SetEnc}(\cdot)$ denotes a set-attention encoder. This design produces a compact, permutation-invariant representation that captures both per-event attributes and relations among events. The resulting $\mathbf{C}_{\mathrm{sem}}(t_{\mathrm{CA}})$ serves as the event-level conditioning representation for layer-aligned control.

Control Branch. The Control Branch transforms the semantic features $\mathbf{C}_{\mathrm{sem}}(t_{\mathrm{CA}})$ into layer-aligned control signals $\{\mathbf{c}_\ell(t_{\mathrm{CA}})\}_{\ell=1}^{L_c}$ for the frozen Backbone Denoiser. To match the injection sites, it adopts a lightweight trainable structure aligned with the core blocks of the Backbone Denoiser, so that its intermediate features correspond to the hidden representations where control is injected. Starting from the current denoising state $\mathbf{x}_n(t_{\mathrm{CA}})$, each control block incorporates the time-step embedding $\mathrm{TE}(t_{\mathrm{CA}})$ and attends to the semantic features $\mathbf{C}_{\mathrm{sem}}(t_{\mathrm{CA}})$, producing the control signal for the corresponding injection layer:

$$\mathbf{c}_\ell(t_{\mathrm{CA}}) = \mathrm{CB}_\ell\big(\mathbf{c}_{\ell-1}(t_{\mathrm{CA}}), \mathrm{TE}(t_{\mathrm{CA}}), \mathbf{C}_{\mathrm{sem}}(t_{\mathrm{CA}})\big), \quad \ell = 1, \ldots, L_c. \tag{32}$$

Here, $\mathrm{CB}_\ell(\cdot)$ denotes the ℓ-th control block, $\mathrm{TE}(\cdot)$ denotes the time-step embedding function, and $\mathbf{c}_\ell(t_{\mathrm{CA}})$ is the control signal aligned with the ℓ-th injection layer. In practice, semantic conditioning is implemented through cross-attention, where intermediate control features serve as queries and semantic features serve as keys and values.

Control Gate. Given the layer-aligned control signals $\{\mathbf{c}_\ell(t_{\mathrm{CA}})\}_{\ell=1}^{L_c}$ from the Control Branch, the Control Gate converts them into gated residual offsets for the frozen Backbone Denoiser. At each predefined injection site, the gate produces an offset that matches the shape of the corresponding hidden representation or output prediction, and adaptively regulates its magnitude:

$$\Delta_\ell(t_{\mathrm{CA}}) = \mathrm{CG}_\ell(\mathbf{c}_\ell(t_{\mathrm{CA}})) = \sigma\big(\mathbf{W}_\ell^{(g)}\mathbf{c}_\ell(t_{\mathrm{CA}})\big) \odot \big(\mathbf{W}_\ell^{(\Delta)}\mathbf{c}_\ell(t_{\mathrm{CA}})\big), \quad \ell = 1, \ldots, L_c. \tag{33}$$

Here, $\mathrm{CG}_\ell(\cdot)$ denotes the ℓ-th gating module, $\sigma(\cdot)$ is the sigmoid function, $\odot$ denotes element-wise product, and $\mathbf{W}_\ell^{(g)}$ and $\mathbf{W}_\ell^{(\Delta)}$ project the control feature to the gate value and residual offset, respectively. The resulting offsets are injected as residual updates at the corresponding sites of the Backbone Denoiser. For example, when injecting into a hidden representation or the output prediction, we have

$$\tilde{\mathbf{x}}_h(t_{\mathrm{CA}}) = \mathbf{x}_h(t_{\mathrm{CA}}) + \Delta_h(t_{\mathrm{CA}}), \quad \tilde{\hat{\mathbf{x}}}_0(t_{\mathrm{CA}}) = \hat{\mathbf{x}}_0(t_{\mathrm{CA}}) + \Delta_0(t_{\mathrm{CA}}), \tag{34}$$

where $\Delta_h(t_{\mathrm{CA}})$ and $\Delta_0(t_{\mathrm{CA}})$ are selected from $\{\Delta_\ell(t_{\mathrm{CA}})\}_{\ell=1}^{L_c}$ according to the predefined injection layers. This gated residual design adaptively strengthens extreme-event formation while preserving background temporal patterns.

L Implementation Details

L.1 Dataset Description

In this section, we provide a detailed description of the six datasets used in this work. Our evaluation is conducted on one synthetic dataset and five real-world datasets spanning diverse domains, including

Temperature, Precipitation, ECG, Electricity, and Transportation. An overview of these datasets is presented in Table 3.

Table 3: Dataset statistics.

Datasets	# of Samples	Time Series Length	Categories	Extreme Events	Link
SYN-DATA	1000	200	Synthetic	Synthetic Extreme Event Sequence	E4GEN/Data
WEA-TEMP	1520	168	Climate	Cold-Temperature Event	Dewey/CustomWeather
WEA-PREC	1056	90	Climate	Heavy-Rainfall Event	Dewey/CustomWeather
LTST-ECG	1000	250	Healthcare	ST-Segment Elevation Event	PhysioNet LTSTDB
HH-POWER	1096	144	Energy	High Electricity Consumption Event	UCI Household Power
PEMS-SF	1320	144	Transportation	Traffic Congestion Event	UCI PEMS-SF

- **SYN-DATA**: For the synthetic dataset, we design a toy example to explicitly illustrate the generation mechanism of extreme events in time series. Here, an extreme event is defined as a contiguous temporal segment whose values remain above a predefined threshold. Each sequence consists of four components: a trend component, a seasonal component, a noise component, and an extreme-event component. To facilitate separate analysis, we further introduce a threshold so that the extreme-event component can be clearly identified and isolated. The full sequence is defined as

$$x_i(t) = \text{trend}_i(t) + \text{season}_i(t) + \text{noise}_i(t) + \text{extreme}_i(t), \quad t = 0, \ldots, L-1. \tag{35}$$

The trend component is modeled as

$$\text{trend}_i(t) = b_i + a_i \frac{t}{L-1}, \tag{36}$$

where $a_i \sim U(0.05, 0.1)$ and $b_i \sim U(-0.02, 0.02)$. The seasonal component is defined as the sum of three sinusoidal waves with fixed frequencies $k \in \{5, 8, 11\}$:

$$\text{season}_i(t) = \sum_{k \in \{5,8,11\}} A_{i,k} \sin\left(2\pi k \frac{t}{L} + \phi_{i,k}\right). \tag{37}$$

Here, $A_{i,5} \in \{0.04, 0.06, 0.08, 0.10\}$, $A_{i,8} \in \{0.03, 0.05, 0.07, 0.09\}$, $A_{i,11} \in \{0.02, 0.04, 0.06, 0.08\}$, and $\phi_{i,k} \in \{0, \frac{\pi}{2}, \pi, \frac{3\pi}{2}\}$. The noise component is sampled from a Gaussian distribution:

$$\text{noise}_i(t) \sim \mathcal{N}(0, 0.01^2). \tag{38}$$

For the extreme-event component, we first define the background signal as the sum of the trend, seasonal, and noise components:

$$\text{base}_i(t) = \text{trend}_i(t) + \text{season}_i(t) + \text{noise}_i(t). \tag{39}$$

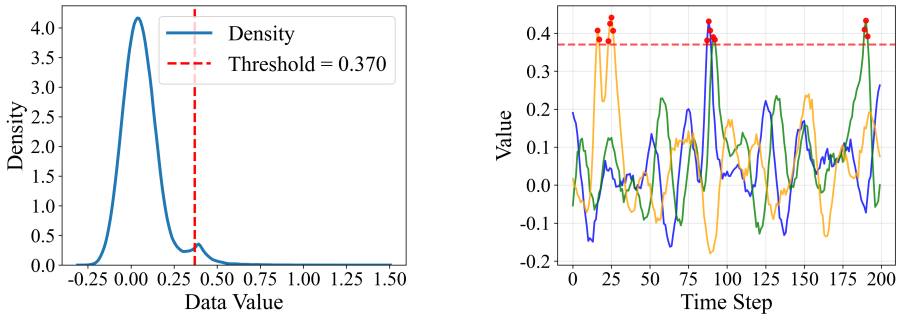
We further introduce the threshold $\theta = 0.3711$ to distinguish extreme events from normal background fluctuation, and the signal with the extreme-event component satisfies: $\text{base}_i(t) + \text{extreme}_i(t) > u$. As a result, the threshold can cleanly separate normal background behavior from extreme events.

To make extreme events follow a predefined intrinsic pattern, we align their occurrence locations with specific phase combinations of $\text{base}_i(t)$. In this way, the generated extremes are structurally coupled with the underlying temporal pattern rather than inserted at arbitrary positions. Each extreme event is modeled as a smooth Gaussian-shaped spike, whose amplitude is adaptively determined by the local background level. The final sequence is therefore constructed as

$$x_i(t) = \text{base}_i(t) + \sum_{j=1}^{n_i} \text{spike}_{i,j}(t). \tag{40}$$

For our experiments, we set the sequence length to 200 and generate 1000 samples, resulting in a final dataset of shape $(1000, 200, 1)$. This toy dataset provides a controlled testbed for studying the event-level generation mechanism of extremes. Notably, this design induces a visible concentration of values near the threshold region in the tail of the overall distribution, producing a local bump rather than a purely conventional long-tailed shape. As

a result, the synthetic distribution exhibits a structured tail pattern, making it particularly suitable for evaluating whether generation models can capture tail-specific enhancement beyond simple heavy-tail behavior. The estimated probability density function (PDF) and sample visualizations are shown in Fig. 15a and Fig. 15b, respectively. In particular, three representative samples are displayed in yellow, blue, and green. The red dashed line denotes the threshold, while the red dots indicate the extreme-value points. All subsequent figures across all datasets follow the same sample setting.



(a) PDF of SYN-DATA (b) Data Samples in SYN-DATA

Fig. 15: Visualizations for SYN-DATA Dataset

- **WEA-TEMP**: We use the *Hourly Weather Data* provided by Dewey [66], which contains hourly climate observations across the United States since 2018. Here, an extreme event refers to a low-temperature event, represented by a consecutive period with temperature below a predefined threshold. We select five nearby stations around Jacksonville in northeastern Florida, a region frequently affected by extreme weather such as heavy rainfall and hurricanes. The temperature records are divided into weekly samples, each containing 168 hourly observations (24×7). Using data from these five stations over the period from January 1, 2021 to December 31, 2025, we obtain 1,520 samples, resulting in a dataset of shape $(1520, 168, 1)$. The extreme threshold is defined as the lower 5% quantile of the temperature distribution, corresponding to $8.3°\mathrm{C}$. Temperatures below this value are treated as anomalously low-temperature extremes. Unlike the other datasets, this benchmark focuses on lower-tail extremes, which further demonstrates the flexibility of our model across different extreme-event types. This dataset captures both long-term temporal variation and local spatial variation across nearby stations. We visualize the temperature series of the five stations over one month in Fig. 16. The PDF and sample visualizations are shown in Fig. 17a and Fig. 17b, respectively.

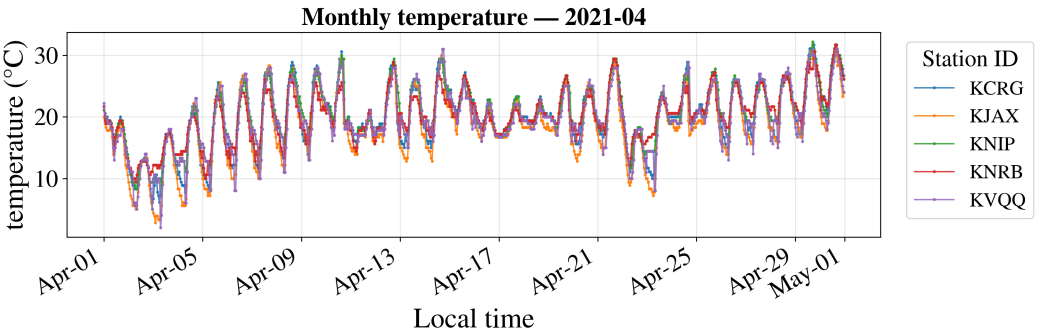
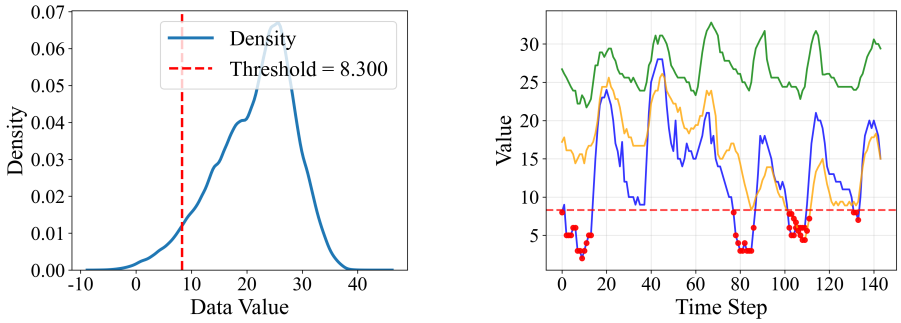


Fig. 16: Temperature series for five stations in April 2021.

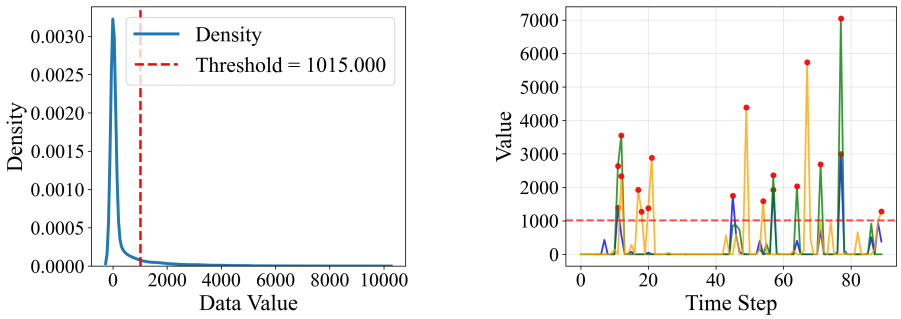
- **WEA-PREC**: We use the *Daily Weather Data* provided by Dewey [67]. In this dataset, an extreme event refers to an extreme rainfall event, characterized by a consecutive period with daily precipitation above a predefined threshold. We select 88 observation stations



(a) PDF of WEA-TEMP Dataset (b) Data Samples in WEA-TEMP Dataset

Fig. 17: Visualizations for WEA-TEMP Dataset

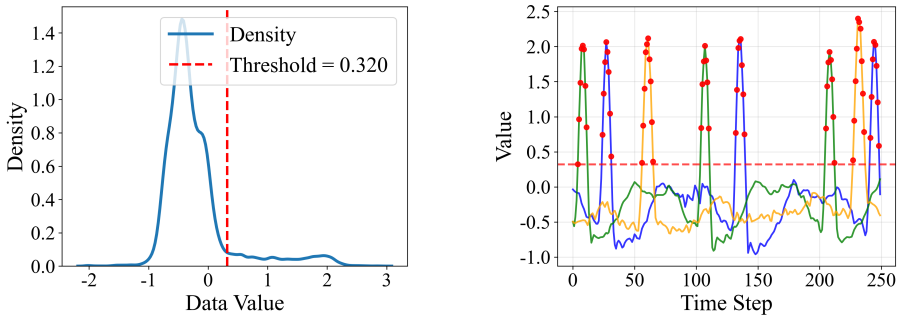
across Florida and record the daily total precipitation at each station. Daily precipitation is used instead of hourly precipitation because the hourly records contain substantial missing values. The data are partitioned into 90-day samples. Using records from January 1, 2023 to December 31, 2025, we construct a dataset of shape $(1056, 90, 1)$. The extreme threshold is set to 1015, corresponding to the top 10% of the data. The PDF and sample visualizations are shown in Fig. 18a and Fig. 18b, respectively.



(a) PDF of WEA-PREC Dataset (b) Data Samples in WEA-PREC Dataset

Fig. 18: Visualizations for WEA-PREC Dataset

- **LTST-ECG**: We use the *Long-Term ST Database* (LTST) from PhysioNet [68, 69], which contains long-duration ambulatory ECG recordings with expert-provided ST-related annotations. In this dataset, we first identify abnormal ST-related intervals based on the provided annotations, and then define extreme events within these intervals as consecutive periods during which the signal values remain above a predefined threshold. During preprocessing, we load the raw ECG signals and annotation files, and extract the first ECG lead as a one-dimensional time series. The annotations are used to identify abnormal ST-related intervals, which provide clinically meaningful references for locating informative regions. Based on these regions, we segment the signal into fixed-length windows of 250 time steps, corresponding to one second at 250 Hz, and generate candidate samples using a sliding-window strategy. To obtain representative windows with stronger signal intensity, we rank candidates by the mean absolute signal amplitude within each window and randomly select 1,000 samples from the high-score set, yielding a final dataset of shape $(1000, 250, 1)$. The extreme threshold is set to 0.3205, corresponding to the top 10% of the data. The PDF and sample visualizations are shown in Fig. 19a and Fig. 19b, respectively.
- **HH-POWER** : We use the *UCI Individual Household Electric Power Consumption Dataset* [70], which records minute-level household electricity usage over an extended period. In this dataset, an extreme event refers to a high-consumption electricity event, characterized

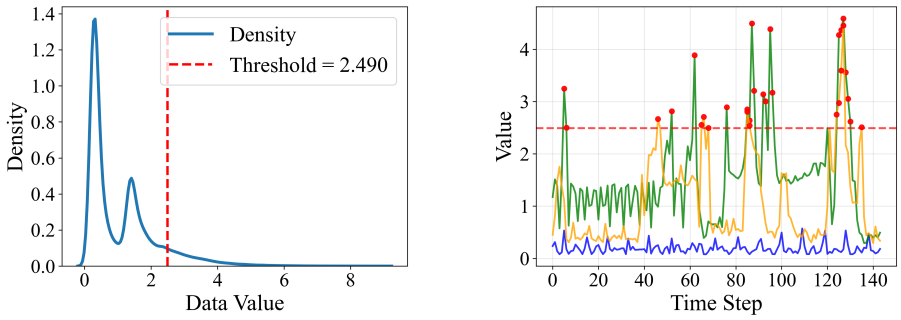


(a) PDF of LTST-ECG Dataset (b) Data Samples in LTST-ECG Dataset

Fig. 19: Visualizations for LTST-ECG Dataset

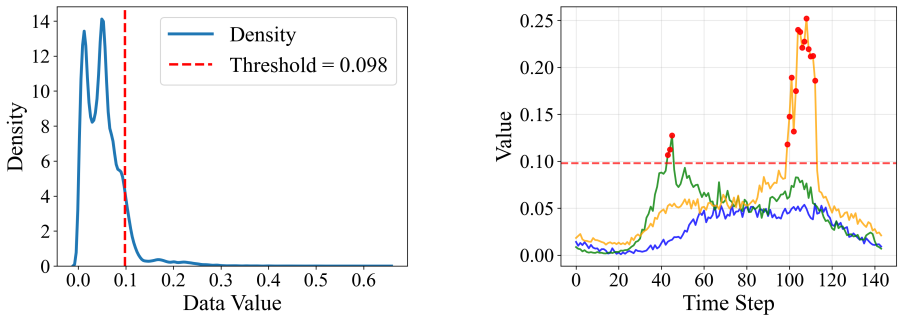
by a consecutive period during which the power values remain above a predefined threshold. We focus on the `Global_active_power` variable and select a continuous three-year period from January 1, 2007 to December 31, 2009. After temporal alignment and missing-value imputation, the data are resampled at 10-minute intervals, so that each day contains 144 observations. We then group the data into daily samples and retain only complete sequences, resulting in a dataset of shape $(1096, 144, 1)$. The extreme threshold is set to $2.4905\,\mathrm{kW}$, corresponding to the top 10% of the data. This dataset provides a realistic benchmark for modeling daily electricity consumption patterns and their extreme behaviors in a single-household setting. The PDF and sample visualizations are shown in Fig. 20a and Fig. 20b, respectively.

- **PEMS-SF**: We use the *PEMS-SF* dataset derived from the California Performance Measurement System (PeMS) [71, 72], which records freeway traffic occupancy rates from monitoring stations in the San Francisco Bay Area. In this dataset, an extreme event refers to a high-occupancy traffic event, characterized by a consecutive period during which the occupancy values remain above a predefined threshold. The occupancy value ranges from 0 to 1 and indicates the proportion of time that a sensor is occupied by vehicles within a given interval. The data are sampled every 10 minutes, so each day is represented by a sequence of length 144. Following the provided train-test split, we combine the `PEMS_train` and `PEMS_test` sets, obtaining 440 daily sequences in total. To build our benchmark, we focus on three nearby monitoring stations and extract their daily occupancy sequences separately, treating each station-day pair as one sample. This results in a dataset of shape $(1320, 144, 1)$. The extreme threshold is set to 0.097, corresponding to the top 10% of the occupancy values. This construction captures realistic daily traffic dynamics while also reflecting local spatial variation across nearby sensor locations. The PDF and sample visualizations are shown in Fig. 21a and Fig. 21b, respectively.



(a) PDF of HH-POWER Dataset (b) Data Samples in HH-POWER Dataset

Fig. 20: Visualizations for HH-POWER Dataset



(a) PDF of PEMS-SF Dataset (b) Data Samples in PEMS-SF Dataset

Fig. 21: Visualizations for PEMS-SF Dataset

L.2 Descriptions of Metrics

To comprehensively evaluate the performance of our generation model, we utilize seventeen distinct metrics assessed from two primary perspectives: overall generation quality and extreme-event generation quality. Below are the detailed definitions and implementations of each metric.

For **overall generation quality**, we employ eight metrics to assess the general characteristics of the generated time series, which are further divided into fidelity metrics and downstream utility metrics. Following TSGBench [40], we use five fidelity metrics to measure the distributional and temporal similarity between the generated and real data:

- **Wasserstein Distance (Wass) [35]:** Measures the distance between two probability distributions by quantifying the minimum cost to transform P into Q. We use the 1-Wasserstein distance:
$$W_1(P,Q) = \inf_{\gamma \in \Pi(P,Q)} \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\gamma} \left[\|\mathbf{x}-\mathbf{y}\|\right], \tag{41}$$
where $\Pi(P,Q)$ denotes the set of all joint distributions with marginals P and Q.
- **Kolmogorov-Smirnov Test (KS) [36]:** Computes the maximum absolute distance between the empirical cumulative distribution functions (CDFs) of the real dataset (F_{real}) and the generated dataset (F_{gen}):
$$D_{\text{KS}} = \sup_x \left|F_{\text{real}}(x) - F_{\text{gen}}(x)\right|. \tag{42}$$
- **Jensen-Shannon Divergence (JS) [37]:** A symmetric and smoothed version of the Kullback-Leibler (KL) divergence, defined as:
$$JS(P \parallel Q) = \frac{1}{2} D_{\text{KL}}(P \parallel M) + \frac{1}{2} D_{\text{KL}}(Q \parallel M), \tag{43}$$
where $M = \frac{1}{2}(P+Q)$.
- **Maximum Mean Discrepancy (MMD) [38]:** Evaluates the distance between distributions mapped into a Reproducing Kernel Hilbert Space (RKHS) using a kernel $k(\cdot,\cdot)$:
$$\begin{aligned}\text{MMD}^2(P,Q) = \mathbb{E}_{\mathbf{x},\mathbf{x}'\sim P}\left[k(\mathbf{x},\mathbf{x}')\right] + \mathbb{E}_{\mathbf{y},\mathbf{y}'\sim Q}\left[k(\mathbf{y},\mathbf{y}')\right] \\ - 2\mathbb{E}_{\mathbf{x}\sim P,\mathbf{y}\sim Q}\left[k(\mathbf{x},\mathbf{y})\right].\end{aligned} \tag{44}$$
- **Autocorrelation Discrepancy (ACD) [39]:** Assesses the preservation of temporal dynamics by computing the absolute difference between the autocorrelation functions:
$$\text{ACD} = \frac{1}{L}\sum_{l=1}^{L} \left|\rho_{\text{real}}(l) - \rho_{\text{gen}}(l)\right|, \tag{45}$$
where $\rho(l)$ is the autocorrelation at lag l.

Furthermore, we evaluate the general downstream utility of the generated data through three Train-on-Synthetic, Test-on-Real (TSTR) tasks:

- **Context-FID ([41]):** Measures the Fréchet distance between the contextual representations of real and generated time series. We first train a TS2Vec [73] encoder on the real data and then encode each real and generated sequence into a full-series representation. Let μ_r, Σ_r and μ_g, Σ_g denote the mean vectors and covariance matrices of the real and generated representations, respectively:

$$\text{Context-FID} = \|\mu_r - \mu_g\|_2^2 + \text{Tr}\left(\Sigma_r + \Sigma_g - 2(\Sigma_r\Sigma_g)^{1/2}\right). \tag{46}$$

- **Predictive Score (Pred [1]):** Evaluates the prediction utility of generated time series. A forecasting model f_θ is trained only on generated data and tested on real data. For each channel d and prediction horizon h, the model predicts the future segment from a fixed-length lookback window:

$$\hat{\mathbf{x}}_{t:t+h,d} = f_\theta(\mathbf{x}_{t-L_b:t,d}). \tag{47}$$

  The score is computed as the average prediction error on real data:

$$\text{Pred} = \frac{1}{|\mathcal{D}|}\sum_{(d,h)}\frac{1}{m_{d,h}h}\sum_{i=1}^{m_{d,h}}\sum_{j=1}^{h}\left|\hat{x}_{i,j}^{(d,h)} - x_{i,j}^{(d,h)}\right|, \tag{48}$$

  where $m_{d,h}$ is the number of valid test windows for channel d and horizon h. Lower values indicate that the generated data better preserves predictive temporal dynamics.

- **Reconstructive Score (Recon [42]):** Evaluates reconstruction-based utility. An autoencoder, consisting of an encoder E_ϕ and a decoder D_ψ, is trained on generated data and tested on real data. For each real sequence $\mathbf{x}^{(i)} \in \mathbb{R}^{T\times D}$, the reconstruction is obtained as

$$\hat{\mathbf{x}}^{(i)} = D_\psi\left(E_\phi(\mathbf{x}^{(i)})\right). \tag{49}$$

  The score is the mean squared reconstruction error over all test sequences, time steps, and channels:

$$\text{Recon} = \frac{1}{NTD}\sum_{i=1}^{N}\sum_{t=1}^{T}\sum_{d=1}^{D}\left(\hat{x}_{t,d}^{(i)} - x_{t,d}^{(i)}\right)^2. \tag{50}$$

For **extreme-event generation quality**, we specifically evaluate the model's ability to capture and synthesize extreme phenomena. Let $\mathcal{E}_{\text{real}}$ and $\mathcal{E}_{\text{gen}}$ denote the sets of extracted extreme events from the real and generated time series, respectively. Here, $P(\cdot)$ denotes the corresponding distribution. We propose seven event-level fidelity metrics:

- **EM-W1:** The Wasserstein-1 distance between the distributions of event-level mean values from real and generated extreme events:

$$\text{EM-W1} = W_1\left(P(\text{Mean}(\mathcal{E}_{\text{real}})), P(\text{Mean}(\mathcal{E}_{\text{gen}}))\right), \tag{51}$$

  where $\text{Mean}(\mathcal{E})$ denotes the collection of mean values computed over the temporal span of each event in $\mathcal{E}$.

- **EC-Diff:** The absolute difference between the numbers of extracted extreme events in real and generated time series:

$$\text{EC-Diff} = \left||\mathcal{E}_{\text{real}}| - |\mathcal{E}_{\text{gen}}|\right|. \tag{52}$$

  where $|\mathcal{E}|$ denotes the number of extracted extreme events in the event set $\mathcal{E}$.

- **ED-W1:** The Wasserstein-1 distance between the duration distributions of real and generated extreme events:

$$\text{ED-W1} = W_1\left(P(\text{Dur}(\mathcal{E}_{\text{real}})), P(\text{Dur}(\mathcal{E}_{\text{gen}}))\right), \tag{53}$$

  where $\text{Dur}(\mathcal{E})$ denotes the collection of durations of all events in $\mathcal{E}$.

- **EP-W1:** The Wasserstein-1 distance between the peak-value distributions of real and generated extreme events:

$$\text{EP-W1} = W_1\left(P(\text{Peak}(\mathcal{E}_{\text{real}})), P(\text{Peak}(\mathcal{E}_{\text{gen}}))\right), \tag{54}$$

  where $\text{Peak}(\mathcal{E})$ denotes the collection of peak values of all events in $\mathcal{E}$.

- **EWass ([35]):** The Wasserstein-1 distance between the distributions of pooled extreme values from real and generated extreme events:

$$\mathrm{EWass} = W_1\left(P(\mathcal{V}(\mathcal{E}_{\text{real}})), P(\mathcal{V}(\mathcal{E}_{\text{gen}}))\right), \tag{55}$$

  where $\mathcal{V}(\mathcal{E})$ denotes the collection of all values at the extreme time points within the event set $\mathcal{E}$.

- **EJS ([37]):** The Jensen-Shannon distance between the distributions of pooled extreme values from real and generated extreme events:

$$\mathrm{EJS} = \sqrt{JS\left(P(\mathcal{V}(\mathcal{E}_{\text{real}})) \parallel P(\mathcal{V}(\mathcal{E}_{\text{gen}}))\right)}. \tag{56}$$

- **EMMD ([38]):** The RBF-kernel Maximum Mean Discrepancy between real and generated extreme-event snippet distributions. Each snippet is a fixed-length segment centered at the peak of an identified extreme event. The final score further includes a count penalty for the mismatch in the number of extracted snippets:

$$\mathrm{EMMD} = \mathrm{MMD}_{\mathrm{RBF}}\left(P(\mathcal{S}_{\text{real}}), P(\mathcal{S}_{\text{gen}})\right) + \lambda_{\mathrm{cnt}}\Delta_{\mathrm{cnt}}. \tag{57}$$

Finally, we assess the extreme-oriented downstream utility to evaluate the practical value of the generated extreme events for critical downstream tasks:

- **EPred:** Evaluates whether generated data can support prediction under extreme-event contexts. A forecasting model is trained on generated windows whose prediction target corresponds to an extreme time point, and is then tested on real windows selected by the same rule. Specifically, the model predicts future values from 1 to 5 steps ahead using a fixed-length lookback window:

$$\hat{x}_{t+h,d} = f_\theta^{(h)}(\mathbf{x}_{t-L_b:t,d}), \quad h = 1, \ldots, 5, \tag{58}$$

  where d denotes the target channel, L_b is the lookback length, and h denotes the prediction horizon. The score is computed as the average prediction error over real extreme-target windows and prediction horizons:

$$\mathrm{EPred} = \frac{1}{5|\mathcal{W}_{\mathrm{ext}}|}\sum_{h=1}^{5}\sum_{(t,d)\in\mathcal{W}_{\mathrm{ext}}} \ell\left(\hat{x}_{t+h,d}, x_{t+h,d}\right), \tag{59}$$

  where $\mathcal{W}_{\mathrm{ext}}$ denotes the set of real test windows whose prediction targets are extreme time points, and $\ell(\cdot,\cdot)$ is the prediction loss, such as MAE or MSE. Lower values indicate that the generated data better preserves local predictive dynamics around extreme events.

- **ERecon:** Evaluates whether generated data preserves reconstructive patterns around extreme-event regions. An autoencoder, consisting of an encoder E_ϕ and a decoder D_ψ, is trained on generated windows that contain at least one extreme time point. It is then tested on real sequences, with the reconstruction error evaluated only around real extreme-event regions. For each real sequence, the reconstruction is obtained as:

$$\hat{\mathbf{x}}^{(i)} = D_\psi\left(E_\phi(\mathbf{x}^{(i)})\right). \tag{60}$$

  The score is computed as the average reconstruction error over real extreme-event regions:

$$\mathrm{ERecon} = \frac{1}{|\mathcal{E}_{\text{real}}|}\sum_{e\in\mathcal{E}_{\text{real}}} \mathrm{ReconErr}(e), \tag{61}$$

  where $\mathrm{ReconErr}(e)$ denotes the mean squared reconstruction error over the temporal region of extreme event e. Lower values indicate better reconstruction of extreme-event patterns.

L.3 Baseline Description

In E4GEN, we consider nine baseline methods from six different categories. An overview of these baselines is provided in Table 4. Their implementations with detailed parameter settings are included in the `Baseline` directory of our codebase. The details are summarized as follows:

Table 4: Overview of baseline methods compared in this paper.

Baseline	Category	Venue	Link
TimeGAN [1]	GAN	NeurIPS'19	Official Implementation
TimeVAE [11]	VAE	ArXiv'21	Official Implementation
koVAE [12]	VAE	ICLR'24	Official Implementation
F-Flow [13]	Flow	ICLR'21	Official Implementation
DiffWave [15]	Diffusion	ICLR'21	Official Implementation
Diffusion-TS [4]	Diffusion	ICLR'24	Official Implementation
SDForger [5]	LLM	NeurIPS'25	Official Implementation
FIDE [3]	Extreme-Aware	NeurIPS'24	Official Implementation
HeavyDiff [23]	Extreme-Aware	ICLR'25	Official Implementation

- **TimeGAN** [1]: For the TimeGAN baseline, we adopt the implementation provided by the TSGM library [7], a unified framework for synthetic time-series generation and evaluation. This choice is mainly motivated by reproducibility and engineering compatibility, since the original TimeGAN codebase is built in an early TensorFlow style and imposes strict constraints on software versions, dependencies, and training pipeline organization.

  In our implementation, the TimeGAN model is built using the TSGM library through `TimeGAN(...)`. The sequence length T and feature dimension C are automatically inferred from the shape of the input data. The model adopts a GRU-based architecture with hidden dimension 24, 3 recurrent layers, batch size 128, and $\gamma = 1.0$. After initialization, the model is configured through the `compile` interface, where separate Adam optimizers with learning rate 10^{-3} are assigned to the discriminator, generator, embedding module, supervised component, and autoencoder component. The loss design follows the standard TimeGAN formulation, using mean squared error for embedding and reconstruction objectives and binary cross-entropy for adversarial discrimination.

  TSGM encapsulates the multi-stage optimization strategy of TimeGAN, and the actual training process proceeds through embedding network training, supervised training, and joint adversarial training for 100 epochs. Once training is completed, the generator weights are stored as the reusable component for later synthesis. During generation, random noise sampled from a uniform distribution is fed into the generator batch by batch, which supports arbitrary output sample sizes without requiring exact batch divisibility. For the sample-only setting, the generator graph is first built through a zero-epoch initialization step before loading pretrained weights, after which synthetic sequences are produced directly. The generated outputs are finally mapped back to the original scale through inverse min-max normalization and saved in the same shape as the input series.

- **TimeVAE** [11]: For the TimeVAE baseline, we use a PyTorch reimplementation of the original TimeVAE model proposed in [11]. The corresponding code link is provided in Table 4. Similar to TimeGAN, we do not use the original TensorFlow implementation and instead adopt a more recent PyTorch version in our experiments.

  In our implementation, TimeVAE is used as a variational autoencoder for time-series generation. The encoder maps each input sequence into a latent representation through stacked 1D convolutional layers, while the decoder reconstructs the sequence through an additive decomposition including a global level term, a polynomial trend term, and a residual branch for non-structured variations. The model is trained with a weighted VAE objective consisting of the reconstruction loss and the standard KL regularization. During sampling, latent variables are drawn from a standard Gaussian prior and decoded into synthetic sequences, which are then transformed back to the original data scale.

  For the parameter settings, each dataset is organized as sliding-window samples of shape (N, T, C) in `.npz` format. We use a 90%/10% train-validation split and fit MinMax normalization only on the training set. The latent dimension is set to 8, the hidden dimensions are $[50, 100, 200]$, the reconstruction weight is 3.0, and the batch size is 16. We enable the residual connection and use a second-order polynomial trend term, while disabling the

seasonal component. The model is trained for 200 epochs with Adam and a fixed random seed of 123, and the number of generated samples is matched to the size of the training set.

- **koVAE** [12]: For the koVAE baseline, we reproduce the model using the authors' Official Implementation. KoVAE is a variational autoencoder for time-series generation that replaces the conventional static latent prior with a Koopman-inspired dynamical prior. Specifically, it models the latent prior evolution through a linear transition map, so that temporal dynamics in latent space follow a structured linear system. This design enables koVAE to better capture temporal regularities in time series and supports both regular and irregularly sampled data. By combining the VAE framework with Koopman-inspired latent dynamics, koVAE provides a principled approach to realistic time-series generation.

  For the model configuration, following the Official Implementation, all channels are independently normalized to $[0, 1]$ using Min–Max scaling before training, and the generated samples are mapped back to the original scale afterward. We adopt the regular setting of koVAE, in which both the encoder and decoder are implemented as bidirectional GRUs. The encoder uses 3 layers with hidden dimension 20 and maps inputs to a latent space of dimension 16, while the decoder uses a linear output head with sigmoid activation to match the normalized target space. Batch normalization is enabled by default. The training objective consists of a reconstruction term, a KL regularization term, and a Koopman prior prediction term, with weights $w_{\mathrm{rec}} = 1.0$, $w_{\mathrm{kl}} = 0.007$, and $w_{\mathrm{pred_prior}} = 0.005$, respectively. The prior prediction loss is computed with 1-step rollout, and the Koopman operator is estimated using the default QR-based solver. For optimization, we train the model with Adam for 100 epochs using a learning rate of 7×10^{-4}, zero weight decay, a batch size of 64, and random seed 10. During generation, the trained model samples latent trajectories and decodes them into synthetic sequences with the same sample size as the training set.

- **F-Flow** [13]: For the F-Flow baseline, we reproduce the model using the authors' Official Implementation. F-Flow is a normalizing-flow-based model for time-series generation that operates in the frequency domain instead of directly modeling raw sequences in the time domain. Specifically, it first transforms each time series into its Fourier representation, and then learns an explicit likelihood over the resulting spectral coefficients using invertible flows. By modeling data in the spectral domain, F-Flow is better suited to capturing global temporal patterns and periodic structures in time series. This design provides a principled explicit-density framework for realistic time-series generation.

  For the model configuration, following the Official Implementation, all channels are independently normalized to $[0, 1]$ using Min–Max scaling before training. Each time series is then flattened into a one-dimensional vector and transformed into the frequency domain, where F-Flow performs density estimation over the spectral representation. To satisfy the model requirement for the Fourier transform size, an additional zero is prepended when the flattened sequence length is even. We adopt the default Fourier Flow setting, with 5 flow layers and hidden dimension 200. During training, the spectral coefficients are standardized using dataset-level mean and standard deviation statistics. For optimization, we train the model with Adam for 1000 epochs using a learning rate of 10^{-3}, together with an exponential learning rate decay. During generation, samples are first drawn from a standard Gaussian distribution, then mapped through the inverse flow and inverse Fourier transform back to the time domain, and finally reshaped to the original sequence format. We generate the same number of synthetic samples as in the training set for downstream evaluation.

- **DiffWave** [15]: For the DiffWave baseline, we adapt the authors' Official Implementation from raw speech waveform synthesis to unconditional univariate time-series generation. DiffWave is a diffusion-based generative model that progressively transforms Gaussian noise into realistic sequences through a predefined reverse diffusion process, using a one-dimensional convolutional denoising network as its backbone. Since the original implementation is designed for speech data, we replace the audio-based pipeline with direct `.npy` loading and use samples of shape $(N, T, 1)$ as both input and output. With these changes, DiffWave serves as a diffusion-based baseline for unconditional time-series generation in our benchmark.

  For the model configuration, the training data are stored in a single `.npy` file and normalized using dataset-level mean and standard deviation statistics computed over all values. During training, a random subsequence of length T is extracted from each series, while shorter sequences are right-padded with zeros before cropping. We retain the original DiffWave

backbone, including 30 residual layers, 64 residual channels, a dilation cycle length of 10, and the default diffusion noise schedule with 50 training steps. The model is trained in the unconditional setting with the spectrogram input fixed to `None`. For optimization, we use Adam with a learning rate of 2×10^{-4} and an L_1 loss for noise prediction. During generation, samples are initialized from standard Gaussian noise, refined through reverse diffusion, and then transformed back to the original scale using the stored normalization statistics. For downstream evaluation, we generate the same number of synthetic samples as in the training set.

- **Diffusion-TS** [4]: For the Diffusion-TS baseline, we adopt the official open-source implementation and retain its original diffusion training and sampling procedure. In our codebase, the model is instantiated directly from the YAML configuration and trained on sequences normalized to $[-1, 1]$. The denoising network follows the interpretable decomposition design of Diffusion-TS, where the model reconstructs the clean target x_0 from noisy inputs and explicitly decomposes the prediction into trend and seasonality components. During training, a timestep is randomly sampled for each batch, noise is injected according to the forward diffusion process, and the network is optimized to recover the target sequence under the corresponding denoising objective. For generation, we use the EMA model to perform the reverse diffusion process and synthesize time series samples.

  For the model configuration, the sequence length and feature dimension are specified according to the input data. The Transformer backbone is built with 2 encoder layers and 2 decoder layers, a hidden dimension of 64, and 4 attention heads. The diffusion process uses 1000 timesteps with a cosine noise schedule, and the same number of steps is used during sampling. We adopt the L_1 loss as the reconstruction objective. For optimization, the model is trained with Adam using a base learning rate of 10^{-5}, a batch size of 64, and gradient accumulation every 2 steps. To further improve training stability, we employ EMA with decay 0.995 and update interval 10, together with a warmup-based ReduceLROnPlateau scheduler, where the warmup learning rate is set to 8×10^{-4} for the first 500 steps. The total training budget is 10,000 optimization steps.

- **SDForger** [5]: For the SDForger baseline, we follow its original framework for synthetic time-series generation with large language models. Its central idea is to map raw time series into compact functional embeddings, cast sequence generation as structured text generation, and then decode the generated embeddings back into time-series samples. By operating in a lower-dimensional representation space, SDForger avoids directly generating long raw sequences and instead leverages the generative capability of language models for time-series synthesis.

  In our implementation, the input sequences are processed under the multisample setting and embedded using FastICA, where the embedding dimension is determined automatically under a variance retention target of 0.7. These embeddings are then used to fine-tune an autoregressive language model, and the generated embeddings are mapped back to the original time-series space through inverse embedding and inverse normalization. The number of generated samples is set to match the size of the training set, and `norms_diversity_threshold` is set to 0.

- **FIDE** [3]: For the FIDE baseline, we follow its original extreme-aware diffusion framework for time-series generation. Its core idea is to improve tail preservation through frequency inflation in the frequency domain and further enhance extreme-event modeling by conditioning generation on block maxima. Since the original FIDE codebase is not directly runnable on some datasets, we make a lightweight revision to adapt it to our benchmark.

  In our implementation, FIDE is used as a conditional diffusion model, where each training window is paired with its block maximum and a GEV model is fitted on the observed maxima to characterize the extreme-value distribution. During training, the model takes the noisy sequence together with the corresponding block-maximum condition as input, and is optimized by the DDPM loss with an additional GEV-based regularization term. During sampling, new block-maxima conditions are drawn from the fitted GEV model to guide reverse diffusion, so that the generated sequences better preserve realistic peak behavior.

  For the parameter settings, we train the model for 400 epochs with batch size 2000, using hidden dimension 64 on univariate sequences. The diffusion process uses 100 steps with a linear noise schedule from $\beta_{\text{start}} = 10^{-4}$ to $\beta_{\text{end}} = 0.2$, together with correlated Gaussian-process noise with $\sigma = 0.05$. We also retain the original frequency-enhancement strategy,

where the top 20% high-frequency components are amplified by a factor of 1.1 before training and inversely transformed after generation.

- **HeavyDiff** [23]: As no Official Implementation of HeavyDiff is publicly available, and the original method is not specifically designed for time-series generation, we adapt its core idea within Diffusion-TS, a state-of-the-art framework for general time-series generation. Specifically, following the key idea of HeavyDiff, we replace the commonly used light-tailed Gaussian distribution in diffusion models with a heavy-tailed Student-t distribution in Diffusion-TS, thereby obtaining our HeavyDiff baseline.

  In our implementation, the heavy-tailed Student-t noise is applied consistently in both the forward diffusion and reverse sampling processes. Instead of directly sampling Gaussian noise, we construct the injected noise through a Gaussian scale mixture:

$$\epsilon = \frac{z}{\sqrt{\kappa/\nu}}.$$

  Here, ϵ denotes the sampled heavy-tailed noise, $z \sim \mathcal{N}(0, I)$ is a standard Gaussian random variable, $\kappa \sim \chi^2_\nu$ is a chi-square random variable, and ν is the degree of freedom controlling the tail heaviness of the Student-t distribution, where a smaller ν produces heavier tails. To reduce the variance shift introduced by heavy-tailed noise, we further apply an optional variance correction:

$$\epsilon \leftarrow \epsilon \cdot \sqrt{\frac{\nu - 2}{\nu}}, \qquad \nu > 2.$$

  This correction rescales the sampled noise so that its variance is numerically closer to that of the standard Gaussian setting, which improves training stability and comparability. The same Student-t-based noise mechanism is used during reverse stochastic sampling to maintain consistency between training and inference.

  For the parameter settings, we enable heavy-tailed diffusion in the main experiments by setting the degree of freedom of the Student-t distribution to $\nu = 2.5$, and apply variance correction by default.

L.4 Interpretable Generation Dynamics

In this section, we provide an interpretable analysis of how extreme events emerge and are controlled during the denoising process, which answers RQ4 in Section 5. To provide an intuitive illustration of the effect of the control signal in E4GEN, we show an example and trace the generation trajectory. Specifically, we examine a denoising process in E4GEN in which the control signal is activated at **step 100**, and visualize how the overall distribution and extreme-value statistics evolve across representative denoising steps. As shown in Fig. 22, the trajectory covers both the stages before control activation and the stages after the control signal is introduced, including step 101 immediately before activation and step 100 as the activation step. The results show that the generated distribution is progressively transformed from the initial Gaussian distribution toward the target data distribution, while the extreme values are enhanced after the control signal is activated.

We further analyze how extreme values evolve during the denoising process. For intuitive visualization, we report the proportion of extreme-value points along the denoising trajectory. Although this statistic does not fully capture the formation of complete extreme events, it provides a clear view of how the extreme-event control signal guides generation in E4GEN. As shown in Fig. 23a, after the control signal is activated at $t = 100$, the proportion of extreme-value points increases rapidly and moves toward the target extreme-value level, indicating that E4GEN can effectively amplify extreme regions during the refinement stage. In contrast, Fig. 23b shows that, without the extreme-event control signal, the extreme-value proportion increases only gradually and remains consistently lower than the target distribution. This comparison demonstrates that the proposed control signal provides explicit and reliable guidance for extreme-aware generation, enabling E4GEN to better follow the target extreme-event distribution.

L.5 E-Predictor Intermediate Output Analysis

To answer RQ5, this part analyzes the accuracy of E-Predictor in predicting extreme-event semantics during generation. E-Predictor bridges semantic characterization and control injection by converting

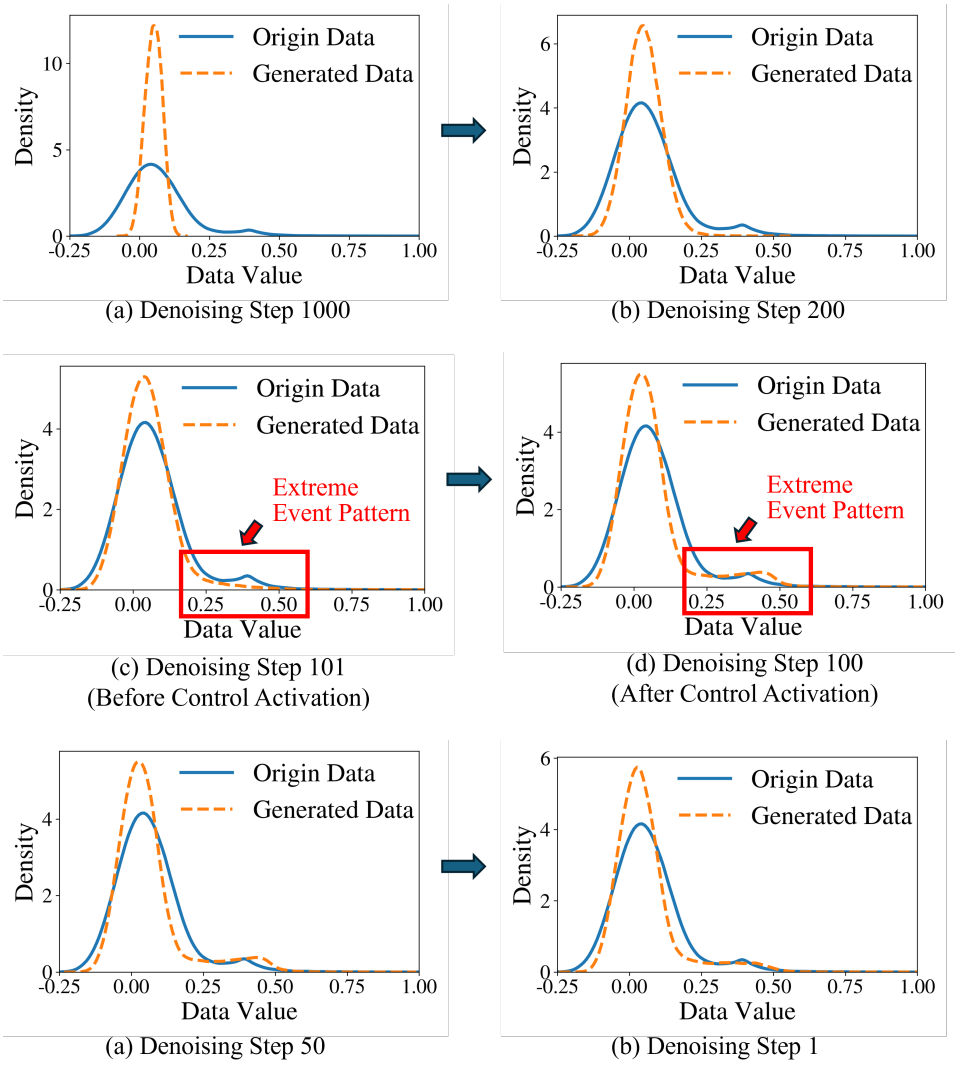
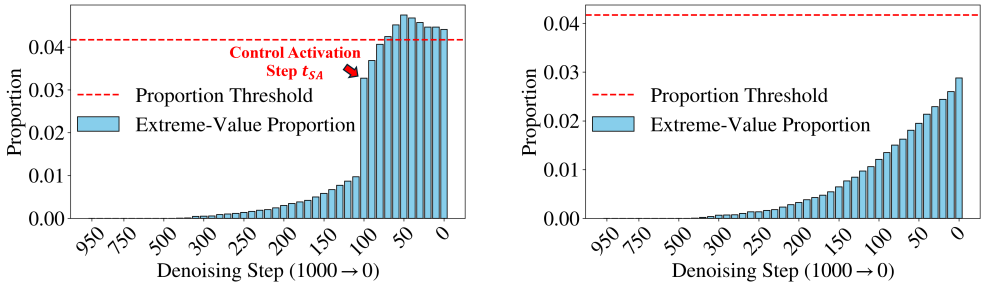


Fig. 22: Example for interpretable generation dynamics. The six figures visualize the evolution of the overall distribution and extreme-value statistics along the denoising trajectory at steps 1000, 200, 101, 100, 50, and 1. The control signal is activated at step 100, where step 101 denotes the stage immediately before control activation and step 100 denotes the activation step. Overall, the trajectory shows how the generated distribution is gradually transformed from the initial Gaussian distribution toward the target data distribution, while extreme values are enhanced after control activation.



(a) Evolution of the extreme-value point proportion with the extreme-event control signal. The curve shows that E4GEN progressively increases the proportion of extreme-value points during denoising, illustrating the guidance effect of the proposed control signal. After the control signal is activated at $t = 100$, the extreme-value proportion rises rapidly, demonstrating the effectiveness and reliability of E4GEN in guiding extreme-event generation.

(b) Evolution of the extreme-value point proportion without the extreme-event control signal. The curve shows that the proportion of extreme-value points increases gradually during denoising, reflecting the model's inherent tendency to recover extreme values. However, without external extreme-event control, the extreme-value proportion remains consistently suppressed, indicating that existing methods struggle to follow the target extreme-event distribution.

Fig. 23: Comparison of the evolution of extreme-value point proportions during the denoising process, with and without the extreme-event control signal.

intermediate denoising states into explicit extreme-event semantics, which are then used by E-Control as control signals. Therefore, we examine whether E-Predictor produces accurate and semantically meaningful intermediate outputs, thereby validating the reliability of the predicted control signals for extreme-event generation.

Figure 24 provides qualitative examples of E-Predictor outputs on three representative datasets, namely SYN-DATA, LTST-ECG, and HH-POWER, with one time-series sample visualized for each dataset. In each example, the black curve denotes the ground-truth signal, red points indicate the ground-truth extreme-event regions, blue shaded areas represent the predicted extreme-event regions, and blue triangles mark the predicted peak positions. Overall, the results show that E-Predictor can effectively capture the presence of extreme events across different datasets. When an extreme event is successfully detected, the predicted event span and peak position are generally well aligned with the ground truth, indicating accurate recovery of the location and intensity semantics in EES. The predicted regions also preserve the main temporal extent of the event, reflecting meaningful temporal-shape semantics for subsequent control injection. Meanwhile, a small number of false negatives and false positives can still be observed, suggesting that E-Predictor is not error-free. Nevertheless, the overall alignment between predicted and ground-truth event semantics supports the reliability of the predicted EES as control signals for E-Control.

Table 5 further quantifies the intermediate prediction quality of E-Predictor across all six datasets. The Event-F1 scores range from 0.713 to 0.872, showing that E-Predictor can reliably identify extreme events in most cases. The Span-IoU values range from 0.572 to 0.774, indicating that the predicted event regions substantially overlap with the ground-truth event spans, although exact boundary recovery remains challenging for some datasets. For intensity semantics, Peak-MAE and Prom-MAE remain within moderate ranges of 0.096–0.248 and 0.121–0.296, respectively, suggesting that E-Predictor can reasonably estimate the peak strength and prominence of extreme events. For temporal-shape semantics, Shape-DTW ranges from 0.214 to 0.487, showing that the predicted semantics preserve the main temporal evolution of extreme events, while more irregular events still lead to larger shape deviations. Overall, these results indicate that E-Predictor provides sufficiently accurate and semantically meaningful EES predictions to serve as control signals for E-Control, which is also consistent with the qualitative examples in Figure 24.

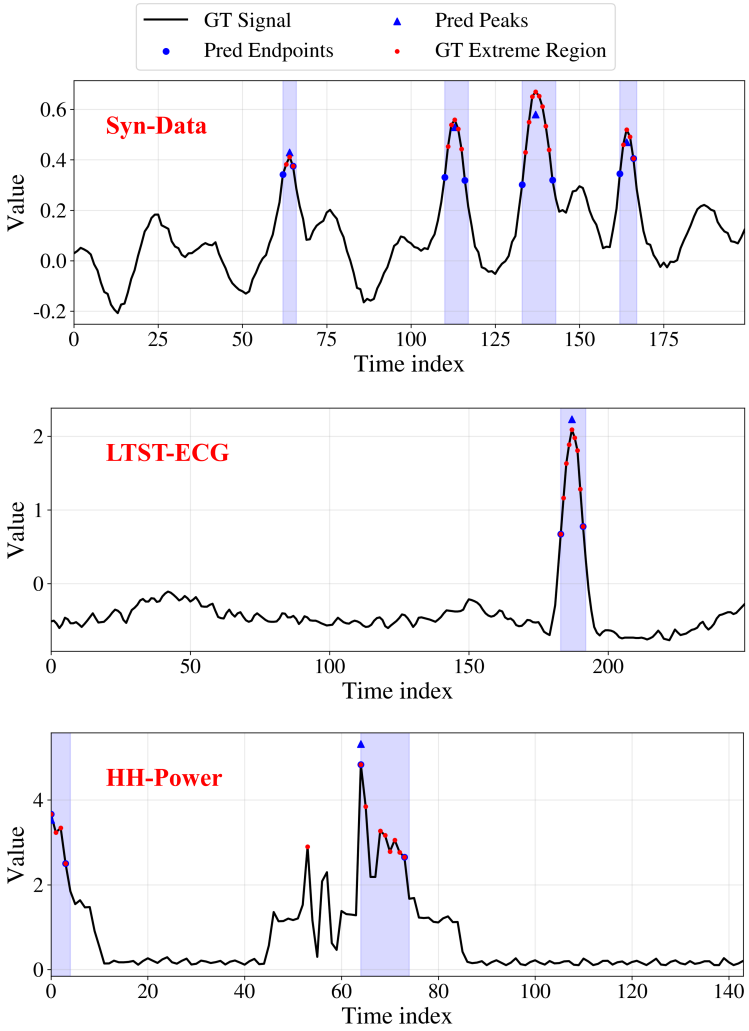


Fig. 24: Visualization of E-Predictor predictions for extreme-event semantics on SYN-DATA, LTST-ECG, and HH-POWER. For each dataset, one time-series sample is shown. The black curve denotes the ground-truth signal, red points indicate the ground-truth extreme-event regions, blue shaded areas represent the predicted extreme-event regions, and blue triangles mark the predicted peak positions. Overall, once E-Predictor identifies the presence of an extreme event, it can accurately localize the event region and estimate the peak position, although a small number of false negatives and false positives may still occur.

Table 5: Intermediate prediction quality of E-Predictor on extreme-event semantics.

Dataset	Event-F1 ↑	Span-IoU ↑	Peak-MAE ↓	Prom-MAE ↓	Shape-DTW ↓
SYN-DATA	0.872	0.774	0.096	0.136	0.214
WEA-TEMP	0.801	0.684	0.157	0.188	0.331
WEA-PREC	0.713	0.572	0.248	0.296	0.487
LTST-ECG	0.834	0.728	0.132	0.121	0.276
HH-POWER	0.768	0.641	0.181	0.221	0.379
PEMS-SF	0.742	0.618	0.204	0.247	0.423

L.6 Ablation Study

To answer RQ6, we evaluate the contribution of each key design (extreme-event semantic representation, E-Activator, and E-Predictor) to the final generation performance. The three ablation variants are defined as follows:

- **E4GEN**$_{\backslash \mathrm{Sem}}$: removes the extreme-event semantic representation. Instead of using the structured EES representation, this ablation variant uses the raw point sequence to represent each extreme event. When the event segment is longer than the fixed representation size, uniform sampling is applied to shorten it.
- **E4GEN**$_{\backslash \mathrm{Act}}$: removes E-Activator. Instead of dynamically selecting the dataset-specific Control Activation Step, this ablation variant disables step selection and keeps extreme-event enhancement active throughout the entire denoising process.
- **E4GEN**$_{\backslash \mathrm{Pred}}$: removes E-Predictor. Instead of predicting extreme-event semantics at the Control Activation Step, this ablation variant samples semantics directly from the empirical extreme-event semantic distribution of the training dataset.

Table 6 and Table 7 report the ablation results on SYN-DATA as a representative example. The full E4GEN achieves the best performance across all overall, downstream, and extreme-event metrics, showing that the three designs are jointly beneficial for extreme-aware generation. Among the ablation variants, E4GEN$_{\backslash \mathrm{Act}}$ suffers the largest degradation, indicating the importance of selecting an appropriate Control Activation Step rather than applying control throughout the entire denoising process. E4GEN$_{\backslash \mathrm{Sem}}$ and E4GEN$_{\backslash \mathrm{Pred}}$ perform better but still lag behind the full model, confirming that both structured EES representation and state-dependent semantic prediction are necessary for reliable control. Overall, the results validate the contribution of extreme-event semantic representation, E-Activator, and E-Predictor to the final generation quality.

Table 6: Overall Generation Fidelity and General Downstream Utility on SYN-DATA

Dataset	Type	Method	Wass.	KS	JS	MMD	ACD	Context_FID	Pred	Recon
Syn-Data	Ablation	E4GEN$_{\backslash \mathrm{Sem}}$	0.0264	0.1097	0.1698	0.1428	0.0784	1.9462	0.1694	0.0548
		E4GEN$_{\backslash \mathrm{Act}}$	0.0418	0.1586	0.2564	0.2247	0.1369	4.1265	0.2137	0.0864
		E4GEN$_{\backslash \mathrm{Pred}}$	0.0241	0.1016	0.1745	0.1247	0.0809	1.6428	0.1648	0.0569
	Ours	**E4GEN**	**0.0144**	**0.0568**	**0.1106**	**0.0447**	**0.0505**	**0.4501**	**0.1518**	**0.0368**

Table 7: Extreme-Event Generation Fidelity and Extreme-Oriented Downstream Utility on SYN-DATA

Dataset	Type	Method	EM-W1	EC-Diff	ED-W1	EP-W1	EWass.	EJS	EMMD	EPred	ERecon
Syn-Data	Ablation	E4GEN$_{\backslash \mathrm{Sem}}$	0.1284	3348	2.3567	0.1642	0.9147	0.4628	0.7042	0.4527	0.1038
		E4GEN$_{\backslash \mathrm{Act}}$	0.2487	5126	3.2864	0.2765	1.6842	0.6938	0.9725	0.6247	0.2148
		E4GEN$_{\backslash \mathrm{Pred}}$	0.1096	2976	2.4389	0.1425	0.7248	0.4875	0.6236	0.3864	0.1147
	Ours	**E4GEN**	**0.0245**	**1232**	**1.5383**	**0.0360**	**0.3451**	**0.2689**	**0.3615**	**0.3129**	**0.0871**

L.7 Sensitivity Analysis of Control-Related Hyperparameters

To answer RQ7, this part provides a sensitivity analysis of the key control-related hyperparameters in E4GEN. Specifically, we examine the effects of the control activation step t_{CA} and the alignment start step t_{AS} introduced in Section 4.2.2.

For t_{CA}, Since E-Activator defines a valid search region $\mathcal{R}_{\mathrm{CA}}$ for the control activation step t_{CA}, we focus on how to select t_{CA} within this region. In E4GEN, we selects t_{CA} through a lightweight validation search within $\mathcal{R}_{\mathrm{CA}}$. To examine whether E4GEN is sensitive to this selection, we compare the default strategy with two alternative strategies for choosing t_{CA} within $\mathcal{R}_{\mathrm{CA}}$:

- **E4GEN**$_{\mathrm{Grid}}$: uses fixed candidate steps that are uniformly spaced within $\mathcal{R}_{\mathrm{CA}}$, and reports the average performance over these candidates. This variant evaluates whether different fixed positions inside the control activation window lead to stable generation performance.

Table 8: Overall Generation Fidelity and General Downstream Utility under Different t_{CA} Selection Strategies on SYN-DATA

Dataset	Type	Method	Wass.	KS	JS	MMD	ACD	Context_FID	Pred	Recon
Syn-Data	Sensitivity	E4GEN$_{\mathrm{Grid}}$	0.0162	0.0619	0.1187	0.0508	0.0534	0.4936	0.1576	0.0395
		E4GEN$_{\mathrm{Rand}}$	0.0187	0.0684	0.1269	0.0583	0.0578	0.5724	0.1643	0.0431
	Ours	E4GEN	**0.0144**	**0.0568**	**0.1106**	**0.0447**	**0.0505**	**0.4501**	**0.1518**	**0.0368**

Table 9: Extreme-Event Generation Fidelity and Extreme-Oriented Downstream Utility under Different t_{CA} Selection Strategies on SYN-DATA

Dataset	Type	Method	EM-W1	EC-Diff	ED-W1	EP-W1	EWass.	EJS	EMMD	EPred	ERecon
Syn-Data	Sensitivity	E4GEN$_{\mathrm{Grid}}$	0.0318	1396	1.6824	0.0437	0.3895	0.2916	0.3982	0.3265	0.0948
		E4GEN$_{\mathrm{Rand}}$	0.0396	1584	1.8467	0.0521	0.4368	0.3194	0.4375	0.3417	0.1029
	Ours	E4GEN	**0.0245**	**1232**	**1.5383**	**0.0360**	**0.3451**	**0.2689**	**0.3615**	**0.3129**	**0.0871**

- **E4GEN$_{\mathrm{Rand}}$**: randomly samples t_{CA} from $\mathcal{R}_{\mathrm{CA}}$ during each sampling process, and reports the average performance over multiple random trials. This variant evaluates whether stochastic control activation within the identified window affects generation stability.

Tables 8 and 9 report the sensitivity analysis of t_{CA} based on different selection strategy. Both E4GEN$_{\mathrm{Grid}}$ and E4GEN$_{\mathrm{Rand}}$ maintain competitive performance, suggesting that E4GEN is not highly sensitive to the exact activation step once a valid control activation window is identified. E4GEN$_{\mathrm{Grid}}$ consistently outperforms E4GEN$_{\mathrm{Rand}}$, indicating that fixed activation steps provide more stable control than stochastic step selection. The default E4GEN achieves the best performance across all metrics, demonstrating the advantage of selecting a better fixed t_{CA} through lightweight validation search.

We further analyze the sensitivity of E4GEN to the alignment start step t_{AS}. Since t_{AS} determines the starting point of the short reverse trajectory used to construct data-conditioned training states, we focus on its distance from t_{CA}, denoted as $\Delta = |t_{\mathrm{AS}} - t_{\mathrm{CA}}|$. Specifically, we select ten candidate values of t_{AS} and progressively slide t_{CA} within the control activation window $\mathcal{R}_{\mathrm{CA}}$, thereby increasing the distance $\Delta = |t_{\mathrm{AS}} - t_{\mathrm{CA}}|$. For each pair of $(t_{\mathrm{AS}}, t_{\mathrm{CA}})$, we measure the distributional discrepancy between the resulting data-conditioned state $\mathbf{x}_d(t_{\mathrm{CA}})$ and the corresponding noise-initiated sampling

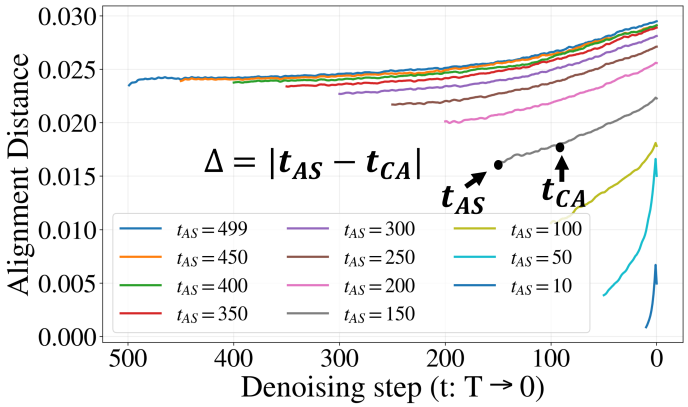


Fig. 25: Sensitivity analysis of E4GEN with respect to the alignment start step t_{AS}. For each candidate value of t_{AS}, we progressively vary t_{CA} within the search region $\mathcal{R}_{\mathrm{CA}}$ to increase the temporal offset $\Delta = |t_{\mathrm{AS}} - t_{\mathrm{CA}}|$. We measure the resulting alignment distance between the data-conditioned state distribution $\mathbf{x}_d(t_{\mathrm{CA}})$ and the corresponding noise-initiated sampling state distribution $\mathbf{x}_n(t_{\mathrm{CA}})$, where a smaller distance indicates better train–sample state alignment.

state $\mathbf{x}_n(t_{\mathrm{CA}})$. This analysis evaluates how t_{AS} affects training-sampling alignment and whether the alignment remains stable across different values of Δ.

As shown in Figure 25, smaller values of t_{AS}, which are closer to the sampling endpoint, generally lead to lower alignment distance between $\mathbf{x}_d(t_{\mathrm{CA}})$ and $\mathbf{x}_n(t_{\mathrm{CA}})$. However, the alignment distance also increases more rapidly with the offset $\Delta = |t_{\mathrm{AS}} - t_{\mathrm{CA}}|$ under these settings. This indicates that later alignment start step t_{AS} reduce the initial train–sample discrepancy, but are more sensitive to mismatches between $\mathbf{x}_d(t_{\mathrm{CA}})$ and $\mathbf{x}_n(t_{\mathrm{CA}})$. Since t_{CA} is usually located at a relatively small diffusion step within the control activation window, this sensitivity analysis further supports the E4GEN design choice of selecting t_{AS} close to t_{CA}, which helps reduce train–sample state discrepancy and preserve reliable state alignment.

L.8 Efficiency Analysis

This section analyzes the computational efficiency of E4GEN in terms of training time and per-sample sampling time. As shown in Table 10, E4GEN is computationally efficient across all six datasets. The training time ranges from 0.09 to 0.12 seconds per epoch, while the sampling time ranges from 0.09 to 0.24 seconds per sample. For most datasets, E4GEN requires only about 0.09–0.10 seconds per training epoch and 0.13 seconds or less to generate one sample. The efficient training and sampling speeds demonstrate that the proposed control-based design is both practical and scalable, making E4GEN an effective and efficient framework for extreme-aware time-series generation.

Table 10: Training Time and Per-Sample Sampling Time Across Datasets of E4GEN.

Dataset	**Training Time** (s / epoch)	**Sampling Time** (s / sample)
SYN-DATA	0.10	0.19
WEA-TEMP	0.09	0.09
WEA-PREC	0.10	0.13
LTST-ECG	0.12	0.24
HH-POWER	0.09	0.13
PEMS-SF	0.09	0.13

L.9 Controllable Extreme-Event Generation with User-Specified Semantics

For a time series of length 200, we manually define three user-specified extreme-event configurations as external control signals:

- **Configuration 1:** two extreme events, where Event 1 spans time indices 66–73 with a target peak intensity of 0.78, and Event 2 spans time indices 115–130 with a target peak intensity of 0.76.
- **Configuration 2:** two extreme events, where Event 1 spans time indices 10–30 with a target peak intensity of 0.47, and Event 2 spans time indices 150–180 with a target peak intensity of 0.50.
- **Configuration 3:** three extreme events, where Event 1 spans time indices 52–58 with a target peak intensity of 0.52, Event 2 spans time indices 92–101 with a target peak intensity of 0.60, and Event 3 spans time indices 160–163 with a target peak intensity of 0.41.

As shown in Figure 26, E4GEN successfully generates time series that follow the three user-specified extreme-event configurations. Across all configurations, the generated samples exhibit clustered extreme values around the prescribed event locations, and the aggregated event regions are generally well aligned with the specified temporal spans. The generated peak intensities also remain close to the target intensity levels, indicating that E4GEN can effectively translate external semantic control signals into localized extreme-event patterns. These results demonstrate that E4GEN supports controllable extreme-event generation beyond self-predicted semantics, enabling users to guide both the occurrence and intensity of extreme events.

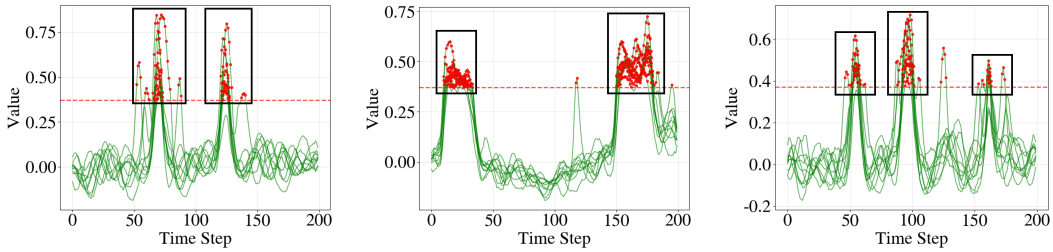


Fig. 26: Controllable extreme-event generation results under three user-specified semantic configurations on SYN-DATA. For each configuration, 20 generated time-series samples are visualized to assess whether E4GEN follows the specified event-level control signals. The red dashed line denotes the extreme threshold, red points indicate extreme values, and black squares highlight the aggregated extreme-event regions.

M Additional Experimental Results on Generation Fidelity and Utility

In this part, we present additional experimental results on the remaining five datasets, including SYN-DATA, WEA-PREC, LTST-ECG, HH-POWER, and PEMS-SF. We also evaluate each dataset from three perspectives: overall generation fidelity, extreme-event generation fidelity, and downstream utility. In these tables, ∞ indicates that the corresponding metric cannot be computed due to an insufficient number of generated extreme events.

Table 11: Overall Generation Fidelity and General Downstream Utility on SYN-DATA

Dataset	Type	Method	Wass.	KS	JS	MMD	ACD	Context-FID	Pred	Recon
SYN-DATA	**GAN**	TimeGAN (NeurIPS'19) [1]	0.0267	0.1038	0.1995	0.1018	0.1592	2.0077	0.2258	0.1547
	VAE	TimeVAE (ArXiv'21) [11]	0.0650	0.3359	0.4424	0.2340	0.3998	16.284	0.2456	0.0764
		koVAE (ICLR'24) [12]	0.0425	0.1425	0.2669	0.2086	0.0779	6.7766	0.1746	0.0422
	Flow	F-Flow (ICLR'21) [13]	0.1395	0.6424	0.5579	0.6539	0.0819	17.130	0.1833	0.0514
	Diffusion	DiffWave (ICLR'21) [15]	0.0557	0.2348	0.3767	0.2191	0.1630	10.798	0.2264	0.0706
		Diffusion-TS (ICLR'24) [4]	0.0236	0.1032	0.1719	0.3052	0.0920	1.9910	0.1641	0.0557
	LLM	SDForger (NeurIPS'25) [5]	0.0203	0.0628	0.1369	0.0708	**0.0427**	1.2159	0.1548	0.0525
	Extreme-Aware	FIDE (NeurIPS'24) [3]	0.0309	0.1279	**0.1075**	0.1558	0.1614	4.4028	0.6670	0.8808
		HeavyDiff (ICLR'25) [23]	0.0231	0.0971	0.1718	0.2470	0.0784	1.5271	0.1609	0.0675
	Ours	**E4GEN (Ours)**	**0.0144**	**0.0568**	0.1106	**0.0447**	0.0505	**0.4501**	**0.1518**	**0.0368**

Table 12: Extreme-Event Generation Fidelity and Extreme-Oriented Downstream Utility on SYN-DATA

Dataset	Type	Method	EM-W1	EC-Diff	ED-W1	EP-W1	EWass.	EJS	EMMD	EPred	ERecon
SYN-DATA	**GAN**	TimeGAN [1]	0.4755	6218	3.1726	0.4779	∞	∞	∞	∞	∞
	VAE	TimeVAE [11]	0.4378	5820	3.2847	0.4619	2.1274	0.8318	1.2573	0.5816	2.3121
		koVAE [12]	0.4334	6439	3.2525	0.4570	2.1200	0.8256	0.7437	0.5727	0.1014
	Flow	F-Flow [13]	0.3511	5513	2.5745	0.3731	1.8070	0.7059	1.1253	0.5055	0.1012
	Diffusion	DiffWave [15]	0.4381	6463	3.2765	0.4618	∞	∞	∞	∞	∞
		Diffusion-TS [4]	0.1139	3061	3.0921	0.1696	0.6257	0.4534	0.5606	0.3017	0.0942
	LLM	SDForger [5]	0.3661	5212	2.5214	0.3872	1.7097	0.6607	0.9180	0.4600	0.1224
	Extreme-Aware	FIDE [3]	0.1499	3864	2.5898	0.1678	9.7145	0.5724	0.7376	0.4647	10.997
		HeavyDiff [23]	0.0784	2200	2.9818	0.1216	0.5113	0.4026	0.5528	**0.2709**	0.0978
	Ours	**E4GEN (Ours)**	**0.0245**	**1232**	**1.5383**	**0.0360**	**0.3451**	**0.2689**	**0.3615**	0.3129	**0.0871**

Table 13: Overall Generation Fidelity and General Downstream Utility on WEA-PREC Dataset

Dataset	Type	Method	Wass.	KS	JS	MMD	ACD	Context-FID	Pred	Recon
WEA-PREC	GAN	TimeGAN (NeurIPS'19) [1]	484.09	0.7788	0.7501	0.2408	242.06	219.36	0.5706	1.0104
	VAE	TimeVAE (ArXiv'21) [11]	423.34	0.7357	0.6436	0.2279	211.75	182.57	0.6858	0.9595
		koVAE (ICLR'24) [12]	463.50	0.8015	0.7459	0.2420	231.83	210.14	0.9813	2.1157
	Flow	F-Flow (ICLR'21) [13]	327.02	0.4024	0.2930	0.3219	163.52	251.63	0.4624	1.1018
	Diffusion	DiffWave (ICLR'21) [15]	271.15	0.5122	0.2098	0.2824	135.58	174.39	0.4119	0.9321
		Diffusion-TS (ICLR'24) [4]	172.16	0.5574	0.1558	**0.0876**	86.088	301.04	0.4053	1.4584
	LLM	SDForger (NeurIPS'25) [5]	611.65	0.3527	0.4921	0.1592	305.84	545.03	0.5214	0.8450
	Extreme-Aware	FIDE (NeurIPS'24) [3]	137.61	0.4410	0.2329	0.0898	68.818	311.95	0.4466	0.8916
		HeavyDiff (ICLR'25) [23]	158.32	0.5673	0.2240	0.1306	79.172	317.88	0.4117	0.8670
	Ours	**E4GEN (Ours)**	**133.20**	**0.2114**	**0.1425**	0.0886	**66.607**	**130.68**	**0.4033**	**0.7913**

Table 14: Extreme-Event Generation Fidelity and Extreme-Oriented Downstream Utility on WEA-PREC Dataset

Dataset	Type	Method	EM-W1	EC-Diff	ED-W1	EP-W1	EWass.	EJS	EMMD	EPred	ERecon
WEA-PREC	GAN	TimeGAN [1]	2.5847	9661	1.3385	2.7133	∞	∞	∞	∞	∞
	VAE	TimeVAE [11]	2.5234	8532	1.3185	2.7128	16.450	0.8323	5.4374	1.6613	2.2534
		koVAE [12]	2.5088	9427	1.3157	2.6977	16.443	0.8311	5.4416	1.7204	5.1512
	Flow	F-Flow [13]	2.5163	9434	1.3223	2.7052	∞	∞	∞	∞	∞
	Diffusion	DiffWave [15]	1.7680	7834	0.5415	1.9528	13.436	0.7506	4.3148	1.5536	2.3741
		Diffusion-TS [4]	1.2734	4331	0.1525	1.3156	8.1159	0.4793	2.0414	2.3124	3.0427
	LLM	SDForger [5]	0.7438	13462	0.4953	0.7883	13.625	0.5380	2.1673	1.3387	3.9173
	Extreme-Aware	FIDE [3]	1.1417	**3130**	0.1693	1.2455	48.397	**0.2672**	1.4808	**1.2212**	17.326
		HeavyDiff [23]	1.3451	3990	0.1436	1.3779	6.9423	0.4110	**1.1951**	1.4888	3.1175
	Ours	**E4GEN (Ours)**	**0.4286**	3619	**0.1004**	**0.4341**	**6.4997**	0.4056	1.7371	1.3368	**2.2081**

Table 15: Overall Generation Fidelity and General Downstream Utility on HH-POWER Dataset

Dataset	Type	Method	Wass.	KS	JS	MMD	ACD	Context-FID	Pred	Recon
HH-POWER	GAN	TimeGAN (NeurIPS'19) [1]	0.1904	0.1721	0.2317	0.2122	0.2285	1.6236	0.3206	0.7460
	VAE	TimeVAE (ArXiv'21) [11]	0.3103	0.2509	0.2976	0.1898	0.2678	4.3785	0.3498	0.2720
		koVAE (ICLR'24) [12]	0.3206	0.1882	0.3334	0.2714	0.2572	2.8463	0.3294	0.2671
	Flow	F-Flow (ICLR'21) [13]	0.9752	0.7554	0.6198	0.6760	0.4951	22.1413	0.5614	0.4091
	Diffusion	DiffWave (ICLR'21) [15]	0.4087	0.2659	0.3705	0.3140	0.2245	4.6475	0.3439	0.2495
		Diffusion-TS (ICLR'24) [4]	0.1737	0.1584	**0.1350**	0.2473	0.1273	1.0810	0.3023	0.2535
	LLM	SDForger (NeurIPS'25) [5]	0.2182	0.1636	0.3021	0.1727	0.1463	136.4964	0.3385	0.3391
	Extreme-Aware	FIDE (NeurIPS'24) [3]	0.1769	0.1540	0.2170	0.1414	0.1944	1.8600	0.5878	0.4356
		HeavyDiff (ICLR'25) [23]	0.1856	0.1420	0.1872	0.2218	0.1286	**0.9149**	0.3123	0.2328
	Ours	**E4GEN (Ours)**	**0.1698**	**0.1315**	0.1741	**0.1375**	**0.1246**	1.0035	**0.2999**	**0.2278**

Table 16: Extreme-Event Generation Fidelity and Extreme-Oriented Downstream Utility on HH-POWER Dataset

Dataset	Type	Method	EM-W1	EC-Diff	ED-W1	EP-W1	EWass.	EJS	EMMD	EPred	ERecon
HH-POWER	GAN	TimeGAN [1]	1.0383	3817	10.9922	1.2187	3.0033	0.5337	5.3360	0.8896	2.1582
	VAE	TimeVAE [11]	2.2889	10623	2.8694	2.5049	8.6922	0.6534	6.0586	0.9083	0.8072
		koVAE [12]	0.8455	13305	1.7526	1.0439	10.0234	0.6979	4.2831	0.6331	1.0922
	Flow	F-Flow [13]	2.7518	15812	2.6638	3.9851	∞	∞	∞	∞	∞
	Diffusion	DiffWave [15]	2.9345	15783	2.7963	3.1761	∞	∞	∞	∞	∞
		Diffusion-TS [4]	0.8144	4505	1.8059	1.0288	2.2833	0.3284	**0.2816**	0.5794	0.9291
	LLM	SDForger [5]	0.5677	1383	4.1994	0.6450	1.4516	0.2082	4.1405	0.6264	1.0743
	Extreme-Aware	FIDE [3]	1.2496	3902	1.7325	0.4221	3.0098	0.3105	2.8839	0.6496	11.6615
		HeavyDiff [23]	0.8059	3943	1.1517	0.9038	2.0354	0.3142	0.5353	0.5600	0.7368
	Ours	**E4GEN (Ours)**	**0.0862**	**886**	**0.7834**	**0.1606**	**0.6672**	**0.1458**	1.4301	**0.5239**	**0.7284**

Table 17: Overall Generation Fidelity and General Downstream Utility on LTST-ECG Dataset

Dataset	Type	Method	Wass.	KS	JS	MMD	ACD	Context-FID	Pred	Recon
LTST-ECG	GAN	TimeGAN (NeurIPS'19) [1]	0.2190	0.2115	0.2985	0.1540	0.1988	3.5454	0.1660	0.0521
	VAE	TimeVAE (ArXiv'21) [11]	0.2526	0.2971	0.3580	0.2177	0.3487	16.310	0.1885	0.0580
		koVAE (ICLR'24) [12]	0.2531	0.2790	0.3423	0.2952	0.3276	10.033	0.1126	0.0217
	Flow	F-Flow (ICLR'21) [13]	0.7239	0.8665	0.7669	0.6604	0.3803	31.992	0.3033	0.0501
	Diffusion	DiffWave (ICLR'21) [15]	0.2864	0.3803	0.4826	0.2535	0.4048	17.473	0.1877	0.0601
		Diffusion-TS (ICLR'24) [4]	0.0538	0.0934	0.1064	0.2368	0.0557	1.1740	0.0945	0.0171
	LLM	SDForger (NeurIPS'25) [5]	0.2390	0.1830	0.2973	0.1137	0.1794	3.5274	0.1008	0.0192
	Extreme-Aware	FIDE (NeurIPS'24) [3]	0.2624	0.2200	0.1659	0.2371	0.2434	69.932	0.5078	0.6218
		HeavyDiff (ICLR'25) [23]	0.0443	0.0700	0.1144	0.2054	0.0599	1.1824	0.0708	**0.0159**
	Ours	**E4GEN (Ours)**	**0.0352**	**0.0383**	**0.0866**	**0.1067**	**0.0357**	**0.4920**	**0.0703**	0.0171

Table 18: Extreme-Event Generation Fidelity and Extreme-Oriented Downstream Utility on LTST-ECG Dataset

Dataset	Type	Method	EM-W1	EC-Diff	ED-W1	EP-W1	EWass.	EJS	EMMD	EPred	ERecon
LTST-ECG	GAN	TimeGAN [1]	0.6881	33741	7.4728	1.2060	3.0622	0.6407	2.2042	0.7011	0.1413
	VAE	TimeVAE [11]	1.3423	20290	10.199	1.9551	3.7131	0.7479	2.8147	0.7452	0.5056
		koVAE [12]	1.0624	22875	9.5807	1.5670	3.9207	0.7441	2.0463	0.4008	0.0604
	Flow	F-Flow [13]	0.9259	24703	9.9093	1.4716	4.5078	0.7858	2.3123	0.9146	0.2565
	Diffusion	DiffWave [15]	1.3480	24996	11.475	1.9541	4.3085	0.8322	2.7932	∞	0.5797
		Diffusion-TS [4]	0.0936	4960	3.1839	**0.0444**	0.9857	0.3071	0.3888	0.1893	0.0693
	LLM	SDForger [5]	0.6669	22481	3.9868	1.1220	3.4591	0.5351	1.4759	0.2774	**0.0460**
	Extreme-Aware	FIDE [3]	0.6834	16922	10.406	1.0252	75.5862	0.6080	1.9742	0.9475	3.4025
		HeavyDiff [23]	0.0331	4754	3.1284	0.0988	0.9545	**0.2783**	0.3306	0.1885	0.0703
	Ours	**E4GEN (Ours)**	**0.0328**	**4633**	**2.4458**	0.0561	**0.8911**	0.2823	**0.2650**	**0.1674**	0.0558

Table 19: Overall Generation Fidelity and General Downstream Utility on PEMS-SF Dataset

Dataset	Type	Method	Wass.	KS	JS	MMD	ACD	Context-FID	Pred	Recon
PEMS-SF	**GAN**	TimeGAN (NeurIPS'19) [1]	0.0097	0.1139	0.1910	0.3917	0.0793	0.9527	0.2490	0.2349
	VAE	TimeVAE (ArXiv'21) [11]	0.0061	0.0810	0.1405	0.2063	0.0355	0.5406	0.2255	0.1386
		koVAE (ICLR'24) [12]	0.0106	0.1227	0.2120	0.3148	0.0901	0.6285	0.2159	0.1437
	Flow	F-Flow (ICLR'21) [13]	0.0272	0.2830	0.2540	0.3814	0.0254	0.4721	0.1985	0.1212
	Diffusion	DiffWave (ICLR'21) [15]	0.0194	0.2237	0.3908	0.4820	0.0585	2.2068	0.2550	0.1681
		Diffusion-TS (ICLR'24) [4]	0.0032	0.0749	0.0858	0.1138	0.0237	0.0758	0.1879	0.1218
	LLM	SDForger (NeurIPS'25) [5]	0.0049	0.0744	0.1334	0.1045	0.0285	0.2002	**0.1848**	0.1252
	Extreme-Aware	FIDE (NeurIPS'24) [3]	0.0203	0.2319	0.1561	0.3026	0.0612	0.7426	0.2444	0.1729
		HeavyDiff (ICLR'25) [23]	0.0051	0.0940	0.0785	0.1328	**0.0189**	0.0842	0.1942	0.1202
	Ours	**E4GEN (Ours)**	**0.0031**	**0.0293**	**0.0554**	**0.0969**	0.0382	**0.0755**	0.1862	**0.1167**

Table 20: Extreme-Event Generation Fidelity and Extreme-Oriented Downstream Utility on PEMS-SF Dataset

Dataset	Type	Method	EM-W1	EC-Diff	ED-W1	EP-W1	EWass.	EJS	EMMD	EPred	ERecon
PEMS-SF	**GAN**	TimeGAN [1]	0.0649	11176	3.3566	0.0732	0.6028	0.5959	0.3982	0.8045	0.9457
	VAE	TimeVAE [11]	0.0321	5517	1.6787	0.0400	0.3642	0.3446	0.2809	0.6911	0.9729
		koVAE [12]	0.0710	16200	2.4943	0.0785	0.8908	0.7015	0.4640	0.6735	0.8006
	Flow	F-Flow [13]	0.0323	42717	9.5380	0.0378	1.1185	0.5896	0.1772	0.4918	0.6230
	Diffusion	DiffWave [15]	0.1012	19121	4.1817	0.1104	∞	∞	∞	∞	∞
		Diffusion-TS [4]	0.0185	2100	2.5681	0.0200	0.3262	0.4283	0.2523	0.4730	0.6254
	LLM	SDForger [5]	0.0290	5365	5.9978	0.0287	0.3428	**0.3364**	0.2174	0.4759	0.6466
	Extreme-Aware	FIDE [3]	0.0242	25025	2.0519	0.0279	4.1184	0.5097	0.2424	0.5760	0.8926
		HeavyDiff [23]	0.0217	1767	2.7466	0.0373	**0.2817**	0.3951	0.2133	0.4791	0.6213
	Ours	**E4GEN (Ours)**	**0.0138**	**1473**	**0.7813**	**0.0154**	0.2980	0.3400	**0.1505**	**0.4667**	**0.6046**

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 main claims are stated in the Abstract and Section 1, where the paper presents E4GEN as an event-level explainable diffusion framework with three core designs for when, what, and how to control extreme-event generation. These claims are supported by the method description in Section 4 and the evaluation scope in Section 5, which reports experiments on six datasets with 17 metrics against nine baselines across overall fidelity, extreme-event fidelity, and downstream utility.

   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: The paper includes a limitations discussion in the Appendix C. It discusses the scope of the empirical evaluation, noting that the experiments are conducted on six representative datasets across several domains, and that broader validation on additional domains and longer time-series settings would further support generality. It also discusses the dependence on domain-specific extreme-event thresholds and the additional computational cost introduced by semantic prediction and control injection modules.

   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: The theoretical results justifying the DCT-NIS training protocol (Lemma 1 and Lemma 2) and their underlying assumptions are explicitly stated in Section 4.2. The complete and rigorous proofs for these lemmas, alongside supporting empirical validations, are provided in Appendix J.

   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 provides the main information needed to reproduce the experimental results, and our code is open-sourced to support reproducibility. Specifically, we provide the implementation code for E4GEN and all baseline methods on all datasets, together with detailed configuration files that record the parameters used for each experiment. Section 4 describes the E4GEN architecture and control mechanism, while Section 5 specifies the datasets, baselines, metrics, and train-on-generated/test-on-original evaluation protocol. Additional reproducibility details are provided in the appendices, including dataset descriptions in Appendix L.1, metric definitions in Appendix L.2, baseline implementation details in Appendix L.3, and full experimental results in Appendix M.

   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: We provide a code repository at `https://github.com/LinJiang18/E4GEN` to support reproducibility. The repository includes the implementation of E4GEN, data processing scripts, training and evaluation scripts, and instructions for reproducing the main experimental results. It also provides guidance for accessing and preparing the datasets used in the experiments, together with scripts for running the proposed method and baseline comparisons. These materials are intended to reproduce the main results reported across the six datasets, nine baselines, and seventeen evaluation metrics considered in the paper.

   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: Section 5 specifies key training and sampling details of E4GEN, including three core components:E-Activator, E-Predictor and E-Control. Section 5 further describes

the experimental protocol, including the evaluated datasets, baselines, metrics, and train-on-generated/test-on-original downstream evaluation setting, with additional implementation and metric details provided in the appendices.

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: [No]

Justification: The paper reports mainly point-estimate results across multiple datasets, baselines, and evaluation metrics, and does not include error bars, confidence intervals, or statistical significance tests in the tables or figures. For our proposed method, the reported results are averaged over 5 independent runs with different random seeds, which helps reduce variance and demonstrates the robustness of the model across stochastic training conditions. However, we do not additionally report standard deviation or confidence intervals due to space limitations in the main tables and figures.

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: [Yes]

Justification: All experiments were conducted on an AWS EC2 instance running Amazon Linux 2023, equipped with an NVIDIA A10G GPU with 24GB GPU memory, an AMD EPYC 7R32 CPU with 8 vCPUs, and 30GB system memory. The experimental environment used CUDA 12.8 and NVIDIA driver version 570.86.15. The storage configuration included a 200GB NVMe system disk and an additional 419GB NVMe disk for datasets and checkpoints. The paper additionally reports training and inference efficiency metrics, including GPU memory usage, training time, inference latency, and parameter counts for all compared

methods. No additional large-scale preliminary experiments requiring substantially more compute resources than the reported experiments were conducted.

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 authors have reviewed the NeurIPS Code of Ethics and believe that the research conforms to it. The paper studies time-series generation using existing datasets and reports the experimental scope in Section 5; it does not involve new human-subject data collection, direct interaction with participants, or intended applications involving surveillance, deception, or harmful deployment.

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: [Yes]

Justification: We discuss both the positive societal applications of extreme-event time-series generation (e.g., in healthcare and climate resilience) and the potential negative impacts (e.g., malicious data spoofing and risk of over-reliance) in Appendix D.

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 pretrained model, image generator, or scraped dataset requiring special safeguards. The experiments in Section 5 are conducted on existing time-series datasets across climate, healthcare, energy, and transportation domains, and they pose no such risks.

    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: [Yes]

    Justification: All existing assets, including datasets and baseline codebases, are properly credited in the main text and references. Additionally, we provide a dedicated section in the Appendix that explicitly details the sources, URLs, and respective licenses or terms of use for each utilized asset.

    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 a code package including all codes and implementation details of the proposed E4GEN.

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 paper does not involve crowdsourcing experiments or research with human subjects. The experiments in Section 5 are conducted on existing time-series datasets across climate, healthcare, energy, and transportation domains, so participant instructions, screenshots, and compensation details are not applicable.

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: The paper does not involve crowdsourcing or direct research with human subjects, so IRB approval and participant-risk disclosure are not applicable. The experiments in Section 5 are conducted on existing time-series datasets rather than newly collected participant data.

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 development does not use LLMs as an important, original, or non-standard component. E4GEN is described in Section 4 as a diffusion-based framework with E-Activator, E-Predictor, and E-Control, while the LLM-based method SDForger appears only as a baseline in Section 5. Any LLM use for writing, editing, or formatting does not affect the core methodology, scientific rigor, or originality of the research.

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.