arXiv:2610.17407v2 [cs.LG] 14 Oct 2026

CROSS-VIEW TIME-FREQUENCY DIFFUSION FOR EXTREME-AWARE TIME-SERIES GENERATION

Lin Jiang
Department of Computer Science
Florida State University
Tallahassee, FL 32306, USA
lin.jiang@fsu.edu

Rongchao Xu
Department of Computer Science
Florida State University
Tallahassee, FL 32306, USA
rx21a@fsu.edu

Guang Wang*
Department of Computer Science
Florida State University
Tallahassee, FL 32306, USA
guang@cs.fsu.edu

ABSTRACT

Generating realistic time-series requires faithfully reproducing rare but high-impact extreme events. Existing approaches often rely on specialized representations or extreme-focused training objectives, which may restrict the diversity of captured extremes or compromise overall distributional fidelity. We propose **Cross-View Time-Frequency Diffusion (CrossTF)**, a framework that exploits discrepancies between time- and frequency-view denoisers for extreme-aware time-series generation. Our key empirical finding is *extreme-sensitive cross-view asymmetry*: moving along the direction connecting the two views' clean-sample estimates substantially changes extreme fidelity while having limited effects on overall generation quality. We further identify a late-stage *cross-view window* in which this discrepancy stabilizes and cross-view adjustments have a particularly strong effect on extreme fidelity. These findings motivate a staged optimization strategy in CrossTF: after training and freezing both denoisers, it only needs to learn step-specific fusion parameters within this window using an Extreme Value Theory-based objective under an energy-distance constraint. This formulation supports interpolation between and extrapolation beyond the view-specific estimates while limiting degradation in overall generation quality. Extensive experiments on seven real-world datasets against 11 baselines show that CrossTF improves extreme fidelity by 26.1% and overall performance by 7.4% relative to the strongest baselines. These results highlight cross-view discrepancy as an effective basis for low-dimensional optimization in extreme-aware time-series generation.

1 INTRODUCTION

Time-series generation has emerged as an important research direction (Yuan & Qiao, 2024; Naiman et al., 2024a;b; Rousseau et al., 2025) because it can support a wide range of applications, including data augmentation (Lou et al., 2023), system simulation (Nikitin et al., 2024), and stress testing (Rizzato et al., 2022). Within this field, extreme-aware time-series generation has attracted increasing attention (Galib et al., 2024; Pandey et al., 2025), driven by the need to faithfully reproduce high-impact events such as severe heatwaves, intense rainfall, and abrupt demand surges (Gu et al., 2025). However, these events are often rare and temporally localized, making their distinctive patterns particularly susceptible to smoothing or distortion during generation (Jiang et al., 2026).

Existing approaches to improving extreme-aware time-series generation broadly fall into two categories. **Representation-based methods** leverage specialized representations, such as high-frequency components in FIDE (Galib et al., 2024) and time-domain shapelets in E4GEN (Jiang

*Corresponding author.

et al., 2026). While effective for certain patterns, these predefined representations introduce inductive biases that may limit their ability to capture the diversity of extreme events. **Objective-based methods**, such as ExGAN (Bhatia et al., 2021) and Tail-GAN (Cont et al., 2026), emphasize extremes through customized training objectives. However, explicitly prioritizing sparse extremes may divert optimization from modeling the full data distribution, potentially reducing overall generation fidelity. This trade-off motivates an alternative optimization strategy that improves the generation fidelity of extremes without sacrificing the overall generation fidelity.

To this end, we investigate whether discrepancies between time- and frequency-domain generative models can guide optimization toward better extreme-aware generation (Kollovieh et al., 2023; Crabbé et al., 2024). Although these models operate on representations that are information-equivalent under an invertible Fourier transform (Shatkay, 1995), they encode extremes through distinct structures. An abrupt peak is localized in time, while its spectral representation distributes this structure across the amplitudes and phases of multiple frequency components, whose constructive superposition produces the peak, as illustrated in Figure 1. These structural differences may give the time-view and frequency-view complementary strengths in capturing extremes, suggesting that their **cross-view discrepancy** may encode informative signals beyond any single view. This prompts a fundamental question: *Can cross-view discrepancy define an optimization direction that improves extreme modeling while largely preserving overall generation quality?*

To investigate this question, we develop **Cross-View Time-Frequency Diffusion (CrossTF)**, a framework that characterizes cross-view discrepancy and exploits it for extreme-aware time-series generation. We present the framework through three key questions.

How is the cross-view direction defined? At each denoising step along a shared diffusion trajectory, CrossTF constructs two views of the same intermediate sample using delay embedding (Packard et al., 1980) for the time view and STFT (Allen & Rabiner, 1977) for the frequency view. Each view-specific denoiser produces a clean-sample estimate that is then mapped back to the time domain. We define the vector from the frequency-view estimate to the time-view estimate as the **cross-view direction**, as in Figure 2. A step-specific fusion parameter α_i controls movement along this direction, with $\alpha_i = 0$ and $\alpha_i = 1$ denoting the frequency- and time-view estimates.

Why and when is this direction useful? We probe the cross-view direction by varying α_i between the pure frequency view and time view, revealing a pronounced asymmetry: *the denoising loss remains nearly flat, while metrics of extreme fidelity change substantially*. Across seven datasets, the median relative variation of the extreme-fidelity metric is 16.74× that of the denoising loss and 11.56× that of the overall-quality metric. This asymmetry creates an opportunity to improve extreme fidelity with limited impact on overall generation quality.

We further examine how the cross-view discrepancy evolves during denoising. In the early stages, the discrepancy is largely driven by noise and does not yet reliably reflect the systematic differences between the two views. As denoising progresses, the discrepancy stabilizes as the influence of noise diminishes, increasingly reflecting view-specific differences. Interestingly, the onset of this stable stage is highly consistent across the evaluated datasets. We define this late-stage interval as the **cross-view window**. Further experiments show that optimization within this window has a substantially stronger effect on extreme fidelity than optimization outside it. These motivate optimizing extreme fidelity along the cross-view direction specifically within this window.

How do we optimize along this direction? CrossTF uses a staged strategy that decouples distribution learning from extreme-focused optimization. We first train the time-view and frequency-view denoisers using the standard MSE-based denoising objective and then freeze their parameters. Then,

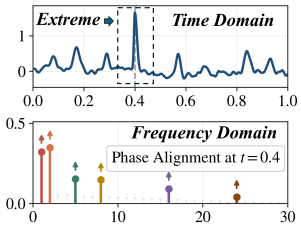


Fig. 1: Extreme Representations

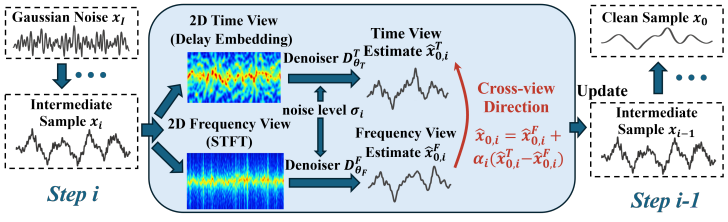


Fig. 2: Pipeline of CrossTF

a lightweight constrained cross-view optimizer learns the fusion parameters α_i within the cross-view window using an extreme-focused objective inspired by Extreme Value Theory (EVT) (Coles, 2001). Notably, CrossTF allows α_i to take values outside $[0, 1]$, enabling extrapolation beyond the two view-specific estimates. This flexibility is particularly useful when both views overestimate or both underestimate the extremes, as interpolation alone may not provide a sufficient correction.

The key contributions of this paper are summarized as follows:

(i) **Conceptually**, we identify a time-frequency cross-view optimization direction for extreme-aware modeling to which the standard denoising loss is largely insensitive. Along this cross-view direction, we observe a pronounced asymmetry: the denoising loss remains nearly flat, while metrics of extreme fidelity change substantially. This finding reveals an underexplored degree of freedom for improving extreme-aware time-series generation while largely preserving overall generation quality.

(ii) **Technically**, we propose CrossTF, a diffusion framework that turns the time–frequency discrepancy into an optimizable cross-view direction. CrossTF identifies a **cross-view window** and uses a constrained EVT-based optimizer to learn step-specific movement along the cross-view direction, supporting both interpolation and extrapolation while keeping the pretrained denoisers frozen.

(iii) **Experimentally**, we evaluate CrossTF on seven real-world datasets against 11 representative baselines using 10 metrics. Across all 770 (dataset, metric, baseline) comparisons, CrossTF wins 730 (94.8%). Relative to the strongest baseline on each dataset, it improves extreme fidelity by 26.1% and overall performance by 7.4% on average. These results demonstrate the effectiveness and broad applicability of CrossTF. Our code and resources are available at **Anonymous Repository**.

2 PRELIMINARIES

2.1 EXTREMES IN TIME SERIES

We first examine how extremes manifest in time series. Across different scenarios, extremes take diverse forms, such as impulse spikes, sustained high values, and abrupt drops, with varying durations and local structures, as shown in Figure 8 in Appendix C.1. Such diversity makes it challenging to characterize extremes using a single event representation. The Peaks-Over-Threshold (POT) framework in Extreme Value Theory (EVT) provides a general approach to identifying extremes as exceedances above a high threshold. However, this pointwise definition primarily captures extreme magnitudes and overlooks extreme changes between consecutive observations. To account for both, we characterize extremes along two complementary dimensions: **extreme magnitude** and **extreme change**, as illustrated in Figure 3.

Given a time series $\mathbf{x} = \{x_t\}_{t=1}^{T}$, we define normalized scores for the two dimensions:

$$s_t^{\text{mag}} = |x_t - \text{median}(\mathbf{x})|/\text{std}(\mathbf{x}), \qquad s_t^{\text{chg}} = |x_t - x_{t-1}|/\text{std}(\mathbf{x}). \tag{1}$$

The former measures deviations from the sequence baseline, capturing both unusually high and unusually low values, while the latter measures abrupt temporal transitions that magnitude alone may not capture. For each score type, we identify extremes as exceedances above an upper-tail threshold estimated from the real-data scores, using the 95th percentile as a representative setting. These two complementary dimensions guide extreme-aware optimization throughout CrossTF.

2.2 TIME AND FREQUENCY VIEWS

As illustrated in Figure 2, CrossTF maintains the intermediate sample $\mathbf{x}_i$ as a one-dimensional (1D) time-domain sequence throughout the denoising process. At each denoising step from $\mathbf{x}_i$ to $\mathbf{x}_{i-1}$, $\mathbf{x}_i$ is transformed into both time and frequency views, yielding view-specific representations X_i^T and X_i^F. To ensure comparable computational budgets across the two views, both are encoded as equally sized two-dimensional (2D) representations (Naiman et al., 2024a). Specifically, as shown in Figure 4, the time view is constructed using delay embedding (Packard et al., 1980), where each row represents the sequence under a different temporal delay. For example, with a one-step delay, the first row contains (t_1, t_2, t_3), while the next contains (t_2, t_3, t_4). In parallel, the frequency view is constructed using the short-time Fourier transform (STFT) (Allen & Rabiner, 1977), which decomposes each local time frame into frequency components. For example, the first frame (t_1, t_2, t_3)

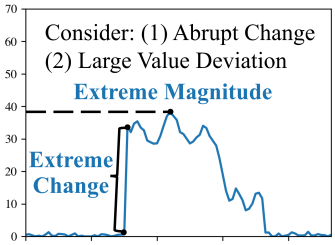


Fig. 3: Two Different Aspects of Extremes

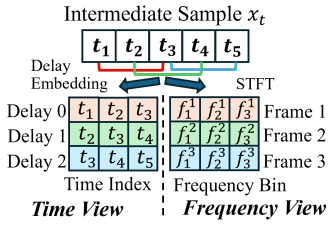


Fig. 4: Two-Dimensional Time and Frequency Views

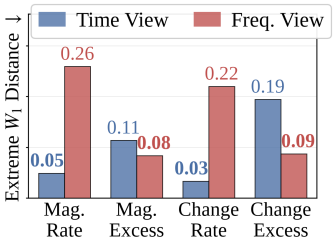


Fig. 5: Complementary Advantages for Extreme-Aware Generation

is mapped to its frequency coefficients $(f_1^1, \ldots, f_K^1)$, where f_j^k denotes the coefficient at frequency bin j in frame k, and K is the number of frequency bins. The transformation is then applied to each subsequent overlapping frame.

We further examine the complementary advantages of the two views for extreme-aware generation. The time view directly represents extreme magnitudes and local changes, whereas the frequency view captures extremes through distributed cross-frequency patterns. Empirically, the two views exhibit complementary characteristics for extreme-aware generation across datasets. Taking the Precipitation dataset as an example, Figure 5 compares the two views using four magnitude- and change-based evaluation metrics defined in Section 4, where a lower W_1 distance indicates better fidelity. The time view performs better on Magnitude Rate and Change Rate, whereas the frequency view achieves lower errors on Magnitude Excess and Change Excess. This complementarity motivates the key design in CrossTF: modeling the same intermediate sample through both views to leverage their advantages and exploiting the cross-view direction to improve extreme fidelity.

2.3 EDM-Based Diffusion for Time-Series Generation

CrossTF adopts EDM (Karras et al., 2022) as its underlying diffusion framework. During sampling, EDM starts from Gaussian noise $\mathbf{x}_I$ and progressively denoises it along decreasing noise levels until reaching the final sample $\mathbf{x}_0$. At each denoising step i, the denoiser produces a clean-sample estimate $\hat{\mathbf{x}}_{0,i}$ given the current state $\mathbf{x}_i$ and noise level σ_i. Based on this estimate, EDM computes the local direction of the probability-flow ODE and performs an Euler prediction toward the next noise level:

$$\mathbf{d}_i = (\mathbf{x}_i - \hat{\mathbf{x}}_{0,i})/\sigma_i, \qquad \tilde{\mathbf{x}}_{i-1} = \mathbf{x}_i + (\sigma_{i-1} - \sigma_i)\mathbf{d}_i. \tag{2}$$

Rather than using this first-order prediction, EDM applies a second-order Heun correction (Hairer et al., 1993) when $\sigma_{i-1} > 0$. It recomputes the direction at $(\tilde{\mathbf{x}}_{i-1}, \sigma_{i-1})$ and averages the two directions to obtain

$$\mathbf{x}_{i-1} = \mathbf{x}_i + (\sigma_{i-1} - \sigma_i)(\mathbf{d}_i + \tilde{\mathbf{d}}_{i-1})/2. \tag{3}$$

When $\sigma_{i-1} = 0$ at the final step, the Heun correction is omitted and the Euler prediction (Karras et al., 2022) is used directly. Together, these updates progressively transform $\mathbf{x}_I$ into $\mathbf{x}_0$.

3 Proposed CrossTF

3.1 Cross-View Direction Formulation

CrossTF uses the discrepancy between time- and frequency-view estimates to define a direction for adjusting the clean-sample estimate $\hat{\mathbf{x}}_{0,i}$. As shown in Figure 2, at each denoising step i, the same intermediate sample $\mathbf{x}_i \in \mathbb{R}^T$ is transformed into time and frequency representations X_i^T and X_i^F through delay embedding and STFT, respectively, as introduced in Section 2.2. The view-specific denoisers $D_{\theta_T}^T$ and $D_{\theta_F}^F$ then independently produce the time- and frequency-view clean-sample estimates:

$$\hat{\mathbf{x}}_{0,i}^T = D_{\theta_T}^T(X_i^T, \sigma_i) \in \mathbb{R}^T, \qquad \hat{\mathbf{x}}_{0,i}^F = D_{\theta_F}^F(X_i^F, \sigma_i) \in \mathbb{R}^T. \tag{4}$$

Here, each denoiser processes its corresponding 2D view and maps the output back to the 1D time domain through a fixed, non-learned inverse transform, making the two estimates directly comparable. We define their difference $\Delta_i = \hat{\mathbf{x}}_{0,i}^T - \hat{\mathbf{x}}_{0,i}^F$, as the cross-view direction from the frequency-view estimate to the time-view estimate, with the two estimates as its anchors. At each denoising step,

Table 1: **Extreme-sensitive cross-view asymmetry** across seven datasets.

	Relative Max–Min Range ($\alpha_i \in [0, 1]$)			Ratio	
Dataset	Loss	Overall	Extreme	Ext./Loss	Ext./Overall
Traffic	4.5%	8.4%	290.4%	65.22×	34.44×
Transportation	7.5%	51.1%	254.2%	34.11×	4.97×
Temperature	7.3%	12.6%	145.6%	19.91×	11.52×
Electricity	8.7%	78.4%	85.2%	9.82×	1.09×
Precipitation	33.3%	339.9%	160.3%	4.81×	0.47×
ECG	16.7%	3.9%	38.2%	2.29×	9.93×
Building	12.9%	10.8%	22.8%	1.77×	2.10×
Median	8.7%	12.6%	145.6%	**16.74**×	**11.56**×

a step-specific fusion parameter α_i determines the position along this direction, yielding the cross-view clean-sample estimate $\hat{\mathbf{x}}_{0,i}$:

$$\hat{\mathbf{x}}_{0,i} = \hat{\mathbf{x}}^F_{0,i} + \alpha_i \Delta_i = \hat{\mathbf{x}}^F_{0,i} + \alpha_i \left(\hat{\mathbf{x}}^T_{0,i} - \hat{\mathbf{x}}^F_{0,i} \right). \tag{5}$$

Under this parameterization, $\alpha_i = 0$ and 1 correspond to the pure frequency-view and time-view estimates, respectively. The resulting estimate is then passed to the EDM sampler in Section 2.3, which updates $\mathbf{x}_i$ to $\mathbf{x}_{i-1}$ using Equations 2 and 3. Further architectural details of the view-specific denoisers are provided in Appendix C.4.

3.2 Cross-View Asymmetry and Window Identification

We next examine why the cross-view direction is useful for extreme-aware generation and when to apply it during denoising. Although other effective directions may exist in the high-dimensional sample space, this cross-view direction offers two practical advantages. First, the time and frequency views represent the same underlying sequence in different forms, providing complementary perspectives on its structure. Second, their clean-sample estimates anchor a sample-dependent direction along which adjustments can be controlled by a single scalar at each denoising step. Beyond these advantages, our empirical analysis reveals a key property particularly relevant to extreme-focused optimization, which we term *Extreme-Sensitive Cross-View Asymmetry*.

Extreme-Sensitive Cross-View Asymmetry. At denoising step i, varying the fusion parameter α_i moves the clean-sample estimate along the cross-view direction. Empirically, this movement produces substantial changes in extreme-fidelity metrics, while the denoising loss and overall-quality metrics vary considerably less. Thus, extreme fidelity is markedly more sensitive to movement along this direction than overall generation quality is. This differential sensitivity creates a key optimization opportunity for CrossTF: extreme-focused optimization can be performed separately along this direction after overall generation capability has stabilized, avoiding direct competition with the primary objective during joint optimization. The two objectives can thus be largely decoupled, enabling improvements in extreme fidelity with limited impact on the learned overall distribution.

To examine this asymmetry, Table 1 reports results across seven datasets. Here, the *Relative Max–Min Range* measures the difference between the maximum and minimum values observed for each quantity while varying α_i over $[0, 1]$, normalized by its pure time-view value. For simplicity, we use the same α_i across the denoising steps considered in this analysis. We calculate the *denoising loss* using the standard MSE-based one-step denoising objective in EDM-based diffusion, the *overall-quality metric* using the Wasserstein distance, and the *extreme-fidelity metric* using the average of four EVT-based metrics introduced in Section 4. Across the seven datasets, the median relative range of the extreme-fidelity metric is **16.74**× that of the denoising loss and **11.56**× that of the overall-quality metric. Notably, the extreme-fidelity metric exhibits a larger relative range than the denoising loss on all seven datasets and the overall-quality metric on six. The only exception is Precipitation, where extremes dominate the signal variation, making changes in extreme behavior more directly reflected in the overall distribution. These findings provide empirical support for optimizing extreme fidelity along the cross-view direction. Further details are provided in Appendix C.2.

Identifying the Cross-View Window. Having established the utility of the cross-view direction for extreme-focused optimization, we next ask when to apply this optimization during denoising. Because the direction is defined by the discrepancy between the two views, we examine how this

discrepancy evolves along the denoising process and identify when it stabilizes sufficiently to provide a reliable cross-view direction.

At each denoising step, we keep the real samples and noise level fixed and evaluate the cross-view discrepancy over 16 independent noise realizations for each of three independently trained seeds, with the results aggregated across seeds. We decompose the discrepancy into a *stable component*, which remains consistent across noise realizations, and a *deviation component*, which varies with the noise realization. We then quantify the proportion of the total discrepancy contributed by the stable component, denoted as the *Stable Discrepancy Share*. From Figure 6, we observe that although the share can be high at the earliest denoising stages, it varies substantially across training seeds, indicating that the apparent stability is not reproducible and thus not yet meaningful. The share then drops to a low level, indicating that a stable cross-view component has not yet emerged. In the later stages, however, the share rises sharply and remains high, indicating that a persistent cross-view discrepancy begins to emerge across different noise realizations. Interestingly, this pattern is broadly observed across all seven datasets. Based on this observation, we define this late-stage interval as the *cross-view window*, where a reliable cross-view discrepancy emerges, and restrict cross-view optimization to this window. In our setting with $I = 18$ denoising steps, the cross-view window corresponds to the final six steps ($i = 6, \ldots, 1$), where $\sigma_i \leq 0.35$.

To further validate the effectiveness of the cross-view window, we conduct an ablation study by switching the generation view from the pure time view to the pure frequency view either within or outside the window, while keeping all other settings unchanged, and compare the resulting changes in extreme fidelity. As shown in Figure 7, the changes within the window are substantially larger than those outside the window across all seven datasets, with differences of up to $38\times$. This confirms that the cross-view direction is substantially more effective within the cross-view window. Restricting optimization to this window also reduces the search space of the fusion parameters and simplifies subsequent optimization. Additional analyses and discussion are provided in Appendix C.3.

3.3 Lightweight Cross-View Optimization

Finally, we determine how far to move along the cross-view direction by learning step-wise fusion parameters α_i within the cross-view window. Since fusion outside this window offers little benefit to extreme fidelity (Figure 7), we use the time view alone for the remaining steps by setting $\alpha_i = 1$.We first independently train the time- and frequency-view denoisers, $D^T_{\theta_T}$ and $D^F_{\theta_F}$ from Eq. 4, with the EDM denoising objective and freeze their parameters before optimizing α_i. Here, X^v_σ denotes the corresponding 2D view representation at noise level σ. The training objective is a noise-weighted MSE between the denoised estimate and the clean sample:

$$\mathcal{L}^v_{\text{denoise}} = \mathbb{E}_{\mathbf{x}_0,\sigma,\boldsymbol{\epsilon}}\left[\lambda(\sigma)\left\|D^v_{\theta_v}\left(X^v_\sigma, \sigma\right) - \mathbf{x}_0\right\|_2^2\right], \qquad v \in \{T, F\}, \tag{6}$$

where $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and $\lambda(\sigma)$ is the noise-dependent weighting function in EDM. Further details on the training and sampling configurations of the EDM-based denoiser are in Appendix C.4.

After training and freezing both denoisers, the only learnable variables are six scalar fusion parameters, $\boldsymbol{\alpha} = \{\alpha_i\}_{i=1}^6$, one for each denoising step in the six-step cross-view window. To optimize these parameters, we formulate a Peaks-Over-Threshold (POT)-based objective grounded in EVT (Coles, 2001). For each extreme dimension $q \in \{\text{mag}, \text{chg}\}$, POT approximates the distribution of excesses

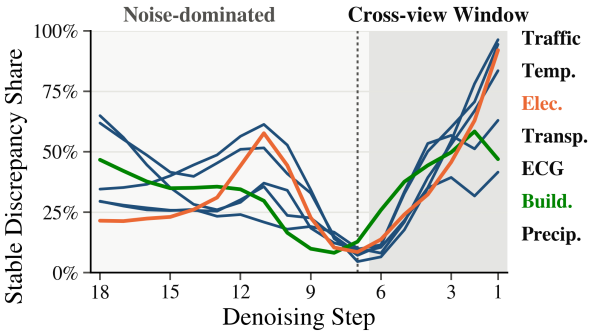


Fig. 6: Cross-View Window Identification.

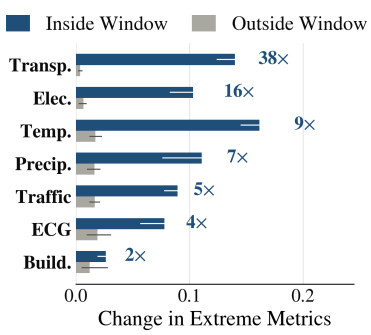


Fig. 7: Cross-View Window Sensitivity.

above a sufficiently high threshold with a Generalized Pareto Distribution (GPD), whose shape parameter ξ^q characterizes tail behavior across pooled exceedances. Rather than relying on this pooled tail descriptor alone, we further characterize extremes at the sequence level through their occurrence frequency and severity. Specifically, the per-sequence exceedance count C^q is modeled by a Negative Binomial distribution (Hilbe, 2011), while the mean excess M^q is modeled by a Gamma distribution (Johnson et al., 1994). The resulting EVT-based objective is formulated as:

$$\mathcal{L}_{\mathrm{EVT}}(\boldsymbol{\alpha}) = \sum\nolimits_{q\in\{\mathrm{mag},\mathrm{chg}\}} \Big[|\xi_g^q - \xi_r^q| + \widetilde{W}_1(\mathrm{NegBin}_g^q, \mathrm{NegBin}_r^q) + \widetilde{W}_1(\Gamma_g^q, \Gamma_r^q)\Big], \tag{7}$$

where subscripts g and r denote generated and real data, respectively, NegBin^q and Γ^q denote the corresponding Negative Binomial and Gamma distributions, and $\widetilde{W}_1$ denotes the Wasserstein distance normalized by the mean of the corresponding real data. The three terms quantify discrepancies in ***tail shape***, ***extreme occurrence frequency***, and ***extreme severity***, respectively. All three terms are dimensionless and are combined with equal weights.

Because CrossTF allows extrapolation beyond the two view-specific estimates with $\alpha_i \notin [0,1]$, we further constrain the energy distance $\mathcal{E}$ between the generated and real distributions to limit the potential degradation in overall generation quality. The resulting optimization problem is:

$$\boldsymbol{\alpha}^* = \arg\min_{\boldsymbol{\alpha}} \mathcal{L}_{\mathrm{EVT}}(\boldsymbol{\alpha}) \quad \text{s.t.} \quad \mathcal{E}(\boldsymbol{\alpha}) \le \mathcal{E}_T + \kappa\,\mathrm{SE}(\mathcal{E}_T). \tag{8}$$

Here, $\mathcal{E}_T$ denotes the energy distance of the pure time-view baseline, and $\mathrm{SE}(\mathcal{E}_T)$ denotes its estimated standard error accounting for randomness in training and generation. We set $\kappa = 2$, allowing the optimized energy distance to exceed the baseline by at most two standard errors, thereby limiting the overall performance degradation introduced by cross-view extrapolation to the baseline's natural statistical fluctuation range. We solve the resulting constrained nonlinear optimization problem using Sequential Least Squares Programming (SLSQP) (Kraft, 1988), which iteratively solves local quadratic subproblems with finite-difference gradient estimates. With both denoisers frozen, this stage reduces to a lightweight optimization problem involving only six scalar fusion parameters.

4 EXPERIMENTS

Research Questions (RQs). Our evaluation addresses seven key research questions. **RQ1:** Does CrossTF improve extreme fidelity? **RQ2:** Does CrossTF preserve overall generation fidelity? **RQ3:** Does CrossTF generate useful synthetic data for downstream tasks? **RQ4:** How does cross-view fusion compare with generation using either view alone? **RQ5:** Do the learned step-wise fusion parameters α_i exhibit both interpolation and extrapolation? **RQ6:** Does cross-view complementarity provide benefits beyond model diversity? **RQ7:** How computationally efficient is CrossTF?

Datasets. We evaluate CrossTF on **seven real-world datasets** across diverse domains: TRAFFIC, PRECIP., BUILD., TEMP., ELECT., TRANSP., and ECG. Here, TRAFFIC, PRECIP., BUILD., TEMP., ELECT., TRANSP., and ECG denote Traffic Speed, Precipitation, Building Power, Temperature, Electricity, Traffic Occupancy, and Electrocardiogram, respectively. These datasets capture diverse real-world extremes, including abrupt traffic-speed changes, intense precipitation, building-load surges, unusual temperature deviations, power surges, high freeway occupancy, and pronounced ECG deviations. Detailed descriptions and sources are provided in Appendix D.1.

Metrics. We evaluate CrossTF with **ten metrics** for extreme fidelity and overall performance. Extreme fidelity uses four metrics over extreme magnitude and change: MAG-RATE and CHG-RATE measure occurrence-frequency discrepancies, while MAG-EXCESS and CHG-EXCESS measure exceedance severity. Overall performance uses three fidelity metrics (WASS. (Villani, 2009), MARG. (Naiman et al., 2024a), and ACD (Ang et al., 2023)) and three downstream metrics (CONTEXT-FID (Yuan & Qiao, 2024), PRED (Naiman et al., 2024a), and DISC (Naiman et al., 2024a)) for representation, prediction, and discrimination, respectively. PRED follows the Train-on-Synthetic, Test-on-Real protocol. All metrics are lower-is-better. Details are in Appendix D.2.

Baselines. We compare CrossTF with **11 representative baselines** spanning GAN, VAE, LLM, diffusion, time-frequency-based, and extreme-aware generation approaches, including TimeGAN (Yoon et al., 2019), KoVAE (Naiman et al., 2024b), SDForger (Rousseau et al., 2025), DiffTime (Coletta et al., 2023), Diffusion-TS (Yuan & Qiao, 2024), FourierDiff (Crabbé et al., 2024), ImagenTime (Naiman et al., 2024a), RF-Diffusion (Chi et al., 2024), FIDE (Galib et al., 2024), HeavyDiff (Pandey et al., 2025), and E4GEN (Jiang et al., 2026). Details are in Appendix D.3.

Table 2: Summary of comparisons on seven datasets. *Best* denotes the strongest baseline. *W–L* counts Wins-Losses against 11 baselines across 10 metrics, yielding 110 pairwise comparisons per dataset. Lower is better.

Dataset	Extreme Fidelity				Overall Performance				W–L
	Best	Ours	Baseline	Improve	Best	Ours	Baseline	Improve	
TRAFFIC	ImagenTime	**0.0343**	0.0591	**+42.0%**	FourierDiff	**0.0083**	0.0138	**+39.9%**	110–0
PRECIP.	ImagenTime	**0.0755**	0.1086	**+30.5%**	ImagenTime	**0.0050**	0.0051	**+2.0%**	103–7
BUILD.	ImagenTime	**0.0762**	0.1012	**+24.7%**	ImagenTime	**0.0040**	0.0046	**+13.0%**	102–8
TEMP.	ImagenTime	**0.0884**	0.1222	**+27.7%**	FIDE	**0.0131**	0.0165	**+20.6%**	101–9
ELECT.	ImagenTime	**0.0691**	0.0734	**+5.9%**	ImagenTime	0.0220	**0.0193**	−14.0%	102–8
TRANSP.	ImagenTime	**0.0477**	0.0691	**+31.0%**	ImagenTime	0.0120	**0.0114**	−5.3%	107–3
ECG	ImagenTime	**0.1421**	0.1807	**+21.4%**	ImagenTime	0.0201	**0.0193**	−4.1%	105–5
Summary	–	–	–	**+26.1%**	–	–	–	**+7.4%**	**730–40**

Table 3: Generation performance on the TRAFFIC dataset. All metrics are lower-is-better.

Dataset	Type	Method	Extreme Fidelity				Overall Fidelity		
			Mag-Rate	Chg-Rate	Mag-Excess	Chg-Excess	Wass.	Marg.	ACD
TRAFFIC	**GAN**	TimeGAN	0.6852	0.6682	1.2587	0.7178	0.1240	0.0027	0.1393
	VAE	KoVAE	0.6755	0.9931	0.6919	1.0784	0.3751	0.0100	0.1384
	LLM	SDForger	0.1641	0.7892	0.2229	0.9543	0.2521	0.0053	0.0690
	Diffusion	DiffTime	0.2229	0.2625	0.2053	0.2194	0.1098	0.0020	0.0258
		Diffusion-TS	0.2756	0.6143	0.2531	0.3993	0.1817	0.0036	0.0388
		FourierDiff	0.0749	0.1563	0.1359	0.6281	0.0373	0.0017	0.0064
		ImagenTime	0.0634	0.0532	0.0681	0.0532	0.0471	0.0023	0.0059
	TF-Based	RF-Diffusion	0.4571	7.2173	0.5824	1.3235	0.2573	0.0082	0.2225
	Extreme-Aware	FIDE	0.5573	5.3031	0.9103	2.2618	0.3703	0.0072	0.1041
		HeavyDiff	0.3541	0.3541	0.4583	0.9180	0.1088	0.0022	0.0273
		E4GEN	0.0901	0.5107	0.0644	0.1965	0.1133	0.0048	0.0725
	Ours	**CrossTF**	**0.0323**	**0.0416**	**0.0282**	**0.0367**	**0.0208**	**0.0014**	**0.0041**

4.1 OVERALL PERFORMANCE

To answer RQs 1–3, Table 2 summarizes comparisons between CrossTF and 11 baselines across seven real-world datasets and ten metrics, yielding 770 pairwise comparisons. CrossTF achieves 730 wins, outperforming the baselines in 94.8% of these comparisons. Relative to the strongest baseline on each dataset, CrossTF improves extreme fidelity by 26.1% overall, with gains across all seven datasets, and overall performance by 7.4%. Among the seven datasets, PRECIPITATION and ELECTRICITY, which exhibit the weakest Ext./Overall ratios (0.47× and 1.09×) in Table 1, also show some of the clearest degradations in individual overall-fidelity metrics. In contrast, TRAFFIC exhibits the strongest asymmetry (34.44×) and achieves the largest overall-performance improvement (+39.9%). This correspondence supports our interpretation that stronger asymmetry provides greater room to improve extreme fidelity without degrading overall performance, whereas weaker asymmetry may require a greater trade-off between extreme and overall performance. We first use TRAFFIC as a representative case to examine extreme and overall fidelity in detail, followed by a cross-dataset analysis. Detailed fidelity results for the remaining datasets are provided in Appendix D.8, while downstream utility is reported separately in Table 19 in Appendix D.9.

As shown in Table 3, CrossTF achieves the best results on TRAFFIC across all four extreme-fidelity and three overall-fidelity metrics. Compared with the strongest baseline for each metric, CrossTF reduces the four extreme-fidelity metrics by 21.8%–56.2% and the three overall-fidelity metrics by 17.6%–44.2%. Among the baselines, ImagenTime and E4GEN are competitive in extreme fidelity, while FourierDiff and ImagenTime perform strongly in overall fidelity. CrossTF improves upon these methods across both dimensions. Across all seven datasets, CrossTF shows its most consistent advantage on CHG-EXCESS, ranking first on every dataset with an average relative improvement of 32.6% over the strongest baseline and at least a 22.1% gain. In contrast, its gains on CHG-RATE

are more modest, with an average relative improvement of 1.1% and the best result on five of seven datasets. This suggests that cross-view optimization is more effective at controlling how far abrupt changes exceed the threshold than how often they occur.

For downstream utility, we utilize commonly adopted public evaluators in time-series generation studies to compute DISC, PRED, and CONTEXT-FID. As shown in Table 19, CrossTF achieves the best or tied-best result in 18 of 21 dataset–task combinations, including all seven CONTEXT-FID, six PRED, and five DISC cases. These results demonstrate strong downstream utility of CrossTF-generated data across representation learning, prediction, and real-vs-synthetic discrimination tasks.

4.2 FURTHER ANALYSIS

To answer RQ4, we compare CrossTF with TIME-ONLY and FREQUENCY-ONLY to examine the effect of cross-view fusion. Compared with the stronger single-view variant, TIME-ONLY, CrossTF improves extreme fidelity by **18.0**% on average, while overall fidelity changes by only **0.1**%. This shows that cross-view fusion substantially improves extreme generation while largely maintaining overall distributional fidelity. Details are shown in Appendix D.4.

To answer RQ5, we analyze the learned step-wise fusion parameters α_i within the cross-view window across seven datasets. Both interpolation and extrapolation appear frequently, with α_i generally biased toward the time view and **52.4%** of the parameters satisfying $\alpha_i > 1$, corresponding to extrapolation beyond the time view. This shows that cross-view fusion is not a simple convex combination and that allowing extrapolation can further exploit the cross-view discrepancy to improve extremes. Details are shown in Appendix D.5.

To answer RQ6, we compare time–frequency cross-view fusion with time–time fusion using two independently trained time-view denoisers, thereby controlling for the effect of same-view model diversity. The cross-view direction exhibits an extreme-fidelity variation **11.25**$\times$ larger than the same-view direction, while all same-view variations remain within sampling noise. These results show that the *Extreme-Sensitive Cross-View Asymmetry* is a characteristic of the cross-view complementarity between the time and frequency views, rather than a generic effect of model ensemble or model diversity. Details are shown in Appendix D.6.

To answer RQ7, we compare the training and sampling efficiency of CrossTF and all baselines on a single NVIDIA RTX PRO 6000 Blackwell Server GPU. CrossTF requires only **3.7 seconds** to generate 1,000 samples, whereas the more competitive ImagenTime, FourierDiff, and E4GEN require approximately 6.03$\times$, 6.27$\times$, and 20.14$\times$ longer sampling time, respectively. These results show that CrossTF maintains fast sampling and practical computational efficiency while achieving strong generation performance. Details are shown in Appendix D.7.

5 RELATED WORK

Recent studies have explored diverse approaches to time-series generation, including GANs, VAEs, and diffusion models, with a primary focus on reproducing overall data distributions. **Extreme-aware generation** instead focuses on rare events. FIDE emphasizes high-frequency components, E4GEN uses event-level shapelets (Jiang et al., 2026), and Tail-GAN focuses on tail statistics (Cont et al., 2026). In contrast, CrossTF freezes denoisers trained with the standard denoising objective and optimizes only their fusion to improve extreme fidelity under an overall-fidelity constraint. **Time-frequency modeling** incorporates spectral structure into time-series generation: FourierDiff denoises spectral coefficients (Crabbé et al., 2024), while RF-Diffusion couples time and frequency domains through a prescribed process for radio signals (Chi et al., 2024). CrossTF instead produces complete time- and frequency-view clean-sample estimates from the same intermediate sample at each denoising step and learns how far to move along the direction defined by their discrepancy to improve extreme fidelity. Further discussion is provided in Appendix A.

6 CONCLUSION

In this paper, we presented CrossTF, a diffusion framework that exploits time–frequency cross-view discrepancy for extreme-aware time-series generation. Our empirical analysis revealed *extreme-*

sensitive cross-view asymmetry: movement along the cross-view direction substantially affects extreme fidelity while having comparatively limited effects on overall generation quality. We further identified a late-stage *cross-view window* in which the discrepancy stabilizes and cross-view adjustments become particularly effective. Building on these findings, CrossTF decouples denoiser training from extreme-focused optimization by freezing both denoisers and learning only six fusion parameters. An EVT-based objective guides interpolation and extrapolation, while an energy-distance constraint limits degradation in overall generation quality. Experiments on seven real-world datasets against 11 baselines demonstrated improvements in extreme fidelity and overall performance. These findings establish time–frequency cross-view discrepancy as an effective optimization mechanism for improving extreme-event time-series generation without retraining the underlying denoisers.

AI USE STATEMENT

In this work, we used generative AI tools, including large language models, to assist with experimental code generation and editing, parts of mathematical derivations, manuscript editing for improved readability, literature retrieval, and reference formatting. The authors reviewed all AI-assisted work, including verifying AI-assisted code and mathematical derivations for correctness, checking that language edits preserved the intended scientific meaning, and confirming the relevance and bibliographic accuracy of suggested references. The authors take full responsibility for the final content of this work and its associated materials, including the text, mathematical arguments, references, implementation, experimental results, and scientific claims, regardless of whether generative AI assistance was involved.

ETHICS STATEMENT

All authors have read and agree to abide by the ICLR Code of Ethics. This study does not involve the collection of private or personally identifiable information. The work focuses on methodological development for extreme-aware time-series generation and does not involve direct interaction with human participants.

REPRODUCIBILITY STATEMENT

To support full reproduction of our results, we release the complete implementation and experimental materials of CrossTF in an anonymous repository: **Anonymous Repository**. The repository covers the complete experimental pipeline, from raw data and model configurations to training, sampling, evaluation, and table generation. Specifically, `data/` provides the raw data used for the seven datasets; `configs/` contains the complete CrossTF configurations for each dataset; `baselines/` provides the implementations and corresponding configurations of all 11 baseline methods; `results/` contains the generated samples from CrossTF and all 11 baselines; and `evaluation/` provides the implementations of the evaluation metrics and result aggregation code. We further provide `evaluation/make_table.py`, which can directly reproduce the main experimental results for all seven datasets reported in Tables 3, 13, 14, 15, 16, 17, and 18 from the released generated samples. With these materials, readers can either rerun the full training and sampling pipeline from scratch or directly reproduce the main quantitative results from the provided samples.

Our experiments were conducted on multiple high-performance computing clusters, primarily using NVIDIA RTX PRO 6000 Blackwell, NVIDIA B200, and NVIDIA A100 GPUs. To ensure fair comparison across methods, all methods follow the unified data splits, evaluation metrics, and experimental protocols reported in the paper. The main hyperparameters and implementation details required for model architecture, training, sampling, optimization, and evaluation are provided in the main text, appendices, and anonymous code repository.

REFERENCES

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

Jont B Allen and Lawrence R Rabiner. A unified approach to short-time fourier analysis and synthesis. *Proceedings of the IEEE*, 65(11):1558–1564, 1977.

Yihao Ang, Qiang Huang, Yifan Bao, Anthony KH Tung, and Zhiyong Huang. Tsgbench: Time series generation benchmark. *Proceedings of the VLDB Endowment*, 17(3):305–318, 2023.

Yogesh Balaji, Seungjun Nah, Xun Huang, Arash Vahdat, Jiaming Song, Qinsheng Zhang, Karsten Kreis, Miika Aittala, Timo Aila, Samuli Laine, Bryan Catanzaro, Tero Karras, and Ming-Yu Liu. eDiff-I: Text-to-image diffusion models with an ensemble of expert denoisers. *arXiv preprint arXiv:2211.01324*, 2022.

Siddharth Bhatia, Arjit Jain, and Bryan Hooi. Exgan: Adversarial generation of extreme samples. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 6750–6758, 2021.

California Department of Transportation. Performance measurement system (PeMS) data source. Caltrans. URL https://dot.ca.gov/programs/traffic-operations/mpr/pems-source. Accessed September 25, 2026.

Guoxuan Chi, Zheng Yang, Chenshu Wu, Jingao Xu, Yuchong Gao, Yunhao Liu, and Tony Xiao Han. Rf-diffusion: Radio signal generation via time-frequency diffusion. In *Proceedings of the 30th Annual International Conference on Mobile Computing and Networking*, pp. 77–92, 2024.

Stuart Coles. *An Introduction to Statistical Modeling of Extreme Values*. Springer Series in Statistics. Springer, London, 2001.

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

Rama Cont, Mihai Cucuringu, Renyuan Xu, and Chao Zhang. Tail-gan: Learning to simulate tail risk scenarios. *Management Science*, 72(4):2917–2936, 2026.

Jonathan Crabbé, Nicolas Huynh, Jan Stanczuk, and Mihaela Van Der Schaar. Time series diffusion in the frequency domain. In *Proceedings of the 41st International Conference on Machine Learning*, pp. 9407–9438, 2024.

Custom Weather. Daily weather data. Dewey Data, dataset, 2022a. URL https://www.deweydata.io/doi/10-82551-vbwq-aq20.

Custom Weather. Hourly weather data. Dewey Data, dataset, 2022b. URL https://www.deweydata.io/doi/10-82551-yx6h-k352.

Marco Cuturi. Fast global alignment kernels. In *Proceedings of the 28th International Conference on Machine Learning*, pp. 929–936, 2011. URL https://mlanthology.org/icml/2011/cuturi2011icml-fast/.

Abhimanyu Das, Weihao Kong, Rajat Sen, and Yichen Zhou. A decoder-only foundation model for time-series forecasting. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 10148–10167. PMLR, 2024.

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.

Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34:8780–8794, 2021.

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

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

Hongfan Gao, Wangmeng Shen, Bin Yang, and Jilin Hu. Hyfad: Hybrid time-frequency diffusion with frequency-aware embedding for time series imputation. *arXiv preprint arXiv:2606.05239*, 2026.

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. doi: 10.1161/01.CIR.101.23.e215.

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.

Ernst Hairer, Gerhard Wanner, and Syvert P Nørsett. *Solving ordinary differential equations I: Nonstiff problems*. Springer, 1993.

Georges Hebrail and Alice Berard. Individual household electric power consumption. UCI Machine Learning Repository, dataset, 2006. URL https://archive.ics.uci.edu/dataset/235/individual+household+electric+power+consumption.

Joseph M Hilbe. *Negative binomial regression*, volume 2. Cambridge University Press New York, NY, USA, 2011.

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

Franc Jager, Alessandro Taddei, George B. Moody, Michele Emdin, Gorazd Antolic, Roman Dorn, Ales 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 and Biological Engineering and Computing*, 41(2):172–182, 2003. doi: 10.1007/BF02344885.

Daniel Jarrett, Ioana Bica, and Mihaela van der Schaar. Time-series generation by contrastive imitation. *Advances in neural information processing systems*, 34:28968–28982, 2021.

Jinsung Jeon, Jeonghak Kim, Haryong Song, Seunghyeon Cho, and Noseong Park. GT-GAN: General purpose time series synthesis with generative adversarial networks. *Advances in Neural Information Processing Systems*, 35, 2022.

Seungwoo Jeong, Junghyo Sohn, Jaehyun Jeon, and Heung-Il Suk. Frequency-conditioned diffusion models for time series generation. In *Proceedings of the 34th ACM International Conference on Information and Knowledge Management*, pp. 1114–1123, 2025.

Lin Jiang, Dahai Yu, Ximiao Li, and Guang Wang. E4gen: Event-level explainable extreme-enhanced time-series generation. In *Advances in Neural Information Processing Systems*, 2026.

Norman L Johnson, Samuel Kotz, and Narayanaswamy Balakrishnan. *Continuous univariate distributions, volume 1*, volume 1. John wiley & sons, 1994.

Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. *Advances in neural information processing systems*, 35:26565–26577, 2022.

Tero Karras, Miika Aittala, Tuomas Kynkäänniemi, Jaakko Lehtinen, Timo Aila, and Samuli Laine. Guiding a diffusion model with a bad version of itself. *Advances in Neural Information Processing Systems*, 37, 2024.

Marcel Kollovieh, Abdul Fatir Ansari, Michael Bohlke-Schneider, Jasper Zschiegner, Hao Wang, and Yuyang Bernie Wang. Predict, refine, synthesize: Self-guiding diffusion models for probabilistic time series forecasting. *Advances in Neural Information Processing Systems*, 36:28341–28364, 2023.

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.

Dieter Kraft. A software package for sequential quadratic programming. *Forschungsbericht-Deutsche Forschungs- und Versuchsanstalt fur Luft- und Raumfahrt*, 1988.

Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. In *International Conference on Learning Representations*, 2018. URL https://mlanthology.org/iclr/2018/li2018iclr-diffusion/.

Nan Liu, Shuang Li, Yilun Du, Antonio Torralba, and Joshua B Tenenbaum. Compositional visual generation with composable diffusion models. In *European Conference on Computer Vision*, 2022.

Hang Lou, Siran Li, and Hao Ni. Pcf-gan: generating sequential data via the characteristic function of measures on the path space. *Advances in Neural Information Processing Systems*, 36:39755–39781, 2023.

Clayton Miller, Anjukan Kathirgamanathan, Bianca Picchetti, Pandarasamy Arjunan, June Young Park, Zoltan Nagy, Paul Raftery, Brodie W. Hobson, Zixiao Shi, and Forrest Meggers. The building data genome project 2, energy meter data from the ASHRAE great energy predictor iii competition. *Scientific Data*, 7:368, 2020. doi: 10.1038/s41597-020-00712-x.

Ilan Naiman, Nimrod Berman, Itai Pemper, Idan Arbiv, Gal Fadlon, and Omri Azencot. Utilizing image transforms and diffusion models for generative modeling of short and long time series. *Advances in Neural Information Processing Systems*, 37:121699–121730, 2024a.

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 *The Twelfth International Conference on Learning Representations*, 2024b.

Alexander Nikitin, Letizia Iannucci, and Samuel Kaski. Tsgm: a flexible framework for generative modeling of synthetic time series. *Advances in Neural Information Processing Systems*, 37: 129042–129061, 2024.

Norman H Packard, James P Crutchfield, J Doyne Farmer, and Robert S Shaw. Geometry from a time series. *Physical review letters*, 45(9):712, 1980.

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.

Matteo Rizzato, Nicolas Morizet, William Maréchal, and Christophe Geissler. Stress testing electrical grids: Generative adversarial networks for load scenario generation. *Energy and AI*, 9:100177, 2022.

Cécile Rousseau, Tobia Boschi, Giandomenico Cornacchia, Dhaval Salwala, Alessandra Pascale, and Juan Moreno. Forging time series with language: A large language model approach to synthetic data generation. In *Advances in Neural Information Processing Systems*, volume 38, pp. 87120–87150, 2025.

Amirmojtaba Sabour, Sanja Fidler, and Karsten Kreis. Align your steps: Optimizing sampling schedules in diffusion models. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 42947–42975, 2024.

Hagit Shatkay. The fourier transform-a primer. *Brown University*, pp. 1–20, 1995.

Cédric Villani. *Optimal transport: old and new*, volume 338. Springer, 2009.

Yu-Hsiang Wang and Olgica Milenkovic. Waveletdiff: Multilevel wavelet diffusion for time series generation. *Transactions on Machine Learning Research*, 2026.

Tianlin Xu, Li Kevin Wenliang, Michael Munn, and Beatrice Acciaio. COT-GAN: Generating sequential data via causal optimal transport. *Advances in Neural Information Processing Systems*, 33, 2020.

Xinyu Yang, Yu Sun, Xiaojie Yuan, and Xinyang Chen. Frequency-aware generative models for multivariate time series imputation. *Advances in Neural Information Processing Systems*, 37: 52595–52623, 2024.

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

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

A Related Work

A.1 Time-series Generation

Time-series generation aims to synthesize sequences that preserve the temporal dynamics and distributional properties of real data (Nikitin et al., 2024). In the early years, GAN-based methods were widely applied to this task, combining recurrent generators and discriminators to capture temporal dependencies (Esteban et al., 2017). TimeGAN (Yoon et al., 2019) augments adversarial training with a supervised loss in a jointly learned embedding space. Subsequent methods introduce alternative distribution-matching objectives based on causal optimal transport (Xu et al., 2020) or characteristic functions on path space (Lou et al., 2023). GT-GAN (Jeon et al., 2022) extends the paradigm to irregularly sampled time-series through a continuous-time formulation. Although effective, adversarial approaches can suffer from unstable training and incomplete mode coverage.

Likelihood-based approaches offer an alternative through explicit or variational objectives. TimeVAE (Desai et al., 2021) reconstructs sequences through interpretable components such as level, trend, and seasonality, while Koopman VAE (Naiman et al., 2024b) introduces structured latent dynamics based on Koopman operators. Normalizing flows enable exact likelihood evaluation, with Fourier flows (Alaa et al., 2021) modeling sequences through spectral representations. These approaches support likelihood-based learning, although preserving fine-grained local structure remains challenging, particularly for models with restrictive latent or reconstruction assumptions.

Diffusion models have become a prominent approach following the success of iterative denoising in waveform synthesis (Kong et al., 2021). Diffusion-TS (Yuan & Qiao, 2024) combines a decomposition-based denoiser with interpretable trend and seasonal components, while self-guided formulations reuse an unconditional model for generation and refinement (Kollovieh et al., 2023). Other methods change the representation in which denoising occurs, either transforming sequences into two-dimensional representations (Naiman et al., 2024a) or modeling spectral coefficients directly (Crabbé et al., 2024). Iterative refinement supports strong generation fidelity but typically requires multiple denoiser evaluations during sampling. Beyond these paradigms, time-series generation has also been explored through contrastive imitation (Jarrett et al., 2021) and language modeling over tokenized sequences (Rousseau et al., 2025; Das et al., 2024).

A.2 Extreme-Aware Time-Series Generation

Existing extreme-aware generation methods can be broadly grouped into representation-based and objective-based approaches.

Representation-based methods. These methods incorporate extreme-aware structures into the data representation or generative process. FIDE (Galib et al., 2024) amplifies high-frequency components to retain spectral patterns associated with extremes during denoising, while E4GEN (Jiang et al., 2026) uses time-domain shapelets to represent extremes as event-level patterns that can be individually interpreted and enhanced. Heavy-tailed diffusion (Pandey et al., 2025) instead introduces extreme-aware assumptions into the generative process by replacing the Gaussian prior and perturbation kernel with their Student-t counterparts to better accommodate heavy-tailed data. These design choices introduce inductive biases that may not fully capture the diversity of extreme patterns across datasets.

Objective-based methods. These methods emphasize extreme behavior through tailored learning objectives. ExGAN (Bhatia et al., 2021) conditions adversarial generation on a target extremeness level, enabling sampling at a prescribed severity, while Tail-GAN (Cont et al., 2026) uses a risk-aware objective based on tail statistics to improve the reproduction of tail risk. Prioritizing rare extremes, however, may introduce a trade-off with fidelity to the broader data distribution.

CrossTF differs from both categories in the variables it optimizes and the stage at which extreme-focused optimization occurs. It first trains the view-specific denoisers using the standard denoising objective and then freezes them, without introducing a predefined extreme-specific representation or generative assumption. Extreme fidelity is subsequently refined through a small set of fusion parameters that control movement along the cross-view direction, to which the denoising loss is comparatively insensitive. This avoids committing to a particular form of extremes and allows the

optimization to adapt to diverse extreme patterns across datasets, while an explicit distributional constraint limits degradation in overall generation quality.

A.3 Time-Frequency Modeling for Time-Series Generation

The time and frequency domains provide mathematically equivalent descriptions of the same sequence (Shatkay, 1995), but organize its structure differently for generative modeling. Existing approaches broadly fall into two categories: representation-centric and integration-centric methods.

Representation-centric methods. These methods choose a spectral or time–frequency representation as the primary generative space. Frequency-domain diffusion, for example, directly denoises Fourier coefficients (Crabbé et al., 2024), whereas WaveletDiff operates on multilevel wavelet coefficients and exchanges information across resolutions (Wang & Milenkovic, 2026). Such representations expose compact spectral or multiscale structure, but the roles of time and frequency information are determined by the chosen transform rather than modeled as two parallel generative views.

Integration-centric methods. These methods combine the two domains within a model while explicitly specifying how they interact. Frequency-conditioned diffusion retains a time-domain denoising path and injects adaptively separated spectral components as conditions (Jeong et al., 2025). Related generative models for imputation similarly fuse frequency-derived features into a temporal modeling pipeline (Yang et al., 2024). More tightly coupled designs incorporate both domains directly into the diffusion process. RF-Diffusion develops a time–frequency formulation specialized for RF signals (Chi et al., 2024), while HyFAD performs sequential cross-domain denoising with predefined coarse-to-fine roles (Gao et al., 2026). Despite differences in coupling strength, these methods specify in advance how and where each domain contributes, such as through conditioning or stage-specific processing.

CrossTF instead treats the two domains as parallel generative views of the same intermediate sample. At each denoising step, the time and frequency views independently produce complete clean-sample estimates along the same sampling trajectory, without assigning either view a permanent primary or auxiliary role. Their discrepancy therefore defines a sample-dependent cross-view direction rather than a fixed backbone–auxiliary hierarchy. This design leverages the complementary properties characterized in Section 2.2 without designating either domain as the primary pathway.

B How CrossTF Differs from Guidance and Multi-Model Combination Methods

CrossTF's cross-view fusion is algebraically similar to guidance-based sampling (Dhariwal & Nichol, 2021; Ho & Salimans, 2022; Karras et al., 2024). As defined in Equation 5, CrossTF linearly combines two clean-sample estimates and allows extrapolation along their difference when $\alpha_i \notin [0, 1]$. This naturally raises the question of whether CrossTF is simply another form of guidance or multi-model combination (Balaji et al., 2022; Liu et al., 2022). The key distinction lies not in the linear combination itself, but in how the two estimates are obtained, why their difference is useful, and how the fusion parameters are determined.

Difference from guidance-based sampling. Classifier guidance steers the generation process using the gradient of an auxiliary classifier (Dhariwal & Nichol, 2021), while classifier-free guidance uses the difference between conditional and unconditional predictions to strengthen conditioning (Ho & Salimans, 2022). Autoguidance further uses a deliberately smaller or less-trained model as a reference and guides sampling away from its errors (Karras et al., 2024). CrossTF relies on neither a conditioning difference nor a deliberately weakened reference model. Its time- and frequency-view denoisers are independently trained on the same data using the same EDM denoising objective, noise distribution, and network configuration; they differ only in the representations of the same intermediate sample they receive. Therefore, the cross-view direction in CrossTF represents neither a conditional–unconditional difference nor a strong–weak model difference, but rather the discrepancy between two parallel representation views of the same sample.

Importantly, CrossTF does not assume that one view is uniformly superior to the other. On the Precipitation dataset, for example, the time view performs better on magnitude and change rates,

whereas the frequency view performs better on magnitude and change excesses, indicating complementary strengths for different types of extreme behavior. Moreover, Table 1 shows that, across seven datasets, varying the fusion parameter along the cross-view direction changes extreme fidelity substantially more than the denoising loss. CrossTF therefore exploits a direction that naturally arises from representation discrepancy and is particularly sensitive to extreme fidelity, rather than relying on a predefined guidance direction.

Difference from post-training sampler optimization. CrossTF is also related to methods that optimize the sampling procedure after model training. Align Your Steps, for example, optimizes the discretization schedule of a fixed diffusion model (Sabour et al., 2024). CrossTF likewise freezes the trained denoisers, but optimizes six step-wise fusion parameters, $\boldsymbol{\alpha} = \{\alpha_i\}_{i=1}^{6}$, within the cross-view window rather than the noise schedule. These parameters are optimized using the EVT-based extreme objective under an energy-distance constraint that limits degradation in overall distributional fidelity. Consequently, whether a step performs interpolation or extrapolation is determined by the constrained optimization rather than specified in advance as a guidance scale. Outside the cross-view window, CrossTF uses the time view alone.

Difference from multi-model combination. CrossTF also differs from methods in which multiple denoisers are assigned different generative roles. For example, eDiff-I trains experts for different noise-level intervals and activates the corresponding expert at each stage (Balaji et al., 2022), while compositional diffusion combines models associated with different conditions to generate samples satisfying their conjunction (Liu et al., 2022). In CrossTF, the two denoisers correspond neither to different noise stages nor to different semantic conditions. Both model the same unconditional data distribution and simultaneously produce complete clean-sample estimates of the same intermediate sample within the cross-view window. Their representation-specific discrepancy then defines the cross-view direction used to optimize extreme fidelity.

In summary, the similarity between CrossTF and guidance or multi-model methods is primarily algebraic: all involve combining multiple estimates. Their underlying mechanisms are different. In CrossTF, the two endpoints arise from parallel time and frequency views of the same data distribution, the cross-view direction is induced by representation discrepancy and empirically exhibits strong sensitivity to extreme fidelity, and the step-wise fusion parameters are learned through post-training optimization under an overall-fidelity constraint. CrossTF therefore does not rely on conditioning differences, deliberately weakened reference models, or predefined expert roles.

C Additional Preliminaries

C.1 Visualization of Representative Extremes in Time-series

We visualize representative extremes from five real-world datasets, covering diverse patterns such as burst surges, sustained bursts, impulse spikes, abrupt dips, and trend breaks. As shown in Figure 8, these extremes exhibit substantially different shapes, durations, and local structures, making it difficult to describe them through a single event pattern. Despite this diversity, their extreme behavior can be largely reflected through two fundamental aspects: unusually large deviations from the typical level and abrupt changes between consecutive time steps. These examples further motivate our characterization of extremes through *extreme magnitude* and *extreme change*.

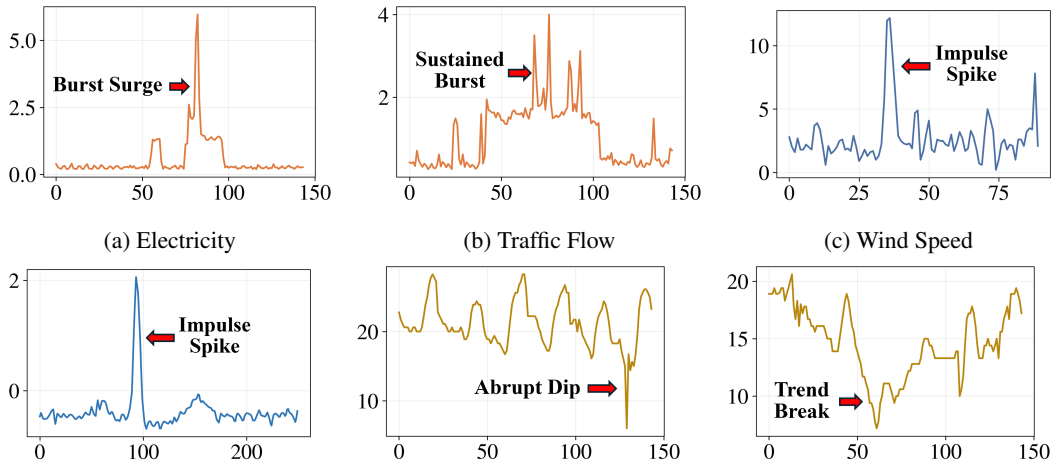
C.2 Additional Analysis of Cross-View Asymmetry

This section provides additional details on the experimental setup and statistical procedure used for the Extreme-Sensitive Cross-View Asymmetry analysis in Table 1.

For the asymmetry analysis, we follow the cross-view formulation in Section 3.1. At denoising step i, the fused clean-sample estimate is

$$\hat{\mathbf{x}}_{0,i} = \hat{\mathbf{x}}_{0,i}^{F} + \alpha_i \left(\hat{\mathbf{x}}_{0,i}^{T} - \hat{\mathbf{x}}_{0,i}^{F} \right), \tag{9}$$

where $\alpha_i = 0$ and $\alpha_i = 1$ correspond to the pure frequency-view and time-view estimates, respectively. In this analysis, we use a shared fusion parameter $\alpha_i = \alpha$ across all steps within the cross-view window and consider $\alpha \in \{0, 0.25, 0.5, 0.75, 0.9, 1.0\}$.



(a) Electricity (b) Traffic Flow (c) Wind Speed

(d) ECG (e) Temperature (f) Temperature

Fig. 8: Representative extremes across different real-world time-series datasets.

We use three independently trained time/frequency denoiser pairs with training seeds 4, 5, and 6, and evaluate their EMA weights (Karras et al., 2022). Generation uses two fixed sampling seeds, 101 and 102. For each training and sampling seed, we first execute the denoising steps preceding the cross-view window and cache the resulting intermediate state. All α values then continue sampling from the same cached state, ensuring that their differences arise only from cross-view fusion within the window rather than from different sampling trajectories.

For any quantity q, we define its Relative Max–Min Range as

$$\mathrm{Range}(q) = \frac{\max_\alpha q(\alpha) - \min_\alpha q(\alpha)}{|q(\alpha = 1)|}, \tag{10}$$

where the pure time-view value is used for normalization. We first compute the range for each individual α-response curve and then average the resulting ranges, rather than averaging the curves before computing the range. Specifically, *Loss* is evaluated separately for each training seed and each noise level σ within the cross-view window; *Extreme* is evaluated separately for each of the four EVT-based metrics and each training seed; and *Overall* is evaluated for each independent generation run. The metrics used for *Extreme* and *Overall* follow the definitions in the main evaluation.

The denoising loss is evaluated on real data. For each noise level σ_i within the cross-view window, the time and frequency branches use the same noise realization and produce clean-sample estimates $\hat{\mathbf{x}}^T_{0,i}$ and $\hat{\mathbf{x}}^F_{0,i}$. We define their denoising errors as

$$e_T = \hat{\mathbf{x}}^T_{0,i} - \mathbf{x}_0, \qquad e_F = \hat{\mathbf{x}}^F_{0,i} - \mathbf{x}_0. \tag{11}$$

Following the cross-view fusion in Section 3.1, the fused error is therefore

$$e_\alpha = e_F + \alpha(e_T - e_F), \tag{12}$$

and the corresponding loss is

$$L(\alpha) = \|e_F + \alpha(e_T - e_F)\|^2. \tag{13}$$

Since $L(\alpha)$ is quadratic in α, we compute its coefficients once and evaluate it analytically over the entire α grid, eliminating the need for additional forward passes at each α value.

In Table 1, Ext./Loss and Ext./Overall denote, for each dataset, the ratios of the Extreme range to the Loss and Overall ranges, respectively. In the Median row, the first three entries report the median Loss, Overall, and Extreme ranges across the seven datasets, and the last two entries divide the median Extreme range by the median Loss and Overall ranges, respectively, using the unrounded values.

To further assess whether the observed variation is attributable to α rather than ordinary seed-level variability, Table 4 compares the α-induced range with the variation across training seeds. For

Table 4: **Comparison between α-induced variation and seed variation** across seven datasets.

	Loss		Overall		Extreme	
Dataset	α Var.	Seed Var.	α Var.	Seed Var.	α Var.	Seed Var.
Traffic	4.5%	1.45%	8.4%	9.21%	290.4%	17.10%
Transportation	7.5%	1.70%	51.1%	12.50%	254.2%	10.81%
Temperature	7.3%	1.70%	12.6%	14.49%	145.6%	9.62%
Electricity	8.7%	0.28%	78.4%	3.76%	85.2%	4.50%
Precipitation	33.3%	1.01%	339.9%	4.63%	160.3%	14.73%
ECG	16.7%	1.20%	3.9%	14.08%	38.2%	9.15%
Building	12.9%	2.76%	10.8%	9.87%	22.8%	6.10%

Extreme, the variation induced by α is consistently larger than the seed variation across all seven datasets, often by a substantial margin, indicating that the observed sensitivity is not simply caused by model instability. For *Overall*, the α-induced variation is comparable to or smaller than the seed variation on Traffic, Temperature, ECG, and Building, while larger changes are observed on Transportation, Electricity, and Precipitation, suggesting that overall quality is relatively stable on some datasets but not uniformly invariant. For *Loss*, the absolute variation remains modest on most datasets, with Precipitation showing the largest change. Taken together, these results reinforce that the cross-view direction has a substantially stronger and more consistent effect on extreme fidelity than would be expected from seed-level variability.

C.3 Details of the Cross-View Window

We provide the definition and estimation of Stable Discrepancy Share, followed by the experimental settings of the view-switching ablation.

Stable Component: For a fixed real sample $\mathbf{x}_0^{(n)}$, noise level σ_i, and trained denoiser pair, we construct time and frequency representations from the same noisy input. Let $\boldsymbol{\Delta}_i^{(n)}(\boldsymbol{\epsilon})$ denote the difference between their clean-sample estimates, where n indexes real samples, i denotes the denoising step, and $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ is Gaussian noise. We define the stable component as the expected discrepancy over noise realizations:

$$\boldsymbol{\mu}_i^{(n)} = \mathbb{E}_{\boldsymbol{\epsilon}} \left[\boldsymbol{\Delta}_i^{(n)}(\boldsymbol{\epsilon}) \right]. \tag{14}$$

Here, $\boldsymbol{\mu}_i^{(n)}$ captures the mean cross-view discrepancy for a fixed sample and noise level, independently of any particular noise realization.

Deviation Component: Each discrepancy can then be expressed as the sum of its stable component and a deviation component:

$$\boldsymbol{\xi}_i^{(n)}(\boldsymbol{\epsilon}) = \boldsymbol{\Delta}_i^{(n)}(\boldsymbol{\epsilon}) - \boldsymbol{\mu}_i^{(n)}, \qquad \mathbb{E}_{\boldsymbol{\epsilon}} \left[\boldsymbol{\xi}_i^{(n)}(\boldsymbol{\epsilon}) \right] = \mathbf{0}. \tag{15}$$

The residual $\boldsymbol{\xi}_i^{(n)}(\boldsymbol{\epsilon})$ captures fluctuations across noise realizations. Its zero expectation implies that the expected squared norm of the discrepancy decomposes into the energies of the stable and deviation components.

Stable Discrepancy Share: We therefore define the proportion of total discrepancy energy contributed by the stable component as

$$s_i = \frac{\sum_{n=1}^{N} \left\| \boldsymbol{\mu}_i^{(n)} \right\|_2^2}{\sum_{n=1}^{N} \mathbb{E}_{\boldsymbol{\epsilon}} \left[\left\| \boldsymbol{\Delta}_i^{(n)}(\boldsymbol{\epsilon}) \right\|_2^2 \right]}. \tag{16}$$

Here, N is the number of real samples, $\| \cdot \|_2^2$ denotes the squared Euclidean norm, and s_i is the Stable Discrepancy Share at step i. When the total energy is nonzero, $s_i \in [0, 1]$, with larger values indicating a greater stable proportion. Energies are summed across samples before taking the ratio.

In practice, we fix $N = 128$ real series and draw $K = 16$ noise realizations per series at each noise level, using identical noisy inputs for both views. We estimate the stable component by the sample

mean of these discrepancies. To correct the upward bias from finite sampling, we subtract $1/K$ of the summed unbiased sample variances across dimensions from the squared norm of this mean. The share is computed separately for each training seed before aggregation, allowing us to assess both stability across noise realizations and the reproducibility of this stability across training seeds.

View-Switching Ablation: The above statistic measures discrepancy stability at individual noise levels. To examine how view selection within and outside the window affects the final generated samples, we evaluate four configurations along complete sampling trajectories: Time-Only, Frequency-Only, time outside and frequency inside the window, and frequency outside and time inside the window. We use the window defined in the main text, $\mathcal{W} = \{i : \sigma_i \leq 0.35\} = \{6, \ldots, 1\}$.

Let M^q_{ab} denote extreme-fidelity metric q when view a is used outside the window and view b inside it, where $a, b \in \{T, F\}$ denote the time and frequency views. We compute the switching effect for each segment by averaging over the two view assignments in the other segment:

$$\delta^q_{\text{in}} = \frac{(M^q_{TT} - M^q_{TF}) + (M^q_{FT} - M^q_{FF})}{2}, \tag{17}$$

$$\delta^q_{\text{out}} = \frac{(M^q_{TT} - M^q_{FT}) + (M^q_{TF} - M^q_{FF})}{2}. \tag{18}$$

Here, δ^q_{in} and δ^q_{out} are the average switching effects inside and outside the window, respectively. Within each run, we average their absolute values over Mag-Rate, Chg-Rate, Mag-Excess, and Chg-Excess to prevent cancellation across metrics. We then aggregate over nine runs formed by three training seeds and three sampling seeds. The ratio of the resulting mean effect magnitudes inside and outside the window gives the fold difference reported in the main text. Within each run, all four configurations share the initial noise, denoisers, and deterministic sampler, differing only in view assignment. This ablation measures the magnitude of changes in extreme fidelity, rather than improvements from switching views.

C.4 Implementation Details of EDM-Based Diffusion

We provide additional implementation details of the EDM-based diffusion framework used in CrossTF. For each view $v \in \{T, F\}$, the denoiser processes the corresponding 2D representation X^v_σ and maps its output back to the 1D time domain through a fixed, non-learned inverse transform to obtain the clean-sample estimate. Both the time- and frequency-view denoisers follow the standard EDM preconditioning and sampling configuration (Karras et al., 2022), while differing in their view-specific 2D representations.

EDM Preconditioning. For each view, EDM preconditioning is applied to the corresponding 2D representation X^v_σ:

$$\widetilde{D}^v_{\theta_v}(X^v_\sigma, \sigma) = c_{\text{skip}} X^v_\sigma + c_{\text{out}} F^v_{\theta_v}\left(c_{\text{in}} X^v_\sigma, c_{\text{noise}}\right), \tag{19}$$

where

$$c_{\text{skip}} = \frac{\sigma^2_{\text{data}}}{\sigma^2 + \sigma^2_{\text{data}}}, \qquad c_{\text{out}} = \frac{\sigma \sigma_{\text{data}}}{\sqrt{\sigma^2 + \sigma^2_{\text{data}}}}, \tag{20}$$

and

$$c_{\text{in}} = \frac{1}{\sqrt{\sigma^2 + \sigma^2_{\text{data}}}}, \qquad c_{\text{noise}} = \frac{1}{4}\log\sigma. \tag{21}$$

Following the standard EDM configuration (Karras et al., 2022), we set $\sigma_{\text{data}} = 0.5$. The preconditioned output $\widetilde{D}^v_{\theta_v}$ remains in the corresponding 2D representation space. It is then mapped back to the 1D time domain through the fixed, non-learned inverse transform, yielding $D^v_{\theta_v}(X^v_\sigma, \sigma) \in \mathbb{R}^T$, the clean-sample estimate used in Section 3.1.

View-Specific Denoiser Architecture. The time- and frequency-view denoisers are implemented as two independently trained models with the same backbone configuration. Each denoiser takes the corresponding 2D representation X^v_σ as input and processes it using a 2D DhariwalUNet (Dhariwal & Nichol, 2021). The two branches are trained independently and stored as separate checkpoints, and are combined only during CrossTF sampling.

Both branches share the same DhariwalUNet backbone configuration, with a base channel width of 64, four resolution levels with channel multipliers $[1, 2, 2, 2]$, and one residual block per level. For

Table 5: View-specific representation configurations across datasets.

Dataset	T	Delay	Emb.	Time Res.	(n_{fft}, h)	Frequency Res.
TRAFFIC	144	3	48	48×48	$(95, 3)$	48×48
PRECIP.	90	2	32	32×32	$(63, 3)$	32×32
BUILD.	144	3	48	48×48	$(95, 3)$	48×48
TEMP.	144	3	48	48×48	$(95, 3)$	48×48
ELECT.	144	3	48	48×48	$(95, 3)$	48×48
TRANSP.	144	3	48	48×48	$(95, 3)$	48×48
ECG	250	3	64	64×64	$(127, 4)$	64×64

the 48×48 configuration, self-attention is applied at spatial resolutions 24×24, 12×12, and 6×6, with 64 channels per attention head. The network uses a dropout rate of 0.1 and adaptive scale-and-shift conditioning based on the EDM noise embedding. For each dataset, the time and frequency branches use matching spatial resolutions, with their view-specific configurations summarized in Table 5.

Training Configuration. Both view-specific denoisers are trained using the EDM denoising objective in Equation 6. During training, the noise level is sampled as

$$\log \sigma \sim \mathcal{N}(-1.2, 1.2^2), \tag{22}$$

with the standard EDM weighting function

$$\lambda(\sigma) = \frac{\sigma^2 + \sigma_{\text{data}}^2}{(\sigma \sigma_{\text{data}})^2}. \tag{23}$$

The same noise distribution, loss formulation, and network configuration are used for both views, while their view-specific representation configurations follow Table 5.

Sampling Configuration. For generation, we use an 18-step Karras noise schedule (Karras et al., 2022) with $\rho = 7$, $\sigma_{\max} = 80$, and $\sigma_{\min} = 0.002$. We set $S_{\text{churn}} = 0$ and use deterministic sampling with an Euler prediction followed by a second-order Heun correction (Karras et al., 2022; Hairer et al., 1993). The Heun correction is omitted for the final step ending at $\sigma = 0$. Thus, an 18-step trajectory requires 35 denoiser evaluations for a single view.

D ADDITIONAL EXPERIMENTAL DETAILS AND RESULTS

D.1 DATASETS

- **Traffic**: We use the *PEMS-BAY* dataset released with DCRNN (Li et al., 2018; California Department of Transportation), which records freeway traffic speeds from 325 loop detectors in the San Francisco Bay Area. The extremes in this dataset are the congestion events, e.g., the abrupt collapses of speed at the onset of a jam, which take a detector from free-flow velocity to stop-and-go within a few intervals and hold it there through the peak period. Speed is reported in km/h and represents the mean vehicle velocity observed by a detector within a given interval. The data are sampled every 5 minutes from January 1, 2017 to June 30, 2017. To construct our benchmark, we randomly select 10 detectors (seed 42) whose fraction of missing readings is below 0.001, partition each detector record into daily windows, and subsample to a 10-minute resolution, so each day is represented by a sequence of length 144. Since zero denotes a missing reading, any day containing zero values is discarded (20 of 1810). Treating each detector-day pair as one sample yields a dataset of shape $(1790, 144, 1)$. This construction retains the strong daily commuting pattern while preserving the sharp speed drops associated with congestion. The sample visualization is shown in Fig. 9a.
- **Precipitation**: We use the *Daily Weather Data* provided by Dewey (Custom Weather, 2022a), which contains daily weather observations from ground stations. Extreme events in this dataset consist of heavy-rainfall episodes, including those associated with tropical systems, that produce substantial precipitation over one or a few days amid otherwise dry

conditions. We select 88 observation stations across Florida and use daily precipitation totals, as the hourly records contain substantial missing data. Records from January 1, 2023, to December 31, 2025, are partitioned into 90-day samples, yielding a dataset of shape $(1056, 90, 1)$. Among the seven datasets, Precipitation exhibits the strongest zero inflation, with 59.8% of all time steps equal to zero. Its marginal distribution therefore combines a large point mass at zero with a long right tail, and its extremes typically appear as isolated bursts rather than smooth excursions. The sample visualization is shown in Fig. 9b.

- **Building Power**: We use the *Building Data Genome Project 2* dataset (Miller et al., 2020), which records whole-building electricity consumption from non-residential buildings on university and institutional campuses. The extremes in this dataset are the peak-demand periods, e.g., the hours of heaviest HVAC and occupancy load, typically on hot weekday afternoons, during which a building draws far more power than it does at night or over a weekend. The value is reported in kWh and represents the electricity consumed by an entire building within a given interval. The data are sampled hourly from January 1, 2016 to December 31, 2017. To construct our benchmark, we randomly select 16 meters (seed 42) from the 113 meters with complete and non-degenerate records over the full period, and partition each record into non-overlapping 144-hour (six-day) windows; flat or incomplete windows are discarded (98 of 1936). Treating each meter-window pair as one sample yields a dataset of shape $(1838, 144, 1)$. This construction retains both daily occupancy patterns and weekday–weekend variation within each six-day window, at a finer spatial aggregation than a grid-level load series. The sample visualization is shown in Fig. 9c.
- **Temperature**: We use the *Hourly Weather Data* provided by Dewey (Custom Weather, 2022b), which contains hourly climate observations across the United States since 2018. The extremes in this dataset are the cold snaps and heat spells that interrupt the ordinary diurnal and seasonal rhythm, lasting from several hours to a few days. We select five nearby stations around Jacksonville in northeastern Florida, a region frequently affected by extreme weather such as heavy rainfall and hurricanes. The value is reported in °C and stored at a resolution of $0.1\,^{\circ}$C, inherited from an original reporting granularity of $1\,^{\circ}$F. The temperature records are divided into six-day samples, each containing 144 hourly observations (24×6). Using data from these five stations over the period from January 1, 2021 to December 31, 2025, we obtain 1,520 samples, yielding a dataset of shape $(1520, 144, 1)$, with values ranging from -7.0 to $44.4\,^{\circ}$C. Among the seven datasets, Temperature exhibits the clearest diurnal periodicity, and its extremes appear as broad, sustained excursions rather than isolated spikes. This construction captures both long-term temporal variation and local spatial variation across nearby stations. The sample visualization is shown in Fig. 9d.
- **Electricity**: We use the *UCI Individual Household Electric Power Consumption* dataset (Hebrail & Berard, 2006), which records minute-level household electricity usage from a single residence near Paris, France over an extended period. Extreme events appear as short-duration load spikes, consistent with the operation of high-power appliances, that substantially exceed typical household demand. We focus on the `Global_active_power` variable, reported in kW, which represents the household's instantaneous active power demand, 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, yielding a dataset of shape $(1096, 144, 1)$. The spikes are not tightly phase-locked to the daily pattern they ride on, which makes the extremes of this dataset hard to anticipate from its periodic component alone. The sample visualization is shown in Fig. 9e.
- **Transportation**: We use the *PEMS-SF* dataset derived from the California Performance Measurement System (PeMS) (Cuturi, 2011; California Department of Transportation), which records freeway traffic occupancy from monitoring stations in the San Francisco Bay Area. The extremes in this dataset are the heavily congested stretches of the day, during which a sensor stays covered by vehicles for a large fraction of each interval instead of the brief moments typical of free flow. The occupancy value ranges from 0 to 1 and represents the fraction 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 construct our benchmark, we select three

nearby monitoring stations and extract their daily occupancy sequences separately, treating each station-day pair as one sample. This yields a dataset of shape $(1320, 144, 1)$. This construction captures realistic daily traffic dynamics while retaining local variation across nearby monitoring locations. The sample visualization is shown in Fig. 9f.

- **Electrocardiogram**: We use the *Long-Term ST Database* (LTST) (Jager et al., 2003; Goldberger et al., 2000), which contains long-duration ambulatory ECG recordings with expert ST-related annotations. Extreme values in the selected signals appear as sharp, short-duration amplitude excursions, particularly the prominent peaks of QRS complexes. During preprocessing, we load the raw signals and annotation files and extract the first ECG lead as a one-dimensional time series. The annotations locate the abnormal ST-related intervals, which provide clinically meaningful references for 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 the candidates by the mean absolute amplitude within each window and randomly select 1,000 samples from the high-score set, yielding a dataset of shape $(1000, 250, 1)$. The value is reported in mV, stored at a 0.005 mV quantization step, and ranges from -2.045 to 2.940 mV. Electrocardiogram is the only physiological dataset in our benchmark and carries its most impulsive extremes. The sample visualization is shown in Fig. 9g.

D.2 METRICS

To comprehensively evaluate the performance of our CrossTF, we use ten metrics from two perspectives: extreme fidelity and overall performance. All metrics are lower-is-better. Below, we provide the definitions and implementation details of each metric. Throughout, let $\mathcal{X}_{\text{real}} = \{\mathbf{x}^{(i)}\}_{i=1}^{N}$ and $\mathcal{X}_{\text{gen}} = \{\tilde{\mathbf{x}}^{(i)}\}_{i=1}^{N}$ denote the real and generated sets of time-series of length T, respectively, and let $P(\cdot)$ denote the corresponding distribution.

For **extreme fidelity**, we introduce four metrics that characterize extremes along two dimensions: *how often* a series becomes extreme and *how far* it exceeds the extreme threshold. We evaluate both dimensions for two notions of extremes: magnitude and change. For a series $\mathbf{x} \in \mathbb{R}^T$, we define the magnitude and change scores as

$$z_t(\mathbf{x}) = |x_t - \text{med}(\mathbf{x})|, \quad t = 1, \dots, T, \qquad w_t(\mathbf{x}) = |x_{t+1} - x_t|, \quad t = 1, \dots, T-1, \tag{24}$$

where $\text{med}(\cdot)$ denotes the temporal median. Median centering provides a robust reference for measuring deviations from the typical level of each series. The scores remain in the original data units so that systematic magnitude discrepancies between the real and generated data remain visible.

We determine a single pair of thresholds from the pooled *real* scores using a tail fraction of $\tau = 0.05$:

$$u_{\text{mag}} = Q_{1-\tau}\left(\left\{z_t(\mathbf{x}^{(i)})\right\}_{i,t}\right), \qquad u_{\text{chg}} = Q_{1-\tau}\left(\left\{w_t(\mathbf{x}^{(i)})\right\}_{i,t}\right), \tag{25}$$

where $Q_{1-\tau}(\cdot)$ denotes the empirical $(1-\tau)$-quantile. These thresholds are estimated only from the real data and are subsequently fixed for all generated sets. This design allows the exceedance frequency of generated samples to differ from that of the real data. In contrast, using a separate rank-based threshold for each series would fix its exceedance frequency at approximately τ by construction and therefore obscure frequency mismatches.

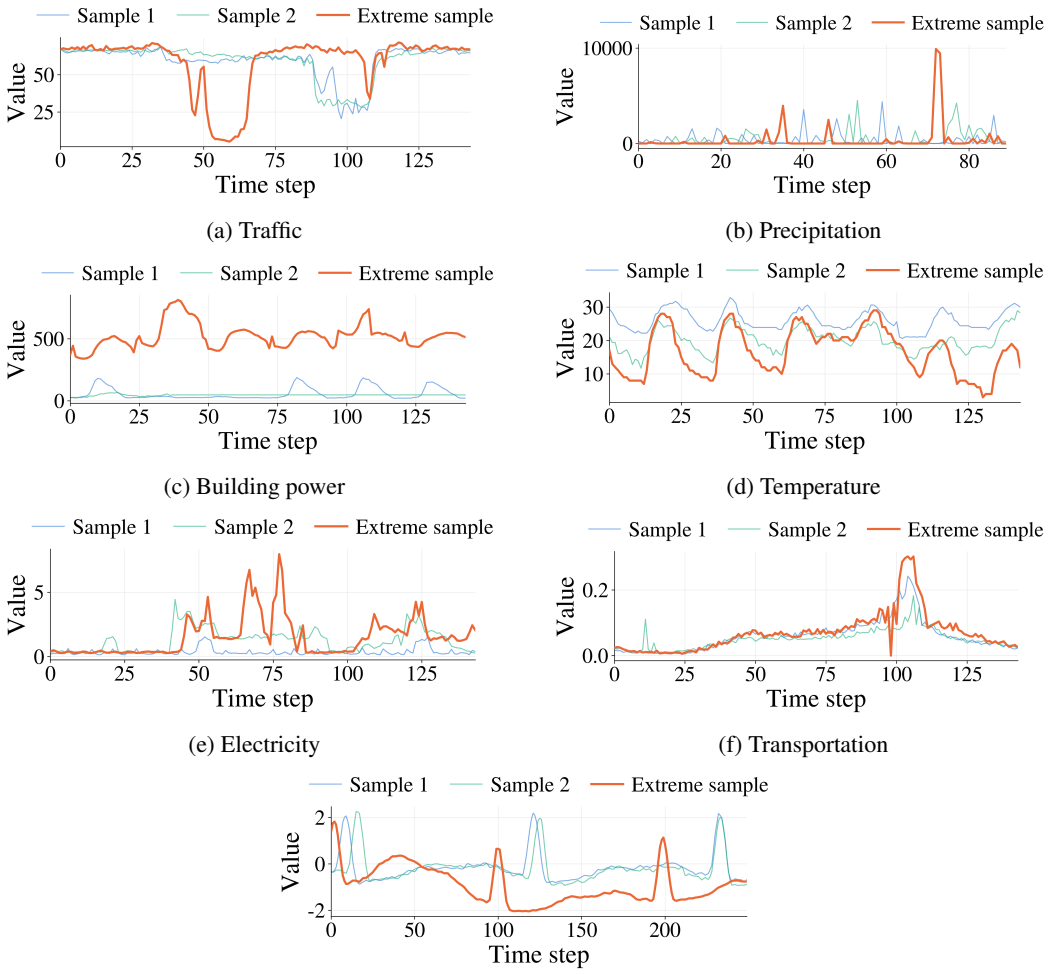
Each of the four quantities below produces one value per series. We compare the distributions of these per-series values using the 1-Wasserstein distance, normalized by the standard deviation of the corresponding real values:

$$\text{Dist}(q) = \frac{W_1(P(q(\mathcal{X}_{\text{gen}})), P(q(\mathcal{X}_{\text{real}})))}{\text{std}(q(\mathcal{X}_{\text{real}}))}, \tag{26}$$

where $q(\mathcal{X})$ denotes the collection of per-series values of quantity q over the set $\mathcal{X}$.

- **Mag-Rate:** Measures the discrepancy in how frequently a series exceeds the magnitude threshold. For each series, we compute a smooth exceedance rate:

$$r_{\text{mag}}(\mathbf{x}) = \frac{1}{T}\sum_{t=1}^{T}\sigma\left(\frac{z_t(\mathbf{x}) - u_{\text{mag}}}{\kappa\, u_{\text{mag}}}\right), \qquad \text{Mag-Rate} = \text{Dist}(r_{\text{mag}}), \tag{27}$$



(a) Traffic

(b) Precipitation

(c) Building power

(d) Temperature

(e) Electricity

(f) Transportation

(g) Electrocardiogram

Fig. 9: Representative sequences from the seven datasets described above. Each panel presents two typical samples and one sample exhibiting pronounced extreme behavior.

where $\sigma(\cdot)$ is the logistic function and κ controls the relative width of the transition around the threshold. The hard exceedance rate $|\{t : z_t > u_{\text{mag}}\}|/T$ is piecewise constant with respect to the data and therefore provides no useful gradient. The logistic relaxation provides a smooth approximation, allowing the same quantity to be used both for evaluation and as a differentiable optimization signal.

- **Chg-Rate:** Measures the corresponding discrepancy in the frequency of extreme changes by replacing z_t and u_{mag} with w_t and u_{chg}:

$$r_{\text{chg}}(\mathbf{x}) = \frac{1}{T-1}\sum_{t=1}^{T-1}\sigma\left(\frac{w_t(\mathbf{x}) - u_{\text{chg}}}{\kappa\, u_{\text{chg}}}\right), \qquad \text{Chg-Rate} = \text{Dist}(r_{\text{chg}})\,. \tag{28}$$

- **Mag-Excess:** Measures the discrepancy in how far magnitude extremes extend beyond the threshold. For each series, we compute the mean excess over all exceeding time points:

$$e_{\text{mag}}(\mathbf{x}) = \frac{1}{\max(|\mathcal{T}_{\text{mag}}|, 1)}\sum_{t\in\mathcal{T}_{\text{mag}}}(z_t(\mathbf{x}) - u_{\text{mag}})\,, \qquad \text{Mag-Excess} = \text{Dist}(e_{\text{mag}})\,, \tag{29}$$

where $\mathcal{T}_{\text{mag}} = \{t : z_t(\mathbf{x}) > u_{\text{mag}}\}$ denotes the set of exceeding time points. A series with no exceedance is assigned a value of 0 rather than being discarded. This retains the absence

of extreme occurrences as part of the series-level distribution and ensures that the real and generated quantities are evaluated over the same populations.

- **Chg-Excess:** Measures the corresponding discrepancy in the severity of extreme changes, with $\mathcal{T}_{\text{chg}} = \{t : w_t(\mathbf{x}) > u_{\text{chg}}\}$:

$$e_{\text{chg}}(\mathbf{x}) = \frac{1}{\max(|\mathcal{T}_{\text{chg}}|, 1)} \sum_{t \in \mathcal{T}_{\text{chg}}} (w_t(\mathbf{x}) - u_{\text{chg}}), \qquad \text{Chg-Excess} = \text{Dist}(e_{\text{chg}}). \quad (30)$$

For **overall performance**, we first report three fidelity metrics that measure the distributional and temporal similarity between the generated and real data, following common practice in time-series generation:

- **Wasserstein Distance (Wass.) (Villani, 2009)**: Measures the discrepancy between the pooled value distributions of the real and generated data. We use the 1-Wasserstein distance and normalize it by the standard deviation of the real values to improve comparability across datasets:

$$\text{Wass.} = \frac{1}{\text{std}(\mathcal{X}_{\text{real}})} \inf_{\gamma \in \Pi(P,Q)} \mathbb{E}_{(x,y)\sim\gamma} [|x - y|], \quad (31)$$

  where $P = P(\mathcal{X}_{\text{gen}})$ and $Q = P(\mathcal{X}_{\text{real}})$ denote the distributions of pooled generated and real values, respectively, and $\Pi(P, Q)$ is the set of joint distributions with marginals P and Q.
- **Marginal Distribution Difference (Marg.) (Naiman et al., 2024a):** Measures the discrepancy between the marginal value distributions of the generated and real data using histogram densities. We divide the range of the real data into B equal-width bins and use the same fixed bins for both sets. The score is the mean absolute difference between their binned densities:

$$\text{Marg.} = \frac{1}{B} \sum_{b=1}^{B} \left| \hat{p}_b^{\text{gen}} - \hat{p}_b^{\text{real}} \right|, \quad (32)$$

  where $\hat{p}_b$ denotes the density of the b-th bin. We use $B = 50$.
- **Autocorrelation Discrepancy (ACD) (Ang et al., 2023):** Measures the discrepancy in temporal dependencies by comparing the autocorrelation functions of the real and generated data:

$$\text{ACD} = \frac{1}{L} \sum_{l=1}^{L} |\rho_{\text{real}}(l) - \rho_{\text{gen}}(l)|, \quad (33)$$

  where $\rho(l)$ denotes the sample autocorrelation at lag l, averaged across all series. We use $L = 50$.

We further evaluate the downstream utility of the generated data using three metrics. For these metrics, all data are min–max normalized using the per-channel range of the real set, following the evaluation conventions adopted in prior work:

- **Discriminative Score (Disc) (Naiman et al., 2024a):** Evaluates how distinguishable the generated sequences are from the real ones. A GRU classifier g_ϕ is trained to distinguish real from generated sequences, and the score is defined as the deviation of its held-out classification accuracy from chance:

$$\text{Disc} = \left| \text{Acc}\big(g_\phi; \mathcal{X}_{\text{real}}^{\text{test}}, \mathcal{X}_{\text{gen}}^{\text{test}}\big) - 0.5 \right|, \quad (34)$$

  where $\text{Acc}(\cdot)$ denotes classification accuracy on the held-out split. A score of 0 indicates that the classifier cannot distinguish the generated data from the real data better than chance.
- **Predictive Score (Pred) (Naiman et al., 2024a):** Evaluates whether the generated data preserve predictive temporal dynamics under the Train-on-Synthetic, Test-on-Real protocol. A GRU forecaster f_θ is trained exclusively on generated data to predict the next value from the preceding sequence:

$$\hat{x}_{t+1} = f_\theta(\mathbf{x}_{1:t}). \quad (35)$$

The trained model is then evaluated on real data using the mean absolute one-step-ahead prediction error:

$$\text{Pred} = \frac{1}{N(T-1)} \sum_{i=1}^{N} \sum_{t=1}^{T-1} \left| \hat{x}_{t+1}^{(i)} - x_{t+1}^{(i)} \right|. \tag{36}$$

Lower values indicate that models trained on generated data generalize better to the predictive dynamics of real data.

- **Context-FID (Yuan & Qiao, 2024):** Measures the Fréchet distance between contextual representations of the real and generated time series. We first train a TS2Vec encoder on the real data and then encode each real and generated sequence into a sequence-level 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. The score is computed as

$$\text{Context-FID} = \|\mu_r - \mu_g\|_2^2 + \text{Tr}\left(\Sigma_r + \Sigma_g - 2(\Sigma_r \Sigma_g)^{1/2}\right). \tag{37}$$

D.3 BASELINES

We compare CrossTF with eleven baseline methods from six different categories. An overview of these baselines is provided in Table 6. Their implementations with detailed parameter settings are included in the **baselines** directory of our codebase. The details are summarized as follows:

Table 6: Overview of the baselines.

Baseline	Category	Venue	Link
TimeGAN (Yoon et al., 2019)	GAN	NeurIPS'19	Implementation Used
KoVAE (Naiman et al., 2024b)	VAE	ICLR'24	Official Implementation
SDForger (Rousseau et al., 2025)	LLM	NeurIPS'25	Official Implementation
DiffTime (Coletta et al., 2023)	Diffusion	NeurIPS'23	Implementation Used
Diffusion-TS (Yuan & Qiao, 2024)	Diffusion	ICLR'24	Official Implementation
FourierDiff (Crabbé et al., 2024)	Diffusion	ICML'24	Official Implementation
ImagenTime (Naiman et al., 2024a)	Diffusion	NeurIPS'24	Official Implementation
RF-Diffusion (Chi et al., 2024)	TF-Based	MobiCom'24	Official Implementation
FIDE (Galib et al., 2024)	Extreme-Aware	NeurIPS'24	Official Implementation
HeavyDiff (Pandey et al., 2025)	Extreme-Aware	ICLR'25	Implementation Used
E4GEN (Jiang et al., 2026)	Extreme-Aware	NeurIPS'26	Official Implementation

Unless otherwise stated, every baseline is trained from scratch on the full array of each dataset with the hyperparameters of its public repository, normalizes the data internally and inverts that transform before writing, so all generated series are evaluated in the original units. Each baseline is run with three seeds and generates as many series as the real set contains; no sample is ever duplicated to reach that count.

- **TimeGAN** (Yoon et al., 2019): TimeGAN embeds sequences into a latent space using a recurrent autoencoder and trains a generator with an adversarial objective together with a supervised next-step prediction loss. The original implementation is based on TensorFlow 1.x and is incompatible with our current GPU environment, so we use the `tsgm` port linked in Table 6. We use it without architectural modifications and follow the original repository defaults: a three-layer GRU with hidden size 24, $\gamma = 1.0$, batch size 128, and Adam with a learning rate of 10^{-3} for all five optimizers. Each of the three training phases runs for 1000 iterations. We keep the training budget fixed in iterations rather than epochs so that datasets of different sizes receive the same number of updates.
- **KoVAE** (Naiman et al., 2024b): KoVAE is a variational autoencoder whose latent sequence is constrained to evolve under a linear Koopman operator. Its objective combines reconstruction, KL regularization, and a prior-prediction term that aligns the latent dynamics

with the learned operator. We use the official implementation with its default settings: 600 epochs, learning rate 7×10^{-4}, batch size 64, three layers, latent dimension 16, hidden dimension 20, and weights $1.0/0.007/0.005$ for the three loss terms. The paper reports tuned KL and prediction weights for only two datasets, neither of which is included in our evaluation, so we use the default values for all datasets.

- **SDForger** (Rousseau et al., 2025): SDForger compresses each series into a low-dimensional ICA representation, converts the representation into text, fine-tunes a language model on the resulting corpus, and decodes sampled text back into time series. We run IBM's `fms-dgt` databuilder without modification, fine-tuning GPT-2 for 100 epochs with a learning rate of 8×10^{-5} and batch size 32, and sampling with temperature 1.3. The ICA representation is selected to explain 70% of the variance. We modify two generation-side settings to make its output comparable with the other baselines: the upstream limit of 50–100 generated series is increased to N, and the diversity threshold that can terminate generation early is disabled. Because the decoder performs de-duplication and interquartile filtering, the final number of generated samples can be slightly smaller than N; we report the number actually obtained.
- **DiffTime** (Coletta et al., 2023): DiffTime is a denoising diffusion model for time series that supports both soft and hard constraints. We use its unconstrained generator, which is directly comparable with the other baselines. Since no public implementation is available, we take the diffusion hyperparameters from Table 5 of the paper and use the CSDI architecture on which DiffTime is based. Specifically, we use four residual layers, 64 channels, eight attention heads, a diffusion embedding dimension of 128, and 50 noise steps with a quadratic schedule from 10^{-6} to 0.5, trained for 10,000 gradient steps with batch size 16. We use CSDI's learning rate of 10^{-3} rather than the 10^{-4} reported for DiffTime, since the latter leaves the generated mean biased by approximately one standard deviation under this training budget.
- **Diffusion-TS** (Yuan & Qiao, 2024): Diffusion-TS uses an interpretable transformer that decomposes each sequence into trend and seasonal components and predicts the clean signal under a Fourier-weighted objective. We use the official implementation with three encoder and three decoder layers, $d_{\text{model}} = 64$, four attention heads, 1000 diffusion steps with a cosine schedule, an ℓ_1 objective, and 20,000 gradient steps with batch size 64. The released configurations target univariate sequences of length 24 and therefore do not directly cover the sequence lengths of 90–250 used in our evaluation. We extrapolate the model capacity according to how the released configurations scale with sequence length and channel count, and use the upper end of the resulting range to avoid under-parameterizing the baseline.
- **FourierDiff** (Crabbé et al., 2024): FourierDiff trains a transformer-based score model with a variance-preserving SDE over the discrete Fourier coefficients of a series and maps generated samples back to the time domain through an inverse transform. We use the official implementation with $d_{\text{model}} = 72$, ten layers, twelve attention heads, learning rate 10^{-3}, batch size 64, and 1000 sampling steps. We modify two settings from the released configuration. First, we enable the frequency-domain path because the released default corresponds to the time-domain ablation rather than the proposed frequency-domain model. Second, we train for 1000 rather than 200 epochs, since 200 epochs correspond to only a few thousand updates on datasets of our size and lead to divergence on ECG. This training budget is selected once and applied consistently across all datasets rather than tuned per dataset.
- **ImagenTime** (Naiman et al., 2024a): ImagenTime converts each time series into a two-dimensional image through an invertible transformation, generates samples using a standard image diffusion model, and maps the generated images back to time series. We use the official implementation with delay embedding (delay 4 and embedding dimension 32) at a resolution of 32×32, a U-Net with 128 base channels, 18 diffusion steps, batch size 8, and learning rate 10^{-4}. CrossTF is built on the same codebase and backbone architecture, making ImagenTime a direct reference for evaluating the effect of the proposed cross-view design.
- **RF-Diffusion** (Chi et al., 2024): RF-Diffusion is a complex-valued conditional diffusion model whose forward process attenuates frequency bins at different rates, providing its

time-frequency modeling mechanism. Since our data are real-valued, univariate, and unconditional, we fix the conditioning input to a zero vector and represent each real-valued series as a complex signal with zero imaginary component, without modifying the model architecture. We also increase the blur-schedule parameter from the value released for the signal-generation tasks to the value used for the other two tasks. With the former setting, the blur kernel is numerically indistinguishable from one across frequency bins on our data, effectively removing the frequency-dependent attenuation. Because the released data loader normalizes each series independently by its mean and standard deviation, we restore the amplitude of each generated series by sampling a $(\text{mean}, \text{standard deviation})$ pair from the real dataset. Training otherwise follows the published settings: 32 blocks, eight attention heads, batch size 32, learning rate 10^{-3}, and 100 diffusion steps.

- **FIDE** (Galib et al., 2024): FIDE is a conditional diffusion model that amplifies high-frequency components and regularizes the distribution of block maxima using a generalized extreme value (GEV) likelihood. After examining its released implementation, we find that **the code contains multiple implementation issues that prevent the method from being executed as described in the paper**, including: **(1)** the GEV regularization term is detached from the autograd graph before computation and therefore does not contribute gradients during backpropagation; **(2)** the ablation branch without the GEV regularizer cannot run because a returned variable is undefined; **(3)** the clean-signal estimate used for GEV regularization omits the noise-schedule scaling required by DDPM; and **(4)** some scheduled parameters and network layers are defined but never participate in training. Consequently, the training procedure executed by the released code differs from the FIDE method described in the paper, most notably because its core GEV regularization does not actually update the model parameters. To faithfully evaluate the proposed FIDE method, we correct these implementation issues while keeping the remaining model architecture, noise schedule, sampler, and major hyperparameters unchanged.

  Specifically, we recompute the GEV log-density within the autograd graph, correct the clean-signal estimate used in the denoising step, and apply a lower bound to the log-density to avoid numerical divergence on heavy-tailed datasets. The remaining model architecture, noise schedule, and sampler follow the released implementation, including a hidden size of 64, 100 diffusion steps, a regularization weight of 0.05, and a high-frequency boost of 1.1 applied to 20% of the spectrum. We train for 2000 epochs rather than the released 400 epochs because, under its batch-size setting, each epoch contains only one full-batch update for datasets of our scale. We also do not use the released evaluation code. Its histogram-based metrics set both the number of bins and the histogram range according to the number of samples, causing the actual data to occupy only approximately 0.079% of the specified range and roughly four effective bins. Recomputing the same samples over their actual data range increases the Jensen–Shannon divergence and KL divergence by approximately $94\times$ and $129\times$, respectively. Therefore, FIDE, like all other baselines, is evaluated using the unified evaluation pipeline described in Appendix D.2.

- **HeavyDiff** (Pandey et al., 2025): HeavyDiff replaces the Gaussian noise in a diffusion model with heavy-tailed Student-t noise to better accommodate heavy-tailed time-series distributions. Since no public implementation is available, we reimplement HeavyDiff on top of Diffusion-TS. The forward process draws Student-t noise with $\nu = 2.5$ and rescales it to match the variance assumed by the backbone's noise schedule, while the reverse process is adapted accordingly. All remaining settings follow the Diffusion-TS backbone, including 1000 diffusion steps, an ℓ_1 objective, a cosine schedule, and the same training budget, so the primary difference from the corresponding diffusion baseline is the noise distribution.

- **E4GEN** (Jiang et al., 2026): E4GEN is an event-level extreme-enhanced generator that first trains a diffusion backbone, predicts extreme events, and then steers generation through a ControlNet conditioned on the predicted events. We use the released pipeline without modification except for the threshold used to define extreme events. The released configurations use the 90th percentile, which we change to the 95th percentile to match the tail fraction used by our extreme-fidelity metrics. This aligns the definition of extremes between generation and evaluation; all other method-specific settings are retained as released.

D.4 ABLATION STUDY

To answer RQ4, we conduct an ablation study to examine whether the performance gains of CrossTF arise from our cross-view fusion design rather than from either view alone. All settings use the same time and frequency denoisers, training configurations, and sampling procedure; the only difference is the fusion parameters α_i used during sampling. The full CrossTF model uses the learned step-wise fusion parameters α_i; TIME-ONLY fixes $\alpha_i = 1$ at all denoising steps and uses only the time view; FREQUENCY-ONLY fixes $\alpha_i = 0$ and uses only the frequency view. This comparison directly evaluates the gains brought by cross-view fusion over single-view generation.

Tables 7 and Table 8 summarize the performance of CrossTF and the two single-view variants in terms of extreme fidelity and overall fidelity, respectively. Together, the two tables show that cross-view fusion substantially improves extreme fidelity while largely maintaining overall generation quality. Specifically, for extreme fidelity, CrossTF achieves an average improvement of 18.0% over TIME-ONLY and performs better on six of the seven datasets. Compared with FREQUENCY-ONLY, the average improvement further increases to 48.0%, with gains observed on all seven datasets. These results indicate that the performance gains of CrossTF do not arise from either individual view alone, but from the complementarity between the two views and their cross-view fusion. In contrast, for overall fidelity, CrossTF achieves only a 0.1% average improvement over TIME-ONLY, indicating that its overall performance remains essentially unchanged. Compared with FREQUENCY-ONLY, however, CrossTF still achieves an average improvement of 25.1% and performs better on all seven datasets. These results are consistent with the design goal of CrossTF: leveraging cross-view complementarity to improve extreme generation while maintaining overall distributional fidelity. Table 9 further reports the detailed extreme- and overall-fidelity metrics across all seven datasets.

Table 7: Detailed cross-view ablation on **extreme fidelity** across seven datasets. Extreme fidelity is summarized by the geometric mean of the four extreme-fidelity metrics. "Gain" denotes the relative improvement of CrossTF over each single-view baseline, where positive values indicate improvement. Lower is better. Best and second-best fidelity values are shown in bold and underlined, respectively.

Dataset	Time-Only	Freq.-Only	CrossTF	Gain vs. Time	Gain vs. Freq.
ELECT.	0.1531	<u>0.1018</u>	**0.0691**	+54.9%	+32.1%
BUILD.	<u>0.1031</u>	0.1280	**0.0762**	+26.0%	+40.4%
PRECIP.	<u>0.0963</u>	0.1899	**0.0755**	+21.7%	+60.3%
TEMP.	<u>0.0979</u>	0.2217	**0.0884**	+9.7%	+60.1%
ECG	<u>0.1534</u>	0.1667	**0.1421**	+7.4%	+14.8%
TRANSP.	<u>0.0512</u>	0.1627	**0.0477**	+7.0%	+70.7%
TRAFFIC	**0.0341**	0.0815	<u>0.0343</u>	-0.7%	+57.9%
Avg. Gain	–	–	–	**+18.0%**	**+48.0%**
Datasets Improved	–	–	–	**6/7**	**7/7**

Table 8: Detailed cross-view ablation on **overall fidelity** across seven datasets. Overall fidelity is summarized by the geometric mean of the three overall-fidelity metrics. "Gain" denotes the relative improvement of CrossTF over each single-view baseline, where positive values indicate improvement. Lower is better. Best and second-best fidelity values are shown in bold and underlined, respectively.

Dataset	Time-Only	Freq.-Only	CrossTF	Gain vs. Time	Gain vs. Freq.
ELECT.	<u>0.0142</u>	0.0184	**0.0139**	+2.3%	+24.5%
BUILD.	**0.0025**	<u>0.0037</u>	**0.0025**	+0.3%	+32.7%
PRECIP.	**0.0015**	<u>0.0031</u>	**0.0015**	+0.0%	+53.3%
TEMP.	**0.0111**	0.0133	<u>0.0111</u>	-0.6%	+16.3%
ECG	**0.0179**	0.0187	<u>0.0180</u>	-0.9%	+3.7%
TRANSP.	<u>0.0139</u>	0.0189	**0.0135**	+2.4%	+28.5%
TRAFFIC	**0.0048**	0.0059	<u>0.0049</u>	-2.7%	+16.9%
Avg. Gain	–	–	–	+0.1%	**+25.1%**
Datasets Improved	–	–	–	3/7	**7/7**

Table 9: Detailed cross-view ablation results on extreme and overall fidelity across seven datasets. All metrics are lower-is-better. Best results are shown in bold and second-best results are underlined.

Dataset	Method	Extreme Fidelity				Overall Fidelity		
		Mag-Rate	Mag-Excess	Chg-Rate	Chg-Excess	Wass.	Marg.	ACD
ELECT.	Time-Only	0.1070	0.1991	0.1669	0.1546	0.0327	**0.0107**	0.0082
	Frequency-Only	**0.0772**	**0.0564**	0.0818	0.3009	0.0456	0.0194	0.0070
	CrossTF	0.0802	0.1329	**0.0345**	**0.0619**	**0.0315**	0.0127	**0.0067**
BUILD.	Time-Only	0.0518	0.0840	0.1632	0.1587	0.0294	**0.0000**	0.0177
	Frequency-Only	0.0746	0.0952	0.1624	0.2326	0.0388	**0.0000**	0.0370
	CrossTF	**0.0401**	**0.0664**	**0.1047**	**0.1214**	**0.0292**	**0.0000**	**0.0162**
PRECIP.	Time-Only	**0.0521**	0.1496	**0.0597**	0.1849	0.0283	**0.0000**	**0.0032**
	Frequency-Only	0.2823	0.1429	0.2150	0.1498	0.1308	0.0001	0.0042
	CrossTF	0.0555	**0.0778**	0.0663	**0.1134**	**0.0247**	**0.0000**	0.0035
TEMP.	Time-Only	0.0586	0.0503	**0.1221**	0.2550	**0.0461**	**0.0026**	**0.0112**
	Frequency-Only	0.1238	0.1386	0.2227	0.6319	0.0505	0.0027	0.0173
	CrossTF	**0.0535**	**0.0486**	0.1703	**0.1379**	0.0463	**0.0026**	0.0114
ECG	Time-Only	0.0787	**0.3569**	0.0993	0.1988	**0.0500**	**0.0142**	0.0081
	Frequency-Only	**0.0733**	0.5107	0.1291	0.1598	0.0514	0.0160	**0.0080**
	CrossTF	0.0738	0.3607	**0.0976**	**0.1568**	0.0503	0.0143	0.0081
TRANSP.	Time-Only	**0.0257**	0.0437	**0.0601**	0.1022	0.0152	**0.0401**	0.0044
	Frequency-Only	0.0731	0.1077	0.3410	0.2607	0.0237	0.0702	**0.0041**
	CrossTF	0.0270	**0.0389**	0.0613	**0.0804**	**0.0149**	**0.0401**	0.0042
TRAFFIC	Time-Only	0.0327	**0.0262**	**0.0387**	0.0407	**0.0203**	**0.0014**	**0.0039**
	Frequency-Only	0.0401	0.0322	0.1359	0.2518	0.0216	0.0015	0.0063
	CrossTF	**0.0323**	0.0282	0.0416	**0.0367**	0.0208	**0.0014**	0.0041

D.5 INTERPOLATION AND EXTRAPOLATION OF LEARNED STEP-WISE FUSION PARAMETERS α_i

Table 10 presents the learned step-wise fusion parameters α_i within the cross-view window across seven datasets. For clarity, interpolation and extrapolation parameters are highlighted in blue and orange, respectively. The learned parameters span both interpolation and extrapolation: among the 42 parameters, 20 fall within $0 \leq \alpha_i \leq 1$, while the remaining 22 satisfy $\alpha_i > 1$. Notably, we do not observe any case with $\alpha_i < 0$. This behavior can be explained by the different extreme-generation behaviors observed for the two views in our experiments: the time view generally achieves better extreme fidelity, whereas the frequency view tends to underestimate extremes, making optimization less likely to move beyond the frequency-view estimate along the cross-view direction. When the two views respectively overestimate and underestimate extremes, the learned parameter typically lies between them, corresponding to interpolation; when both views underestimate extremes, optimization may move beyond the time-view estimate, resulting in extrapolation with $\alpha_i > 1$. Extrapolation appears on six of the seven datasets except ELECT., while interpolation also appears on six datasets except BUILD., indicating that the two modes are adaptively selected across datasets and denoising steps. This suggests that restricting α_i to $[0, 1]$ would unnecessarily constrain the search space of cross-view optimization. Furthermore, compared with using a single shared α across the cross-view window, the learned step-wise parameters improve extreme fidelity on all seven datasets, with gains ranging from 0.7% to 45.4%. These results further support the use of step-wise interpolation and extrapolation in CrossTF.

D.6 CROSS-VIEW COMPLEMENTARITY BEYOND MODEL DIVERSITY

To answer RQ 6, we further investigate whether the cross-view effect observed in CrossTF simply arises from general model diversity between two different denoisers, or is essentially a model ensemble effect. In other words, if we do not introduce distinct time and frequency views and instead fuse two independently trained models from the same view, can we obtain a similar extreme-sensitive direction? To examine this, we construct a same-view ensemble control. Since the time view generally

Table 10: Learned step-wise fusion parameters α_i within the cross-view window across seven datasets. The final six denoising steps ($i = 6, \ldots, 1$) form the cross-view window. Parameters within the interpolation range ($0 \leq \alpha_i \leq 1$) and extrapolation range ($\alpha_i < 0$ or $\alpha_i > 1$) are shown in blue and orange, respectively. The last column reports the relative gain in extreme fidelity over using a single shared α across the window.

Dataset	Cross-View Window						Gain vs. Single α
	α_1	α_2	α_3	α_4	α_5	α_6	
ELECT.	0.546	0.440	0.480	0.846	0.654	0.681	+45.4%
BUILD.	1.271	1.544	1.871	1.905	1.812	1.735	+20.3%
PRECIP.	0.954	1.006	1.101	0.979	0.701	0.404	+5.8%
TEMP.	1.502	1.460	1.335	0.939	0.804	0.779	+8.0%
ECG	0.571	0.666	0.977	1.143	0.911	0.830	+0.7%
TRANSP.	1.061	1.163	0.934	1.152	1.358	1.207	+6.0%
TRAFFIC	1.014	1.096	0.823	1.075	1.252	1.138	+3.6%

achieves better extreme fidelity in our previous experiments, we perform time-time fusion using two independently trained time denoisers and compare it with time-frequency cross-view fusion. Specifically, we evaluate both settings using three independent training seeds and four sampling seeds. The cross-view and same-view settings use the same cross-view window, sampler, and α grid, so that the comparison mainly reflects the difference between cross-view and same-view fusion. Consistent with the aggregation used in Figure 7, we summarize extreme fidelity using the geometric mean of the four extreme-fidelity metrics. Along each direction, we measure the relative max–min range as α varies, as well as the endpoint gap normalized by its sampling standard error.

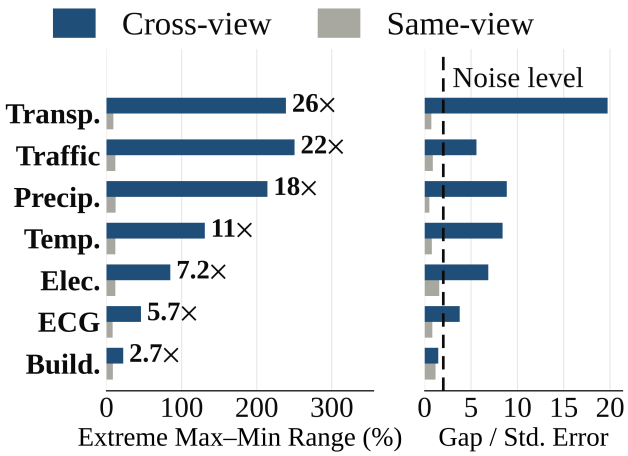


Fig. 10: **Cross-view complementarity versus same-view model diversity.** Left: relative max–min range of extreme fidelity along cross-view and same-view directions, with annotations showing their ratio. Right: endpoint gap normalized by sampling standard error; the dashed line marks the 2-standard-error noise threshold. Cross-view directions exhibit substantially stronger and more stable extreme-sensitive variation.

Figure 10 and Table 11 present the comparison. The same-view ensemble direction exhibits limited variation in extreme fidelity, with a median relative range of 11.6%, whereas the cross-view direction reaches 130.8%, approximately 11.25× larger, and shows a greater range across all seven datasets. More importantly, the endpoint gaps along all same-view directions remain below the two-standard-error noise threshold, with a median of 0.82, indicating no distinguishable change in extreme fidelity beyond generation noise. In contrast, the cross-view direction has a median endpoint gap of 6.86 standard errors and exceeds the noise threshold on six of the seven datasets. These results show that same-view model ensembling does not produce a meaningful extreme-sensitive direction. Therefore, the *Extreme-Sensitive Cross-View Asymmetry* observed in CrossTF is a characteristic of the time-frequency cross-view direction rather than a generic effect of model ensemble or model diversity.

Table 11: Comparison of cross-view and same-view sensitivity across seven datasets. The range ratio is computed as the cross-view range divided by the same-view range, while Gap / Standard Error measures the observed gap relative to sampling variability. Values of Gap / Standard Error ≥ 2 are considered beyond sampling noise. For the median row, the range ratio is computed from the median cross-view and same-view ranges using the unrounded values.

Dataset	Sensitivity Range			Gap / Standard Error	
	Cross-View	Same-View	Ratio	Cross-View	Same-View
TRANSP.	238.8%	9.1%	26.28×	19.73	0.72
TRAFFIC	250.3%	11.6%	21.54×	5.60	0.87
PRECIP.	214.3%	11.9%	18.02×	8.85	0.51
TEMP.	130.8%	11.7%	11.14×	8.39	0.78
ELECT.	85.0%	11.8%	7.17×	6.86	1.59
ECG	45.9%	8.0%	5.73×	3.78	0.82
BUILD.	22.0%	8.3%	2.66×	1.47	1.17
Median	**130.8%**	**11.6%**	**11.25×**	**6.86**	**0.82**
Summary	Cross-View > Same-View: **7/7**			Beyond Noise (≥ 2): **6/7** vs. **0/7**	

D.7 COMPUTATIONAL EFFICIENCY

To evaluate the computational efficiency of CrossTF, we measure the training time and the sampling time required to generate 1,000 samples on the PRECIP. dataset, using a single **NVIDIA RTX PRO 6000 Blackwell Server** GPU for all methods. All baselines follow their original public implementations and configurations as closely as possible, without additional modifications for the efficiency comparison. As shown in Table 12, **CrossTF requires only 29.5 minutes for training and 3.70 seconds to generate 1,000 samples**. Although CrossTF invokes both the time and frequency denoisers during sampling, it remains fast in practice, largely because it uses only 18 denoising steps, whereas diffusion-based methods such as E4GEN may require up to 1,000 sampling steps. Notably, methods with performance closer to CrossTF generally incur substantially higher sampling costs. For example, ImagenTime, FourierDiff, and E4GEN require approximately **6.03×, 6.27×, and 20.14×** the sampling time of CrossTF, respectively. DiffTime is one of the few diffusion baselines that samples substantially faster than CrossTF, but its training takes 60.0 minutes, approximately 2.0× longer than CrossTF. In contrast, several lightweight models with faster sampling incur a noticeable gap in generation performance. Together with the preceding extreme- and overall-fidelity results, these findings show that **CrossTF achieves the best generation performance while maintaining practical computational efficiency**, with fast sampling and moderate training time.

Table 12: Training and sampling efficiency comparison. Total training time is reported in minutes, and sampling time is measured for generating 1,000 samples. Relative sampling time is normalized by CrossTF, with CrossTF set to 1.00×; values above 1× indicate slower sampling than CrossTF.

Method	Training (min) ↓	Sampling 1K (s) ↓	Relative Sampling
TimeGAN	108.0	2.90	0.78×
KoVAE	6.2	0.03	0.01×
SDForger	19.7	151.10	40.84×
DiffTime	**60.0**	0.42	0.11×
Diffusion-TS	8.1	24.10	6.51×
FourierDiff	2.7	23.20	**6.27**×
ImagenTime	144.0	22.30	**6.03**×
RF-Diffusion	15.9	74.60	20.16×
FIDE	1.4	0.54	0.15×
HeavyDiff	4.5	17.90	4.84×
E4GEN	6.9	74.50	**20.14**×
CrossTF	**29.5**	**3.70**	**1.00**×

D.8 Additional Results on Extreme and Overall Fidelity

Table 13: Generation performance on the PRECIP. dataset. All metrics are lower-is-better.

Dataset	Type	Method	Extreme Fidelity				Overall Fidelity		
			Mag-Rate	Chg-Rate	Mag-Excess	Chg-Excess	Wass.	Marg.	ACD
PRECIP.	**GAN**	TimeGAN	0.2923	0.4502	1.0694	1.0721	0.1259	2.18e-05	0.0136
	VAE	KoVAE	0.9404	0.9779	0.7589	0.7514	0.2005	2.23e-05	0.0063
	LLM	SDForger	0.9623	0.4326	0.9122	0.7071	0.6871	1.04e-04	0.0162
	Diffusion	DiffTime	0.2816	0.1978	0.2209	0.1667	0.0607	1.76e-05	0.0074
		Diffusion-TS	0.9686	0.8395	0.4960	0.4518	0.2537	4.83e-05	0.0063
		FourierDiff	0.5341	0.4998	0.9973	0.7184	0.5366	9.89e-05	0.0041
		ImagenTime	0.0815	0.0767	0.1335	0.1664	0.0347	4.11e-05	**0.0028**
	TF-Based	RF-Diffusion	1.7815	1.5310	1.5139	1.1972	0.6161	1.04e-04	0.0112
	Extreme-Aware	FIDE	0.1792	0.1140	0.2004	0.2291	0.0668	3.70e-05	0.0101
		HeavyDiff	0.4075	0.2475	2.4150	2.4176	0.1590	**1.31e-05**	0.0087
		E4GEN	0.6380	0.3563	0.9420	0.7839	0.1896	3.20e-05	0.0073
	Ours	**CrossTF**	**0.0555**	**0.0663**	**0.0778**	**0.1134**	**0.0247**	3.53e-05	0.0035

Table 14: Generation performance on the BUILD. dataset. All metrics are lower-is-better.

Dataset	Type	Method	Extreme Fidelity				Overall Fidelity		
			Mag-Rate	Chg-Rate	Mag-Excess	Chg-Excess	Wass.	Marg.	ACD
BUILD.	**GAN**	TimeGAN	0.3299	0.4519	0.1554	1.1183	0.1900	1.82e-04	0.2302
	VAE	KoVAE	0.3504	0.5416	0.2563	0.4590	0.4135	3.00e-04	0.1417
	LLM	SDForger	0.2244	0.5211	0.2174	0.4442	0.3577	2.23e-04	0.0910
	Diffusion	DiffTime	0.1902	0.2676	0.1616	0.2347	0.1552	6.52e-05	0.0478
		Diffusion-TS	0.2428	0.4228	0.2093	0.4149	0.2102	6.74e-05	0.0777
		FourierDiff	0.1278	0.1830	0.1783	0.2630	0.1411	6.86e-05	0.0209
		ImagenTime	0.0862	**0.0638**	0.1162	0.1642	0.0570	3.40e-05	0.0203
	TF-Based	RF-Diffusion	0.0606	3.4284	**0.0612**	1.1259	**0.0292**	**2.85e-05**	0.2370
	Extreme-Aware	FIDE	0.1465	1.2235	0.1503	0.3541	0.3941	1.50e-04	0.0589
		HeavyDiff	0.0838	0.2901	0.1696	0.3793	0.2409	1.90e-04	0.1667
		E4GEN	0.2377	0.4180	0.2229	0.4254	0.2414	8.82e-05	0.1268
	Ours	**CrossTF**	**0.0401**	0.1047	0.0664	**0.1214**	**0.0292**	3.36e-05	**0.0162**

Table 15: Generation performance on the TEMP. dataset. All metrics are lower-is-better.

Dataset	Type	Method	Extreme Fidelity				Overall Fidelity		
			Mag-Rate	Chg-Rate	Mag-Excess	Chg-Excess	Wass.	Marg.	ACD
TEMP.	**GAN**	TimeGAN	1.4295	15.7082	0.6670	5.7697	0.2666	0.0089	0.2837
	VAE	KoVAE	0.6441	1.0221	0.7101	1.5098	0.2467	0.0077	0.1475
	LLM	SDForger	0.1175	0.6380	0.1528	1.2320	0.2247	0.0058	0.0804
	Diffusion	DiffTime	0.1903	0.1545	0.1360	0.3976	0.3028	0.0065	0.0308
		Diffusion-TS	0.1410	0.3274	0.2299	0.4479	0.1726	0.0040	0.0253
		FourierDiff	0.0824	0.2359	0.0899	0.9614	0.0710	0.0033	0.0185
		ImagenTime	0.1018	**0.1210**	0.0620	0.2915	0.1347	0.0033	0.0168
	TF-Based	RF-Diffusion	0.2914	12.0715	0.5461	3.3805	**0.0435**	0.0031	0.2795
	Extreme-Aware	FIDE	0.1382	0.6491	0.1187	0.5057	0.0779	0.0036	0.0197
		HeavyDiff	0.2722	0.6402	0.1739	0.3053	0.1698	0.0040	0.0872
		E4GEN	0.1672	0.2352	0.1527	0.6367	0.1741	0.0038	0.0354
	Ours	**CrossTF**	**0.0535**	0.1703	**0.0486**	**0.1379**	0.0463	**0.0026**	**0.0114**

Table 16: Generation performance on the ELECT. dataset. All metrics are lower-is-better.

Dataset	Type	Method	Extreme Fidelity				Overall Fidelity		
			Mag-Rate	Chg-Rate	Mag-Excess	Chg-Excess	Wass.	Marg.	ACD
ELECT.	**GAN**	TimeGAN	0.2936	1.0724	0.8155	1.3465	0.1261	0.0522	0.0694
	VAE	KoVAE	0.9229	1.1475	0.7631	0.9310	0.3385	0.0778	0.0422
	LLM	SDForger	0.4007	1.4693	0.7653	1.8697	0.2040	0.0575	0.0408
	Diffusion	DiffTime	0.2456	0.1872	0.1802	0.1525	0.1143	0.0256	0.0246
		Diffusion-TS	0.3145	0.6035	0.3155	0.4491	0.3212	0.0510	0.0286
		FourierDiff	0.6082	0.8399	1.0678	1.4358	0.2036	0.0607	0.0091
		ImagenTime	**0.0445**	0.0621	**0.1278**	0.0821	0.0356	**0.0089**	**0.0060**
	TF-Based	RF-Diffusion	0.6494	9.0197	1.0747	0.5879	0.1287	0.0433	0.1610
	Extreme-Aware	FIDE	0.2682	0.7331	0.1928	0.1588	0.0738	0.0262	0.0220
		HeavyDiff	1.3662	0.6383	1.5234	0.8290	0.1751	0.0382	0.0247
		E4GEN	0.4506	0.3388	0.3829	0.1602	0.2145	0.0521	0.0258
	Ours	**CrossTF**	0.0802	**0.0345**	0.1329	**0.0619**	**0.0315**	0.0127	0.0067

Table 17: Generation performance on the TRANSP. dataset. All metrics are lower-is-better.

Dataset	Type	Method	Extreme Fidelity				Overall Fidelity		
			Mag-Rate	Chg-Rate	Mag-Excess	Chg-Excess	Wass.	Marg.	ACD
TRANSP.	**GAN**	TimeGAN	0.2161	0.8154	0.4333	0.7877	0.1951	0.5131	0.0298
	VAE	KoVAE	0.6015	0.8455	0.6152	0.7580	0.2056	0.5539	0.0635
	LLM	SDForger	0.1201	0.3637	0.4108	0.6160	0.1127	0.2715	0.0268
	Diffusion	DiffTime	0.2473	0.2002	0.2661	0.1386	0.1560	0.3159	0.0194
		Diffusion-TS	0.2835	0.5090	0.2599	0.3218	0.1393	0.1938	0.0391
		FourierDiff	0.4169	0.5539	0.5616	0.7058	0.0974	0.1928	0.0591
		ImagenTime	0.0760	0.0687	0.0425	0.1032	0.0272	0.0431	**0.0027**
	TF-Based	RF-Diffusion	0.6752	9.3573	0.4048	1.0099	0.1680	0.7449	0.3004
	Extreme-Aware	FIDE	0.2395	0.5854	0.2632	0.2071	0.1054	0.1259	0.0221
		HeavyDiff	0.1595	0.1009	0.3101	0.4885	0.0569	0.1948	0.0332
		E4GEN	0.1359	0.3536	0.1970	0.2078	0.0539	0.2168	0.0285
	Ours	**CrossTF**	**0.0270**	**0.0613**	**0.0389**	**0.0804**	**0.0149**	**0.0401**	0.0042

Table 18: Generation performance on the ECG dataset. All metrics are lower-is-better.

Dataset	Type	Method	Extreme Fidelity				Overall Fidelity		
			Mag-Rate	Chg-Rate	Mag-Excess	Chg-Excess	Wass.	Marg.	ACD
ECG	**GAN**	TimeGAN	0.8675	0.4654	0.9233	1.7869	0.0886	0.0581	0.0282
	VAE	KoVAE	2.7715	1.9206	3.7157	2.1064	0.3909	0.1586	0.1084
	LLM	SDForger	1.9948	1.8672	3.4791	2.1706	0.3934	0.1243	0.0546
	Diffusion	DiffTime	0.5198	0.6658	0.3663	0.7390	0.2807	0.0723	0.0298
		Diffusion-TS	0.4689	0.5354	0.2841	0.5571	0.1496	0.0440	0.0153
		FourierDiff	2.2049	1.7647	3.6607	1.8019	0.3620	0.1252	0.0153
		ImagenTime	0.1214	0.1249	**0.2705**	0.2603	0.0637	0.0149	0.0157
	TF-Based	RF-Diffusion	2.7087	19.3703	4.6889	11.1231	0.3463	0.1566	0.1122
	Extreme-Aware	FIDE	0.7168	21.8706	0.6882	14.4202	0.4429	0.1240	0.1232
		HeavyDiff	0.2136	0.7832	0.7308	0.7663	0.0822	0.0500	0.0515
		E4GEN	0.2054	0.5869	0.8813	0.4068	0.1212	0.0446	0.0331
	Ours	**CrossTF**	**0.0738**	**0.0976**	0.3607	**0.1568**	**0.0503**	**0.0143**	**0.0081**

D.9 Additional Results about Downstream Utility

Table 19 reports the downstream utility results across seven datasets. Since DISC, PRED, and CONTEXT-FID involve stochastic training, their scores can vary across random seeds. To avoid over-interpreting small stochastic differences, we additionally consider the performance range across seeds: a result is regarded as tied with the best method when their ranges overlap, as indicated by † in the table. This is particularly important for DISC, where high-quality generated samples can make the classifier approach random guessing at 50% accuracy; consequently, small seed-level variations in classification accuracy can noticeably affect the reported score. Under this criterion, CrossTF achieves the best or tied-best result in 18 of the 21 dataset–metric comparisons, including all seven datasets for CONTEXT-FID, six for PRED, and five for DISC.

Table 19: **Downstream task utility** across seven real-world datasets. **Bold** denotes the best or tied-best result. A result is regarded as tied with the best method (†) when their performance ranges across random seeds overlap. The last column reports the number of best or tied-best results.

Metric	Method	Traffic	Precip.	Build.	Temp.	Elect.	Transp.	ECG	#Best
DISC ↓	**CrossTF (Ours)**	**0.0126**†	**0.0212**†	0.0466	0.0088	**0.0470**†	**0.0189**†	**0.0342**†	5
	ImagenTime	**0.0396**†	**0.0157**†	**0.0086**†	0.0214	**0.0295**†	**0.0076**†	**0.0067**†	6
	DiffTime	0.0968	**0.1053**†	0.0784	0.1266	**0.1000**†	0.1307	**0.1867**†	3
	FIDE	0.3897	0.0354	0.2595	**0.0016**	**0.0364**†	0.1913	0.2550	2
	FourierDiff	**0.0209**†	0.4929	0.0082	0.0263	0.1091	0.1989	0.3025	1
	E4GEN	0.3240	0.3325	0.0992	0.1053	0.2038	**0.1780**†	0.2658	1
	Diffusion-TS	0.1280	0.3176	0.0466	0.0482	0.2023	0.0928	0.1425	0
	HeavyDiff	0.1676	0.3514	0.1889	0.1941	0.3682	0.2557	0.3175	0
	SDForger	**0.0140**†	0.4929	**0.0054**†	0.0082	0.3318	0.1648	**0.0125**†	3
	KoVAE	0.3198	0.1108	0.0815	0.2056	0.2341	0.1761	0.4700	0
	RF-Diffusion	0.3687	0.4882	0.3614	0.4737	0.4318	0.4886	0.0325	0
	TimeGAN	0.4763	**0.0283**†	0.4728	0.4342	0.4364	0.3769	0.0750	1
PRED ↓	**CrossTF (Ours)**	**0.0198**†	**0.0325**†	**0.0026**†	0.0161	**0.0298**†	**0.0095**†	**0.0069**	6
	ImagenTime	**0.0199**†	**0.0326**†	**0.0026**†	0.0161	**0.0298**†	**0.0095**†	0.0072	5
	DiffTime	0.0201	0.0327	**0.0027**†	0.0160	0.0300	0.0096	0.0080	1
	FIDE	0.0378	0.0333	0.0036	0.0169	0.0338	0.0103	0.0644	0
	FourierDiff	0.0199	0.0439	0.0027	0.0158	0.0310	0.0097	0.0070	0
	E4GEN	**0.0199**†	0.0328	0.0028	0.0159	0.0299	0.0096	0.0078	1
	Diffusion-TS	0.0200	0.0329	0.0027	**0.0158**	**0.0298**†	0.0096	0.0077	2
	HeavyDiff	0.0202	0.0328	0.0031	0.0161	0.0299	0.0096	0.0083	0
	SDForger	0.0226	0.0442	0.0031	0.0176	0.0368	0.0100	0.0100	0
	KoVAE	0.0229	0.0346	0.0028	0.0185	0.0310	0.0109	0.0111	0
	RF-Diffusion	0.0501	0.0427	0.0085	0.0620	0.0726	0.0433	0.0675	0
	TimeGAN	0.0283	0.0329	0.0040	0.1077	0.0362	0.0149	0.0085	0
CONTEXT-FID ↓	**CrossTF (Ours)**	**0.0112**	**0.0074**†	**0.0019**	**0.0254**	**0.0303**†	**0.0068**†	**0.0485**†	7
	ImagenTime	0.0325	**0.0089**†	0.0109	0.1110	**0.0306**†	**0.0098**†	**0.0710**†	4
	DiffTime	0.1379	0.0462	0.0372	0.6744	0.1356	0.1525	0.3918	0
	FIDE	1.8391	0.0590	0.1642	0.1308	0.1071	0.2239	5.3355	0
	FourierDiff	0.0394	0.6732	0.1911	0.0697	1.4993	0.4780	1.4689	0
	E4GEN	0.1550	0.0480	0.0517	0.2065	0.2987	0.0841	0.2622	0
	Diffusion-TS	0.4602	0.3729	0.0470	0.2125	0.7585	0.2326	0.1749	0
	HeavyDiff	0.5811	1.0210	0.0344	0.6680	0.7900	0.1154	0.5741	0
	SDForger	11.6116	1.5288	0.0996	0.9860	7.3520	0.4737	2.4966	0
	KoVAE	3.1085	0.3765	0.1826	0.8379	0.9892	0.9565	4.4922	0
	RF-Diffusion	0.7904	1.0919	0.0106	1.9811	0.9143	1.5682	2.0688	0
	TimeGAN	2.0369	0.6872	0.0485	4.5985	0.6189	0.6625	0.5203	0

D.10 Visualization of Generated Distributions

For each dataset, we compare the distribution of real sequences with those generated by CrossTF and four representative baselines, using three complementary views: a PCA projection, a t-SNE embedding, and a kernel density estimate of the marginal value distribution. We visualize a representative subset of baselines for clarity, while quantitative results for the full set are reported in Appendix D.8. Real data are shown in blue and generated data in orange in the projection plots, and as solid and dashed curves, respectively, in the density plots.

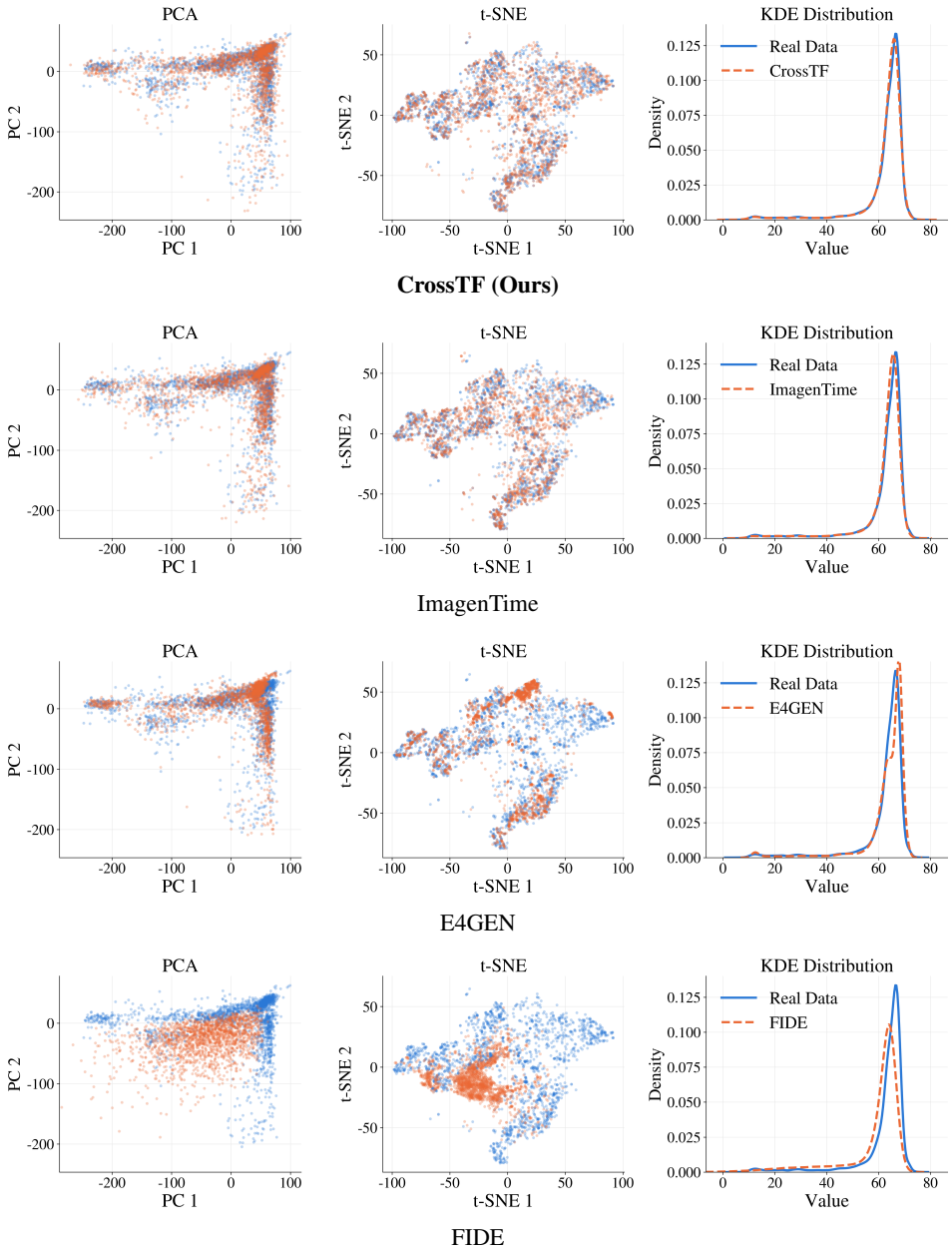


Fig. 11: Distribution visualization on the Traffic dataset. Each panel triplet shows the PCA projection, t-SNE embedding, and kernel density estimate for one method against the real data.

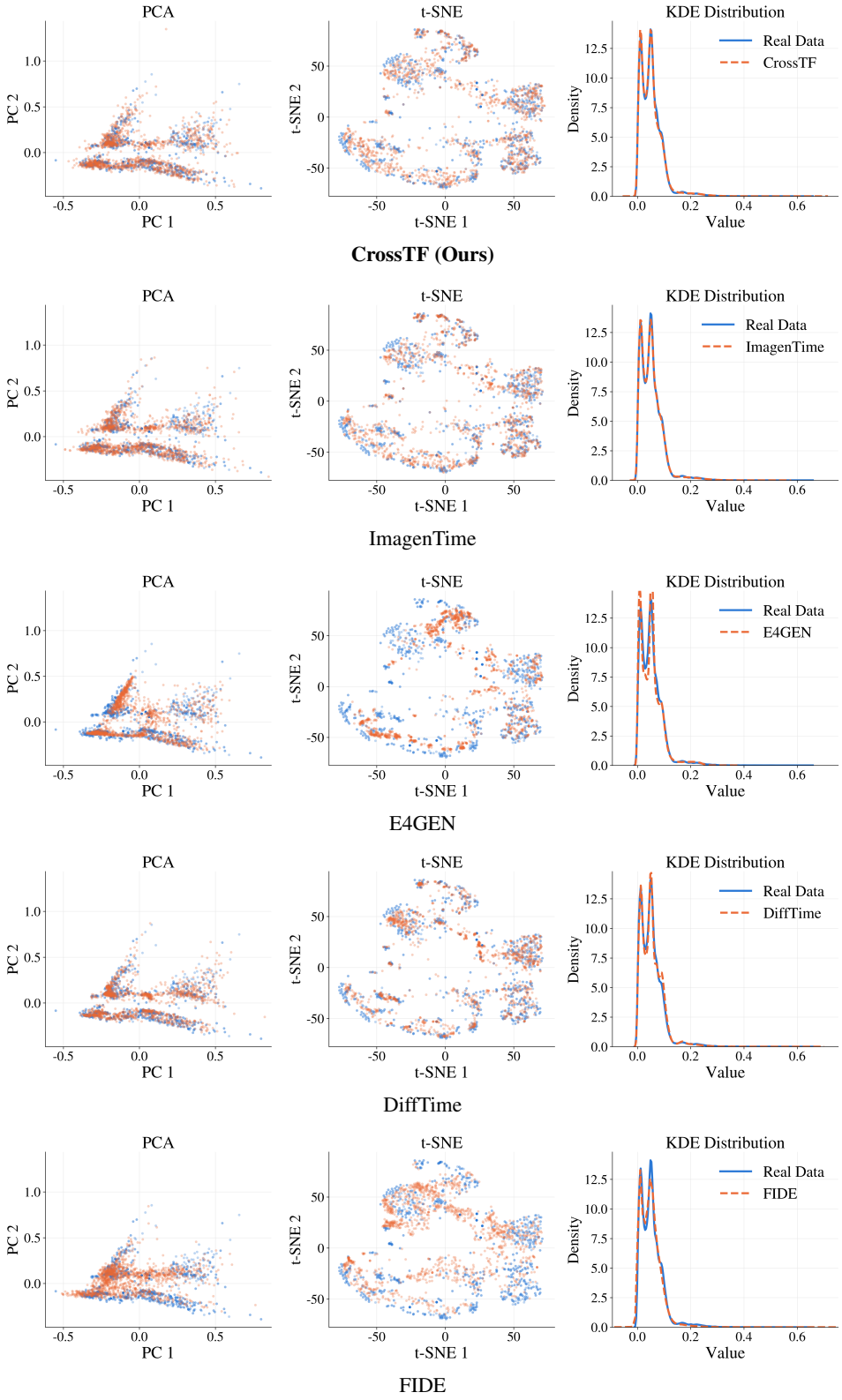


Fig. 12: Distribution visualization on the Transportation dataset. Each panel triplet shows the PCA projection, t-SNE embedding, and kernel density estimate for one method against the real data.

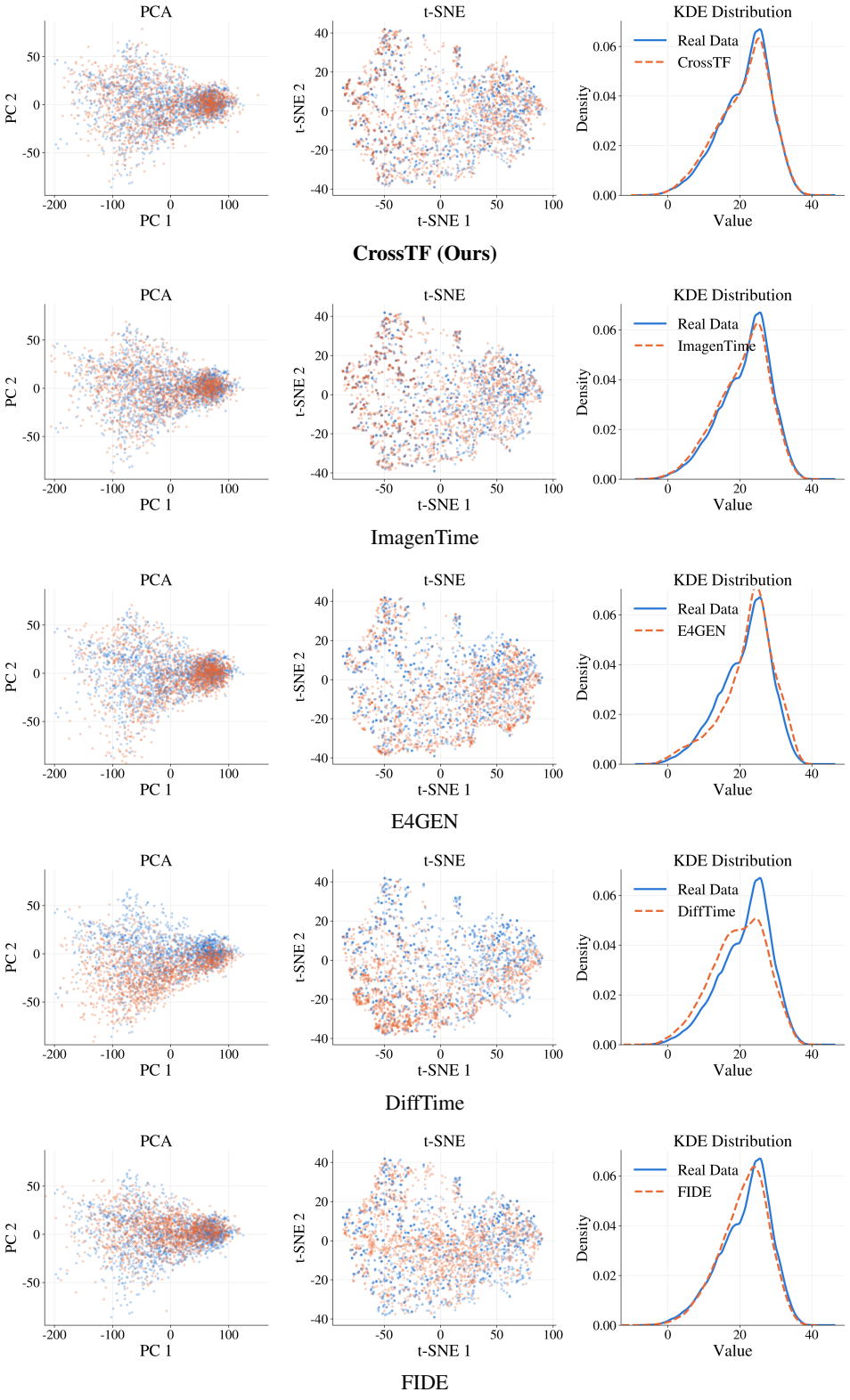


Fig. 13: Distribution visualization on the Temperature dataset. Each panel triplet shows the PCA projection, t-SNE embedding, and kernel density estimate for one method against the real data.

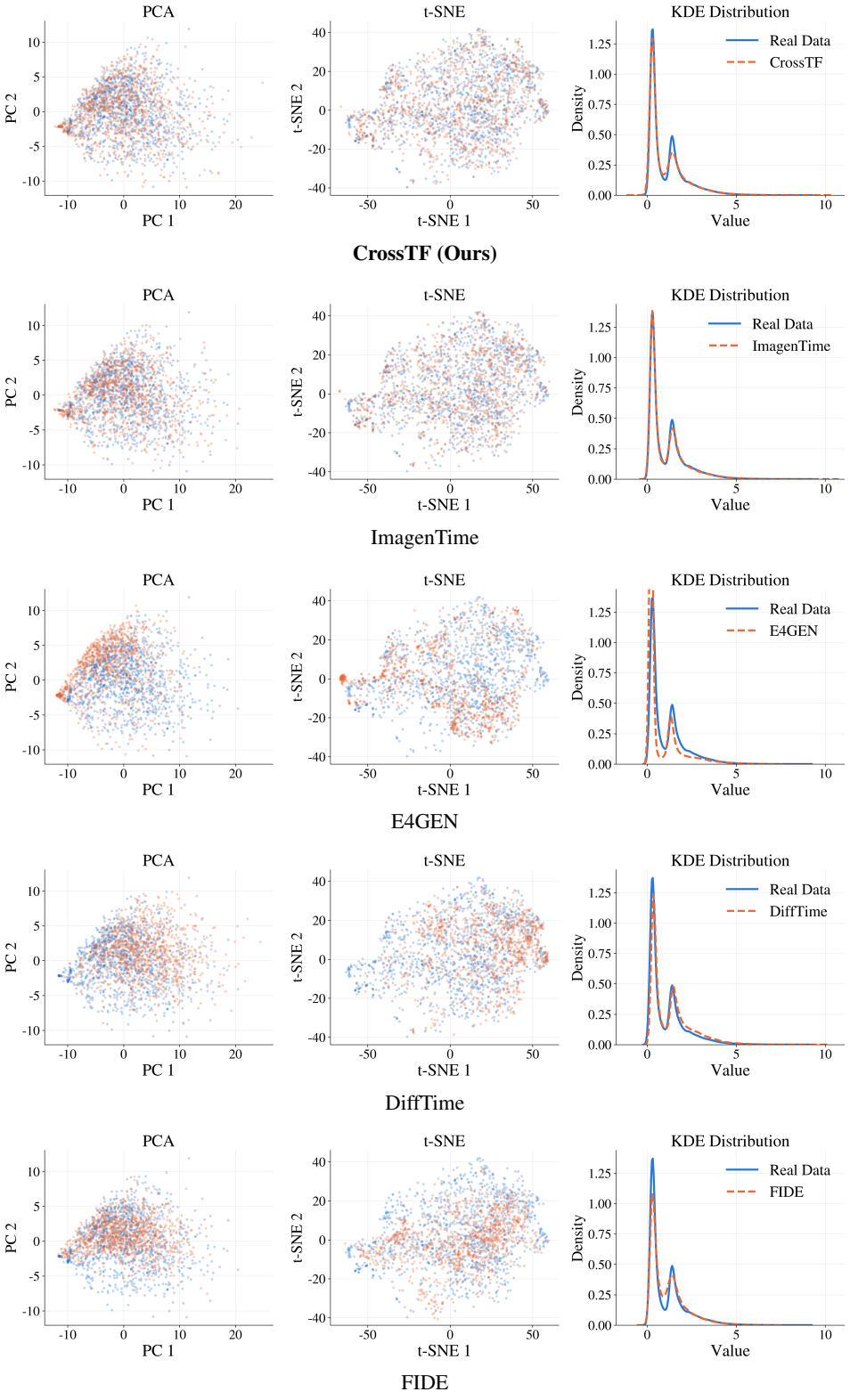


Fig. 14: Distribution visualization on the Electricity dataset. Each panel triplet shows the PCA projection, t-SNE embedding, and kernel density estimate for one method against the real data.

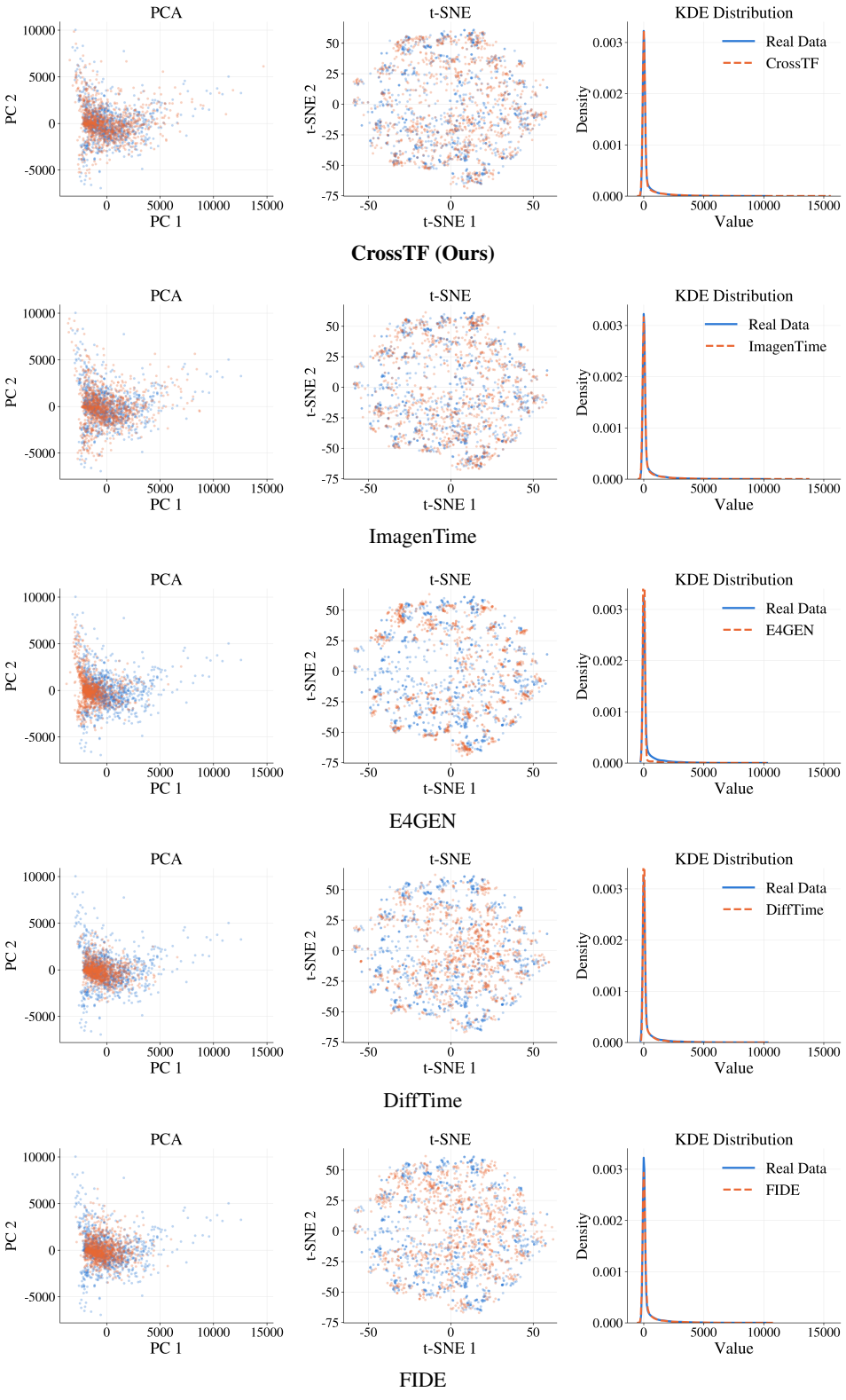


Fig. 15: Distribution visualization on the Precipitation dataset. Each panel triplet shows the PCA projection, t-SNE embedding, and kernel density estimate for one method against the real data.

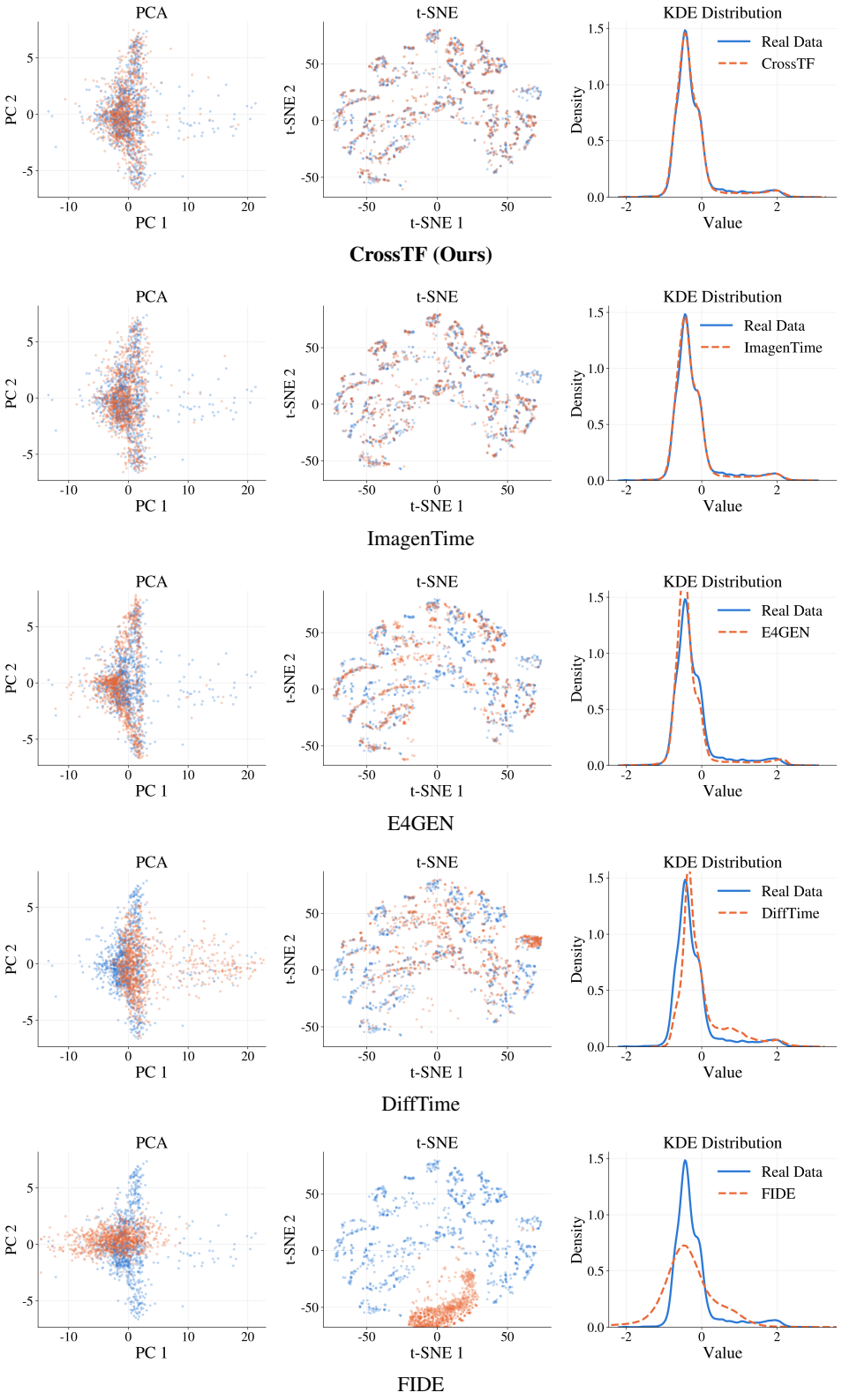


Fig. 16: Distribution visualization on the ECG dataset. Each panel triplet shows the PCA projection, t-SNE embedding, and kernel density estimate for one method against the real data.

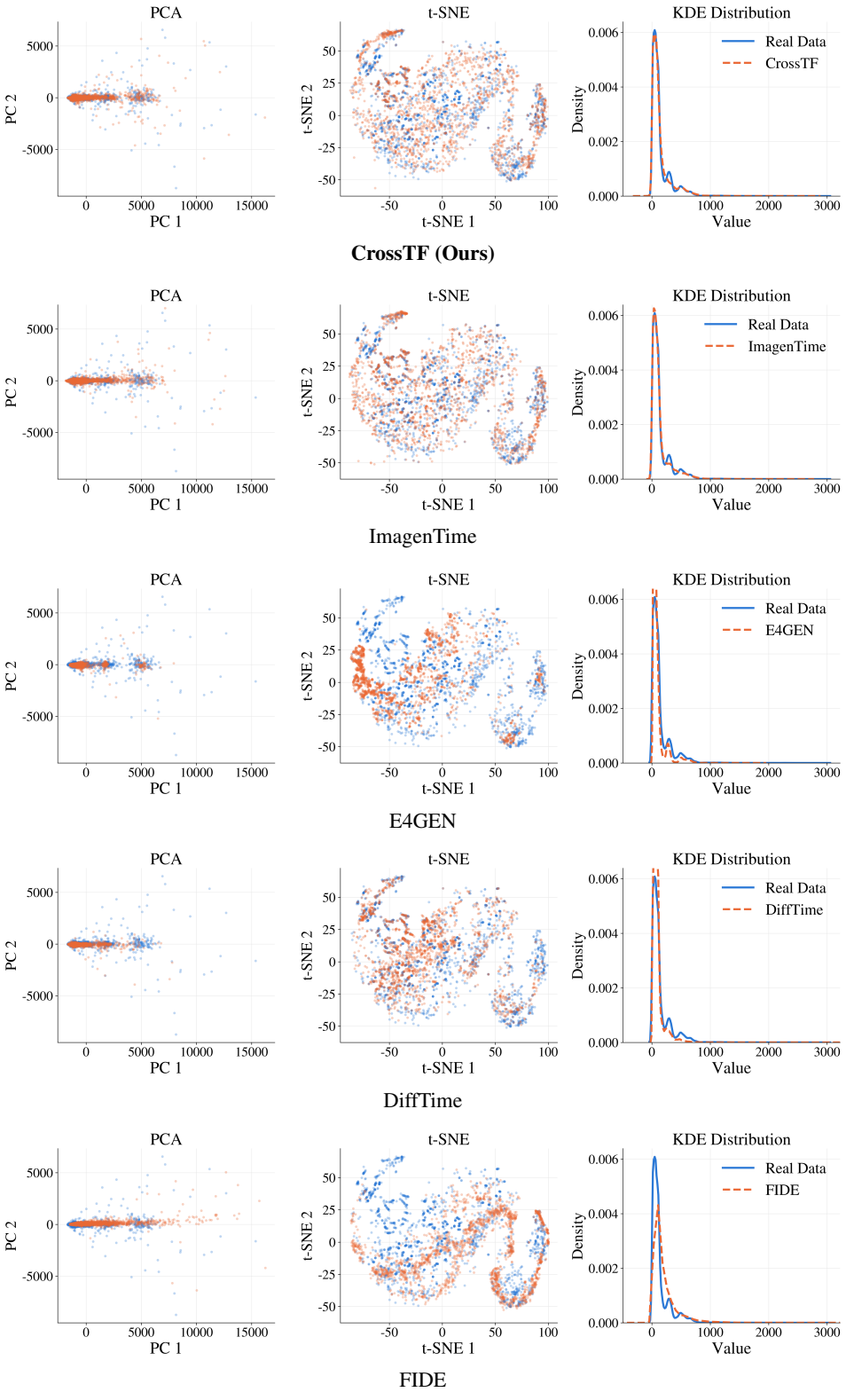


Fig. 17: Distribution visualization on the Building dataset. Each panel triplet shows the PCA projection, t-SNE embedding, and kernel density estimate for one method against the real data.