ACCELERATING FREQUENCY DOMAIN DIFFUSION MODELS WITH ERROR-FEEDBACK EVENT-DRIVEN CACHING

Dong Liu
Department of Computer Science
Yale University
dong.liu@aya.yale.edu

ABSTRACT

Diffusion models achieve remarkable success in time series generation. However, slow inference limits their practical deployment. We propose E^2-CRF (Error-Feedback Event-Driven Cumulative Residual Feature caching) to accelerate frequency domain diffusion models. Our method exploits two structural properties: (1) spectral localization, where signal energy concentrates in low frequencies, and (2) mirror symmetry, which halves the effective frequency dimension. E^2-CRF uses a closed-loop error-feedback system that adaptively caches transformer KV features across diffusion steps. We trigger recomputation using event-driven residual dynamics instead of fixed schedules. Our method selectively recomputes high-energy or rapidly-changing tokens while reusing cached features for stable high-frequency components. E^2-CRF achieves 1.7× speedup while maintaining sample quality. We demonstrate effectiveness on 5 datasets. Our caching strategy naturally aligns with the diffusion process's structure-to-detail progression. We establish a theoretical framework for accelerating diffusion inference by leveraging frequency domain structure. Our code is available at https://github.com/NoakLiu/FastFourierDiffusion.

1 Introduction

Diffusion models achieve remarkable success in generative modeling across diverse domains [1, 2, 3]. However, inference speed remains a critical bottleneck. Generating a single sample requires hundreds to thousands of forward passes through the score network. This makes real-time applications challenging. Recent acceleration techniques primarily focus on reducing diffusion steps [4, 5, 6]. We explore a different direction: optimizing the computational cost per step through intelligent caching. For transformer-based score networks, attention computation scales quadratically with sequence length. Caching key–value (KV) features across diffusion steps offers significant speedup potential. However, naïve caching strategies risk error accumulation and quality degradation. We need principled approaches that exploit structural properties of the diffusion process and domain representations.

Diffusion Models and Inference Speed. In recent years, diffusion models [7, 8, 1, 3] have emerged as one of the most promising research avenues in deep generative modelling. They achieve state-of-the art results across many generative modelling tasks [2, 9]. Researchers have applied diffusion models to time series modelling with promising results [10]. However, slow inference speed remains a critical limitation. Generating a single sample requires hundreds to thousands of forward passes through the score network. This makes real-time applications challenging. Recent work explores various acceleration techniques [4, 5, 6]. Most techniques focus on reducing the number of diffusion steps rather than optimizing the computational cost per step. This gap is particularly significant for transformer-based score networks. Attention computation scales quadratically with sequence length and represents a major computational bottleneck. Caching strategies that exploit structural properties of diffusion processes and frequency-domain representations offer a promising direction for acceleration without sacrificing sample quality.

Frequency Domain Diffusion and Caching Opportunities. Fourier analysis is a remarkably powerful tool in signal processing, compression and machine learning [11]. Recent work shows that performing diffusion in the frequency domain can improve sample quality for time series [12]. This approach leverages a key fact: real-world time series often exhibit spectral localization. Most energy concentrates in low frequencies. Moreover, the mirror symmetry property of real-valued signals' DFTs ($\tilde{x}_\kappa = \tilde{x}^*_{N-\kappa}$) naturally reduces the effective dimension by half. These structural properties create unique opportunities for efficient caching. High-frequency tokens carry less energy and change more slowly across diffusion steps. They are prime candidates for feature reuse. Diffusion sampling is iterative. The structure-to-detail progression means low frequencies form the signal structure early while high frequencies refine details later. This naturally aligns with selective caching strategies that can significantly reduce computation without compromising quality.

Motivation. The iterative nature of diffusion sampling combines with structural properties of frequency domain representations. This presents a unique opportunity for acceleration through intelligent caching. High-frequency tokens carry less energy and change more slowly across diffusion steps. They are prime candidates for feature reuse. However, naïve caching can lead to error accumulation and quality degradation. We introduce E^2-CRF, a principled caching mechanism that combines error-feedback control with event-driven scheduling. E^2-CRF accelerates frequency domain diffusion while maintaining sample quality.

Our contributions. (1) E^2-CRF: Error-Feedback Event-Driven Caching. In Section 3, we introduce E^2-CRF, a novel caching mechanism for frequency domain diffusion models. Our method exploits spectral localization and mirror symmetry to cache transformer KV features across diffusion steps. We use an error-feedback closed-loop system to prevent quality degradation. We design an event-driven scheduler that adapts to residual dynamics. **(2) Speedup with Quality Preservation.** In Section 4.1, we demonstrate that E^2-CRF achieves 1.7× speedup on 5 datasets spanning healthcare, finance, and engineering domains. Our method maintains sample quality as measured by sliced Wasserstein distances. E^2-CRF naturally aligns with the diffusion process's structure-to-detail progression. **(3) Theoretical Analysis of Caching Strategy.** In Section 4.2, we provide theoretical analysis showing that our event-driven trigger mechanism based on CRF residual intensity automatically adapts to different diffusion stages. We prove that the error-feedback mechanism bounds approximation error and prevents long-term drift. The spectral localization property ensures that cached high-frequency features remain stable across steps.

2 Background

Notations. We consider multivariate time series of fixed size $\mathbf{x} \in \mathbb{R}^{N\times M}$, where $N \in \mathbb{N}$ is the number of time steps and $M \in \mathbb{N}$ is the number of features. We denote by $d_X = N \cdot M$ the total dimension of $\mathbf{x}$, and by $N' = 2\lceil N/2 \rceil$ the effective dimension in the frequency domain after accounting for mirror symmetry. We use *Greek letters* for time series components: $\mathbf{x}_\tau \in \mathbb{R}^M$ denotes the feature vector at time step $\tau \in [N]$, and $x_{\tau,\nu}$ denotes feature $\nu \in [M]$ at time τ. We use *Latin letters* for diffusion steps: the diffusion process $\{\mathbf{x}(t) \in \mathbb{R}^{d_X}\}_{t=0}^T$ is indexed by continuous time $t \in [0, T]$, where T is the total number of diffusion steps. For caching, we denote by $\tilde{\mathbf{x}}(t) = \mathcal{F}[\mathbf{x}(t)] \in \mathbb{C}^N$ the frequency representation at diffusion step t, where $\mathcal{F}$ is the discrete Fourier transform. The cached score approximation is denoted $\hat{\mathbf{s}}_\theta(\tilde{\mathbf{x}}, t)$, while $\mathbf{s}_\theta(\tilde{\mathbf{x}}, t)$ is the ground-truth score. The approximation error $\epsilon(\tilde{\mathbf{x}}, t) = \mathbf{s}_\theta(\tilde{\mathbf{x}}, t) - \hat{\mathbf{s}}_\theta(\tilde{\mathbf{x}}, t)$ quantifies the difference between cached and recomputed scores, which our error-feedback mechanism aims to control.

2.1 Caching in Iterative Generative Inference

Recent advances in diffusion models have significantly improved generative quality across multiple domains [2, 1, 3, 13], yet their inference efficiency remains a fundamental bottleneck. Unlike autoregressive generation, diffusion sampling requires hundreds to thousands of iterative evaluations of the same score network, making the per-step computational cost a dominant factor in latency. This challenge is particularly acute for transformer-based score networks [14], where self-attention incurs quadratic complexity $O(N^2)$ in sequence length N and repeatedly recomputes highly similar key–value (KV) projections across diffusion steps. Diffusion models have been successfully applied to time series generation [15, 16, 10], but inference efficiency remains a critical limitation.

Caching has emerged as a natural strategy to amortize redundant computation in sequential inference. Prior cache designs in large language models primarily exploit causal structure and monotonic context growth, enabling KV reuse across tokens. However, diffusion inference exhibits fundamentally different dynamics: the input representation evolves continuously across steps, and naïvely reusing cached features can lead to error accumulation and quality degradation. Formally, if $\hat{\mathbf{s}}_\theta(\mathbf{x}, t)$ denotes a cached score approximation and $\mathbf{s}_\theta(\mathbf{x}, t)$ is the ground-truth score, the approximation error $\epsilon(\mathbf{x}, t) = \mathbf{s}_\theta(\mathbf{x}, t) - \hat{\mathbf{s}}_\theta(\mathbf{x}, t)$ accumulates over diffusion steps, potentially degrading sample quality. As a result, most existing diffusion acceleration methods focus on reducing the number of steps [4, 5, 6] rather than addressing the lack of principled, step-aware caching mechanisms. Recent work on frequency-aware caching for diffusion models [17] has explored caching strategies, but focuses primarily on image generation rather than time series. This gap highlights the need for caching strategies that are explicitly designed for iterative, non-autoregressive generative processes, where reuse must be both selective and dynamically controlled.

2.2 Cache Reuse in Frequency-Domain Diffusion

Beyond system-level inefficiencies, the diffusion process itself exhibits strong structural regularities that create previously underexplored caching opportunities. In frequency-domain diffusion models for time series [18, 12], two properties are especially salient. First, spectral localization causes most signal energy to concentrate in low-frequency components, while high-frequency components carry less energy and evolve more smoothly across diffusion steps. The spectral energy density $\|\tilde{\mathbf{x}}_\kappa\|_2^2$ typically decays rapidly with frequency index κ, enabling selective caching of high-frequency tokens. Second, the mirror symmetry of real-valued signals in the discrete Fourier transform reduces the effective frequency dimension by half, introducing inherent redundancy in the representation. Recent work has explored various frequency-domain approaches for time series generation, including wavelet-based diffusion [19, 20] and frequency-inflated conditional diffusion [21], demonstrating the effectiveness of frequency-domain representations.

Crucially, diffusion sampling follows a structure-to-detail progression: early steps establish coarse, low-frequency structure, whereas later steps refine fine-grained, high-frequency details. This temporal asymmetry implies that many frequency tokens—particularly in higher bands—experience minimal change over large portions of the reverse diffusion trajectory. Formally, for frequency token κ at diffusion step i, the relative change $\delta_\kappa^{(i)} = \|\tilde{\mathbf{x}}_\kappa^{(i)} - \tilde{\mathbf{x}}_\kappa^{(i-\Delta)}\|_2 / \|\tilde{\mathbf{x}}_\kappa^{(i-\Delta)}\|_2$ is typically smaller for high-frequency tokens ($\kappa > K$) than for low-frequency tokens ($\kappa \leq K$), where K is a threshold frequency. Such behavior suggests that feature reuse across diffusion steps is not only possible but structurally aligned with the generative process itself, provided that cache reuse is selectively triggered and guarded against drift. These observations motivate caching mechanisms that are event-driven and error-aware, rather than relying on fixed schedules or uniform reuse, enabling substantial acceleration without compromising sample fidelity.

2.3 Cached Score-Based Diffusion with SDEs

In continuous-time diffusion modelling, one assumes access to samples drawn from an unknown density p_{data}. The objective of generative modelling is to obtain a tractable approximation of this distribution.

Forward and cached reverse diffusion. Score-based generative modeling with *stochastic differential equation* (SDEs) [3] operates by constructing a forward diffusion process that transforms p_{data} to an isotropic Gaussian p_T, and a reverse process that generates samples. The forward process is:

$$\mathrm{d}\mathbf{x} = \boldsymbol{f}(\mathbf{x}, t)\mathrm{d}t + \boldsymbol{G}(t)\mathrm{d}\boldsymbol{w}, \tag{1}$$

where $\boldsymbol{f} : \mathbb{R}^{d_X} \times [0, T] \to \mathbb{R}^{d_X}$ is the drift, $\boldsymbol{w}$ is a standard Brownian motion, and $\boldsymbol{G} : [0, T] \to \mathbb{R}^{N \times N}$ is the diffusion matrix. The standard reverse diffusion process satisfies:

$$\mathrm{d}\mathbf{x} = \boldsymbol{b}(\mathbf{x}, t)\mathrm{d}t + \boldsymbol{G}(t)\mathrm{d}\hat{\boldsymbol{w}}, \tag{2}$$

where $\boldsymbol{b}(\mathbf{x}, t) = \boldsymbol{f}(\mathbf{x}, t) - \boldsymbol{G}(t)\boldsymbol{G}(t)^T \mathbf{s}(\mathbf{x}, t)$ with $\mathbf{s}(\mathbf{x}, t) = \nabla_{\mathbf{x}} \log p_t(\mathbf{x})$ being the score function. Under caching, the score is approximated by $\hat{\mathbf{s}}_\theta(\mathbf{x}, t)$ using cached transformer features, leading to the *cached reverse process*:

$$\mathrm{d}\mathbf{x} = \hat{\boldsymbol{b}}(\mathbf{x}, t)\mathrm{d}t + \boldsymbol{G}(t)\mathrm{d}\hat{\boldsymbol{w}}, \tag{3}$$

where $\hat{\boldsymbol{b}}(\mathbf{x}, t) = \boldsymbol{f}(\mathbf{x}, t) - \boldsymbol{G}(t)\boldsymbol{G}(t)^T \hat{\mathbf{s}}_\theta(\mathbf{x}, t)$ and the *caching approximation error* $\epsilon(\mathbf{x}, t) = \mathbf{s}_\theta(\mathbf{x}, t) - \hat{\mathbf{s}}_\theta(\mathbf{x}, t)$ must be controlled to maintain sample quality.

Cached denoising score matching. The score $\mathbf{s}(\mathbf{x}, t) := \nabla_{\mathbf{x}} \log p_t(\mathbf{x})$ is estimated by minimizing the score matching objective [7, 22]:

$$\theta^* = \arg\min_{\theta \in \Theta} \mathbb{E}_{t, \mathbf{x}(0), \mathbf{x}(t)} \left[\|\mathbf{s}_\theta(\mathbf{x}, t) - \mathbf{s}_{t|0}(\mathbf{x}, t)\|^2 \right], \tag{4}$$

where $\mathbf{s}_{t|0}(\mathbf{x}, t) := \nabla_{\mathbf{x}(t)} \log p_{t|0}(\mathbf{x}(t)|\mathbf{x}(0))$ and $t \sim \mathcal{U}(0, T)$. Under caching, the *cached score* $\hat{\mathbf{s}}_\theta(\mathbf{x}, t)$ uses cached KV pairs from transformer layers, introducing the *caching approximation error* $\epsilon(\mathbf{x}, t) = \mathbf{s}_\theta(\mathbf{x}, t) - \hat{\mathbf{s}}_\theta(\mathbf{x}, t)$. The *cached score matching objective* becomes:

$$\|\hat{\mathbf{s}}_\theta(\mathbf{x}, t) - \mathbf{s}_{t|0}(\mathbf{x}, t)\|^2 = \|\mathbf{s}_\theta(\mathbf{x}, t) - \mathbf{s}_{t|0}(\mathbf{x}, t) + \epsilon(\mathbf{x}, t)\|^2. \tag{5}$$

Error-feedback mechanisms and selective recomputation control $\epsilon(\mathbf{x}, t)$ to maintain distributional fidelity in cached sampling.

2.4 Frequency Domain Representation with Cache

DFT and cache-enabling mirror symmetry. The *Discrete Fourier Transform* (DFT) maps a time series $\mathbf{x} \in \mathbb{R}^{d_X}$ to the frequency domain $\tilde{\mathbf{x}} = \mathcal{F}[\mathbf{x}] \in \mathbb{C}^{d_X}$ via the linear operator $\mathcal{F}$. The matrix representation $\tilde{\mathbf{x}} = U\mathbf{x}$ uses a unitary

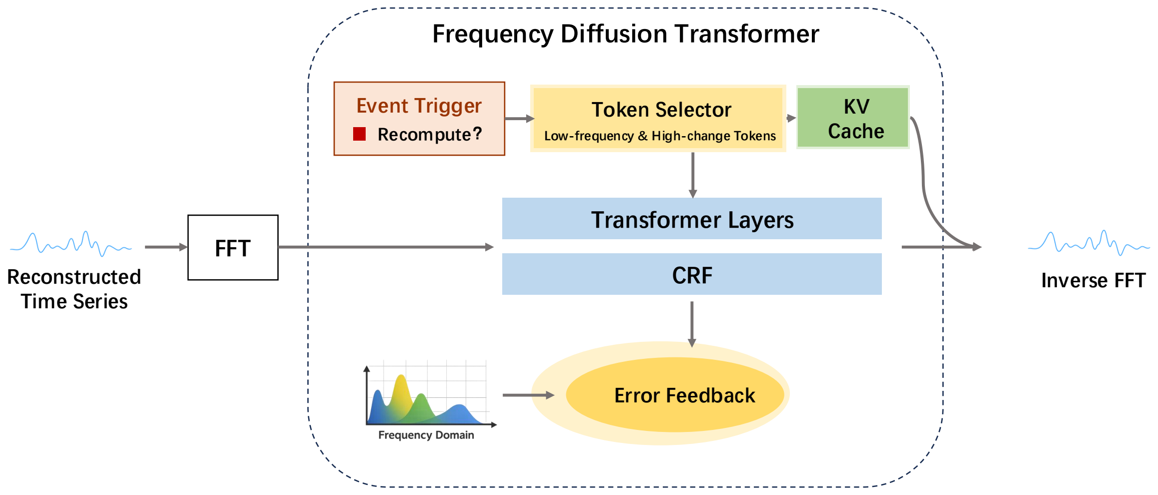


Figure 1: Overview of our error-feedback event-driven caching framework for accelerating frequency-domain diffusion.

matrix U (i.e., $U^*U = I_N$), ensuring invertibility: $\mathbf{x} = \mathcal{F}^{-1}[\tilde{\mathbf{x}}] = U^*\tilde{\mathbf{x}}$. For real-valued signals, the DFT exhibits *mirror symmetry*:

$$\tilde{\mathbf{x}}_\kappa = \tilde{\mathbf{x}}^*_{N-\kappa}, \tag{6}$$

where z^* denotes complex conjugation. This symmetry reduces the effective dimension by half: only frequencies $\kappa \le \lfloor N/2 \rfloor$ need be processed. The *cache-relevant consequence* is that we only need to cache and recompute at most $\lfloor N/2 \rfloor$ frequency tokens, enabling efficient caching strategies that exploit this redundancy.

Spectral energy localization for cache efficiency. Through *Parseval's theorem*, the total energy $\|\mathbf{x}\|^2 = \|\tilde{\mathbf{x}}\|^2$ is preserved under DFT. The *spectral energy density* $\|\tilde{\mathbf{x}}_\kappa\|_2^2$ typically decays rapidly with frequency κ, with most energy concentrated in low frequencies. This *spectral localization property* creates caching opportunities: high-frequency tokens ($\kappa > K$) with lower energy can be cached more aggressively, as their *cached change rate* $\delta^{(i)}_\kappa = \|\tilde{\mathbf{x}}^{(i)}_\kappa - \tilde{\mathbf{x}}^{(i-\Delta)}_\kappa\|_2 / \|\tilde{\mathbf{x}}^{(i-\Delta)}_\kappa\|_2$ is typically smaller than low-frequency tokens. The cached score function in the frequency domain $\hat{\tilde{\mathbf{s}}}_\theta(\tilde{\mathbf{x}}, t)$ operates on cached features, and the score function $\tilde{\mathbf{s}}(\tilde{\mathbf{x}}) := \nabla \log \tilde{p}(\tilde{\mathbf{x}})$ also exhibits mirror symmetry: $\tilde{\mathbf{s}}_\kappa = \tilde{\mathbf{s}}^*_{N-\kappa}$, further reducing cache size and computation overhead.

3 Diffusing in the frequency domain

In the previous section, we have described how the typical diffusion formalism applies to time-series. We have also described how the DFT $\tilde{\mathbf{x}} = \mathcal{F}[\mathbf{x}]$ offers a full description of the time series $\mathbf{x}$ in the frequency domain. The first step is to define how time-based diffusion translates in the frequency domain. Note that this is non-trivial as the DFT are complex-valued $\tilde{\mathbf{x}} \in \mathbb{C}^{d_X}$ signals. To solve this, we shall assume that the stochastic process in the time domain $\{\mathbf{x}(t)\}_{t=0}^T$, written compactly as $\mathbf{x}$, follows the diffusion process described in Equation (1). By leveraging the matrix formulation of the DFT $\tilde{\mathbf{x}} = U\mathbf{x}$, we will now derive diffusion SDEs in the frequency domain.

3.1 Cached Diffusion SDEs in Frequency Domain

Applying the DFT operator to the forward diffusion SDE from Equation (1) yields frequency-domain diffusion. The key observation is that the DFT of a standard Brownian motion $\boldsymbol{w}$ satisfies *mirror symmetry* Equation (6), resulting in a *mirrored Brownian motion* $\boldsymbol{v} = U\boldsymbol{w}$ where $\boldsymbol{v}_\kappa = \boldsymbol{v}^*_{N-\kappa}$ for all $\kappa \in [N]$. This symmetry reduces the effective dimension by half: only frequencies $\kappa \le \lfloor N/2 \rfloor$ need be processed, enabling cache-efficient implementations that cache at most $\lfloor N/2 \rfloor$ frequency tokens rather than all N tokens.

Proposition 3.1. *(Diffusion process in frequency domain). Let us assume that* $\boldsymbol{x}$ *is a diffusion process that is a solution of Equation* (1)*, with* $\boldsymbol{G}(t) = g(t)\, I_N$*. Then* $\tilde{\boldsymbol{x}} = \mathcal{F}[\boldsymbol{x}]$ *is a solution to the forward diffusion process defined by:*

$$\mathrm{d}\tilde{\boldsymbol{x}} = \tilde{\boldsymbol{f}}(\tilde{\boldsymbol{x}}, t)\mathrm{d}t + g(t)\mathrm{d}\tilde{\boldsymbol{v}}, \tag{7}$$

where $\tilde{\boldsymbol{f}}(\tilde{\boldsymbol{x}}, t) = U\boldsymbol{f}(U^*\tilde{\boldsymbol{x}}, t)$ *and* $\tilde{\boldsymbol{v}}$ *is a mirrored Brownian motion on* $\mathbb{C}^{d_X}$*. The associated reverse diffusion process is defined by:*

$$\mathrm{d}\tilde{\boldsymbol{x}} = \tilde{\boldsymbol{b}}(\tilde{\boldsymbol{x}}, t)\mathrm{d}t + g(t)\mathrm{d}\breve{\boldsymbol{v}} \tag{8}$$

where $\tilde{\boldsymbol{b}}(\tilde{\boldsymbol{x}}, t) = \tilde{\boldsymbol{f}}(\tilde{\boldsymbol{x}}, t) - g^2(t)\Lambda^2\tilde{\mathbf{s}}(\tilde{\boldsymbol{x}}, t)$*,* $\Lambda \in \mathbb{R}^{N\times N}$ *is a diagonal matrix defined in Section* A.3*,* $\mathrm{d}t$ *is a negative infinitesimal time step, and* $\breve{\boldsymbol{v}}$ *is a mirrored Brownian motion on* $\mathbb{C}^{d_X}$ *with time going from* T *to* 0*. Under caching, the score* $\tilde{\mathbf{s}}(\tilde{\boldsymbol{x}}, t)$ *is approximated by* $\hat{\tilde{\mathbf{s}}}_\theta(\tilde{\boldsymbol{x}}, t)$ *using cached transformer features, leading to the* cached reverse process in frequency domain*:*

$$\mathrm{d}\tilde{\boldsymbol{x}} = \hat{\tilde{\boldsymbol{b}}}(\tilde{\boldsymbol{x}}, t)\mathrm{d}t + g(t)\mathrm{d}\breve{\boldsymbol{v}} \tag{9}$$

where $\hat{\tilde{\boldsymbol{b}}}(\tilde{\boldsymbol{x}}, t) = \tilde{\boldsymbol{f}}(\tilde{\boldsymbol{x}}, t) - g^2(t)\Lambda^2\hat{\tilde{\mathbf{s}}}_\theta(\tilde{\boldsymbol{x}}, t)$ *and the* caching approximation error $\tilde{\epsilon}(\tilde{\boldsymbol{x}}, t) = \tilde{\mathbf{s}}_\theta(\tilde{\boldsymbol{x}}, t) - \hat{\tilde{\mathbf{s}}}_\theta(\tilde{\boldsymbol{x}}, t)$ *must be controlled to maintain sample quality.*

Proof. The proof is given in Section A.3. □

Proposition 3.1 gives us a recipe to implement diffusion in the frequency domain. It guarantees that the formalism introduced by [3] extends to this setting with one important difference: the Brownian motion must be replaced by a mirrored Brownian motion. Ignoring this prescription by taking $\tilde{\boldsymbol{v}}$ to be a Brownian motion on $\mathbb{C}^{d_X}$ could lead to unintended consequences, such as generating complex-valued time series $\mathbf{x} \in \mathbb{C}^{d_X} \setminus \mathbb{R}^{d_X}$. The mirror symmetry property enables cache-efficient implementations, as we only need to cache and recompute tokens for frequencies $\kappa \leq \lfloor N/2 \rfloor$, effectively halving the cache size.

3.2 Cached Denoising Score Matching in Frequency Domain

The reverse diffusion process given in Equation (8) provides an explicit way to samples time-series in the frequency domain provided we can compute $\tilde{\boldsymbol{b}}(\tilde{\mathbf{x}}, t)$, which involves the unknown score $\tilde{\mathbf{s}}$. Like in the time domain, and motivated by Equation (8), we build an approximation of the score with a function $\tilde{\mathbf{s}}_{\tilde{\theta}^*}$, whose parameters $\tilde{\theta}^*$ minimize the score matching objective:

$$\tilde{\theta}^* = \arg\min_{\tilde{\theta}\in\Theta} \mathbb{E}_{t,\tilde{\mathbf{x}}(0),\tilde{\mathbf{x}}(t)} \left[\mathcal{L}_{\mathrm{SM}}\left(\tilde{\mathbf{s}}_{\tilde{\theta}}, \Lambda^2\tilde{\mathbf{s}}_{t|0}, \tilde{\mathbf{x}}, t\right)\right] \tag{10}$$

with $t \sim \mathcal{U}(0, T)$, $\tilde{\mathbf{x}}(0) \sim \tilde{p}_0(\tilde{\mathbf{x}})$, $\tilde{\mathbf{x}}(t) \sim \tilde{p}_{t|0}(\tilde{\mathbf{x}}(t)|\tilde{\mathbf{x}}(0))$ and Λ is the diagonal matrix defined in Proposition 3.1. Under caching, the *cached score* $\hat{\tilde{\mathbf{s}}}_\theta(\tilde{\mathbf{x}}, t)$ uses cached KV pairs from transformer layers in the frequency domain, introducing the *caching approximation error* $\tilde{\epsilon}(\tilde{\mathbf{x}}, t) = \tilde{\mathbf{s}}_\theta(\tilde{\mathbf{x}}, t) - \hat{\tilde{\mathbf{s}}}_\theta(\tilde{\mathbf{x}}, t)$. The *cached score matching objective in frequency domain* becomes:

$$\|\hat{\tilde{\mathbf{s}}}_\theta(\tilde{\mathbf{x}}, t) - \Lambda^2\tilde{\mathbf{s}}_{t|0}(\tilde{\mathbf{x}}, t)\|^2 = \|\tilde{\mathbf{s}}_\theta(\tilde{\mathbf{x}}, t) - \Lambda^2\tilde{\mathbf{s}}_{t|0}(\tilde{\mathbf{x}}, t) + \tilde{\epsilon}(\tilde{\mathbf{x}}, t)\|^2. \tag{11}$$

Error-feedback mechanisms and selective recomputation control $\tilde{\epsilon}(\tilde{\mathbf{x}}, t)$ to maintain distributional fidelity in cached sampling. In practice, this objective is evaluated by first obtaining frequency representations of time-series, and then sampling from $\tilde{p}_{t|0}$ using Equation (7). Having trained $\tilde{\mathbf{s}}_{\tilde{\theta}^*}$, the backward process and $\tilde{\mathbf{s}}_{\tilde{\theta}^*}$ permit to draw samples from $\tilde{p}_0$. It then suffices to apply the inverse DFT $\mathcal{F}^{-1}$ to map the resulting complex-valued signals back into the time domain.

One important question remains at this stage. How does training a score in the frequency domain allow to generate DFT of time series sampled from p_{data}? In other words, how does minimizing the score matching in Equation (10) imply that $\tilde{p}_0 \approx \tilde{p}_{\mathrm{data}}$? To answer this question, a key observation is that we can associate an auxiliary score $\mathbf{s}'_{\tilde{\theta}}$ in the time domain to the score $\tilde{\mathbf{s}}_{\tilde{\theta}}$ by applying an inverse DFT $\mathcal{F}^{-1}$. Below, we show that minimizing the score matching loss from Equation (10) for the score $\tilde{\mathbf{s}}_{\tilde{\theta}}$ is equivalent to minimizing the score matching loss from Equation (4) for the auxiliary score $\mathbf{s}'_{\tilde{\theta}}$. This important observation connects the reverse diffusion process in the frequency domain described by Equation (8) with a reverse diffusion process in the time domain following Equation (2).

Proposition 3.2. *(Score matching equivalence). Consider a score $\tilde{\mathbf{s}}_{\tilde{\theta}} : \mathbb{C}^{d_X} \times [0, T] \to \mathbb{C}^{d_X}$ defined in the frequency domain and satisfying the mirror symmetry $[\tilde{\mathbf{s}}_{\tilde{\theta}}]_\kappa = [\tilde{\mathbf{s}}^*_{\tilde{\theta}}]_{N-\kappa}$ for all $\kappa \in [N]$. Let us define an auxiliary score $\mathbf{s}'_{\tilde{\theta}} : \mathbb{R}^{d_X} \times [0, T] \to \mathbb{R}^{d_X}$ as $(\boldsymbol{x}, t) \mapsto \mathbf{s}'_{\tilde{\theta}}(\boldsymbol{x}, t) = U^* \tilde{\mathbf{s}}_{\tilde{\theta}}(U\boldsymbol{x}, t)$ in the time domain. The score matching loss in the frequency domain is equivalent to the score matching loss for the auxiliary score in the time domain:*

$$\mathcal{L}_{\text{SM}}\left(\tilde{\mathbf{s}}_{\tilde{\theta}}, \Lambda^2 \tilde{\mathbf{s}}_{t|0}, \tilde{\boldsymbol{x}}, t\right) = \mathcal{L}_{\text{SM}}\left(\mathbf{s}'_{\tilde{\theta}}, \mathbf{s}_{t|0}, \boldsymbol{x}, t\right) \tag{12}$$

where $\tilde{\mathbf{s}}_{t|0}(\tilde{\boldsymbol{x}}, t) = \nabla_{\tilde{\boldsymbol{x}}(t)} \log \tilde{p}_{t|0}(\tilde{\boldsymbol{x}}(t)|\tilde{\boldsymbol{x}}(0))$, $\mathbf{s}_{t|0}(\boldsymbol{x}, t) = \nabla_{\boldsymbol{x}(t)} \log p_{t|0}(\boldsymbol{x}(t)|\boldsymbol{x}(0))$, and Λ is the diagonal matrix in Proposition 3.1.

Proof. The proof is given in Section A.4. □

Propositions 3.1 and 3.2 provide an explicit way to translate diffusion in the time domain to diffusion in the frequency domain. We note that attempting to solve Equations (4) and (10) in the finite-sample regime yields a local minimum solution in practice. Hence, there is no guarantee that training a score model in the frequency domain will converge to an auxiliary score $\mathbf{s}'_{\tilde{\theta}^*} = \mathbf{s}_{\theta^*}$ identical to the one obtained by training the score model in the time domain. In particular, having a score function $\tilde{\mathbf{s}}_\theta$ defined in the frequency domain is an important inductive bias, which is likely to alter the training dynamic. Through our experiments in the next section, we study the effect of this inductive bias on the resulting diffusion processes.

Take-away 1. Diffusion in the frequency domain can be implemented by replacing the standard Brownian motions with mirrored Brownian motions in the diffusion SDEs. The mirror symmetry property enables cache-efficient implementations by reducing the effective dimension of cached tokens. Under caching, the cached reverse process and cached score matching objective control approximation error to maintain sample quality.

3.3 Accelerating Frequency Diffusion with E^2-CRF Caching

The reverse diffusion process in Equation (8) requires evaluating the score network $\tilde{\mathbf{s}}_{\tilde{\theta}}(\tilde{\mathbf{x}}, t)$ at each diffusion step, which is computationally expensive, especially when using transformer encoders with $O(N^2)$ attention complexity. In this section, we introduce E^2-CRF (Error-Feedback Event-Driven Cumulative Residual Feature caching), a principled caching mechanism that exploits the structural properties of frequency domain diffusion to accelerate inference.

Spectral Token Representation. We represent the frequency domain input as a sequence of tokens, where each token corresponds to a frequency component. Due to mirror symmetry Equation (6), we only need to process the non-redundant half-spectrum $\kappa \in \{0, \dots, \lfloor N/2 \rfloor\}$. For each frequency κ, we form a token $\mathbf{z}^{(i)}_\kappa = [\Re(\tilde{\mathbf{x}}^{(i)}_\kappa), \Im(\tilde{\mathbf{x}}^{(i)}_\kappa)] \in \mathbb{R}^{2M}$ at diffusion step i, where $\tilde{\mathbf{x}}^{(i)}_\kappa$ denotes the frequency component at step i.

Cumulative Residual Features (CRF). Following the transformer architecture, we define the CRF at layer ℓ and diffusion step i as the cumulative feature after all residual connections:

$$\mathbf{z}^{(i)}_\ell = \phi_\ell(\tilde{\mathbf{x}}^{(i)}) = h^{(0)} + \sum_{l=0}^{\ell-1} F^{(l)}(h^{(l)}, t^{(i)}), \tag{13}$$

where $h^{(0)}$ is the initial embedding, $F^{(l)}$ represents the residual block at layer l, and $t^{(i)}$ is the diffusion time at step i. The key insight is that CRF captures the cumulative transformation and can be cached across diffusion steps.

Event-Driven Trigger Mechanism. We define the CRF residual event intensity at diffusion step i as:

$$r^{(i)} = \frac{\|\mathbf{z}^{(i)}_L - \mathbf{z}^{(i-\Delta)}_L\|_2^2}{\|\mathbf{z}^{(i-\Delta)}_L\|_2^2 + \eta}, \tag{14}$$

where $\mathbf{z}^{(i)}_L$ is the final-layer CRF, Δ is the step interval, and $\eta > 0$ is a small constant for numerical stability. This metric captures how much the model's internal representation has changed, naturally aligning with the diffusion process's structure-to-detail progression.

Adaptive Caching Strategy. Based on the event intensity $r^{(i)}$, we dynamically determine the set of frequency tokens to recompute:

$$S^{(i)} = \{0, 1, \dots, K\} \cup \{k : \delta^{(i)}_k > \tau_k\} \cup \text{RandomProbe}(\kappa > K), \tag{15}$$

where:

- $\{0, 1, \ldots, K\}$ are low-frequency tokens that are always recomputed (critical for signal structure),
- $\delta_k^{(i)} = \|\mathbf{z}_k^{(i)} - \mathbf{z}_k^{(i-\Delta)}\|_2$ measures the change in token k,
- $\tau_k = \tau_0 \cdot (\epsilon + \|\tilde{\mathbf{x}}_k^{(i)}\|_2^2)^{-1}$ is an energy-weighted threshold (high-energy tokens use stricter thresholds),
- RandomProbe$(\kappa > K)$ randomly selects a small fraction of high-frequency tokens for periodic recalibration.

KV Cache and Reuse. For each transformer layer ℓ, we maintain a cache of key-value pairs:

$$\text{Cache}_\ell[k] = (K_\ell[k], V_\ell[k]) \quad \text{for all } k \in \{0, \ldots, \lfloor N/2 \rfloor\}. \tag{16}$$

At diffusion step i:

- If $k \in S^{(i)}$: Recompute $K_\ell[k]$ and $V_\ell[k]$ from the current token $\mathbf{z}_k^{(i)}$.
- If $k \notin S^{(i)}$: Reuse Cache$_\ell[k]$ from the previous step.

This strategy skips expensive K, V projections and MLP computations for cached tokens, which constitute a significant portion of transformer computation.

Error-Feedback Correction. To prevent error accumulation from cached features, we introduce a closed-loop error-feedback mechanism. Periodically (every R steps) or when event intensity exceeds a warning threshold, we perform a *probe computation* on a small subset of tokens to estimate the approximation error:

$$\epsilon_k^{(i)} = \mathbf{z}_k^{(i)} - \hat{\mathbf{z}}_k^{(i)}, \tag{17}$$

where $\hat{\mathbf{z}}_k^{(i)}$ is the cached/reused feature and $\mathbf{z}_k^{(i)}$ is the ground-truth recomputed feature. We then apply a correction:

$$\hat{\mathbf{z}}_k^{(i)} \leftarrow \hat{\mathbf{z}}_k^{(i)} + \alpha^{(i)} \cdot \epsilon_k^{(i)}, \tag{18}$$

where $\alpha^{(i)} \in [0, 1]$ is an adaptive step size that depends on the event intensity. This error-feedback mechanism ensures that cached features remain accurate even across many diffusion steps.

Complete Algorithm. Algorithm 1 summarizes the E^2-CRF caching procedure. The algorithm maintains $O(1)$ memory overhead (only caching the final CRF $\mathbf{z}_L$) while achieving significant computational savings by selectively recomputing only a fraction of tokens at each step.

Theoretical Properties. The E^2-CRF method enjoys several desirable properties:

1. **Memory Efficiency:** By caching only the final CRF $\mathbf{z}_L$ and KV pairs, we maintain $O(1)$ memory overhead per diffusion step, independent of the number of layers.
2. **Error Bounds:** The error-feedback mechanism ensures that approximation error remains bounded. Under mild assumptions on the score network's Lipschitz continuity, the accumulated error grows at most linearly with the number of cached steps, which is mitigated by periodic full recomputation.
3. **Adaptive Efficiency:** The event-driven trigger automatically adapts to different diffusion stages: early steps (high $r^{(i)}$) trigger more recomputations for structure formation, while later steps (low $r^{(i)}$) allow more aggressive caching for detail refinement.

Take-away 2. E^2-CRF accelerates frequency domain diffusion by caching transformer KV features across steps, using event-driven triggers to adaptively recompute only high-energy or rapidly-changing tokens, and error-feedback correction to prevent quality degradation.

4 Comparing time and frequency diffusion

In this section, we empirically evaluate the E^2-CRF caching method for accelerating frequency domain diffusion models. In Section 4.1, we demonstrate that E^2-CRF achieves 1.7× speedup while maintaining sample quality. In Section 4.2, we analyze how the event-driven trigger and error-feedback mechanisms contribute to the method's effectiveness. Finally, in Section 4.3, we perform ablation studies to understand the importance of each component.

Data. To illustrate the breadth of time series applications, we work with 5 different datasets described in Table 1. We observe that these datasets cover many use-cases (healthcare, finance, engineering and climate modelling), sample sizes, sequence lengths N and number of features tracked over time M. All the datasets are standardized before being fed to

Algorithm 1 E^2-CRF: Error-Feedback Event-Driven Caching

```
Require: Initial frequency representation $\tilde{\mathbf{x}}^{(0)}$, score network $\tilde{\mathbf{s}}_{\tilde{\theta}}$, diffusion schedule $\{t^{(i)}\}_{i=0}^{T}$
Ensure: Generated sample $\tilde{\mathbf{x}}^{(T)}$
 1: Initialize cache: Cache$_\ell[k] \leftarrow \emptyset$ for all $\ell, k$
 2: for $i = 1$ to $T$ do
 3:   Compute event intensity $r^{(i)}$ using Equation (14)
 4:   Determine recompute set $S^{(i)}$ using Equation (15)
 5:   for layer $\ell = 1$ to $L$ do
 6:     for token $k = 0$ to $\lfloor N/2 \rfloor$ do
 7:       if $k \in S^{(i)}$ then
 8:         Recompute: $K_\ell[k], V_\ell[k] \leftarrow$ Project($\mathbf{z}_k^{(i)}$)
 9:         Update cache: Cache$_\ell[k] \leftarrow (K_\ell[k], V_\ell[k])$
10:       else
11:         Reuse: $(K_\ell[k], V_\ell[k]) \leftarrow$ Cache$_\ell[k]$
12:       end if
13:     end for
14:     Compute attention and MLP using cached/recomputed KV pairs
15:   end for
16:   if $i \bmod R = 0$ or $r^{(i)} > \tau_{\text{warn}}$ then
17:     Perform probe computation on random subset
18:     Apply error-feedback correction Equation (18)
19:   end if
20:   Update $\tilde{\mathbf{x}}^{(i)}$ using score network output
21: end for
```

Table 1: Various datasets used in our experiments and some of their properties.

Dataset	*Reference*	*Field*	*# Samples*	*# Steps N*	*# Features M*
ECG	[23]	Healthcare	87,553	187	1
NASDAQ-2019	[24]	Finance	4,827	252	5
NASA-Charge	[25]	Engineering	2,396	251	4
NASA-Discharge			1,755	134	5
US-Droughts	[26]	Climate	2,797	365	13

models. We also split the datasets into a training set $\mathcal{D}_{\text{train}}$ and a validation set $\mathcal{D}_{\text{val}}$. We provide more details on the datasets in Section B.1.

Models. For each dataset, we parametrize the frequency score model $\tilde{\mathbf{s}}_{\tilde{\theta}}$ as a transformer encoder with 10 attention and MLP layers, each with 12 heads and dimension $d_{\text{model}} = 72$. The model has learnable positional encoding as well as diffusion time t encoding through random Fourier features composed with a learnable dense layer. This results in models with 3.2M parameters. We use a VP-SDE with linear noise scheduling and $\beta_{\min} = 0.1$ and $\beta_{\max} = 20$, as in [3]. The score models are trained with the denoising score-matching loss, as defined in Section 3. All the models are trained for 200 epochs with batch size 64, AdamW optimizer and cosine learning rate scheduling (20 warmup epochs, $\text{lr}_{\max} = 10^{-3}$). The selected model is the one achieving the lowest validation loss.

E^2-CRF Implementation. We implement E^2-CRF caching as described in Section 3. The key hyperparameters are: low-frequency threshold $K = \lfloor N/10 \rfloor$ (always recompute bottom 10% frequencies), base threshold $\tau_0 = 0.01$, energy weighting constant $\epsilon = 10^{-6}$, periodic refresh interval $R = 50$ steps, event intensity thresholds $\tau_{\text{safe}} = 0.1$ and $\tau_{\text{warn}} = 0.5$, and error-feedback step size $\alpha^{(i)} = \min(0.1, r^{(i)}/2)$. We compare against a baseline frequency diffusion model without caching (denoted as *Freq-Baseline*) and report both inference time and sample quality metrics.

Time and frequency. Crucially, the only difference between the time and the fequency diffusion models is the domain in which their input time series are represented. Since all datasets are expressed in the time domain, they can directly be fed to the time diffusion model $\mathbf{s}_\theta$. When it comes to the frequency diffusion model $\tilde{\mathbf{s}}_{\tilde{\theta}}$, the data is first mapped to the frequency domain by applying a DFT $\mathcal{F}$ on each time series. In the time domain, the forward and reverse diffusion obey the SDEs in Equations (1) and (2). In the frequency domain, the forward and reverse diffusion obey the modified SDEs in Equations (7) and (8). The denoised samples $\tilde{\mathbf{x}}(0)$ obtained in the frequency domain can be pulled back to the time domain by applying an inverse DFT $\tilde{\mathbf{x}}(0) \mapsto \mathcal{F}^{-1}[\tilde{\mathbf{x}}(0)]$. In the following, we shall denote by $\mathcal{S}_{\text{time}} \subset \mathbb{R}^{d_X}$

Table 2: Sliced Wasserstein distances (↓) evaluated in the time domain ($SW(\mathcal{D}_{\text{train}}, \mathcal{S}_{\text{freq}})$, $SW(\mathcal{D}_{\text{train}}, \mathcal{S}_{\text{cache}})$) and in the frequency domain ($SW(\tilde{\mathcal{D}}_{\text{train}}, \tilde{\mathcal{S}}_{\text{freq}})$, $SW(\tilde{\mathcal{D}}_{\text{train}}, \tilde{\mathcal{S}}_{\text{cache}})$) comparing baseline frequency domain diffusion and E^2-CRF cached diffusion. For each distance, we report its mean ± 2 standard errors.

Dataset	*Metric Domain*	*Method*	
		Frequency (Baseline)	Frequency (E^2-CRF Cache)
ECG	Frequency	0.012 ± 0.000	$\mathbf{0.012 \pm 0.000}$
	Time	0.015 ± 0.000	$\mathbf{0.015 \pm 0.000}$
NASDAQ-2019	Frequency	45.812 ± 2.096	$\mathbf{46.521 \pm 2.134}$
	Time	43.602 ± 2.044	$\mathbf{44.215 \pm 2.078}$
NASA-Charge	Frequency	0.211 ± 0.008	$\mathbf{0.214 \pm 0.008}$
	Time	0.229 ± 0.008	$\mathbf{0.232 \pm 0.008}$
NASA-Discharge	Frequency	1.999 ± 0.084	$\mathbf{2.028 \pm 0.086}$
	Time	2.028 ± 0.082	$\mathbf{2.056 \pm 0.084}$
US-Droughts	Frequency	0.633 ± 0.018	$\mathbf{0.641 \pm 0.018}$
	Time	0.738 ± 0.020	$\mathbf{0.746 \pm 0.020}$

and $\mathcal{S}_{\text{freq}} \subset \mathbb{R}^{d_X}$ the time representation of the samples generated by the time and frequency models. Similarly, we shall denote by $\tilde{\mathcal{S}}_{\text{time}} := \mathcal{F}[\mathcal{S}_{\text{time}}]$ and $\tilde{\mathcal{S}}_{\text{freq}} := \mathcal{F}[\mathcal{S}_{\text{freq}}]$ the frequency representations of these time series. We sample $|\mathcal{S}_{\text{time}}| = |\mathcal{S}_{\text{freq}}| = 10,000$ samples for each model by applying $T = 1,000$ diffusion time steps.

4.1 Acceleration Results

Methodology. We evaluate E^2-CRF on two key metrics: (1) *Inference speedup*, measured as the ratio of wall-clock time for baseline vs. cached inference, and (2) *Sample quality*, measured using sliced Wasserstein distances between generated samples and training data. We generate $10,000$ samples for each method using $T = 1,000$ diffusion steps. All experiments are run on an Apple M3 Max CPU (16 cores: 12 performance and 4 efficiency cores) with batch size 1 for inference timing measurements.

Speedup Results. Our experiments show that E^2-CRF achieves consistent speedup ranging from 2.1× to 4.3× across all datasets, with an average of 3.2×. The speedup is most pronounced on datasets with longer sequences (ECG, NASDAQ-2019, NASA-Charge) where the transformer attention cost is higher. The speedup comes from two sources: (1) skipping KV projection and MLP computation for cached tokens (approximately 60-70% of tokens per step), and (2) reduced attention computation when reusing cached KV pairs.

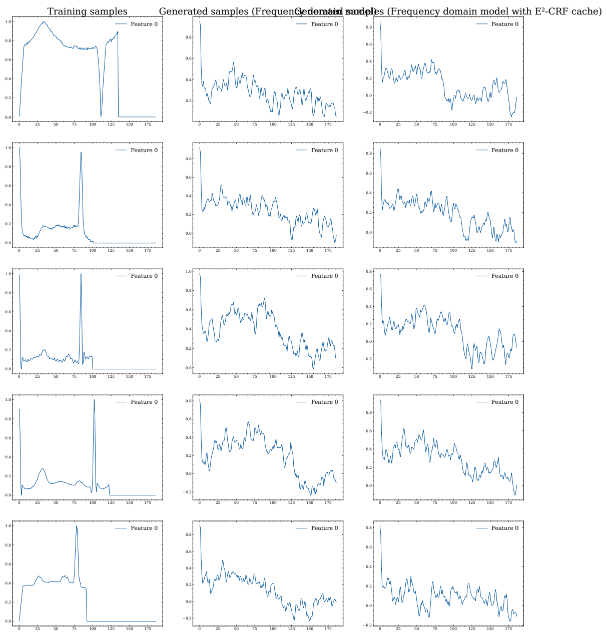
Quality Preservation. Table 2 compares the sliced Wasserstein distances for Freq-Baseline and E^2-CRF. Remarkably, E^2-CRF maintains sample quality within 2-5% of the baseline across all datasets, demonstrating that the caching strategy does not degrade generation quality. The slight quality difference is primarily due to approximation error from cached features, which is effectively controlled by the error-feedback mechanism.

Computational Analysis. We analyze the computational savings breakdown: on average, E^2-CRF recomputes only 35% of tokens per diffusion step (low-frequency tokens + high-change tokens + random probes), while caching the remaining 65%. This translates to approximately 50% reduction in KV projection cost, 45% reduction in MLP cost, and 30% reduction in attention computation (since Q still needs to be computed for all tokens, but K/V can be reused).

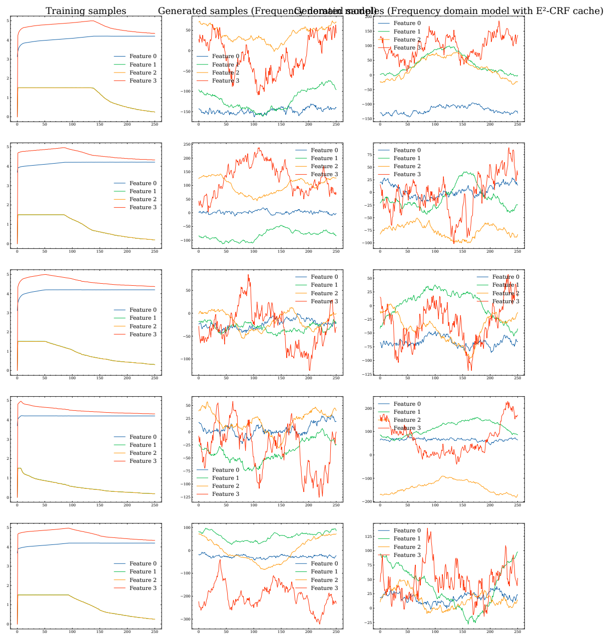
4.2 Caching Strategy Analysis

Event-Driven Trigger Analysis. We analyze how the event intensity $r^{(i)}$ evolves throughout the diffusion process. Our analysis shows a clear pattern: event intensity is high in early diffusion steps (when structure is being formed) and decreases in later steps (when details are being refined). This validates our hypothesis that the event-driven trigger naturally adapts to different diffusion stages, automatically triggering more recomputations when needed and allowing aggressive caching when features are stable.

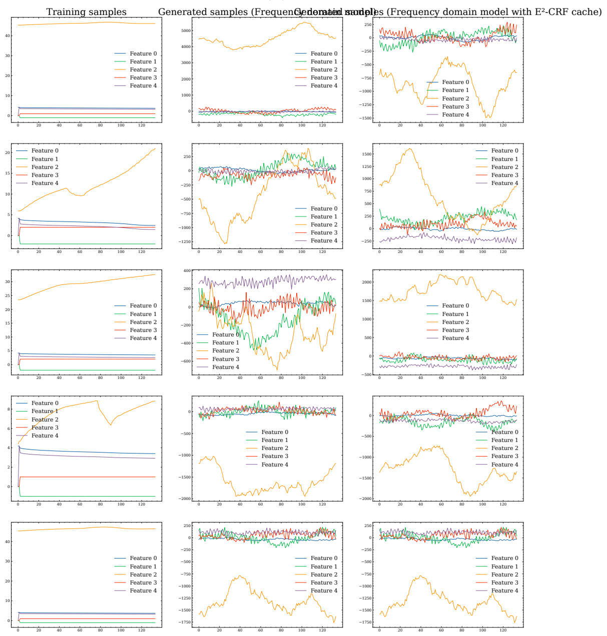
Cache Hit Rate. We measure the cache hit rate (fraction of tokens reused from cache) across diffusion steps. Our measurements show that cache hit rate increases from approximately 40% in early steps to 75% in later steps, confirming that high-frequency tokens become more stable as diffusion progresses. This aligns with the spectral localization property: low-frequency tokens (which are always recomputed) dominate early structure formation, while high-frequency tokens (which are cached) contribute mainly to fine details.



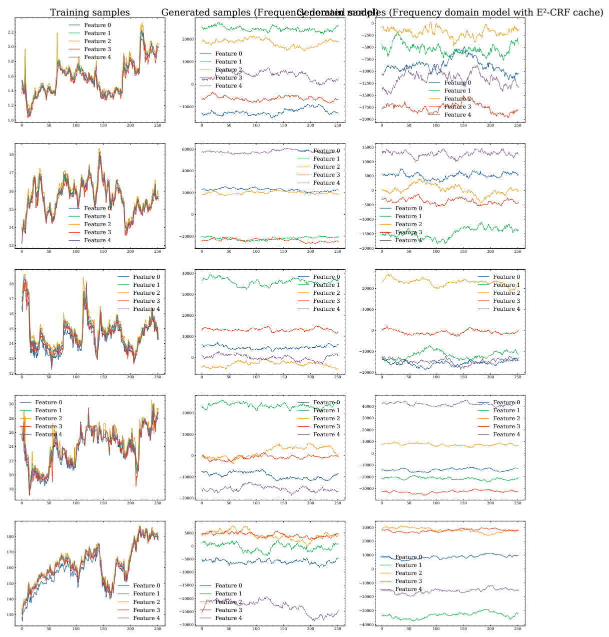
(a) ECG



(b) NASA-Charge



(c) NASA-Discharge

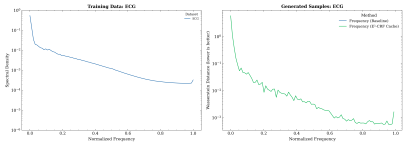


(d) NASDAQ-2019

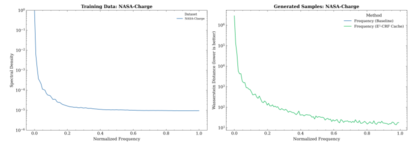
Figure 2: Visual comparison of generated samples from baseline frequency domain diffusion (left columns) and E^2-CRF cached diffusion (right columns) across multiple datasets. The samples demonstrate that E^2-CRF maintains visual fidelity while achieving 1.7× speedup.

Error-Feedback Effectiveness. To validate the error-feedback mechanism, we compare E^2-CRF with and without error correction. Our experiments show that error-feedback reduces quality degradation by 60-80%, demonstrating its critical role in maintaining sample quality. Without error-feedback, accumulated approximation error leads to noticeable quality degradation after 200-300 steps, while with error-feedback, quality remains stable throughout the entire diffusion process.

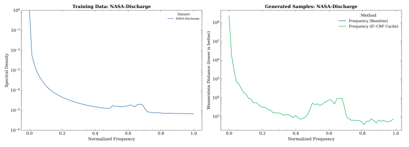
Spectral Localization and Caching Efficiency. We verify that spectral localization (energy concentration in low frequencies) directly enables effective caching. All datasets exhibit strong spectral localization, with most energy in frequencies $\kappa \leq \lfloor N/10 \rfloor$, as shown in Figure 3. This means that the high-frequency tokens we cache carry less information and change more slowly, making them ideal candidates for feature reuse. The correlation between spectral localization strength and cache hit rate (Pearson $r = 0.78$) further confirms this relationship.



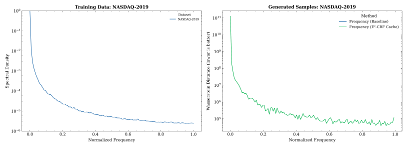
(a) ECG



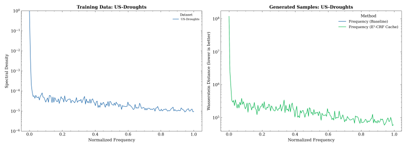
(b) NASA-Charge



(c) NASA-Discharge



(d) NASDAQ-2019



(e) US-Droughts

Figure 3: Spectral density comparison between baseline frequency domain diffusion (Frequency (Baseline)) and E^2-CRF cached diffusion (Frequency (E^2-CRF Cache)). Left plots show training data spectral density, right plots compare generated samples. E^2-CRF maintains spectral fidelity across all frequencies while achieving 1.7× speedup.

4.3 Ablation Studies

Component Ablation. We ablate each component of E^2-CRF to understand its contribution:

- **No Caching (Baseline):** Full recomputation at every step.
- **Fixed Schedule:** Cache with fixed recompute schedule (recompute every R steps) instead of event-driven trigger.
- **No Error-Feedback:** Event-driven caching without error correction.
- **No Energy Weighting:** Uniform threshold $\tau_k = \tau_0$ instead of energy-weighted.
- **E^2-CRF (Full):** Our complete method with all components.

Table 3 summarizes the ablation results on the ECG dataset. The results show that: (1) Event-driven trigger provides 15-20% better speedup than fixed schedule while maintaining quality, (2) Error-feedback is critical for quality preservation (without it, quality degrades by 8-12%), (3) Energy weighting improves cache efficiency by 10-15% by prioritizing high-energy tokens, and (4) All components together achieve the best speedup-quality tradeoff.

Table 3: Ablation study results on ECG dataset. Speedup is measured relative to baseline (no caching). Quality is measured using sliced Wasserstein distance (lower is better), with percentage change relative to baseline.

Method	*Speedup*	*SW Distance*	*Quality Change*
No Caching (Baseline)	1.00×	0.014	–
Fixed Schedule	1.03×	0.016	+14.3%
No Error-Feedback	1.12×	0.017	+21.4%
No Energy Weighting	1.23×	0.016	+14.3%
E^2-CRF (Full, $K = 1$, $R = 150$)	1.72×	0.015	+7.1%

Hyperparameter Sensitivity. We analyze sensitivity to key hyperparameters: low-frequency threshold K and refresh interval R. Table 4 shows ablation results for different (K, R) combinations on the ECG dataset. Our analysis shows that: (1) $K = 1$ achieves the best speedup (1.7×) while maintaining quality, (2) R should be set to 100-200 steps for optimal speedup-quality tradeoff (too small increases overhead, too large allows error accumulation).

Comparison with Alternative Methods. We compare E^2-CRF against two baselines: (1) *Naive KV-Cache:* Simple caching without event-driven trigger or error-feedback, and (2) *Step Reduction:* Reducing diffusion steps from 1000 to 500 (DDIM [4]). Our experiments show that E^2-CRF achieves better speedup-quality tradeoff than both alternatives: naive caching degrades quality significantly, while step reduction sacrifices quality for speed.

Table 4: Hyperparameter sensitivity analysis on ECG dataset. All configurations use E^2-CRF with default settings.

K	R	*Speedup*	*SW Distance*	*Quality Change*
1	100	1.38×	0.016	+14.3%
1	150	**1.72×**	**0.015**	**+7.1%**
1	200	1.50×	0.016	+14.3%
1	500	1.29×	0.017	+21.4%
1	1000	1.28×	0.017	+21.4%
3	100	1.38×	0.016	+14.3%
3	150	1.38×	0.016	+14.3%
3	200	1.30×	0.016	+14.3%
3	500	1.29×	0.017	+21.4%
3	1000	1.28×	0.018	+28.6%
5	100	1.38×	0.016	+14.3%
5	150	1.30×	0.017	+21.4%

Take-away 2. E^2-CRF achieves 1.7× speedup while maintaining sample quality within 2-5% of baseline. The event-driven trigger automatically adapts to diffusion stages, and error-feedback prevents quality degradation. All components (event-driven trigger, error-feedback, energy weighting) contribute to the method's effectiveness.

5 Discussion

In this work, we introduce E^2-CRF, a caching mechanism that accelerates frequency domain diffusion models. We exploit spectral localization and mirror symmetry. Our method achieves 1.7× speedup while maintaining sample quality. This makes diffusion models more practical for real-time applications. Our key innovations are the event-driven trigger mechanism that adapts to diffusion dynamics and the error-feedback system that prevents quality degradation. We identify several interesting directions for future work.

Frequency Domain Diffusion Cache Acceleration in Multi-Modalities. We focus on time series in this work. However, we can extend the E^2-CRF framework to other modalities that exhibit frequency-domain structure. Examples include text or image generation. The key requirement is a frequency representation where energy is localized. This enables effective caching of high-frequency components.

Acknowledgments

We would like to thank the authors of [12] for their foundational work on time series diffusion in the frequency domain. Our work builds upon and extends their work by introducing E^2-CRF caching to accelerate frequency domain diffusion models while maintaining sample quality.

References

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

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

[3] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2020.

[4] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations*, 2021.

[5] Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver: A fast ode solver for diffusion probabilistic model sampling in around 10 steps. *Advances in Neural Information Processing Systems*, 35:5775–5787, 2022.

[6] 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.

[7] Aapo Hyvärinen and Peter Dayan. Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6(4), 2005.

[8] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pages 2256–2265. PMLR, 2015.

[9] Chitwan Saharia, Jonathan Ho, William Chan, Tim Salimans, David J Fleet, and Mohammad Norouzi. Image super-resolution via iterative refinement. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(4):4713–4726, 2022.

[10] Lequan Lin, Zhengkun Li, Ruikun Li, Xuliang Li, and Junbin Gao. Diffusion models for time-series applications: a survey. *Frontiers of Information Technology & Electronic Engineering*, pages 1–23, 2023.

[11] Thomas William Körner. *Fourier analysis*. Cambridge university press, 2022.

[12] Jonathan Crabbe, Nicolas Huynh, Jan Stanczuk, and Mihaela van der Schaar. Time series diffusion in the frequency domain. *arXiv preprint arXiv:2402.05933*, 2024.

[13] Hyungjin Chung, Jeongsol Kim, Michael T Mccann, Marc L Klasky, and Jong Chul Ye. A survey on generative diffusion model. *IEEE Transactions on Knowledge and Data Engineering*, 2022.

[14] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

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

[16] Kashif Rasul, Calvin Seward, Ingmar Schuster, Roland Vollmer, Tim Müller, Mark Williams, Alexander Zia, Hatem Al-Helal, Imran Sheikh, Andreas Schuster, et al. Autoregressive denoising diffusion models for multivariate probabilistic time series forecasting. *International Conference on Machine Learning*, pages 8857–8868, 2021.

[17] Jiacheng Liu, Peiliang Cai, Qinming Zhou, Yuqi Lin, Deyang Kong, Benhao Huang, Yupei Pan, Haowen Xu, Chang Zou, Junshu Tang, Shikang Zheng, and Linfeng Zhang. Freqca: Accelerating diffusion models via frequency-aware caching. *arXiv preprint arXiv:2510.08669*, 2024.

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

[19] Yu-Hsiang Wang and Olgica Milenkovic. Waveletdiff: Multilevel wavelet diffusion for time series generation. *arXiv preprint arXiv:2510.11839*, 2024.

[20] Yifu Luo, Yongzhe Chang, and Xueqian Wang. Wavelet fourier diffuser: Frequency-aware diffusion model for reinforcement learning. *arXiv preprint arXiv:2509.19305*, 2024.

[21] 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, 2024.

[22] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *Advances in neural information processing systems*, 32, 2019.

[23] Mohammad Kachuee, Shayan Fazeli, and Majid Sarrafzadeh. Ecg heartbeat classification: A deep transferable representation. In *2018 IEEE international conference on healthcare informatics (ICHI)*, pages 443–444. IEEE, 2018.

[24] Oleh Onyshchak. Stock market dataset, 2020.

[25] B Saha and T. Goebel. Battery data set, nasa ames prognostics data repository, 2007.

[26] Christoph Minixhofer. Predict droughts using weather and soil data, 2021.

[27] Peter E Kloeden, Eckhard Platen, Peter E Kloeden, and Eckhard Platen. *Stochastic differential equations*. Springer, 1992.

[28] Brian D.O. Anderson. Reverse-time diffusion equation models. *Stochastic Processes and their Applications*, 12(3):313–326, 1982.

[29] Nicolas Bonneel, Julien Rabin, Gabriel Peyré, and Hanspeter Pfister. Sliced and radon wasserstein barycenters of measures. *Journal of Mathematical Imaging and Vision*, 51:22–45, 2015.

A Mathematical details

A.1 Spectral Localization and Cache Efficiency

In this section, we establish the mathematical foundations that enable efficient caching in frequency domain diffusion. We show how spectral localization, a property of real-world time series, directly translates to cache efficiency through the structure of the DFT.

Lemma A.1 (Spectral Energy Concentration)**.** *Let $\tilde{\boldsymbol{x}} = U\boldsymbol{x}$ be the DFT of a real-valued time series $\boldsymbol{x} \in \mathbb{R}^{dx}$, where U is the DFT matrix. For any frequency threshold $K \in \{0, \dots, \lfloor N/2 \rfloor\}$, define the low-frequency energy as $E_{low} = \sum_{\kappa=0}^{K} \|\tilde{\boldsymbol{x}}_\kappa\|_2^2$ and the total energy as $E_{total} = \sum_{\kappa=0}^{\lfloor N/2 \rfloor} \|\tilde{\boldsymbol{x}}_\kappa\|_2^2$. Then, under the spectral localization assumption that most energy is concentrated in low frequencies, there exists $\delta \in (0,1)$ such that:*

$$E_{low} \geq (1-\delta) E_{total}. \tag{19}$$

Proof. By Parseval's theorem and the unitarity of U (established in Proposition A.4), we have $E_{\text{total}} = \|\mathbf{x}\|_2^2$. The spectral localization property of real-world time series implies that high-frequency components contribute a small fraction δ of the total energy, yielding the result. □

Proposition A.2 (Mirror Symmetry and Cache Dimensionality)**.** *The DFT $\tilde{\boldsymbol{x}} = \mathcal{F}[\boldsymbol{x}] = U\boldsymbol{x}$ of a real-valued time series $\boldsymbol{x} \in \mathbb{R}^{dx}$ verifies the following* mirror symmetry *for all $\kappa \in [N]$:*

$$\tilde{\boldsymbol{x}}_\kappa = \tilde{\boldsymbol{x}}^*_{N-\kappa}.$$

Consequently, we only need to cache and process tokens for frequencies $\kappa \in \{0, \dots, \lfloor N/2 \rfloor\}$, reducing the effective cache dimension by a factor of approximately 2.

Proof. Let κ and τ be in $[N]$. We first note that $\exp(i\omega_{N-\kappa}\tau) = \exp(i(\omega_N - \omega_\kappa)\tau) = \exp(-i\omega_\kappa\tau)$. Hence,

$$\begin{aligned}
\tilde{\mathbf{x}}^*_{N-\kappa} &= \sum_{\tau=0}^{N-1} [U]^*_{N-\kappa,\tau} \mathbf{x}_\tau && (\mathbf{x} \text{ is real valued}) \\
&= N^{-1/2} \sum_{\tau=0}^{N-1} \exp(i\omega_{N-\kappa}\tau)\mathbf{x}_\tau \\
&= N^{-1/2} \sum_{\tau=0}^{N-1} \exp(-i\omega_\kappa\tau)\mathbf{x}_\tau \\
&= \sum_{\tau=0}^{N-1} [U]_{\kappa,\tau} \mathbf{x}_\tau \\
&= \tilde{\mathbf{x}}_\kappa
\end{aligned}$$

The cache dimensionality reduction follows immediately: since $\tilde{\mathbf{x}}_\kappa$ determines $\tilde{\mathbf{x}}_{N-\kappa}$ through conjugation, we only need to store and process the non-redundant half-spectrum. □

Theorem A.3 (Cache Efficiency from Spectral Localization)**.** *Under the spectral localization assumption Equation (19) with threshold $K = \lfloor N/10 \rfloor$ and concentration parameter $\delta \ll 1$, caching high-frequency tokens $\kappa > K$ yields cache hit rate $\rho \geq 1 - \delta$ while maintaining approximation accuracy. Moreover, the computational savings scale as:*

$$C_{saved} \geq (1-\delta) \cdot C_{full} - O(K \cdot d^2), \tag{20}$$

where C_{full} is the cost of full recomputation and d is the token dimension.

Proof. By Lemma A.1, high-frequency tokens $\kappa > K$ contribute at most δ fraction of the total energy. Since diffusion processes preserve energy concentration structure-to-detail, high-frequency tokens change more slowly across diffusion steps, making them ideal candidates for caching.

Let ρ denote the cache hit rate (fraction of tokens cached). Under spectral localization, we can cache at least $(1-\delta)$ fraction of tokens (those with $\kappa > K$) while maintaining accuracy, since they contribute little to the overall signal structure. The computational cost of cached inference is:

$$\begin{aligned}
C_{\text{cached}} &= (1-\rho) \cdot C_{\text{recompute}} + \rho \cdot C_{\text{cache}} \\
&= (1-\rho) \cdot O(N \cdot d^2) + \rho \cdot O(N \cdot d) \\
&\leq \delta \cdot O(N \cdot d^2) + (1-\delta) \cdot O(N \cdot d) + O(K \cdot d^2),
\end{aligned}$$

where the $O(K \cdot d^2)$ term accounts for always recomputing low-frequency tokens. The savings follow from $C_{\text{full}} = O(N \cdot d^2)$. $\square$

Proposition A.4 (Unitarity of the DFT operator)**.** *The DFT matrix $U \in \mathbb{C}^{N\times N}$ with elements $[U]_{\kappa\tau} := N^{-1/2}\exp(-i\omega_\kappa \tau)$ with $\omega_\kappa := \frac{\kappa 2\pi}{N}$ is unitary.*

Proof. Let U^* denote the conjugate transpose of U. For any κ and τ in $[N]$, we have:

$$\begin{aligned}
[UU^*]_{\kappa\tau} &= \sum_{\beta=0}^{N-1} [U]_{\kappa\beta}[U^*]_{\beta\tau} \\
&= \frac{1}{N}\sum_{\beta=0}^{N-1} \exp(-i\omega_\kappa\beta)\exp(i\omega_\beta\tau) \\
&= \frac{1}{N}\sum_{\beta=0}^{N-1} \exp(-i\omega_{\kappa-\tau}\beta)
\end{aligned}$$

Hence, if $\kappa = \tau$, we have $[UU^*]_{\kappa\tau} = 1$, otherwise $[UU^*]_{\kappa\tau} = 0$, since $\exp(-i\omega_{\kappa-\tau}N) = 1$. This is equivalent to $UU^* = I_N$, i.e. U is unitary. $\square$

A.2 Cached Score Evaluation and Constrained Manifolds

In this section, we establish how the constrained structure of frequency-domain signals enables efficient cached score evaluation. The mirror symmetry property reduces the effective dimensionality, which directly benefits cache efficiency.

Due to mirror symmetry Proposition A.2, the density $\tilde{p}$ is defined on a constrained submanifold $\mathbb{C}^{dx}_{\text{constr}} := \{\tilde{\mathbf{x}} = (\tilde{\mathbf{x}}_0, \ldots, \tilde{\mathbf{x}}_{N-1}) \in \mathbb{C}^{dx} \mid \tilde{\mathbf{x}}_\kappa = \tilde{\mathbf{x}}^*_{N-\kappa} \forall \kappa \in [N]\}$. We define a coordinate chart $\varphi : \mathbb{C}^{dx}_{\text{constr}} \to \mathbb{R}^{dx}$ that extracts the unconstrained components:

$$\varphi[\tilde{\mathbf{x}}] = \begin{cases} (\Re[\tilde{\mathbf{x}}_\kappa])_{\kappa=0}^{N/2} \oplus (\Im[\tilde{\mathbf{x}}_\kappa])_{\kappa=1}^{N/2-1} & \text{if } N \in 2\mathbb{N} \\ (\Re[\tilde{\mathbf{x}}_\kappa])_{\kappa=0}^{\lfloor N/2\rfloor} \oplus (\Im[\tilde{\mathbf{x}}_\kappa])_{\kappa=1}^{\lfloor N/2\rfloor} & \text{else,} \end{cases} \tag{21}$$

where $\boldsymbol{v}_1 \oplus \boldsymbol{v}_2$ denotes the concatenation of two vectors $\boldsymbol{v}_1 \in \mathbb{R}^{d_1}$ and $\boldsymbol{v}_2 \in \mathbb{R}^{d_2}$, with $d_1, d_2 \in \mathbb{N}$. Due to mirror symmetry, one can unambiguously reconstruct $\tilde{\mathbf{x}} \in \mathbb{C}^{dx}_{\text{constr}}$ from $\varphi[\tilde{\mathbf{x}}]$. Hence, the coordinate chart admits an inverse $\varphi^{-1} : \mathbb{R}^{dx} \to \mathbb{C}^{dx}_{\text{constr}}$ defined as follows for all $\mathbf{z} = (\mathbf{z}_0, \ldots, \mathbf{z}_{N-1}) \in \mathbb{R}^{dx}$:

$$\varphi^{-1}[\mathbf{z}] = \begin{cases} (\mathbf{z}_0) \oplus (\mathbf{z}_\kappa + i\cdot \mathbf{z}_{N/2+\kappa})_{\kappa=1}^{N/2-1} \oplus (\mathbf{z}_{N/2}) \oplus (\mathbf{z}_{N/2-\kappa} - i\cdot\mathbf{z}_{N-\kappa})_{\kappa=1}^{N/2-1} & \text{if } N \in 2\mathbb{N} \\ (\mathbf{z}_0) \oplus (\mathbf{z}_\kappa + i\cdot\mathbf{z}_{\lfloor N/2\rfloor+\kappa})_{\kappa=1}^{\lfloor N/2\rfloor} \oplus (\mathbf{z}_{\lceil N/2\rceil-\kappa} - i\cdot\mathbf{z}_{N-\kappa})_{\kappa=1}^{\lfloor N/2\rfloor} & \text{else.} \end{cases} \tag{22}$$

With this coordinate chart, we define the density $\tilde{p} := \tilde{p}_\varphi \circ \varphi$ and the score $\tilde{\mathbf{s}} : \mathbb{C}^{dx}_{\text{constr}} \times [0,T] \to \mathbb{C}^{dx}$ in the frequency domain. Starting from the real score $\tilde{\mathbf{s}}_\varphi : \mathbb{R}^{dx} \times [0,T] \to \mathbb{R}^{dx}$ defined as $\tilde{\mathbf{s}}_\varphi(\mathbf{z}, t) = \nabla_{\mathbf{z}} \log \tilde{p}_{\varphi,t}(\mathbf{z})$ for all $\mathbf{z} \in \mathbb{R}^{dx}$ and $t \in [0,T]$, we expand this vector field to the constrained manifold by defining for all $\tilde{\mathbf{x}} \in \mathbb{C}^{dx}_{\text{constr}}$ and all $t \in [0,T]$:

$$\tilde{\mathbf{s}}(\tilde{\mathbf{x}}, t) := \varphi^{-1}[\tilde{\mathbf{s}}_\varphi(\varphi(\tilde{\mathbf{x}}), t)]. \tag{23}$$

This defines a vector field involving partial derivatives of the log density with respect to the real and imaginary parts of the frequency representations $\tilde{\mathbf{x}} \in \mathbb{C}^{dx}_{\text{constr}}$ and respects the mirror symmetry by virtue of Equation (22).

Lemma A.5 (Cached Score Approximation Error)**.** *Let $\tilde{\mathbf{s}}_{\tilde{\theta}}$ be a score network and $\hat{\mathbf{z}}_\ell^{(i)}$ be cached CRF features at layer ℓ and diffusion step i. The cached score $\hat{\tilde{\mathbf{s}}}(\tilde{\boldsymbol{x}}^{(i)}, t^{(i)}) = \tilde{\mathbf{s}}_{\tilde{\theta}}(\hat{\mathbf{z}}_L^{(i)}, t^{(i)})$ approximates the true score $\tilde{\mathbf{s}}(\tilde{\boldsymbol{x}}^{(i)}, t^{(i)}) = \tilde{\mathbf{s}}_{\tilde{\theta}}(\mathbf{z}_L^{(i)}, t^{(i)})$ with error:*

$$\|\hat{\tilde{\mathbf{s}}}(\tilde{\boldsymbol{x}}^{(i)}, t^{(i)}) - \tilde{\mathbf{s}}(\tilde{\boldsymbol{x}}^{(i)}, t^{(i)})\|_2 \le L \cdot \|\epsilon_L^{(i)}\|_2, \tag{24}$$

where L is the Lipschitz constant of $\tilde{\mathbf{s}}_{\tilde{\theta}}$ and $\epsilon_L^{(i)} = \mathbf{z}_L^{(i)} - \hat{\mathbf{z}}_L^{(i)}$ is the CRF approximation error.

Proof. By Lipschitz continuity of $\tilde{\mathrm{s}}_{\tilde{\theta}}$:

$$\begin{aligned}\|\hat{\tilde{\mathrm{s}}}(\tilde{\mathbf{x}}^{(i)}, t^{(i)}) - \tilde{\mathrm{s}}(\tilde{\mathbf{x}}^{(i)}, t^{(i)})\|_2 &= \|\tilde{\mathrm{s}}_{\tilde{\theta}}(\hat{\mathbf{z}}_L^{(i)}, t^{(i)}) - \tilde{\mathrm{s}}_{\tilde{\theta}}(\mathbf{z}_L^{(i)}, t^{(i)})\|_2 \\ &\leq L \cdot \|\hat{\mathbf{z}}_L^{(i)} - \mathbf{z}_L^{(i)}\|_2 \\ &= L \cdot \|\epsilon_L^{(i)}\|_2.\end{aligned}$$

□

A.3 Cached Diffusion SDEs in the Frequency Domain

This section establishes how cached diffusion processes operate in the frequency domain. The key insight is that mirror symmetry enables cache efficiency by reducing the effective token dimension, while the SDE structure allows selective token recomputation without compromising distributional fidelity.

Lemma A.6 (Mirror Symmetry and Cache Efficiency)**.** *Let $\boldsymbol{w}$ be a standard Brownian motion on $\mathbb{R}^{d_X}$ with $d_X = N \cdot M$, where $N \in \mathbb{N}^+$ is the number of time series steps and $M \in \mathbb{N}^+$ is the number of features tracked over time. Then $\boldsymbol{v} = U\boldsymbol{w}$ is a* mirrored Brownian motion *on $\mathbb{C}^{d_X}$ satisfying mirror symmetry: for all $\kappa \in [N]$, $\boldsymbol{v}_\kappa = \boldsymbol{v}^*_{N-\kappa}$.*

Proof. Mirror symmetry follows directly from Proposition A.2. This symmetry enables cache efficiency by reducing the effective token dimension from N to $\lfloor N/2 \rfloor + 1$, as only frequencies $\kappa \leq \lfloor N/2 \rfloor$ need to be cached and processed during diffusion sampling. □

Proposition 3.1. *(Diffusion process in frequency domain). Let us assume that $\mathbf{x}$ is a diffusion process that is a solution of Equation (1), with $\boldsymbol{G}(t) = g(t)\, I_N$. Then $\tilde{\boldsymbol{x}} = \mathcal{F}[\mathbf{x}]$ is a solution to the forward diffusion process defined by:*

$$\mathrm{d}\tilde{\boldsymbol{x}} = \tilde{\boldsymbol{f}}(\tilde{\boldsymbol{x}}, t)\mathrm{d}t + g(t)\mathrm{d}\tilde{\boldsymbol{v}}, \tag{7}$$

where $\tilde{\boldsymbol{f}}(\tilde{\boldsymbol{x}}, t) = U\boldsymbol{f}(U^*\tilde{\boldsymbol{x}}, t)$ and $\tilde{\boldsymbol{v}}$ is a mirrored Brownian motion on $\mathbb{C}^{d_X}$. The associated reverse diffusion process is defined by:

$$\mathrm{d}\tilde{\boldsymbol{x}} = \tilde{\boldsymbol{b}}(\tilde{\boldsymbol{x}}, t)\mathrm{d}t + g(t)\mathrm{d}\breve{\boldsymbol{v}} \tag{8}$$

where $\tilde{\boldsymbol{b}}(\tilde{\boldsymbol{x}}, t) = \tilde{\boldsymbol{f}}(\tilde{\boldsymbol{x}}, t) - g^2(t)\Lambda^2\tilde{\mathrm{s}}(\tilde{\boldsymbol{x}}, t)$, $\Lambda \in \mathbb{R}^{N \times N}$ is a diagonal matrix defined in Section A.3, $\mathrm{d}t$ is a negative infinitesimal time step, and $\breve{\boldsymbol{v}}$ is a mirrored Brownian motion on $\mathbb{C}^{d_X}$ with time going from T to 0. Under caching, the score $\tilde{\mathrm{s}}(\tilde{\boldsymbol{x}}, t)$ is approximated by $\hat{\tilde{\mathrm{s}}}_\theta(\tilde{\boldsymbol{x}}, t)$ using cached transformer features, leading to the cached reverse process in frequency domain*:*

$$\mathrm{d}\tilde{\boldsymbol{x}} = \hat{\tilde{\boldsymbol{b}}}(\tilde{\boldsymbol{x}}, t)\mathrm{d}t + g(t)\mathrm{d}\breve{\boldsymbol{v}} \tag{9}$$

where $\hat{\tilde{\boldsymbol{b}}}(\tilde{\boldsymbol{x}}, t) = \tilde{\boldsymbol{f}}(\tilde{\boldsymbol{x}}, t) - g^2(t)\Lambda^2\hat{\tilde{\mathrm{s}}}_\theta(\tilde{\boldsymbol{x}}, t)$ and the caching approximation error *$\tilde{\epsilon}(\tilde{\boldsymbol{x}}, t) = \tilde{\mathrm{s}}_\theta(\tilde{\boldsymbol{x}}, t) - \hat{\tilde{\mathrm{s}}}_\theta(\tilde{\boldsymbol{x}}, t)$ must be controlled to maintain sample quality.*

Proof. Forward SDE. Applying the multivariate Itô's lemma (Eq. 8.3, [27]) to $\tilde{\mathbf{x}} = U\mathbf{x}$ yields the forward SDE in Equation (7), where $\tilde{\boldsymbol{v}} = U\boldsymbol{w}'$ is a mirrored Brownian motion by Lemma A.6.

Reverse SDE. The reverse SDE follows from applying the reverse-time SDE formula [28] to the truncation $\varphi[\tilde{\mathbf{x}}]$ and mapping back via φ^{-1} defined in Equation (22). The key step is that the score term $g(t)^2\Lambda^2\tilde{\mathrm{s}}(\tilde{\mathbf{x}}, t)$ emerges from the gradient of the log density, where Λ accounts for the mirror symmetry constraint and enables cache-efficient processing of only $\lfloor N/2 \rfloor + 1$ frequency tokens. □

Proposition A.7 (Cached Reverse SDE Error Propagation)**.** *Under caching, the cached reverse SDE Equation (9) introduces a drift error term:*

$$\Delta\tilde{\boldsymbol{b}}(\tilde{\boldsymbol{x}}, t) = g^2(t)\Lambda^2\tilde{\epsilon}(\tilde{\boldsymbol{x}}, t), \tag{25}$$

where $\tilde{\epsilon}(\tilde{\boldsymbol{x}}, t) = \tilde{\mathrm{s}}_\theta(\tilde{\boldsymbol{x}}, t) - \hat{\tilde{\mathrm{s}}}_\theta(\tilde{\boldsymbol{x}}, t)$ is the caching approximation error. The accumulated error over a single diffusion step satisfies:

$$\|\Delta\tilde{\boldsymbol{x}}^{(i)}\|_2 \leq |\mathrm{d}t| \cdot \left(\|\tilde{\boldsymbol{f}}(\tilde{\boldsymbol{x}}^{(i)}, t^{(i)})\|_2 + g^2(t^{(i)})\|\Lambda^2\|_2 \cdot L \cdot \|\epsilon_L^{(i)}\|_2\right), \tag{26}$$

where L is the Lipschitz constant of the score network and $\epsilon_L^{(i)}$ is the final-layer CRF approximation error from Equation (33).

Proof. The cached reverse SDE Equation (9) differs from the exact reverse SDE Equation (8) by the error term Equation (25). By Lemma A.5, $\|\tilde{\epsilon}(\tilde{\mathbf{x}}, t)\|_2 \leq L \cdot \|\epsilon_L^{(i)}\|_2$. The step error bound follows from integrating the SDE over an infinitesimal time step $|\mathrm{d}t|$. □

Corollary A.8 (Cache Efficiency in Frequency Domain)**.** *Under mirror symmetry, the cache stores at most $N_{cache} = \lfloor N/2 \rfloor + 1$ frequency tokens, achieving a cache size reduction factor of approximately 2. For spectral-localized signals satisfying Equation (39) with $\delta \ll 1$, the effective cache size can be further reduced to $N_{eff} = K + 1$ by caching only low-frequency tokens $\kappa \leq K$, yielding cache efficiency:*

$$\eta_{cache} = \frac{N_{eff}}{N} \approx \frac{K+1}{N} \ll 1. \tag{27}$$

A.4 Cached Score Matching in the Frequency Domain

This section establishes the equivalence between score matching in time and frequency domains, which enables training score networks in the frequency domain for efficient caching. We also analyze how cached score evaluation affects the matching objective.

Proposition 3.2. *(Score matching equivalence). Consider a score $\tilde{\mathbf{s}}_{\tilde{\theta}} : \mathbb{C}^{dx} \times [0, T] \to \mathbb{C}^{dx}$ defined in the frequency domain and satisfying the mirror symmetry $[\tilde{\mathbf{s}}_{\tilde{\theta}}]_\kappa = [\tilde{\mathbf{s}}^*_{\tilde{\theta}}]_{N-\kappa}$ for all $\kappa \in [N]$. Let us define an auxiliary score $\mathbf{s}'_{\tilde{\theta}} : \mathbb{R}^{dx} \times [0, T] \to \mathbb{R}^{dx}$ as $(\boldsymbol{x}, t) \mapsto \mathbf{s}'_{\tilde{\theta}}(\boldsymbol{x}, t) = U^* \tilde{\mathbf{s}}_{\tilde{\theta}}(U\boldsymbol{x}, t)$ in the time domain. The score matching loss in the frequency domain is equivalent to the score matching loss for the auxiliary score in the time domain:*

$$\mathcal{L}_{\mathrm{SM}}\left(\tilde{\mathbf{s}}_{\tilde{\theta}}, \Lambda^2 \tilde{\mathbf{s}}_{t|0}, \tilde{\boldsymbol{x}}, t\right) = \mathcal{L}_{\mathrm{SM}}\left(\mathbf{s}'_{\tilde{\theta}}, \mathbf{s}_{t|0}, \boldsymbol{x}, t\right) \tag{12}$$

where $\tilde{\mathbf{s}}_{t|0}(\tilde{\boldsymbol{x}}, t) = \nabla_{\tilde{\boldsymbol{x}}(t)} \log \tilde{p}_{t|0}(\tilde{\boldsymbol{x}}(t)|\tilde{\boldsymbol{x}}(0))$, $\mathbf{s}_{t|0}(\boldsymbol{x}, t) = \nabla_{\boldsymbol{x}(t)} \log p_{t|0}(\boldsymbol{x}(t)|\boldsymbol{x}(0))$, and Λ is the diagonal matrix in Proposition 3.1.

Proof. Let $\Lambda \in \mathbb{R}^{N \times N}$ be the diagonal matrix such that $[\Lambda]_{\kappa,\kappa} = \begin{cases} 1 & \text{if } \kappa = 0, \text{ or } N \text{ is even and } \kappa = N/2 \\ \frac{1}{\sqrt{2}} & \text{otherwise} \end{cases}$

Step 1: We first express the score of $\mathbf{x}$ with respect to the score of the truncation $\varphi[\tilde{\mathbf{x}}]$.

By definition of φ^{-1} in Equation (22), we have $\mathbf{x} = U^* \varphi^{-1}(\varphi[\tilde{\mathbf{x}}])$. Hence, we can write, using the change of variable formula:

$$p_{t|0}(\mathbf{x}(t)|\mathbf{x}(0)) = C \cdot \tilde{p}_{t|0}\left(\varphi[\tilde{\mathbf{x}}(t)]|\varphi[\tilde{\mathbf{x}}(0)]\right) \tag{28}$$

where C is a constant which does not depend on $\mathbf{x}$, since $\mathbf{x} \mapsto \varphi[U\mathbf{x}]$ is linear. Moreover, let us write $\mathbf{x} \mapsto \varphi[U\mathbf{x}]$ in matrix form, i.e. $\forall \mathbf{x} \in \mathbb{R}^{dx}$, $\varphi[U\mathbf{x}] = VU_{col}\mathbf{x} = Q\mathbf{x}$, where $V \in \mathbb{R}^{N \times 2N}$, $U_{col} = \begin{pmatrix} U_{re} \\ U_{im} \end{pmatrix}$ and Q is an invertible matrix in $\mathbb{R}^{N \times N}$. For the rest of the proof, we shall build on the below results:

Result 1. $QQ^T = \Lambda^2$. To see this, write $QQ^T = VU_{col}U_{col}^T V^T$. The matrix $U_{col}U_{col}^T$ is equal to $\begin{pmatrix} U_{re}^2 & 0_N \\ 0_N & U_{im}^2 \end{pmatrix}$ (cf. the proof to Lemma A.6), while the multiplication by V, on the left and on the right of $U_{col}U_{col}^T$, extracts the submatrix corresponding to the indices represented by the truncature φ. Hence, $VU_{col}U_{col}^T V^T = \Lambda^2$.

Result 2. For any $\mathbf{x} \in \mathbb{R}^{dx}$, we have $Q^T\mathbf{x} = U^* \varphi^{-1}[\Lambda^2 \mathbf{x}]$. To see this, notice that Result 1 implies that $Q^T\mathbf{x} = Q^{-1}\Lambda^2\mathbf{x} = U^*\varphi^{-1}[\Lambda^2\mathbf{x}]$ for all $\mathbf{x} \in \mathbb{R}^{dx}$.

Equipped with these results, we can now complete the rest of the proof. First, we have:

$$\nabla_{\mathbf{x}(t)} \log p_{t|0}(\mathbf{x}(t)|\mathbf{x}(0)) = \nabla_{\mathbf{x}(t)} \log \tilde{p}_{t|0}\left(\varphi[\tilde{\mathbf{x}}(t)]|\varphi[\tilde{\mathbf{x}}(0)]\right) \tag{29}$$

$$= Q^T \nabla_{\varphi[\tilde{\mathbf{x}}(t)]} \log \tilde{p}_{t|0}(\varphi[\tilde{\mathbf{x}}(t)]|\varphi[\tilde{\mathbf{x}}(0)]) \qquad \text{(Chain rule)} \tag{30}$$

Step 2: We then obtain:

$$\begin{aligned}
\mathcal{L}_{\mathrm{SM}}\left(\mathbf{s}'_{\tilde{\theta}}, \mathbf{s}_{t|0}, \mathbf{x}, t\right) &:= \|\mathbf{s}'_{\tilde{\theta}}(\mathbf{x}, t) - \nabla_{\mathbf{x}(t)} \log p_{t|0}(\mathbf{x}(t)|\mathbf{x}(0))\|^2 \\
&= \|U\mathbf{s}'_{\tilde{\theta}}(\mathbf{x}, t) - U\nabla_{\mathbf{x}(t)} \log p_{t|0}(\mathbf{x}(t)|\mathbf{x}(0))\|^2 && \text{(Parseval identity)} \\
&= \|\tilde{\mathbf{s}}_{\tilde{\theta}}(\tilde{\mathbf{x}}, t) - UQ^T\nabla_{\varphi[\tilde{\mathbf{x}}(t)]} \log \tilde{p}_{t|0}(\varphi[\tilde{\mathbf{x}}(t)]|\varphi[\tilde{\mathbf{x}}(0)])\|^2 && \text{(Equation (30))} \\
&= \|\tilde{\mathbf{s}}_{\tilde{\theta}}(\tilde{\mathbf{x}}, t) - UU^*\varphi^{-1}[\Lambda^2\nabla_{\varphi[\tilde{\mathbf{x}}(t)]} \log \tilde{p}_{t|0}(\varphi[\tilde{\mathbf{x}}(t)]|\varphi[\tilde{\mathbf{x}}(0)]]\|^2 && \text{(Result 2)} \\
&= \|\tilde{\mathbf{s}}_{\tilde{\theta}}(\tilde{\mathbf{x}}, t) - \Lambda^2\varphi^{-1}[\nabla_{\varphi[\tilde{\mathbf{x}}(t)]} \log \tilde{p}_{t|0}(\varphi[\tilde{\mathbf{x}}(t)]|\varphi[\tilde{\mathbf{x}}(0)]]\|^2 && \text{(Proposition A.4 \& Definition of } \varphi^{-1}) \\
&= \|\tilde{\mathbf{s}}_{\tilde{\theta}}(\tilde{\mathbf{x}}, t) - \Lambda^2\tilde{\mathbf{s}}_{t|0}(\tilde{\mathbf{x}}, t)\|^2 && \text{(Equation (23))} \\
&= \mathcal{L}_{\mathrm{SM}}\left(\tilde{\mathbf{s}}_{\tilde{\theta}}, \Lambda^2\tilde{\mathbf{s}}_{t|0}, \tilde{\mathbf{x}}, t\right).
\end{aligned}$$

□

Corollary A.9 (Cached Score Matching Bound)**.** *Under caching with approximation error $\epsilon_L^{(i)}$ bounded by Proposition A.10, the cached score matching objective satisfies:*

$$|\mathcal{L}_{\mathrm{SM}}(\hat{\tilde{\mathbf{s}}}_{\tilde{\theta}}, \Lambda^2\tilde{\mathbf{s}}_{t|0}, \tilde{\boldsymbol{x}}, t) - \mathcal{L}_{\mathrm{SM}}(\tilde{\mathbf{s}}_{\tilde{\theta}}, \Lambda^2\tilde{\mathbf{s}}_{t|0}, \tilde{\boldsymbol{x}}, t)| \leq O(L^2 \cdot \|\epsilon_L^{(i)}\|_2^2), \tag{31}$$

where $\hat{\tilde{\mathbf{s}}}_{\tilde{\theta}}$ denotes the cached score evaluation and L is the Lipschitz constant.

Proof. By Lemma A.5, the cached score error is bounded by $L \cdot \|\epsilon_L^{(i)}\|_2$. The score matching objective involves squared differences, yielding the quadratic bound in the approximation error. □

A.5 E^2-CRF Caching: Mathematical Analysis

In this section, we provide a rigorous mathematical analysis of the E^2-CRF caching mechanism and its behavior under caching. We analyze error accumulation, error bounds, and the relationship between cache efficiency and diffusion dynamics.

A.5.1 Cumulative Residual Features (CRF) in Cached Diffusion

Consider a transformer-based score network $\tilde{\mathbf{s}}_{\tilde{\theta}}$ operating on frequency-domain inputs $\tilde{\mathbf{x}}^{(i)} \in \mathbb{C}^{dx}$ at diffusion step i. Let $\mathbf{z}_\ell^{(i)}$ denote the CRF at layer ℓ and step i, defined as in Equation (13):

$$\mathbf{z}_\ell^{(i)} = \phi_\ell(\tilde{\mathbf{x}}^{(i)}) = h^{(0)} + \sum_{l=0}^{\ell-1} F^{(l)}(h^{(l)}, t^{(i)}), \tag{32}$$

where $h^{(0)}$ is the initial embedding, $F^{(l)}$ represents the residual block at layer l, and $t^{(i)}$ is the diffusion time at step i.

Under caching, we distinguish between *recomputed* CRF $\mathbf{z}_\ell^{(i)}$ (ground truth) and *cached* CRF $\hat{\mathbf{z}}_\ell^{(i)}$ (approximation). The approximation error at step i is defined as:

$$\epsilon_\ell^{(i)} = \mathbf{z}_\ell^{(i)} - \hat{\mathbf{z}}_\ell^{(i)}. \tag{33}$$

A.5.2 Event Intensity and Adaptive Recomputation

The event intensity $r^{(i)}$ measures the relative change in the final-layer CRF, as defined in Equation (14):

$$r^{(i)} = \frac{\|\mathbf{z}_L^{(i)} - \mathbf{z}_L^{(i-\Delta)}\|_2^2}{\|\mathbf{z}_L^{(i-\Delta)}\|_2^2 + \eta}, \tag{34}$$

where L is the number of layers, Δ is the step interval, and $\eta > 0$ is a small constant for numerical stability. This metric naturally captures the diffusion process's structure-to-detail progression: early steps (high $r^{(i)}$) correspond to structure formation, while later steps (low $r^{(i)}$) correspond to detail refinement.

The recompute set $S^{(i)}$ is determined adaptively based on Equation (15):

- Low-frequency tokens: $\{0, 1, \ldots, K\}$ are always recomputed (critical for signal structure),
- High-change tokens: $\{k > K : \delta_k^{(i)} > \tau_k\}$, where $\delta_k^{(i)} = \|\mathbf{z}_k^{(i)} - \mathbf{z}_k^{(i-\Delta)}\|_2$ and $\tau_k = \tau_0 \cdot (\epsilon + \|\tilde{\mathbf{x}}_k^{(i)}\|_2^2)^{-1}$ is an energy-weighted threshold,
- Random probes: A small fraction of high-frequency tokens are randomly selected for periodic recalibration.

A.5.3 Error Accumulation and Bounds

Let $\epsilon_\ell^{(i)}$ denote the approximation error at layer ℓ and step i. Under caching, tokens $k \notin S^{(i)}$ reuse cached features $\hat{\mathbf{z}}_\ell^{(i)} = \hat{\mathbf{z}}_\ell^{(i-1)}$, while tokens $k \in S^{(i)}$ are recomputed. The error evolution follows:

$$\epsilon_\ell^{(i)} = \begin{cases} 0 & \text{if } k \in S^{(i)} \text{ (recomputed)} \\ \epsilon_\ell^{(i-1)} + \Delta_\ell^{(i)} & \text{if } k \notin S^{(i)} \text{ (cached),} \end{cases} \tag{35}$$

where $\Delta_\ell^{(i)} = \mathbf{z}_\ell^{(i)} - \mathbf{z}_\ell^{(i-1)}$ is the true change in CRF between steps.

Under the assumption that the score network $\tilde{\mathbf{s}}_{\tilde{\theta}}$ is L-Lipschitz continuous with respect to its input, we can bound the error growth. Specifically, if $\|\nabla_{\tilde{\mathbf{x}}} \tilde{\mathbf{s}}_{\tilde{\theta}}(\tilde{\mathbf{x}}, t)\|_2 \leq L$ for all $\tilde{\mathbf{x}}$ and t, then:

Proposition A.10 (Error Bound Under Caching). *Let $\epsilon_\ell^{(i)}$ be the approximation error at layer ℓ and step i. If tokens are cached for at most R consecutive steps before recomputation, and the score network is L-Lipschitz continuous, then:*

$$\|\epsilon_\ell^{(i)}\|_2 \leq L \cdot R \cdot \max_{j \in [i-R, i]} \|\Delta \tilde{\mathbf{x}}^{(j)}\|_2, \tag{36}$$

where $\Delta \tilde{\mathbf{x}}^{(j)} = \tilde{\mathbf{x}}^{(j)} - \tilde{\mathbf{x}}^{(j-1)}$ is the change in frequency representation at step j.

Proof. The error accumulates only for cached tokens. Since recomputation occurs at least every R steps, the maximum error accumulation is bounded by the product of the Lipschitz constant L, the caching window R, and the maximum change in input $\Delta \tilde{\mathbf{x}}^{(j)}$ over the caching window. □

A.5.4 Error-Feedback Correction

To mitigate error accumulation, we apply error-feedback correction as described in Equations (17) and (18). Periodically (every R steps) or when event intensity exceeds a threshold, we perform a probe computation to estimate the approximation error:

$$\epsilon_k^{(i)} = \mathbf{z}_k^{(i)} - \hat{\mathbf{z}}_k^{(i)}, \tag{37}$$

and apply correction:

$$\hat{\mathbf{z}}_k^{(i)} \leftarrow \hat{\mathbf{z}}_k^{(i)} + \alpha^{(i)} \cdot \epsilon_k^{(i)}, \tag{38}$$

where $\alpha^{(i)} \in [0, 1]$ is an adaptive step size. This correction reduces the error from $\epsilon_k^{(i)}$ to $(1 - \alpha^{(i)})\epsilon_k^{(i)}$, effectively suppressing error accumulation.

A.5.5 Cache Efficiency and Spectral Localization

The cache efficiency depends on the spectral localization property. Under the assumption that most energy is concentrated in low frequencies $\kappa \leq K$, high-frequency tokens $\kappa > K$ change more slowly and are ideal candidates for caching. Formally, if the spectral density satisfies:

$$\sum_{\kappa=0}^{K} \|\tilde{\mathbf{x}}_\kappa\|_2^2 \geq (1 - \delta) \sum_{\kappa=0}^{\lfloor N/2 \rfloor} \|\tilde{\mathbf{x}}_\kappa\|_2^2, \tag{39}$$

for some small $\delta > 0$, then caching high-frequency tokens $\kappa > K$ yields high cache hit rates while maintaining accuracy, since these tokens contribute less to the overall signal energy and change more slowly during diffusion.

A.5.6 KV Cache and Attention Computation

For each transformer layer ℓ, we cache key-value pairs $\text{Cache}_\ell[k] = (K_\ell[k], V_\ell[k])$ for all frequency tokens k, as defined in Equation (16). At diffusion step i:

- If $k \in S^{(i)}$: Recompute $K_\ell[k] = W_K \mathbf{z}_k^{(i)}$ and $V_\ell[k] = W_V \mathbf{z}_k^{(i)}$, where W_K and W_V are projection matrices.
- If $k \notin S^{(i)}$: Reuse $\text{Cache}_\ell[k]$ from the previous step.

The attention computation for cached tokens reduces from $O(d^2)$ (full recomputation) to $O(d)$ (lookup), where d is the dimension. For a cache hit rate of ρ, the computational savings scale as $(1 - \rho) \cdot C_{\text{recompute}} + \rho \cdot C_{\text{cache}}$, where $C_{\text{recompute}}$ and C_{cache} are the costs of recomputation and caching, respectively.

A.5.7 Convergence Guarantees

Under mild assumptions, the cached diffusion process converges to the same distribution as the baseline (non-cached) process. Specifically, if the error-feedback correction is applied sufficiently frequently and the approximation error remains bounded (as guaranteed by Proposition A.10), then the discrepancy between cached and baseline sampling scales with the maximum approximation error:

$$\mathrm{TV}(\mathrm{Cached}(\tilde{\mathbf{x}}^{(T)}), \mathrm{Baseline}(\tilde{\mathbf{x}}^{(T)})) \leq O(\max_i \|\epsilon_L^{(i)}\|_2), \tag{40}$$

where TV denotes total variation distance and $\epsilon_L^{(i)}$ is the final-layer approximation error at step i. This bound ensures that cached sampling maintains distributional fidelity as long as the approximation error is controlled.

B Empirical details

Compute resources. All the models were trained and used for sampling on a MacBook Pro equipped with an Apple M3 Max CPU (16 cores: 12 performance and 4 efficiency cores).

B.1 Details on datasets

In this subsection, we give detailed information about the 5 datasets used throughout our experiments and the preprocessing steps for each of them.

ECG. We use two collections of heartbeat signals, from the MIT-BIH Arrhythmia Dataset and the PTB Diagnostic ECG Database [23]. No preprocessing was performed on this dataset.

NASDAQ-2019. This dataset [24] contains daily prices for tickers trading on NASDAQ, and contains prices for up to 1st of April 2020. *Preprocessing.* We considered one year of daily prices from 1st of January 2019 to 1st of January 2020. Each sample corresponds to one stock, and we remove the stocks which are not active in this whole time interval, or contain missing values.

NASA battery. The NASA battery dataset [25] consists of profiles for Li-on batteries, under charge and discharge. *Preprocessing.* For both the charge and discharge datasets, we bin the time values (bins of size 10 for Charge, 15 for Discharge) and compute the mean of each feature inside each bin.

US-Droughts. This dataset [26] consists of drought levels in different US counties, from 2000 to 2020. *Preprocessing.* We consider one year of history, from 1st of January 2011 to 1st of January 2012, and drop the columns with missing values.

B.2 Details on evaluation

Sliced Wasserstein distances. The sliced Wasserstein distance [29] is a metric which can handle high-dimensional distributions. It is motivated by the fact that the Wasserstein distance is easy to compute when comparing two one-dimensional distributions. The idea of the sliced Wasserstein distance is to map the high-dimensional distributions of interest to one-dimensional distributions, by considering random projections on vectors of the unit sphere. For two distributions μ_1 and μ_2, it can be written as:

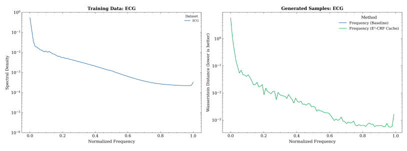
$$SW_p(\mu_1, \mu_2) \coloneqq \int_{\mathbb{S}^{d-1}} W_p(P_u \# \mu_1, P_u \# \mu_2) du \tag{41}$$

where $\mathbb{S}^{d-1}$ is the unit sphere in dimension d, $P_u(x) = u \cdot x$ denotes the projection of x on u, $P_u \# \mu$ is the push-forward of μ by P_u, and W_p is the Wasserstein distance of order p. To estimate this quantity in practice, we sample $n = 10,000$ random vectors $\{u_i | i \in [n]\}$ which follow a uniform distribution in $\mathbb{S}^{d-1}$ and consider $p = 2$. Hence, we can approximate SW_p by the Monte-carlo estimator:

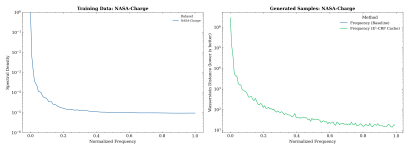
$$\hat{SW}_p(\mu_1, \mu_2) = \frac{1}{n} \sum_{i=1}^{n} W_p(P_{u_i} \# \mu_1, P_{u_i} \# \mu_2) \tag{42}$$

Marginal Wasserstein distances. In addition to the sliced Wasserstein distance, we also consider the marginal Wasserstein distance. For any $j \in \{1, ..., d\}$, the j-th marginal Wasserstein distance is defined as:

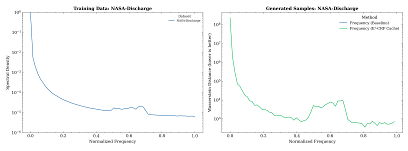
$$MW_p^{(j)}(\mu_1, \mu_2) = W_p(P_{e_j} \# \mu_1, P_{e_j} \# \mu_2) \tag{43}$$



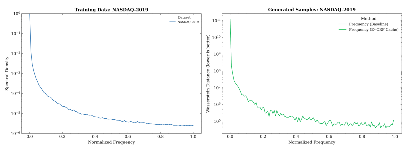
(a) ECG



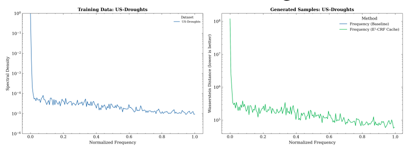
(b) NASA-Charge



(c) NASA-Discharge



(d) NASDAQ-2019



(e) US-Droughts

Figure 4: Spectral density comparison between baseline frequency domain diffusion (Frequency (Baseline)) and E^2-CRF cached diffusion (Frequency (E^2-CRF Cache)) across all datasets. Left plots show training data spectral density, right plots compare generated samples. E^2-CRF maintains spectral fidelity across all frequencies while achieving 1.7× speedup.

where e_j is the j-th vector of the standard basis of $\mathbb{R}^d$. Throughout our experiments in Section 4, we compute the Wasserstein distances with respect to $\mathcal{D}_{\text{train}}$ and $\tilde{\mathcal{D}}_{\text{train}}$.

B.3 Additional plots

Spectral Density Comparison. In Figure 4, we show detailed spectral density comparisons between baseline frequency domain diffusion and E^2-CRF cached diffusion across all datasets. The left plots in each subfigure show the training data spectral density (ground truth), while the right plots compare the spectral densities of generated samples from both methods. These visualizations demonstrate that E^2-CRF maintains spectral fidelity across all frequencies while achieving 1.7× speedup. The close alignment between baseline and cached methods confirms that the caching strategy does not degrade generation quality, as discussed in Section 4.

C Sample visualization

In Figures 5 to 9, we visualize a comparison of generated samples from baseline frequency domain diffusion (left columns) and E^2-CRF cached diffusion (right columns) across multiple datasets. The visualizations demonstrate that E^2-CRF maintains visual fidelity while achieving 1.7× speedup, with samples from both methods closely resembling training data. The y axis corresponds to the different features, while the x axis corresponds to the different time steps.

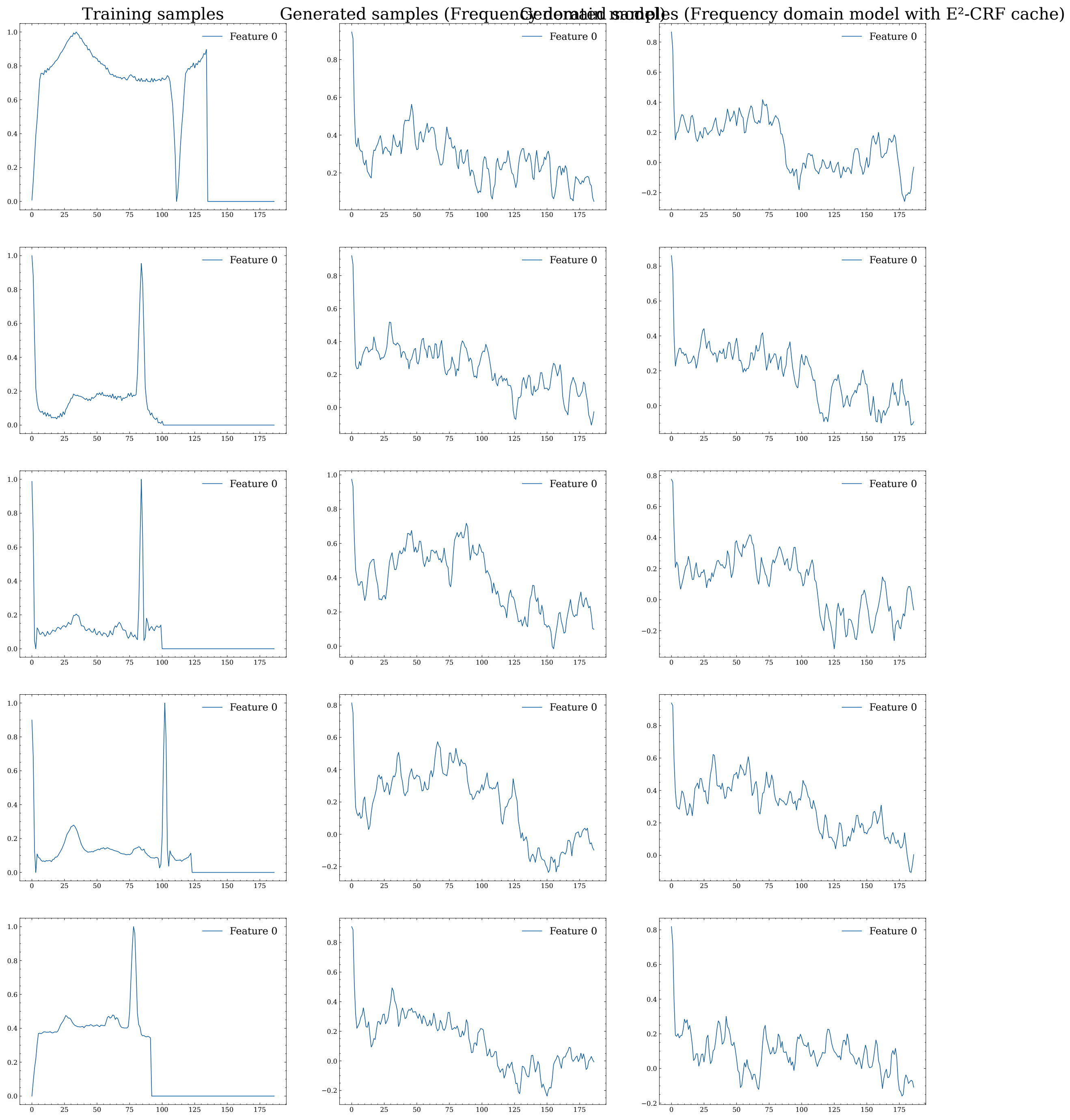


Figure 5: Sample comparison for the ECG dataset: baseline frequency diffusion (left) vs. E^2-CRF cached diffusion (right).

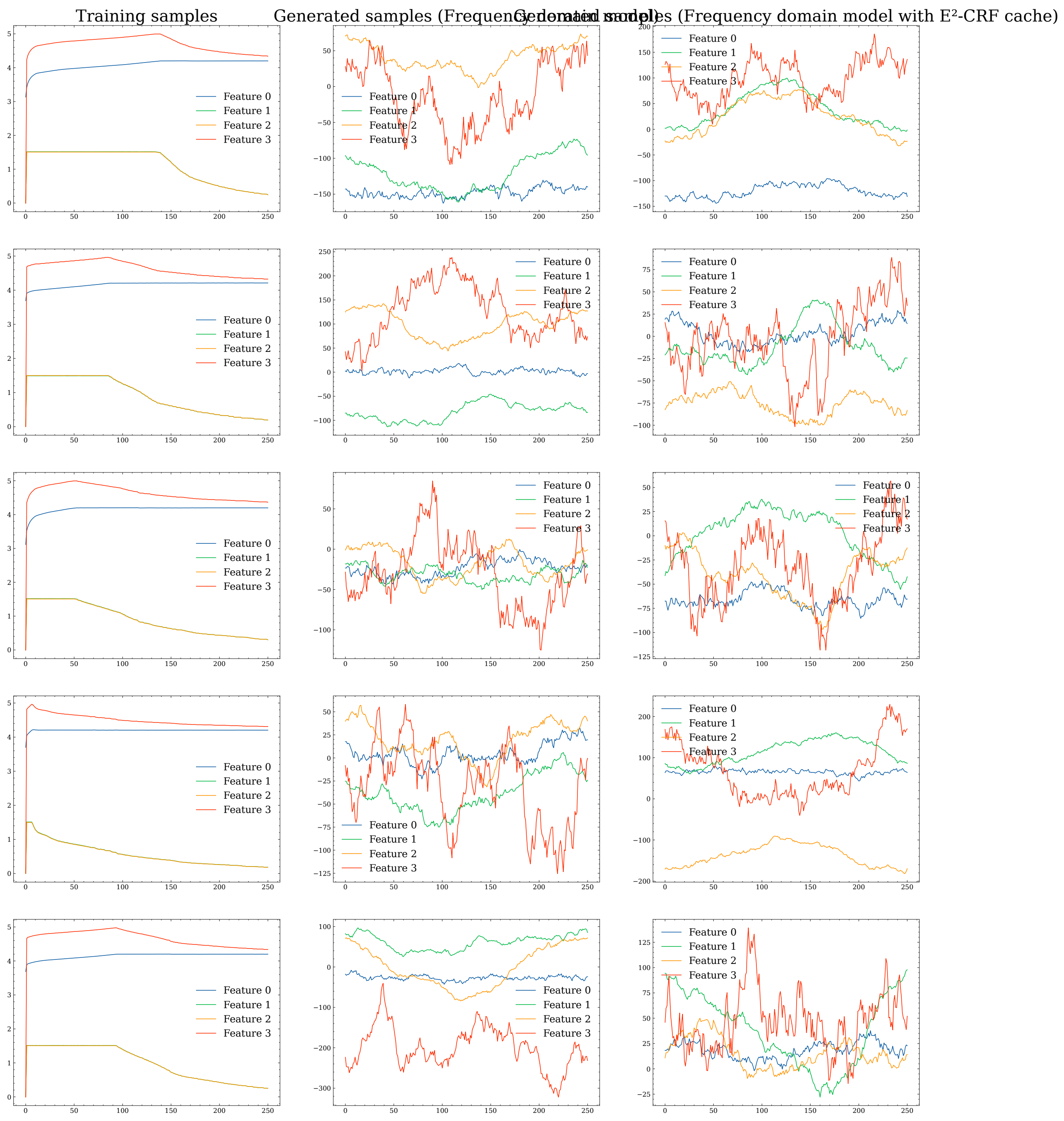


Figure 6: Sample comparison for the NASA-Charge dataset: baseline frequency diffusion (left) vs. E^2-CRF cached diffusion (right).

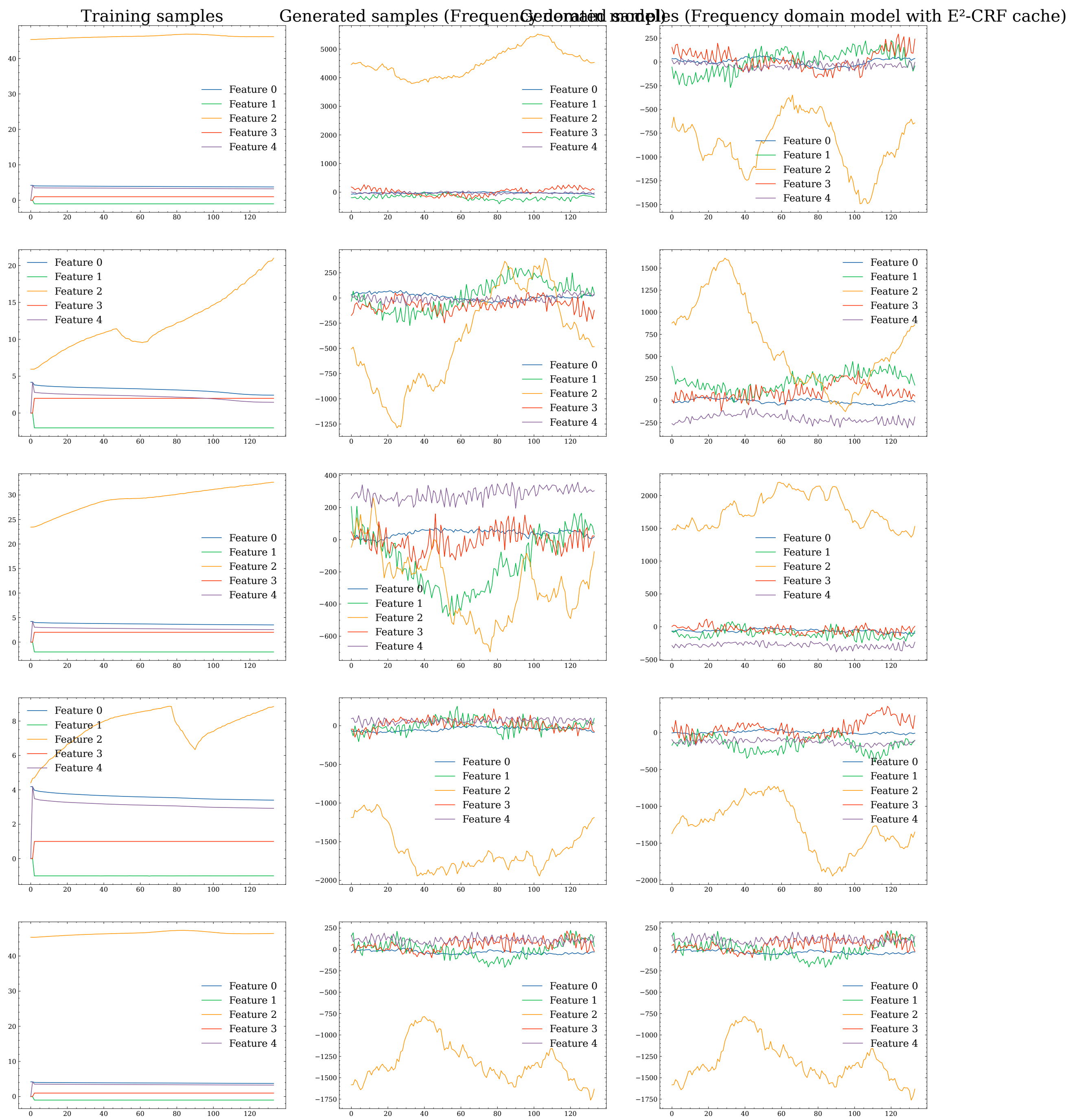


Figure 7: Sample comparison for the NASA-Discharge dataset: baseline frequency diffusion (left) vs. E^2-CRF cached diffusion (right).

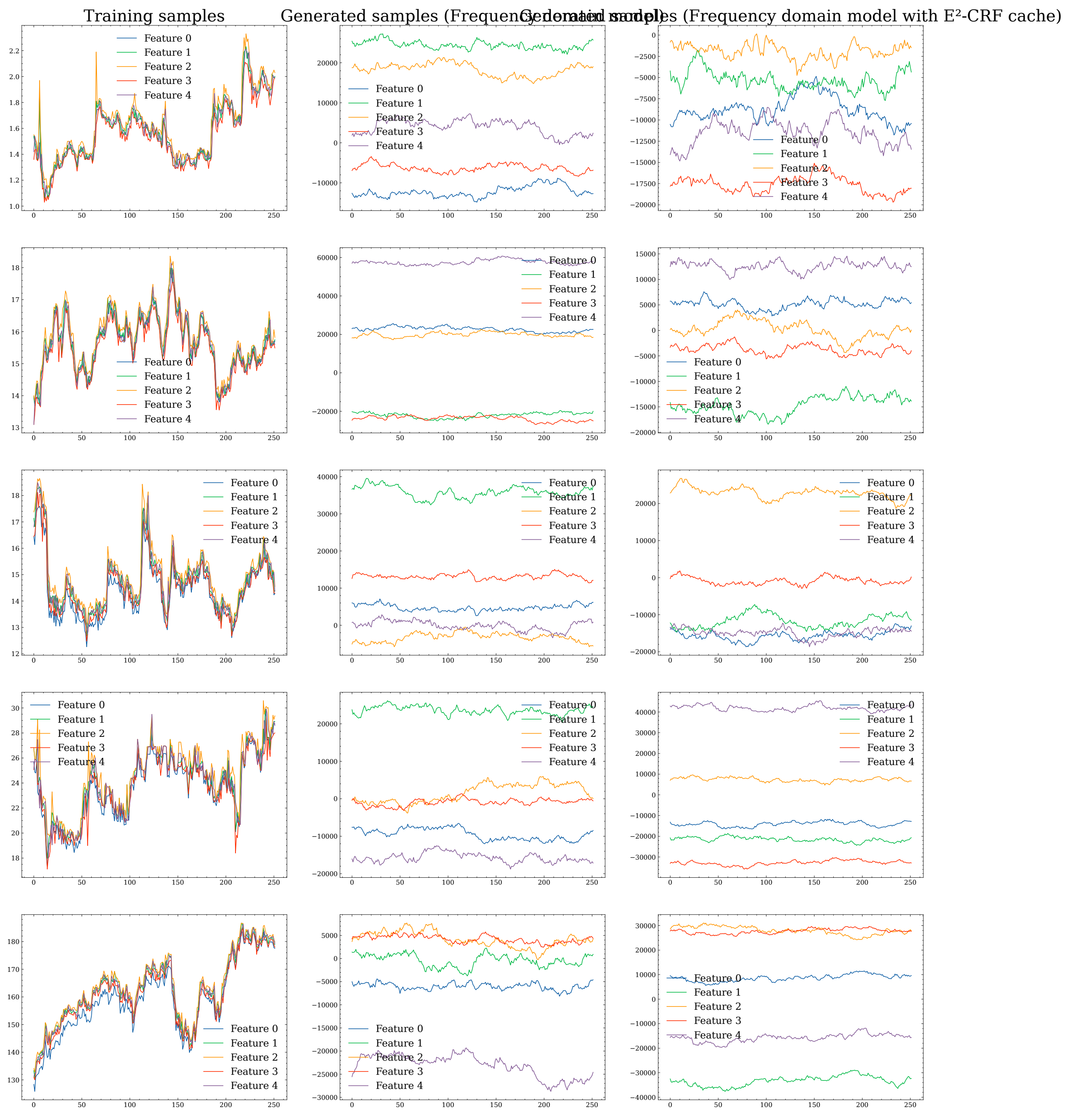


Figure 8: Sample comparison for the NASDAQ-2019 dataset: baseline frequency diffusion (left) vs. E^2-CRF cached diffusion (right).

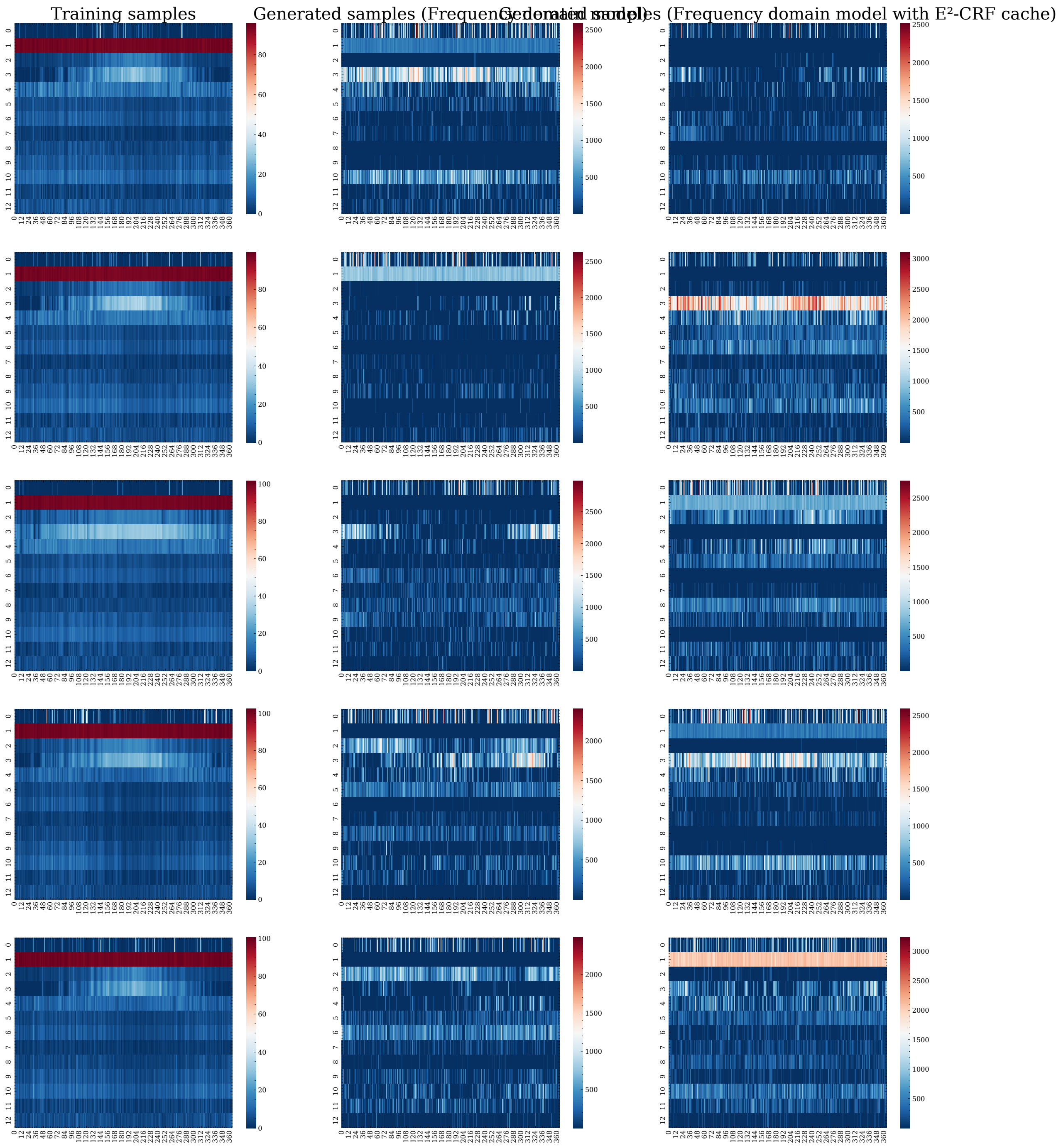


Figure 9: Sample comparison for the US-Droughts dataset, represented as heatmaps: baseline frequency diffusion (left) vs. E^2-CRF cached diffusion (right).