Not All Frequencies Are Equal: Energy-Adaptive Diffusion for Time Series Forecasting

Anonymous Authors[1]

Abstract

Diffusion models have achieved remarkable success in generative modeling, yet their application to time series forecasting remains suboptimal. Existing approaches apply uniform Gaussian noise across all time steps, assuming all frequency components should be corrupted at the same rate. However, energy distribution across frequencies in time series is highly non-uniform: when uniform noise is added, high-frequency components are disproportionately overwhelmed while low-frequency trends remain inadequately diffused. We propose EADIFF, an energy-adaptive diffusion framework operating in the wavelet domain to address this frequency-energy imbalance. Our key insight is that high-energy components require stronger perturbation while low-energy details need gentler corruption to preserve informative structures. We introduce a learnable modulation mechanism that automatically adjusts noise levels for each frequency band on a per-instance basis. Built upon this adaptive scheduler, we design a conditional diffusion framework where low-frequency trends serve as generation conditions, and noise-level-aware loss weighting naturally emphasizes different frequency components according to their signal characteristics. This cohesive design enables the model to respect the intrinsic multi-scale structure throughout both forward and reverse processes. Extensive experiments demonstrate that EADIFF consistently outperforms existing diffusion-based and state-of-the-art deterministic methods.

1. Introduction

Time series forecasting(Hamilton, 2020; Wu et al., 2021; Zeng et al., 2023) is a fundamental task with broad applications spanning energy management, financial markets, traffic planning, and climate science. Accurate predictions of future values based on historical observations enable proactive decision-making and resource optimization across these domains. While traditional statistical methods and recent deep learning approaches have achieved considerable success, capturing the complex multi-scale temporal patterns inherent in real-world time series remains a significant challenge. Diffusion models(Shi et al., 2025a; Song et al., 2021; Tashiro et al., 2021a) have recently emerged as a powerful paradigm for generative modeling, demonstrating remarkable success in image synthesis, audio generation, and beyond. Their ability to model complex data distributions through a gradual denoising process has inspired a growing body of work applying diffusion models to time series tasks, including forecasting, imputation, and generation. However, these methods predominantly operate in the raw temporal domain, applying uniform Gaussian noise across all time steps throughout the diffusion process.

While seemingly natural, this uniform temporal noise strategy has an often-overlooked implication for signals with rich frequency content. Real-world time series are inherently compositions of multiple frequency components(Oppenheim et al., 1999): low-frequency trends capturing slow-varying patterns and seasonality, and high-frequency details encoding rapid fluctuations and fine-grained variations. Crucially, the energy carried by these components is highly non-uniform—low-frequency trends typically dominate the signal energy, while high-frequency details contribute comparatively less. When uniform noise is added in the temporal domain, it distributes evenly across the frequency spectrum (i.e., white noise). This creates a fundamental imbalance: low-energy high-frequency components are quickly overwhelmed and lose their informative structure early in the diffusion process, while high-energy low-frequency trends remain inadequately noised even at large diffusion steps. The consequence is twofold: the model struggles to learn fine-grained details that are prematurely destroyed, and it inefficiently spends capacity on trends that dominate the signal throughout most of the noising trajectory.

This analysis motivates a fundamental question: should a diffusion model treat all frequency components equally? We argue it should not. Instead, the noise schedule should adapt to the energy characteristics of different frequency bands, applying stronger perturbation to high-energy components and gentler corruption to low-energy details. This

frequency-aware approach ensures that each component reaches an appropriate signal-to-noise ratio at each diffusion step, enabling the model to learn and reconstruct multi-scale patterns more effectively.

To achieve this, we propose **EADiff (Energy-Adaptive Diffusion)**, a diffusion framework that operates in the wavelet domain, where the input time series is decomposed into distinct frequency bands. This decomposition exposes the energy distribution across scales, enabling direct manipulation of the noise schedule for each band. We introduce a learnable energy-based modulation mechanism that computes a per-instance, per-band noise multiplier based on the relative energy of wavelet coefficients. Specifically, we estimate the energy of each frequency band via its temporal standard deviation, apply a bounded nonlinear transformation, and modulate the base noise schedule through a learnable scaling factor. This design allows the model to automatically discover the optimal energy-noise relationship for the target dataset during training.

Building upon this adaptive noise scheduler, we design a conditional diffusion framework tailored for forecasting. The low-frequency approximation coefficients, representing the overall trend, serve as a natural condition to guide the generation of higher-frequency detail coefficients. During training, we apply differentiated noise levels to the historical (observed) and future (to-be-predicted) portions of the signal, with light augmentation noise on history and full diffusion noise on the future. The loss function inherits the noise-level-aware weighting from the EDM framework, which, combined with our instance-adaptive, naturally assigns different importance to different frequency bands based on their signal characteristics. At inference time, we employ an SDEdit-style (Meng et al., 2022) conditional generation with a replacement mechanism that preserves the observed history while generating the future through iterative denoising. Our main contributions are summarized as follows:

- We propose a novel noise scheduling mechanism that adapts to the energy distribution of wavelet frequency bands on a per-instance basis. A learnable modulation factor automatically adjusts noise levels, enabling frequency-aware diffusion that respects the multi-scale structure of time series.
- We design a diffusion framework operating in the wavelet domain, where low-frequency trends serve as generation conditions and differentiated noise strategies are applied to observed and predicted regions. This cohesive design ensures consistency between the forward and reverse diffusion processes.
- We conduct extensive experiments on seven benchmarks, demonstrating competitive performance against both time-series and diffusion-based baselines. Comprehensive analysis provides practical guidance for applying frequency-domain diffusion to time series.

2. Preliminary

Let $\mathbf{X} \in \mathbb{R}^{T\times F}$ denote a multivariate time series with T time steps and F variates (features). We use $\mathbf{X}_{a:b} \in \mathbb{R}^{(b-a)\times F}$ to denote the subsequence from time step a to $b-1$ (exclusive of b). For the wavelet-transformed representation, we denote the concatenated wavelet coefficients as $\mathbf{W} \in \mathbb{R}^{F\times L}$, where L is the total number of coefficients across all frequency bands. For a J-level wavelet decomposition, the coefficients are organized as $\mathbf{W} = [\mathbf{W}^{(0)}, \mathbf{W}^{(1)}, \ldots, \mathbf{W}^{(J)}]$, where $\mathbf{W}^{(0)} \in \mathbb{R}^{F\times L_0}$ represents the low-frequency approximation coefficients, and $\mathbf{W}^{(\ell)} \in \mathbb{R}^{F\times L_\ell}$ for $\ell = 1, \ldots, J$ represents the detail coefficients at scale ℓ. The total coefficient length is $L = \sum_{\ell=0}^{J} L_\ell$. We use $k \in [0, 1]$ to denote the normalized diffusion step, where $k = 0$ corresponds to the clean signal and $k = 1$ corresponds to maximum noise. We denote the noise level at diffusion step k as $\sigma(k)$, with $\sigma_{\min}$ and $\sigma_{\max}$ representing the minimum and maximum noise levels.

Diffusion-Based Forecasting. Given historical observations $\mathbf{X}_{\text{his}} = \mathbf{X}_{1:T-H} \in \mathbb{R}^{(T-H)\times F}$, the goal is to predict future values $\mathbf{X}_{\text{pred}} = \mathbf{X}_{T-H+1:T} \in \mathbb{R}^{H\times F}$ over a horizon of H steps. In the diffusion framework (Ho et al., 2020), this is formulated as conditional generation. The forward process gradually corrupts the target signal by adding noise:

$$\mathbf{x}_k = \mathbf{x}_0 + \sigma(k) \cdot \boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \tag{1}$$

where $\mathbf{x}_0$ is the clean signal. The reverse process learns to denoise $\mathbf{x}_k$ back to $\mathbf{x}_0$ through a neural network $f_\theta(\mathbf{x}_k, \sigma(k))$ that predicts the clean signal given the noisy input and noise level. The model learns the conditional distribution $p(\mathbf{X}_{\text{pred}}|\mathbf{X}_{\text{his}})$ by denoising the future portion while keeping the historical observations as context.

Wavelet Transform. The discrete wavelet transform (DWT) (Mallat, 1989) decomposes a signal into multiple frequency bands through recursive filtering. For a signal $\mathbf{x} \in \mathbb{R}^T$, a J-level DWT:

$$\text{DWT}(\mathbf{x}) = [\mathbf{w}^{(0)}, \mathbf{w}^{(1)}, \ldots, \mathbf{w}^{(J)}] \tag{2}$$

where $\mathbf{w}^{(0)}$ contains the low-frequency approximation coefficients capturing the overall trend, and $\mathbf{w}^{(\ell)}$ for $\ell = 1, \ldots, J$ contains the detail coefficients at scale ℓ capturing oscillations at the corresponding frequency band. The original signal can be perfectly reconstructed via the inverse DWT (IDWT):

$$\mathbf{x} = \text{IDWT}(\mathbf{w}^{(0)}, \mathbf{w}^{(1)}, \ldots, \mathbf{w}^{(J)}) \tag{3}$$

The wavelet transform provides a natural multi-resolution representation where different frequency bands are explicitly

separated, enabling targeted manipulation of each band's noise schedule in our proposed framework.

3. Method

We propose EADIFF, an energy-adaptive diffusion framework for time series forecasting that operates in the wavelet domain. Our approach comprises three key components: (1) a wavelet-domain representation that decomposes the input into distinct frequency bands, exposing the underlying energy distribution; (2) an energy-adaptive forward diffusion process that modulates noise levels according to per-instance frequency band characteristics; and (3) a conditional reverse diffusion process that generates future predictions guided by low-frequency trend information. The overall framework illusion as Figure 1.

3.1. Energy-Adaptive Diffusion Process

3.1.1. Frequency-Energy Imbalance in Wavelet Domain

Standard diffusion models apply uniform Gaussian noise across all dimensions:

$$\mathbf{x}_k = \mathbf{x}_0 + \sigma(k) \cdot \boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \tag{4}$$

where $k \in [0, 1]$ denotes the normalized diffusion step and $\sigma(k)$ is the noise level. While effective for many domains, this formulation overlooks the non-uniform energy distribution inherent in time series signals.

We employ the discrete wavelet transform (DWT) to decompose the input $\mathbf{X} \in \mathbb{R}^{T \times F}$ into frequency bands:

$$\text{DWT}(\mathbf{X}) = \{\mathbf{W}^{(0)}, \mathbf{W}^{(1)}, \ldots, \mathbf{W}^{(J)}\} \tag{5}$$

where $\mathbf{W}^{(0)} \in \mathbb{R}^{F \times L_0}$ contains approximation coefficients capturing low-frequency trends, and $\mathbf{W}^{(\ell)} \in \mathbb{R}^{F \times L_\ell}$ for $\ell = 1, \ldots, J$ contains detail coefficients at progressively finer scales.

Operating in this domain reveals a critical observation: the energy carried by different frequency bands is highly non-uniform. Consider the signal-to-noise ratio (SNR) at diffusion step k for frequency band ℓ:

$$\text{SNR}^{(\ell)}(k) = \frac{\mathbb{E}[\|\mathbf{W}^{(\ell)}\|^2]}{\sigma(k)^2} = \frac{\mathcal{E}^{(\ell)}}{\sigma(k)^2} \tag{6}$$

where $\mathcal{E}^{(\ell)}$ denotes the energy of band ℓ. When uniform noise $\sigma(k)$ is applied across all bands, low-energy components (typically high-frequency details) reach critically low SNR early in the diffusion process, destroying informative structure prematurely. Conversely, high-energy components (low-frequency trends) remain inadequately diffused, leading to inefficient learning. This frequency-energy imbalance motivates our adaptive noise scheduling strategy.

3.1.2. Learnable Energy-Adaptive Noise Scheduling

To address this imbalance, we propose a learnable noise scheduling mechanism that adjusts noise levels based on per-instance energy characteristics. The key idea is that frequency bands with higher relative energy should receive proportionally stronger noise to achieve balanced corruption rates across all components.

For each feature f and frequency band ℓ, we estimate band energy via the temporal standard deviation of wavelet coefficients within that band, applying a logarithmic transformation to compress the dynamic range:

$$E_{f,\ell} = \log\left(\text{Std}_t(\mathbf{W}^{(\ell)}_{f,:}) + \epsilon\right) \tag{7}$$

where $\mathbf{W}^{(\ell)}_{f,:} \in \mathbb{R}^{L_\ell}$ denotes the coefficients of band ℓ for feature f. To capture relative energy distribution across bands rather than absolute magnitudes, we center the log-energies by subtracting their mean:

$$\tilde{E}_{f,\ell} = E_{f,\ell} - \frac{1}{J+1}\sum_{j=0}^{J} E_{f,j} \tag{8}$$

The centered energy is broadcast to all coefficient positions within each band, yielding an element-wise energy map $\tilde{E}_{f,i}$. This map is then transformed into a bounded modulation field via the hyperbolic tangent and a learnable parameter γ:

$$f_{f,i} = \exp\left(\gamma \cdot \tanh(\tilde{E}_{f,i}/C)\right) \tag{9}$$

where C is a temperature parameter. The parameter γ is learned end-to-end, allowing the model to discover the optimal energy-noise relationship for each dataset. When $\gamma > 0$, high-energy bands receive amplified noise while low-energy bands receive attenuated noise.

The final noise level combines an EDM-style base schedule (Karras et al., 2022) with the energy-adaptive modulation:

$$\sigma_{\text{base}}(k) = \left(\sigma_{\min}^{1/\rho} + k \cdot (\sigma_{\max}^{1/\rho} - \sigma_{\min}^{1/\rho})\right)^{\rho}, \tag{10}$$

$$\sigma_{f,i}(k, \mathbf{W}) = \sigma_{\text{base}}(k) \cdot f_{f,i}(\mathbf{W}) \tag{11}$$

The energy-adaptive forward diffusion in the wavelet domain becomes:

$$\mathbf{W}_k = \mathbf{W}_0 + \boldsymbol{\sigma}(k, \mathbf{W}) \odot \boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \tag{12}$$

where $\boldsymbol{\sigma}(k, \mathbf{W}) \in \mathbb{R}^{F \times L}$ contains per-instance, per-position noise levels that ensure all frequency bands reach comparable SNR levels at each diffusion step.

3.2. Wavelet-Domain Diffusion Model

3.2.1. Forecasting as Conditional Generation

We formulate forecasting as generating future wavelet coefficients conditioned on observed history. A natural alternative

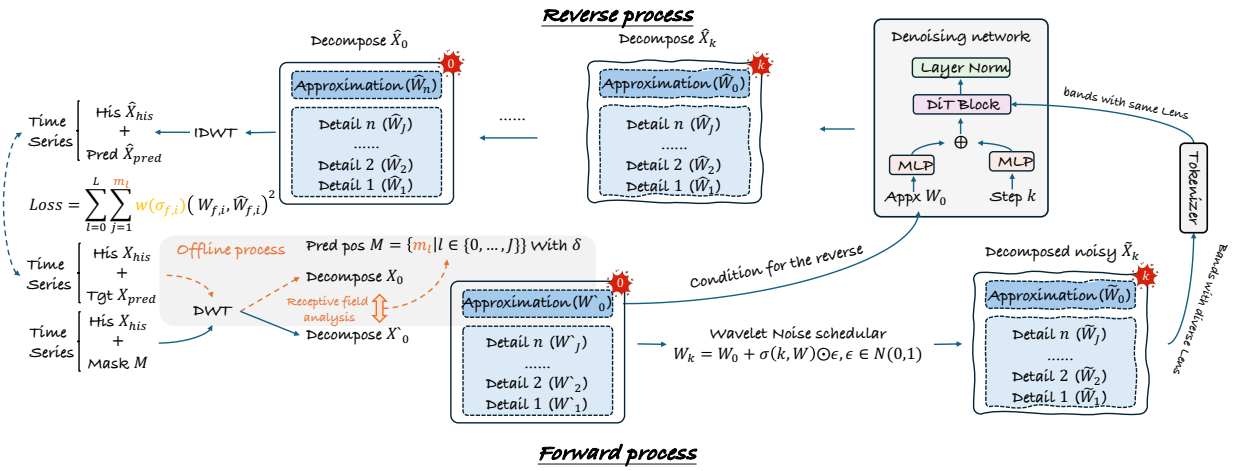

Figure 1. A high-level overview of EADiff.

would be to perform diffusion in the time domain and only use wavelets for the noise schedule. However, operating directly in the wavelet domain offers two advantages: (1) it provides richer supervision signals by enabling the model to learn frequency-specific patterns, and (2) it naturally integrates with our energy-adaptive noise schedule without requiring domain conversion at each diffusion step.

This design introduces a challenge: due to the non-local nature of wavelet filters, the temporal boundary between history and future at time step $T-H$ does not correspond to a sharp boundary in the coefficient domain. We address this by computing a binary mask $\mathbf{M}$ via receptive field analysis that identifies which coefficients are influenced by future time steps. For each frequency band, we determine the first coefficient position affected by any future value (see Appendix B.1 for details).

The diffusion input is constructed as $\tilde{\mathbf{X}} = [\mathbf{X}_{\text{his}}, \hat{\mathbf{X}}_{\text{guess}}]$, concatenating the observed history with an initial guess obtained by repeating the last observed value. This provides a reasonable initialization that the diffusion process iteratively refines. During training, we apply differentiated noise levels: history coefficients receive light augmentation noise ($k \in [0, k_{\text{aug}}]$ with small k_{aug}) to improve robustness while preserving informative content, whereas future coefficients undergo full diffusion ($k \in [0, 1]$) as the prediction target.

The approximation coefficients $\mathbf{W}^{(0)}$, which capture the overall trend, serve as a condition for generation. We project these coefficients through an MLP and inject them into the denoising network via adaptive layer normalization, providing global trend guidance throughout the reverse process.

3.2.2. Denoising Network Architecture

To handle the varying characteristics of different frequency bands, we employ band-specific tokenization with adaptive strides. The low-frequency approximation band uses dense tokenization (stride 1) to preserve trend precision, while detail bands use overlapping tokenization (stride $P/2$ for patch size P) to maintain continuity across patch boundaries (see Appendix B for details).

The tokenized coefficients are processed by a stack of Diffusion Transformer blocks (Peebles & Xie, 2023). Each block employs adaptive layer normalization to inject conditioning information, where the conditioning signal is formed by combining the time embedding (derived from $\log \sigma$) with the projected Approximation coefficient $\mathbf{W}^{(0)}$. This allows the network to modulate its behavior based on both the current noise level and the global trend of the signal.

Following Karras et al. (2022), we apply EDM-style input/output preconditioning with a skip connection to stabilize training across noise levels:

$$\hat{\mathbf{W}}_0 = c_{\text{skip}}(\sigma) \cdot \mathbf{W}_k + c_{\text{out}}(\sigma) \cdot F_\theta(c_{\text{in}}(\sigma) \cdot \mathbf{W}_k, \sigma) \quad (13)$$

where $c_{\text{in}}(\sigma) = 1/\sqrt{\sigma^2+1}$, $c_{\text{out}}(\sigma) = \sigma \cdot c_{\text{in}}(\sigma)$, and $c_{\text{skip}}(\sigma) = 1/(\sigma^2+1)$ ensure the network output magnitude remains stable regardless of the input noise level.

3.3. Training and Inference

3.3.1. Training Objective

We train to predict clean wavelet coefficients from noisy inputs. The loss function incorporates EDM-style(Karras et al., 2022) weighting that assigns higher importance to low-noise predictions where accuracy is most critical:

$$\mathcal{L} = \mathbb{E}_{k, \mathbf{W}_0, \epsilon}\left[\frac{1}{|\mathcal{M}|}\sum_{(f,i)\in\mathcal{M}} w(\sigma_{f,i}) \cdot (\hat{W}_{f,i} - W_{f,i})^2\right] \quad (14)$$

where $\mathcal{M}$ denotes the set of (feature, position) pairs corresponding to the future segment, and $w(\sigma) = (\sigma^2+1)/\sigma^2$

Algorithm 1 EADIFF Training

Require: Training data $\{\mathbf{X}^{(i)}\}$, model F_θ, learnable modulation γ
repeat
  Sample batch $\mathbf{X}$; construct $\tilde{\mathbf{X}} = [\mathbf{X}_{\text{his}}, \hat{\mathbf{X}}_{\text{guess}}]$
  $\mathbf{W}_0 \leftarrow \text{DWT}(\tilde{\mathbf{X}})$; $\mathbf{W}_{\text{target}} \leftarrow \text{DWT}(\mathbf{X})$
  Compute energy modulation factors $\{f_{f,i}\}$ via Eqs. (7)–(9)
  Sample $k_{\text{his}} \sim \mathcal{U}[0, k_{\text{aug}}]$, $k_{\text{fut}} \sim \mathcal{U}[0, 1]$
  Compute adaptive noise: $\sigma_{f,i} \leftarrow \sigma_{\text{base}}(k) \cdot f_{f,i}$
  $\mathbf{W}_k \leftarrow \mathbf{W}_0 + \boldsymbol{\sigma} \odot \boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
  $\hat{\mathbf{W}}_0 \leftarrow F_\theta(\mathbf{W}_k, \boldsymbol{\sigma}, \mathbf{W}_0^{(0)})$
  $\mathcal{L} \leftarrow \sum_{(f,i)\in\mathcal{M}} w(\sigma_{f,i})(\hat{W}_{f,i} - W_{\text{target},f,i})^2$
  Update θ, γ via gradient descent
until converged

Algorithm 2 EADIFF Inference

Require: History $\mathbf{X}_{\text{his}}$, model F_θ, starting step k_0, steps N
Construct $\tilde{\mathbf{X}} = [\mathbf{X}_{\text{his}}, \hat{\mathbf{X}}_{\text{guess}}]$
$\mathbf{W}_{\text{init}} \leftarrow \text{DWT}(\tilde{\mathbf{X}})$
Compute mask $\mathbf{m}$ via receptive field analysis
Compute energy-adaptive schedule $\{\boldsymbol{\sigma}_{k_i}\}_{i=0}^{N}$ from $\mathbf{W}_{\text{init}}$
$\mathbf{W}_{k_0} \leftarrow (1-\mathbf{m}) \odot \mathbf{W}_{\text{init}} + \mathbf{m} \odot (\mathbf{W}_{\text{init}} + \boldsymbol{\sigma}_{k_0} \odot \boldsymbol{\epsilon})$
for $i = 0$ to $N-1$ **do**
  $\hat{\mathbf{W}}_0 \leftarrow F_\theta(\mathbf{W}_{k_i}, \boldsymbol{\sigma}_{k_i}, \mathbf{W}_{\text{init}}^{(0)})$
  $\mathbf{W}_{k_{i+1}}^{\text{raw}} \leftarrow \text{HeunStep}(\mathbf{W}_{k_i}, \hat{\mathbf{W}}_0, k_i, k_{i+1})$
  $\mathbf{W}_{k_{i+1}} \leftarrow (1-\mathbf{m}) \odot \mathbf{W}_{\text{init}} + \mathbf{m} \odot \mathbf{W}_{k_{i+1}}^{\text{raw}}$
end for
$\hat{\mathbf{X}} \leftarrow \text{IDWT}(\mathbf{W}_{k_N})$
return $\hat{\mathbf{X}}_{\text{pred}} = \hat{\mathbf{X}}_{T-H+1:T}$

is the weighting function.

3.3.2. INFERENCE PROCEDURE

In the inference, we employ an SDEdit-style (Meng et al., 2022) conditional generation strategy. Rather than starting from pure noise, we initialize from a partially noised version of the input:

$$\mathbf{W}_{k_0} = (1-\mathbf{m}) \odot \mathbf{W}_{\text{init}} + \mathbf{m} \odot (\mathbf{W}_{\text{init}} + \boldsymbol{\sigma}_{k_0} \odot \boldsymbol{\epsilon}) \quad (15)$$

$\mathbf{W}_{\text{init}}$ contains the wavelet coefficients of the concatenated history and guess, and $k_0 < 1$ is the starting diffusion step.

We solve the reverse diffusion using the Heun sampler(Karras et al., 2022) with a replacement mechanism that enforces hard constraints on the history:

$$\mathbf{W}_{k_{i+1}} = (1-\mathbf{m}) \odot \mathbf{W}_{\text{init}} + \mathbf{m} \odot \text{HeunStep}(\mathbf{W}_{k_i}, k_i, k_{i+1}) \quad (16)$$

This ensures the observed history remains unchanged while the future is iteratively refined through denoising. The complete procedures are summarized in Algorithm 4 and 5.

4. Experiments

We conduct extensive experiments to demonstrate EADiff's performance and provide deep insights into how its adaptive mechanism autonomously resolves the frequency-energy imbalance across diverse temporal regimes.

4.1. Experimental Setup

Datasets We evaluate EADiff on a diverse set of real-world benchmarks, including Exchange(Lai et al., 2018), Weather(Zhou et al., 2021), Beijing Air(Liang et al., 2015), and the ETT family(Zhou et al., 2021) (ETTh1, ETTh2). These datasets represent a wide spectrum of temporal characteristics, from strong periodicity to high non-stationarity.

Baselines We compare EADiff against 7 baselines categorized into two groups. **Time-series Forecasting Models**: TimeMixer(Wang et al., 2024a), iTransformer(Liu et al., 2024b), PatchTST(Nie et al., 2023), and FEDformer(Zhou et al., 2022). **Diffusion-based Models**: We include CSDI(Tashiro et al., 2021a) and adapt the backbones of SOTA tabular diffusion models, TabDiff(Shi et al., 2025b) and TabSyn(Zhang et al., 2024), to the forecasting task.

Settings Standard benchmarks often favor Direct Multi-Step (DMS) models. To ensure a fair comparison of long-term generative dynamics, we establish a unified autoregressive protocol: all models are trained to predict a fixed short horizon ($L_{pred} = 6$) and evaluated by rolling this prediction up to long horizons $H \in \{24, \ldots, 720\}$. To strictly assess the model's intrinsic ability to handle distribution shifts without external aid, we apply standard Z-Score normalization without using learnable affine transformations or reversible normalization modules such as RevIN (Kim et al., 2021).

4.2. Overall Performance

Table 1 summarizes the performance of EADiff against 9 baselines under the unified autoregressive protocol. Our method demonstrates consistent advantages across diverse data characteristics, particularly excelling in long-term forecasting scenarios.

Vs. Time-Series Models. On the non-stationary Exchange dataset, EADiff achieves remarkable improvements, reducing RMSE by 16.9% compared to the best baseline at $H = 720$. This dataset poses a severe challenge for Transformer-based methods like TimeMixer and PatchTST, which suffer dramatic degradation (RMSE > 2.0) without external normalization modules such as RevIN. In contrast, EADiff maintains robust predictions through its intrinsic conditional generation mechanism. On Weather, EADiff achieves the lowest RMSE (0.61) at $H = 720$, reflecting that diffusion models trained with Gaussian objectives naturally capture variance in heavy-tailed distributions. On the periodic ETT-h1, while TimesNet maintains advantages in long-term metrics due to explicit 2D periodicity modeling, EADiff achieves the second-best RMSE (0.837) at $H = 96$,

Table 1. **Performance on Short and Long-Term Forecasting.** We report **MAE** and **RMSE** (lower is better) under the unified autoregressive protocol. The best results are highlighted in **bold**, and the second best are underlined.

Dataset	Len	H	EADiff		TabDiff(2025)		TimeMixer(2024)		iTransformer(2024)		TabSyn(2024)		PatchTST(2023)		FEDformer(2022)		CSDI(2021)	
			MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE
Beijing Air	Short	48	0.2770	0.6087	0.4352	0.8132	0.4973	0.8178	0.4190	0.6566	0.7431	1.0435	0.3235	0.6385	0.4726	0.7562	**0.2712**	**0.6022**
		72	**0.2938**	**0.6255**	0.4378	0.7191	0.5396	0.8791	0.4527	0.6926	0.8803	1.1993	0.3465	0.6739	0.4820	0.7649	0.2967	0.6275
		96	**0.3207**	**0.6412**	0.4426	0.6577	0.5729	0.9179	0.4720	0.7155	1.1403	1.4456	0.3602	0.6952	0.4873	0.7699	0.3240	0.6449
	Long	192	**0.3512**	**0.6737**	0.4799	0.6959	0.6348	0.9790	0.5117	0.7627	1.7557	2.1148	0.3794	0.7262	0.4973	0.7797	0.3519	0.6837
		336	**0.3857**	0.7252	0.6132	1.0593	0.7470	1.1116	0.5361	0.7898	1.8438	2.2803	0.3913	0.7453	0.4149	**0.7152**	0.3863	0.7236
		720	0.4360	0.7986	0.6933	1.1457	1.0261	1.4395	0.5658	0.8286	1.9001	2.2515	0.4133	0.7860	**0.5081**	**0.8585**	0.4446	0.8037
weather	Short	48	0.3561	**0.6299**	OOM	OOM	**0.3522**	0.6419	0.5087	0.8199	NAN	NAN	0.6352	0.9499	0.6110	0.9365	0.3682	0.7863
		72	**0.3979**	**0.6616**	OOM	OOM	0.3984	0.6941	0.5383	0.8561	NAN	NAN	0.7263	1.0599	0.6161	0.9414	0.3989	0.8134
		96	0.4224	**0.6843**	OOM	OOM	0.4384	0.7401	0.5546	0.8744	NAN	NAN	0.7872	1.1328	0.6187	0.9437	**0.4199**	0.8311
	Long	192	**0.4801**	**0.7266**	OOM	OOM	0.7230	1.2424	0.5890	0.9082	NAN	NAN	0.9674	1.3474	0.6224	0.9471	0.4848	0.8975
		336	**0.5374**	**0.8515**	OOM	OOM	1.4022	2.7241	0.6053	0.9231	NAN	NAN	1.0835	1.4676	0.6240	0.9484	0.5549	0.9883
		720	**0.6102**	0.9325	OOM	OOM	2.3338	6.2094	0.6167	**0.9248**	NAN	NAN	1.1788	1.5461	0.6242	0.9479	0.6553	1.1222
ETT-h1	Short	48	0.5447	**0.7537**	0.8040	0.9985	0.7185	0.9244	0.5806	0.7856	1.0308	1.2759	0.6796	0.8927	0.7635	0.9946	**0.5523**	0.8410
		72	0.5810	**0.7938**	0.7915	0.9862	0.7312	0.9418	0.6119	0.8203	0.9906	1.2252	0.7081	0.9282	0.7704	1.0017	**0.5695**	0.8587
		96	0.6183	**0.8367**	0.7735	0.9731	0.7307	0.9418	0.6372	0.8471	1.1319	1.3650	0.7311	0.9565	0.7782	1.0098	**0.5817**	0.8703
	Long	192	0.6675	0.9985	0.9497	1.1410	0.7427	0.9544	0.7127	0.9275	1.1491	1.3692	0.7885	1.0263	0.8116	1.0486	**0.6250**	**0.9125**
		336	0.7312	0.9922	0.9769	1.2056	0.7751	0.9821	0.7749	0.9963	1.2241	1.4884	0.8257	1.0704	0.8470	1.0974	**0.6738**	**0.9615**
		720	0.8375	1.0313	1.0902	1.3009	0.9517	1.2007	0.8540	1.0867	1.3107	1.5684	0.8594	1.1042	0.8709	1.1319	**0.7828**	**1.0723**
ETT-h2	Short	48	0.3400	0.4654	0.7386	0.9248	0.8045	1.0175	**0.2905**	**0.3924**	0.8718	1.0781	0.4157	0.5495	0.3484	0.4795	0.7723	0.9883
		72	0.3670	0.5000	0.7204	0.8834	0.8276	1.0414	**0.3184**	**0.4271**	0.9100	1.1923	0.4457	0.5936	0.3677	0.5058	0.7745	0.9926
		96	0.3941	0.5177	0.8233	1.0073	0.8265	1.0344	**0.3409**	**0.4546**	1.1326	1.3940	0.4761	0.6386	0.3942	0.5219	0.7744	0.9935
	Long	192	0.4777	0.6404	0.9238	1.0820	0.9123	1.1488	**0.4100**	**0.5402**	1.4312	1.6119	0.5677	0.7718	0.4833	0.6470	0.7709	0.9915
		336	0.5346	0.6990	1.0782	1.1929	1.0330	1.2810	**0.4788**	**0.6243**	1.5355	1.7503	0.6467	0.8826	0.7198	0.9639	0.7674	0.9887
		720	0.6081	0.7751	1.2453	1.3630	1.2584	1.5317	**0.5648**	**0.7307**	1.6165	1.7760	0.7594	1.0401	1.1451	1.4383	0.7493	0.9744
Exchange	Short	48	**0.3006**	**0.3710**	0.8135	0.8812	1.9388	2.2386	0.6362	0.7874	1.4457	1.5298	0.7570	0.9941	1.4935	1.7992	0.3444	0.5222
		72	**0.3572**	**0.4400**	0.9098	1.0009	2.1907	2.5150	0.7786	0.9547	1.4699	1.5736	0.8992	1.1982	1.4932	1.7921	0.4253	0.6317
		96	**0.4060**	**0.5004**	0.8734	0.9863	2.5175	2.9420	0.9169	1.1202	1.6232	1.7496	1.0181	1.3658	1.4792	1.7741	0.4873	0.7101
	Long	192	**0.5512**	**0.6823**	0.8796	0.9995	3.4653	4.1635	1.3058	1.5960	2.0518	2.1355	1.3499	1.7936	1.4071	1.6912	0.6205	0.8573
		336	**0.6851**	**0.8522**	1.0037	1.1285	5.2163	6.5516	1.3965	1.7334	2.2289	2.3272	1.6013	2.1144	1.2761	1.5599	0.7415	0.9972
		720	**0.8505**	**1.0534**	1.0192	1.0927	12.5190	17.7049	1.2978	1.6277	2.7411	2.7894	1.7953	2.4468	1.1122	1.3995	0.9134	1.1886

outperforming iTransformer and PatchTST, demonstrating effective capture of local periodic dynamics.

Vs. Diffusion-Based Models. Against the time-series diffusion model CSDI, EADiff achieves substantial improvements across most metrics (e.g., RMSE 0.500 vs 0.710 on Exchange at $H = 96$), validating the efficiency of our wavelet-based latent space over pure time-domain diffusion. The tabular diffusion methods TabDiff and TabSyn, despite their strong performance on structured data generation, either fail to scale to high-dimensional time series (OOM on Weather) or produce suboptimal predictions due to ignoring temporal inductive biases. TabSyn even produces NaN on Weather, indicating fundamental incompatibility with complex temporal dependencies. These results underscore the necessity of our frequency-aware architecture that respects the multi-scale structure inherent in time series data.

4.3. Ablation Studies

We compare EADiff against three variants: removing the adaptive scheduler (w/o Adaptive), removing wavelet decomposition (w/o Wavelet), and a vanilla time-domain baseline (w/o Wav & Adap). Table 2 and Figure 2 jointly demonstrate the effectiveness of our core components from both accuracy and optimization perspectives.

Quantitative Results. Table 2 reveals that both components contribute critically to performance. Removing the wavelet module causes substantial degradation, particularly on non-stationary data where Exchange RMSE increases from 0.500 to 1.301, confirming that frequency disentanglement is essential for handling complex temporal dynamics. The adaptive scheduler is equally critical: without it, Beijing Air RMSE degrades from 0.681 to 1.028, validating that uniform noise injection cannot adequately address the uneven energy distribution across frequency bands.

Table 2. **Ablation Results ($H = 96$).**

Method	Beijing Air MAE / RMSE	Exchange MAE / RMSE	ETTh1 MAE / RMSE	ETTh2 MAE / RMSE
EADiff (Ours)	**0.341 / 0.681**	**0.406 / 0.500**	**0.628 / 0.837**	**0.394 / 0.538**
w/o Adaptive	0.580 / 1.028	0.643 / 0.840	0.832 / 1.137	0.416 / 0.558
w/o Wavelet	0.357 / 0.682	1.056 / 1.301	0.913 / 2.742	0.497 / 0.663
w/o Wav & Adap	0.448 / 1.028	0.945 / 1.153	0.735 / 1.044	0.462 / 0.630

Convergence Analysis. Figure 2 provides complementary insights into optimization behavior. The variant without wavelets exhibits severe instability with high variance throughout training, struggling to converge on chaotic Exchange data. Removing only the adaptive scheduler yields more stable convergence but plateaus at a higher loss, suggesting difficulty in balancing the learning of trend and detail components simultaneously. EADiff achieves both rapid convergence and the lowest stable loss, confirming that our energy-adaptive mechanism effectively accelerates optimization while improving training stability.

4.4. Model Analysis

4.4.1. Adaptivity of Noise Scheduling

Figure 3 illustrates how the learnable modulation factor γ evolves during training, revealing automatic adaptation to

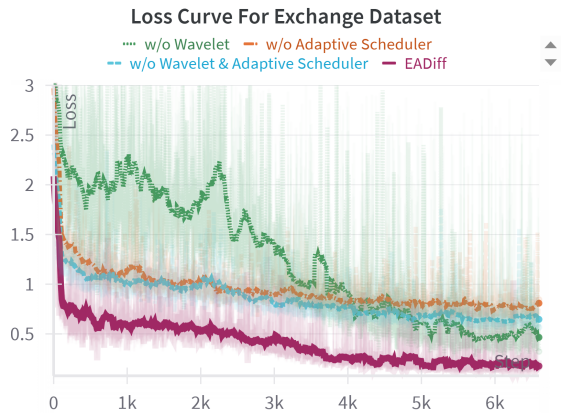

Figure 2. **Training Convergence Analysis on Exchange.** EADiff (Purple) achieves faster convergence and a lower final loss compared to the ablation variants. The "w/o Wavelet" variant (Green) exhibits severe instability, highlighting the difficulty of modeling raw non-stationary data directly.

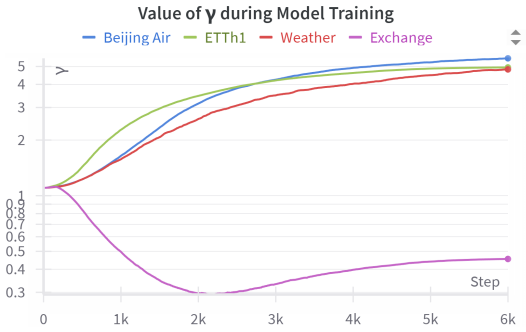

Figure 3. **Evolution of the Learnable Parameter γ during Training.** The parameter adapts differently depending on data characteristics: it increases for structured, periodic datasets (Beijing Air, ETTh1, Weather) to enhance spectral differentiation, but decreases for the highly stochastic Exchange dataset to prevent overfitting to spurious frequency energy.

Table 3. **Impact of Decomposition Levels ($H = 96$).** We fix the wavelet basis to `sym2` and vary the level from 1 to 5. A trade-off is observed: deep decomposition benefits non-stationary data (Exchange) by isolating trends, while shallow decomposition favors noisy data (Beijing Air, ETTh2) by preserving temporal resolution.

Dataset	Level 1		Level 2		Level 3		Level 4		Level 5	
	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE
Beijing Air	**0.341**	**0.681**	0.465	0.759	0.580	0.907	0.583	0.915	0.748	1.106
ETTh1	**0.702**	**0.985**	0.825	1.119	0.749	1.105	0.919	1.366	0.800	1.031
ETTh2	**0.394**	**0.538**	0.441	0.583	0.599	0.931	0.798	1.001	0.656	0.876
Exchange	1.168	1.397	0.938	1.232	0.935	1.194	1.204	1.487	**0.406**	**0.500**

data characteristics. For datasets with strong periodicity and clear temporal patterns (Beijing Air, ETTh1, Weather), γ increases rapidly from the initial value of 1.0 to approximately 5.0, indicating that the model actively learns to amplify differentiation between high-energy trend bands and low-energy detail bands, protecting high-frequency information from being overwhelmed during diffusion.

Conversely, on the stochastic Exchange dataset characterized by random walks, γ drops sharply to approximately 0.3. For such data, energy differences between frequency bands are often dominated by noise rather than meaningful signal; a high γ would erroneously bias the model toward spurious frequency artifacts. By automatically reducing γ, EADiff effectively flattens the modulation toward uniform diffusion, which proves more robust for chaotic data. This bifurcation demonstrates that EADiff autonomously discovers the optimal energy prior for different data domains—a pattern consistently reflected in the divergent optimal wavelet configurations across datasets (Table 3).

4.4.2. Analysis of Wavelet Decomposition

Tables 3 and 4 examine how decomposition level and wavelet family affect performance, revealing that theoretical optimality in wavelet analysis does not directly translate to modeling performance.

Impact of Decomposition Levels. The optimal level is governed by data stationarity. For high-frequency dominated data like Beijing Air and ETTh2, shallow decomposition (level 1-2) significantly outperforms deeper levels, with Beijing Air RMSE degrading from 0.681 at L1 to 1.106 at L5. Deep decomposition on short windows excessively compresses the trend's temporal dimension, destroying local context needed for predicting rapid fluctuations. Conversely, non-stationary Exchange benefits monotonically from deeper decomposition (RMSE: 0.500 at L5 vs 1.397 at L1), as it isolates the shifting trend into a simplified approximation band. This divergence mirrors the adaptive γ behavior in Figure 3: structured data requires strong spectral differentiation while stochastic data benefits from flatter treatment across frequency bands.

Impact of Wavelet Families. Table 5 reveals a critical sparsity-stability trade-off. Although biorthogonal wavelets (bior3.1) achieve the lowest Shannon entropy, they suffer from extreme energy imbalance with 60-150$\times$ scale ratios between bands, challenging numerical stability during optimization and leading to significantly worse RMSE than the balanced sym2. This finding establishes that moderate sparsity is preferable to extreme sparsity in diffusion-based modeling, providing practical guidance: sym2 offers robust performance across datasets, while the simple Haar wavelet suits data with sharp transitions like power grid loads.

4.4.3. The Paradox of Sparsity: Basis Selection

A critical finding is that theoretical optimality in wavelet decomposition does not directly translate to modeling performance. We utilized Shannon Entropy to identify the

Table 4. **Impact of Wavelet Families ($H = 96$).** We compare different wavelet bases at deepest decomposition levels. `sym2` and `db1` generally outperform `bior3.1`, despite the latter having lower theoretical entropy. This highlights the importance of numerical stability and basis symmetry over pure sparsity.

Dataset	`sym2` (L5)		`db1` (L6)		`bior3.1` (L5)		`db4` (L3)	
	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE
Beijing Air	0.748	1.106	0.900	1.172	4.195	6.866	**0.404**	**0.742**
ETTh1	0.800	1.031	**0.628**	**0.837**	2.896	5.732	0.875	1.136
ETTh2	0.656	0.876	**0.417**	**0.564**	1.120	1.633	0.508	0.663
Exchange	**0.406**	**0.500**	0.821	0.995	1.042	1.301	0.900	1.172

Table 5. **Physical Properties vs. Model Performance ($H = 96$).** We compare `sym2` and `bior3.1` at Level 5. **Entropy**: Lower implies better theoretical compression. **Scale Ratio**: $\sigma_{max}/\sigma_{min}$, indicating energy imbalance (Lower is more numerically stable).

Dataset	**`sym2` (L5)**			**`bior3.1` (L5)**		
	Entropy ↓	Ratio ↓	RMSE ↓	Entropy ↓	Ratio ↑	RMSE ↓
ETTh1	5.33	28.9	0.800	5.15	72.3	2.896
Exchange	3.37	88.6	0.406	3.14	140.8	1.042
Beijing Air	5.29	121.5	0.748	5.28	175.8	4.195
ETTm1	5.90	36.5	0.573	5.76	72.4	0.692

sparsest representation, assuming higher sparsity would simplify the diffusion task. As shown in our heuristic analysis, `bior3.1` consistently achieves the lowest entropy due to its biorthogonal property, enabling aggressive energy compaction.

However, empirical results in Table 4 reveal a paradox: `bior3.1` often yields inferior performance compared to `sym2` despite being "sparser." We attribute this failure to spectral energy imbalance: `bior3.1` concentrates energy so aggressively that the standard deviation ratio between trend and detail bands reaches 60-150×, challenging numerical stability and causing gradient dominance by high-energy bands. In contrast, `sym2` strikes a balance with competitive sparsity (Entropy ≈ 5.3 vs. 5.1) but maintains a moderate scale ratio (≈ 20-40×), preserving sufficient signal magnitude in high-frequency bands for effective denoising. This implies a Pareto principle for basis selection: sparsity should be maximized only as long as the inter-band energy gap remains within a numerically stable regime.

5. Related work

Time series forecasting Driven by deep learning architectures, the Transformer (Vaswani, 2017) has become the dominant paradigm for time series forecasting due to its strong global modeling capability. Recent studies primarily focus on three directions. For temporal pattern specialization, MICN (Wang et al., 2023) integrates global and local contextual information, while L3former (Xia et al., 2025) introduces local linear connections to jointly capture temporal dynamics and inter-variable correlations. In attention mechanism evolution, Pathformer (Chen et al., 2024) achieves multi-scale alignment via segment fusion, iTransformer (Liu et al., 2024b) models cross-variable dependencies through channel-independent attention, and Crossformer (Zhang & Yan, 2023) employs a two-stage attention mechanism for temporal and variable dependencies. For multi-scale architectures, TimeMixer (Wang et al., 2024b) learns hierarchical representations through Past-Decomposable-Mixing and aggregates forecasts via Future-Multipredictor-Mixing. Graph neural networks have also shown strong capability in modeling complex spatiotemporal dependencies, such as dynamic spatiotemporal graph networks (Li et al., 2023b). To alleviate data sparsity, GPT-ST (Li et al., 2023a) and Automated Spatiotemporal Graph Contrastive Learning (Zhang et al., 2023), learn generalizable representations from large-scale unlabeled data via pre-training and self-supervised learning.

Diffusion model Recent advancements have demonstrated the effectiveness of diffusion models in time series forecasting. TimeGrad (Rasul et al., 2021) pioneered conditional diffusion in an autoregressive manner but suffers from slow inference. CSDI (Tashiro et al., 2021b) introduced a non-autoregressive masked diffusion framework with costly dual transformers and quadratic complexity. To improve efficiency, SSSD (Alcaraz & Strodthoff, 2023) replaces transformers with structured state space models. Multi-scale diffusion models further decompose temporal structures, such as MG-TSD (Fan et al., 2024) and mr-diff (Shen et al., 2024) which separately model trend and seasonal components. Retrieval-augmented diffusion models (Liu et al., 2024a) enhance forecasting by leveraging similar historical patterns. Recent diffusion-based tabular generative models, including as TabSyn (Zhang et al., 2024) and TabDiff (Shi et al., 2025b), exhibit strong structured-data modeling capacity, motivating their extension to time series forecasting.

6. Conclusion

We present EADiff, an energy-adaptive diffusion framework that addresses the frequency-energy imbalance problem in time series forecasting. By operating in the wavelet domain with a learnable noise modulation mechanism, EADiff automatically adjusts noise levels according to per-instance frequency band characteristics, enabling balanced corruption and reconstruction across multi-scale components. Extensive experiments demonstrate competitive performance against both time-series forecasting and diffusion-based baselines, while our analysis reveals that the optimal wavelet configuration is data-dependent. Structured periodic data benefits from strong spectral differentiation whereas stochastic data requires preserving temporal resolution. These findings provide practical guidance for future work on frequency-domain generative models for time series. Code is available at this repository: https://anonymous.4open.science/r/EADiff-28B2/.

References

Alcaraz, J. M. L. and Strodthoff, N. Diffusion-based time series imputation and forecasting with structured state space models. *Trans. Mach. Learn. Res.*, 2023, 2023.

Chen, P., Zhang, Y., Cheng, Y., Shu, Y., Wang, Y., Wen, Q., Yang, B., and Guo, C. Pathformer: Multi-scale transformers with adaptive pathways for time series forecasting. 2024.

Du, W., Yang, Y., Qian, L., Wang, J., and Wen, Q. PyPOTS: A Python Toolkit for Machine Learning on Partially-Observed Time Series. *arXiv preprint arXiv:2305.18811*, 2023.

Fan, X., Wu, Y., Xu, C., Huang, Y., Liu, W., and Bian, J. MG-TSD: multi-granularity time series diffusion models with guided learning process. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024.

Hamilton, J. D. *Time series analysis*. Princeton university press, 2020.

Ho, J., Jain, A., and Abbeel, P. Denoising diffusion probabilistic models. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

Karras, T., Aittala, M., Aila, T., and Laine, S. Elucidating the design space of diffusion-based generative models. In Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., and Oh, A. (eds.), *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022.

Kim, T., Kim, J., Tae, Y., Park, C., Choi, J.-H., and Choo, J. Reversible instance normalization for accurate time-series forecasting against distribution shift. In *International Conference on Learning Representations*, 2021.

Lai, G., Chang, W., Yang, Y., and Liu, H. Modeling long- and short-term temporal patterns with deep neural networks. In Collins-Thompson, K., Mei, Q., Davison, B. D., Liu, Y., and Yilmaz, E. (eds.), *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, SIGIR 2018, Ann Arbor, MI, USA, July 08-12, 2018*, pp. 95–104. ACM, 2018. doi: 10.1145/3209978.3210006.

Li, Z., Xia, L., Xu, Y., and Huang, C. Gpt-st: Generative pre-training of spatio-temporal graph neural networks. In Oh, A., Naumann, T., Globerson, A., Saenko, K., Hardt, M., and Levine, S. (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 70229–70246. Curran Associates, Inc., 2023a.

Li, Z., Yu, J., Zhang, G., and Xu, L. Dynamic spatio-temporal graph network with adaptive propagation mechanism for multivariate time series forecasting. *Expert Syst. Appl.*, 216:119374, April 2023b.

Liang, X., Zou, T., Guo, B., Li, S., Zhang, H., Zhang, S., Huang, H., and Chen, S. X. Assessing beijing's pm2.5 pollution: severity, weather impact, apec and winter heating. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 471, 2015.

Liu, J., Yang, L., Li, H., and Hong, S. Retrieval-augmented diffusion models for time series forecasting. In Globersons, A., Mackey, L., Belgrave, D., Fan, A., Paquet, U., Tomczak, J. M., and Zhang, C. (eds.), *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024a.

Liu, Y., Hu, T., Zhang, H., Wu, H., Wang, S., Ma, L., and Long, M. itransformer: Inverted transformers are effective for time series forecasting. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024b.

Mallat, S. A theory for multiresolution signal decomposition: The wavelet representation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 11(7):674–693, 1989. doi: 10.1109/34.192463.

Meng, C., He, Y., Song, Y., Song, J., Wu, J., Zhu, J., and Ermon, S. Sdedit: Guided image synthesis and editing with stochastic differential equations. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022.

Nie, Y., Nguyen, N. H., Sinthong, P., and Kalagnanam, J. A time series is worth 64 words: Long-term forecasting with transformers. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023.

Oppenheim, A. V., Willsky, A. S., and Nawab, S. H. *Signals and Systems*. Prentice Hall, 2nd edition, 1999.

Peebles, W. and Xie, S. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 4195–4205, 2023.

Rasul, K., Seward, C., Schuster, I., and Vollgraf, R. Autoregressive denoising diffusion models for multivariate probabilistic time series forecasting. In *International conference on machine learning*, pp. 8857–8868. PMLR, 2021.

Shen, L., Chen, W., and Kwok, J. T. Multi-resolution diffusion models for time series forecasting. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024.

Shi, J., Xu, M., Hua, H., Zhang, H., Ermon, S., and Leskovec, J. Tabdiff: a mixed-type diffusion model for tabular data generation. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025a.

Shi, J., Xu, M., Hua, H., Zhang, H., Ermon, S., and Leskovec, J. Tabdiff: a mixed-type diffusion model for tabular data generation. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025b.

Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., and Poole, B. Score-based generative modeling through stochastic differential equations. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

Tashiro, Y., Song, J., Song, Y., and Ermon, S. CSDI: conditional score-based diffusion models for probabilistic time series imputation. In Ranzato, M., Beygelzimer, A., Dauphin, Y. N., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pp. 24804–24816, 2021a.

Tashiro, Y., Song, J., Song, Y., and Ermon, S. Csdi: Conditional score-based diffusion models for probabilistic time series imputation. In *NeurIPS*, 2021b.

Vaswani, A. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017.

Wang, H., Peng, J., Huang, F., Wang, J., Chen, J., and Xiao, Y. MICN: Multi-scale local and global context modeling for long-term series forecasting. In *The Eleventh International Conference on Learning Representations*, 2023.

Wang, S., Wu, H., Shi, X., Hu, T., Luo, H., Ma, L., Zhang, J. Y., and Zhou, J. Timemixer: Decomposable multiscale mixing for time series forecasting. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024a.

Wang, S., Wu, H., Shi, X., Hu, T., Luo, H., Ma, L., Zhang, J. Y., and ZHOU, J. Timemixer: Decomposable multiscale mixing for time series forecasting. In *The Twelfth International Conference on Learning Representations*, 2024b.

Wu, H., Xu, J., Wang, J., and Long, M. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. In Ranzato, M., Beygelzimer, A., Dauphin, Y. N., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pp. 22419–22430, 2021.

Xia, Y., Wu, C., and Yang, X. L3former: Enhanced multiscale shared transformer with local linear layer for long-term series forecasting. *Information Fusion*, 124:103398, 2025. ISSN 1566-2535. doi: https://doi.org/10.1016/j.inffus.2025.103398.

Zeng, A., Chen, M., Zhang, L., and Xu, Q. Are transformers effective for time series forecasting? In Williams, B., Chen, Y., and Neville, J. (eds.), *Thirty-Seventh AAAI Conference on Artificial Intelligence, AAAI 2023, Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence, IAAI 2023, Thirteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2023, Washington, DC, USA, February 7-14, 2023*, pp. 11121–11128. AAAI Press, 2023. doi: 10.1609/AAAI.V37I9.26317.

Zhang, H., Zhang, J., Shen, Z., Srinivasan, B., Qin, X., Faloutsos, C., Rangwala, H., and Karypis, G. Mixed-type tabular data synthesis with score-based diffusion in latent space. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024.

Zhang, Q., Huang, C., Xia, L., Wang, Z., Li, Z., and Yiu, S. Automated spatio-temporal graph contrastive learning. In *Proceedings of the ACM Web Conference 2023*, WWW '23, pp. 295–305, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9781450394161. doi: 10.1145/3543507.3583304.

Zhang, Y. and Yan, J. Crossformer: Transformer utilizing cross-dimension dependency for multivariate time series forecasting. In *The Eleventh International Conference on Learning Representations*, 2023.

Zhou, H., Zhang, S., Peng, J., Zhang, S., Li, J., Xiong, H., and Zhang, W. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Thirty-Fifth AAAI Conference on Artificial Intelligence,*

AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021, pp. 11106–11115. AAAI Press, 2021. doi: 10.1609/AAAI.V35I12.17325.

Zhou, T., Ma, Z., Wen, Q., Wang, X., Sun, L., and Jin, R. Fedformer: Frequency enhanced decomposed transformer for long-term series forecasting. In Chaudhuri, K., Jegelka, S., Song, L., Szepesvári, C., Niu, G., and Sabato, S. (eds.), *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pp. 27268–27286. PMLR, 2022.

A. Detailed Algorithm Descriptions

A.1. Training Algorithm

Algorithm 4 provides the complete training procedure for EADiff, including the energy-adaptive noise computation, differentiated noise injection for history and future regions, and the weighted loss calculation.

A.2. Inference Algorithm

Algorithm 5 describes the complete inference procedure using the Heun sampler with replacement mechanism for conditional generation.

B. Architecture Details

B.1. Wavelet Mask Computation

Due to the non-local nature of wavelet filters, the temporal boundary between history and future at time step $T - H$ does not correspond to a sharp boundary in the coefficient domain. We compute the mask indices by analyzing the wavelet transform's receptive field, determining which coefficients are affected when the future segment is perturbed. Shown as Algorithm 3.

Algorithm 3 Wavelet Mask Index Computation

Require: Window size T, prediction length H_{pred}, wavelet family, decomposition level J
Ensure: Mask starting indices for each band $\{\text{idx}_\ell\}_{\ell=0}^{J}$
$\mathbf{s}_a \leftarrow \mathbf{0} \in \mathbb{R}^T$ {Zero signal}
$\mathbf{s}_b \leftarrow \mathbf{0} \in \mathbb{R}^T$
$\mathbf{s}_b[T - H_{\text{pred}} : T] \leftarrow 1$ {Perturb future segment}
$\mathbf{C}_a \leftarrow \text{DWT}(\mathbf{s}_a, J)$ {List of $J + 1$ coefficient arrays}
$\mathbf{C}_b \leftarrow \text{DWT}(\mathbf{s}_b, J)$
mask_indices $\leftarrow$ []
for $\ell = 0$ to J **do**
  $\mathbf{d} \leftarrow |\mathbf{C}_a^{(\ell)} - \mathbf{C}_b^{(\ell)}|$
  $\text{idx}_\ell \leftarrow \min\{i : \mathbf{d}[i] > \delta\}$ {First differing position}
  mask_indices.append(idx_ℓ)
end for
return mask_indices

The computed mask has the following properties:

- **Deterministic**: The mask depends only on the window size, prediction length, wavelet family, and decomposition level. It can be precomputed once and reused.
- **Band-specific**: Different bands have different mask starting positions due to varying filter lengths and downsampling factors.
- **Conservative**: The mask may include some coefficients that are only partially affected by the future, ensuring complete coverage of the prediction region.

B.2. Band-Specific Tokenization

The tokenizer converts wavelet coefficients into a sequence of tokens suitable for transformer processing. Unlike standard patch embedding that applies uniform tokenization, we employ band-specific strategies that respect the distinct characteristics of different frequency bands.

Adaptive Stride Design. For a patch size P, we assign different strides to different bands:

- **Low-frequency band** ($\mathbf{W}^{(0)}$, approximation): Stride $s_0 = 1$ (dense tokenization). The approximation coefficients capture overall trends where precise localization is critical. Dense tokenization preserves maximum temporal resolution.
- **Detail bands** ($\mathbf{W}^{(\ell)}$ for $\ell \geq 1$): Stride $s_\ell = P/2$ (50% overlap). Detail coefficients benefit from overlapping receptive fields that maintain continuity across patch boundaries.

For a band with padded length $L_{\text{pad}}^{(\ell)}$ and stride s_ℓ, the number of output tokens is:

$$N_{\text{tokens}}^{(\ell)} = \left\lfloor \frac{L_{\text{pad}}^{(\ell)} - P}{s_\ell} \right\rfloor + 1 \tag{17}$$

Each band is tokenized by a separate 1D convolution:

$$\mathbf{T}^{(\ell)} = \text{Conv1D}(\mathbf{W}_{\text{pad}}^{(\ell)}, \text{kernel} = P, \text{stride} = s_\ell) \in \mathbb{R}^{(B \cdot F) \times N_{\text{tokens}}^{(\ell)} \times D} \tag{18}$$

where D is the token dimension. The tokens from all bands are concatenated along the sequence dimension to form the input to the transformer.

Before tokenization, each band is padded to ensure compatibility with the patch size:

$$L_{\text{pad}}^{(\ell)} = L^{(\ell)} + \begin{cases} P - L^{(\ell)} & \text{if } L^{(\ell)} < P \\ P - (L^{(\ell)} \bmod P) & \text{if } L^{(\ell)} \bmod P \neq 0 \\ 0 & \text{otherwise} \end{cases} \tag{19}$$

We use replicate padding to extend the boundary values, which is more suitable for time series than zero padding.

B.3. Overlap-Add Decoding

The decoder reconstructs wavelet coefficients from transformer output tokens. For bands with overlapping tokenization (stride $< P$), we employ overlap-add with proper normalization.

Each band uses a separate transposed convolution to map tokens back to coefficient space:

$$\hat{\mathbf{W}}_{\text{raw}}^{(\ell)} = \text{ConvTranspose1D}(\mathbf{T}_{\text{out}}^{(\ell)}, \text{kernel} = P, \text{stride} = s_\ell) \tag{20}$$

When stride $s_\ell < P$, the transposed convolution produces overlapping contributions that must be normalized:

$$\mathbf{O}^{(\ell)} = \text{ConvTranspose1D}(\mathbf{1}, \mathbf{1}_P, \text{stride} = s_\ell) \tag{21}$$

where $\mathbf{1}$ is an all-ones tensor matching the token shape, and $\mathbf{1}_P$ is an all-ones kernel of size P. The normalization factor $\mathbf{O}^{(\ell)}$ counts how many patches contribute to each output position.

The final reconstructed coefficients are:

$$\hat{\mathbf{W}}^{(\ell)} = \frac{\hat{\mathbf{W}}_{\text{raw}}^{(\ell)}}{\max(\mathbf{O}^{(\ell)}, \epsilon)} \tag{22}$$

After decoding, we crop the output to the original (unpadded) length:

$$\hat{\mathbf{W}}_{\text{final}}^{(\ell)} = \hat{\mathbf{W}}^{(\ell)}[:, :, : L^{(\ell)}] \tag{23}$$

B.4. Embedding Design

The transformer input incorporates three types of embeddings:

Sinusoidal Position Embedding. Global position information across the entire token sequence:

$$\mathbf{PE}_{pos,2i} = \sin\left(\frac{pos}{10000^{2i/D}}\right), \quad \mathbf{PE}_{pos,2i+1} = \cos\left(\frac{pos}{10000^{2i/D}}\right) \tag{24}$$

Level Embedding. Learnable embeddings that distinguish tokens from different frequency bands:

$$\mathbf{E}_{\text{level}} = \text{Embedding}(J+1, D) \tag{25}$$

where $J+1$ is the total number of bands (one approximation plus J detail bands).

Feature Embedding. For multivariate time series, learnable embeddings distinguish different variates:

$$\mathbf{E}_{\text{feature}} = \text{Embedding}(F, D) \tag{26}$$

The final input to the transformer is:

$$\mathbf{H}^{(0)} = \mathbf{T} + \mathbf{PE} + \mathbf{E}_{\text{level}} + \mathbf{E}_{\text{feature}} \tag{27}$$

C. Experimental Setup

C.1. Datasets

We evaluate EADiff on five diverse real-world benchmarks. Table 6 summarizes the key statistics. To ensure reproducibility, we detail the data preprocessing steps, splitting protocols, and baseline configurations used in our experiments. Our implementation strictly treats all data as multivariate continuous time series; We normalize features using Z-Score (Standard) or MinMax scaling based on the data distribution. No discrete variable embeddings (e.g., time-of-day or day-of-week) are used, forcing the model to learn temporal dynamics solely from the signal values.

Table 6. **Dataset Statistics and Split Protocols.**

Dataset	Variates (F)	Total Samples	Frequency	Split (Train/Val/Test)	Description
Beijing Air	7	43,824	1 hour	0.7 / 0.1 / 0.2	Air quality: PM2.5, DEWP, TEMP, PRES, Iws, Is, Ir
Weather	12	262,968	10 min	0.7 / 0.1 / 0.2	MPI Roof 2021–2025 (filtered features)
ETT-h1	7	17,420	1 hour	0.6 / 0.2 / 0.2	Electricity Transformer Temperature
ETT-h2	7	17,420	1 hour	0.6 / 0.2 / 0.2	Electricity Transformer Temperature
Exchange	8	7,588	Daily	0.7 / 0.1 / 0.2	Daily exchange rates of 8 currencies

Feature Selection and Filtering.

- **Beijing Air**: Contains hourly air quality measurements. We use 7 numerical features: PM2.5, DEWP (dew point), TEMP (temperature), PRES (pressure), Iws (cumulated wind speed), Is (cumulated hours of snow), and Ir (cumulated hours of rain).
- **Weather**: We utilize the Max Planck Institute (MPI) Roof dataset spanning 2021–2025. To rigorously test the model's ability to learn dynamics rather than trivial correlations, we removed derived variables (e.g., T_{pot}, T_{dew}, VP_{max}, VP_{def}, ρ, H_2OC). The retained 12 features are: `p (mbar)`, `T (degC)`, `rh (%)`, `wv (m/s)`, `max. wv (m/s)`, `wd (deg)`, `rain (mm)`, `raining (s)`, `SWDR (W/m²)`, `PAR (µmol/m²/s)`, `max. PAR`, `CO2 (ppm)`.
- **ETT (h1, h2)**: Electricity Transformer Temperature data recorded hourly. Features include HUFL, HULL, MUFL, MULL, LUFL, LULL, and OT (oil temperature).
- **Exchange**: Daily exchange rates of 8 currencies. This dataset is highly non-stationary and exhibits random walk behavior.

Data Preprocessing Pipeline. All datasets are processed using a unified pipeline:

1. **Feature Selection**: Timestamps and index columns are removed. Only numerical measurements are retained.
2. **Missing Value Imputation**: For datasets with missing values (e.g., Beijing Air, Weather), we apply linear interpolation followed by forward-filling and backward-filling to ensure continuity.

3. **Normalization**: We apply per-feature normalization to the raw time domain data:
   - **Standard (Z-Score)**: Used for Beijing Air and Exchange. Maps data to zero mean and unit variance.
   - **MinMax**: Used for ETT and Weather. Maps data to $[-1, 1]$ range, preferred over $[0, 1]$ for diffusion models to avoid mean bias.

   Importantly, we **do not** normalize the wavelet coefficients after decomposition. This preserves the natural energy hierarchy where low-frequency coefficients carry higher variance than high-frequency details, which is crucial for the energy-adaptive scheduler.

4. **Chronological Splitting**: Data is split chronologically into Train, Validation, and Test sets according to the ratios in Table 6.

C.2. Baseline Implementations

Time-Series Forecasting Models. We compare against five state-of-the-art time-series forecasting models, using PyPOTs(Du et al., 2023), default settings, MSE for Training and MAE for validation, training up to 100 epoches:

- **TimeMixer** (2024): Learns hierarchical representations through Past-Decomposable-Mixing.
- **iTransformer** (2024): Models cross-variable dependencies through channel-independent attention.
- **PatchTST** (2023): Applies patch-based tokenization with channel independence.
- **FEDformer** (2022): Uses frequency-enhanced decomposition for long-term forecasting.
- **CSDI** (2021): Conditional score-based diffusion model for time series imputation/forecasting.

All transformer baselines are trained using their official codebases with the standard look-back window of 96.

Diffusion-based Models. We adapt two recent tabular diffusion models for time series forecasting:

- **TabSyn** (2024): TabSyn is a Latent Diffusion Model that uses a VAE to compress data into a latent space. **VAE Backbone**: Transformer-based VAE with $N = 4$ layers. The input window is flattened and tokenized via numerical embedding. **Latent Diffusion**: MLP-based diffusion model in the latent space.**Two-Stage Training**: (1) VAE trained with MSE + KL loss; (2) Diffusion trained on frozen VAE latents.
- **TabDiff** (2025): TabDiff operates directly on the data space with an encoder-decoder architecture. Tokenizer $\rightarrow$ Transformer Encoder $\rightarrow$ MLP Diffusion $\rightarrow$ Transformer Decoder. Formulated as an in-painting task where the future is masked. End-to-end with EDM-style mixed loss. Data consistency enforced by resetting history at each step.

Unified Evaluation Protocol. To ensure fair comparison of long-term generative dynamics, we establish a unified autoregressive protocol:

- All models are trained to predict a fixed short horizon ($L_{\text{pred}} = 6$).
- Evaluation is performed by rolling predictions up to long horizons $H \in \{24, 48, 72, 96, 192, 336, 720\}$.
- Standard Z-Score normalization is applied without learnable affine transformations or reversible normalization modules (e.g., RevIN).

C.3. Hyperparameters

EADiff Configuration. Table 7 summarizes the hyperparameters for EADiff across different datasets.

Baseline Hyperparameters. Table 8 summarizes the hyperparameters for the adapted diffusion baselines.

Table 7. **EADiff Hyperparameters per Dataset.**

Parameter	Beijing Air	Weather	ETT-h1/h2	Exchange	Default
Wavelet Configuration					
Wavelet Family	sym2	sym2	sym2 / db1	sym2	sym2
Decomposition Level	1	1	1 (sym2) / 6 (db1)	5	–
Model Architecture					
Token Dimension D	128	128	128	128	128
Patch Size P	4	4	4	4	4
Transformer Layers	6	6	6	6	6
Attention Heads	4	4	4	4	4
FFN Dimension	512	512	512	512	512
Dropout	0.1	0.1	0.1	0.1	0.1
Noise Scheduler					
$\sigma_{\min}$	0.002	0.002	0.002	0.002	0.002
$\sigma_{\max}$	20.0	20.0	20.0	20.0	20.0
ρ	7.0	7.0	7.0	7.0	7.0
Tanh Scale C	3.0	3.0	3.0	3.0	3.0
Initial γ	0.7	0.7	0.7	0.7	0.7
Training					
Batch Size	512	512	4096	768	–
Learning Rate	2e-3	2e-3	2e-3	2e-3	2e-3
Epochs	300	300	300	300	300
Warmup Epochs	25	25	25	25	25
History Noise k_{aug}	0.2	0.2	0.2	0.2	0.2
EMA Memory (epochs)	10	20	20	3	10
Weight Decay	0.01	0.01	0.01	0.01	0.01
Inference					
Heun Steps	20	20	20	20	20
Starting t	1.0	1.0	1.0	1.0	1.0
Prediction Length	6	6	6	6	6

Table 8. **Hyperparameters for Diffusion Baseline Adaptations.**

Parameter	TabSyn	TabDiff
Architecture		
Model Type	VAE + MLP Diffusion	Transformer + MLP Diffusion
Token Dimension	8	16
Transformer Layers	4	4
Attention Heads	1	4
Latent / Time Dim	VAE Latent	512
Training		
Learning Rate	VAE: 1e-3, Diff: 1e-3	1e-4
Training Batch Size	256	256
Test Batch Size	2048	2048
Epochs	VAE: 500, Diff: 2000	2000
Inference		
Sampling Steps	10 (Heun)	10 (Heun)
Guidance Method	Replacement	In-painting

D. Complexity Analysis

We analyze the computational complexity of EADiff's main components:

Wavelet Transform. The Discrete Wavelet Transform (DWT) and its inverse (IDWT) have linear time complexity:

$$\mathcal{O}_{\text{DWT}} = \mathcal{O}(T \cdot F) \tag{28}$$

where T is the sequence length and F is the number of features. This is because each level of decomposition processes half the samples of the previous level, resulting in a geometric series that sums to $\mathcal{O}(T)$.

Tokenization. The band-specific tokenization involves 1D convolutions:

$$\mathcal{O}_{\text{tokenize}} = \mathcal{O}\left(\sum_{\ell=0}^{J} \frac{L^{(\ell)}}{s_\ell} \cdot P \cdot D \cdot F\right) = \mathcal{O}(N \cdot P \cdot D \cdot F) \tag{29}$$

where $N = \sum_\ell N_{\text{tokens}}^{(\ell)}$ is the total number of tokens, P is the patch size, and D is the token dimension.

Transformer Backbone. The self-attention mechanism dominates the transformer complexity:

$$\mathcal{O}_{\text{transformer}} = \mathcal{O}(L_{\text{layers}} \cdot (N \cdot F)^2 \cdot D) \tag{30}$$

where L_{layers} is the number of transformer layers. The quadratic dependence on sequence length $(N \cdot F)$ is the primary computational bottleneck.

Energy-Adaptive Scheduler. Computing the per-instance energy modulation factors:

$$\mathcal{O}_{\text{scheduler}} = \mathcal{O}((J+1) \cdot F \cdot L) \tag{31}$$

where L is the total coefficient length. This is linear and negligible compared to the transformer.

Total Training Complexity (per step).

$$\mathcal{O}_{\text{train}} = \mathcal{O}(T \cdot F) + \mathcal{O}(L_{\text{layers}} \cdot (N \cdot F)^2 \cdot D) \tag{32}$$

Comparison with Time-Domain Methods. Table 9 compares the complexity of different approaches:

Table 9. **Time Complexity Comparison.** T: sequence length, F: features, D: hidden dim, L: layers.

Method	Complexity	Notes
Vanilla Transformer	$\mathcal{O}(L \cdot T^2 \cdot F \cdot D)$	Quadratic in T
PatchTST	$\mathcal{O}(L \cdot (T/P)^2 \cdot F \cdot D)$	Reduced by patch size P
iTransformer	$\mathcal{O}(L \cdot F^2 \cdot T \cdot D)$	Quadratic in F
CSDI	$\mathcal{O}(L \cdot T^2 \cdot F^2 \cdot D)$	Dual attention
EADiff (Ours)	$\mathcal{O}(L \cdot N^2 \cdot F^2 \cdot D)$	$N < T$ due to DWT compression

The key advantage of EADiff is that the wavelet transform compresses the sequence length from T to $N = \sum_\ell N_{\text{tokens}}^{(\ell)}$, where typically $N < T$ due to downsampling in the DWT. For a J-level decomposition with window size $T = 96$, the approximation band has length $\approx T/2^J$, providing significant compression for deeper decompositions.

.

Algorithm 4 EADIFF Training (Detailed)

Require: Training data $\{\mathbf{X}^{(i)}\}_{i=1}^{N}$, model F_θ, learnable modulation parameter γ, prediction length H_{pred}, history augmentation bound k_{aug}
Ensure: Trained model parameters θ, learned γ

Initialize model parameters θ, set $\gamma \leftarrow \gamma_{\text{init}}$
repeat
  Sample batch $\mathbf{X} \in \mathbb{R}^{B\times T\times F}$ from training data
  // Step 1: Construct input with initial guess
  $\mathbf{X}_{\text{his}} \leftarrow \mathbf{X}[:, : T - H_{\text{pred}}, :]$ {History portion}
  $\mathbf{x}_{\text{last}} \leftarrow \mathbf{X}_{\text{his}}[:, -1, :]$ {Last observed value}
  $\hat{\mathbf{X}}_{\text{guess}} \leftarrow \text{LinearExtrapolate}(\mathbf{X}_{\text{his}}, H_{\text{pred}})$ {or Copy}
  $\tilde{\mathbf{X}} \leftarrow \text{Concat}(\mathbf{X}_{\text{his}}, \hat{\mathbf{X}}_{\text{guess}})$
  // Step 2: Wavelet transform and normalization
  $\tilde{\mathbf{X}}_{\text{scaled}} \leftarrow \text{OriginalScaler}(\tilde{\mathbf{X}})$
  $\mathbf{W}_{\text{base}} \leftarrow \text{DWT}(\tilde{\mathbf{X}}_{\text{scaled}})$ $\{\mathbf{W}_{\text{base}} \in \mathbb{R}^{B\times F\times L}\}$
  $\mathbf{W}_{\text{base}} \leftarrow \text{WaveletScaler}(\mathbf{W}_{\text{base}})$
  // Step 3: Compute ground truth coefficients
  $\mathbf{W}_{\text{target}} \leftarrow \text{WaveletScaler}(\text{DWT}(\text{OriginalScaler}(\mathbf{X})))$
  // Step 4: Compute energy-adaptive modulation factors
  for each frequency band $\ell = 0, \ldots, J$ **do**
    $E^{(\ell)} \leftarrow \log(\text{Std}(\mathbf{W}_{\text{base}}^{(\ell)}, \text{dim} = 2) + \epsilon)$ {Per-instance energy}
  end for
  $\bar{E} \leftarrow \frac{1}{J+1}\sum_{\ell=0}^{J} E^{(\ell)}$ {Mean log-energy}
  $\tilde{E}^{(\ell)} \leftarrow E^{(\ell)} - \bar{E}$ {Centered energy}
  $f^{(\ell)} \leftarrow \exp(\gamma \cdot \tanh(\tilde{E}^{(\ell)}/C))$ {Modulation factor}
  // Step 5: Sample noise levels
  $k_{\text{fut}} \sim \mathcal{U}[0,1]^B, \quad k_{\text{his}} \sim \mathcal{U}[0, k_{\text{aug}}]^B$
  $\sigma_{\text{base,fut}} \leftarrow (\sigma_{\min}^{1/\rho} + k_{\text{fut}} \cdot (\sigma_{\max}^{1/\rho} - \sigma_{\min}^{1/\rho}))^\rho$
  $\sigma_{\text{base,his}} \leftarrow (\sigma_{\min}^{1/\rho} + k_{\text{his}} \cdot (\sigma_{\max}^{1/\rho} - \sigma_{\min}^{1/\rho}))^\rho$
  $\boldsymbol{\sigma}_{\text{fut}} \leftarrow \sigma_{\text{base,fut}} \cdot \mathbf{f}, \quad \boldsymbol{\sigma}_{\text{his}} \leftarrow \sigma_{\text{base,his}} \cdot \mathbf{f}$
  // Step 6: Construct noisy input with mask
  $\mathbf{m} \leftarrow \text{ComputeWaveletMask}(T, H_{\text{pred}}, \text{wavelet}, J)$ {Binary mask}
  $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
  $\mathbf{W}_k \leftarrow \mathbf{W}_{\text{base}} + ((1-\mathbf{m}) \odot \boldsymbol{\sigma}_{\text{his}} + \mathbf{m} \odot \boldsymbol{\sigma}_{\text{fut}}) \odot \boldsymbol{\epsilon}$
  // Step 7: Forward pass with conditioning
  $\mathbf{c}_{W^{(0)}} \leftarrow \mathbf{W}_{\text{base}}[:, :, : L_0]$ {Low-frequency condition}
  $\hat{\mathbf{W}}_0 \leftarrow F_\theta(\mathbf{W}_k, \boldsymbol{\sigma}_{\text{fut}}, \mathbf{c}_{W^{(0)}})$
  // Step 8: Compute weighted loss (future region only)
  $w(\sigma) \leftarrow (\sigma^2 + 1)/\sigma^2$ {EDM weighting}
  $\mathcal{L} \leftarrow \frac{1}{|\mathcal{M}|}\sum_{(f,i)\in\mathcal{M}} \min(w(\sigma_{f,i}), w_{\max}) \cdot (\hat{W}_{f,i} - W_{\text{target},f,i})^2$
  // Step 9: Update parameters
  $\theta \leftarrow \theta - \eta\nabla_\theta\mathcal{L}$
  $\gamma \leftarrow \gamma - \eta\nabla_\gamma\mathcal{L}$
  Update EMA: $\theta_{\text{EMA}} \leftarrow \beta\theta_{\text{EMA}} + (1-\beta)\theta$
until converged

Algorithm 5 EADIFF Inference with Heun Sampler (Detailed)

Require: History $\mathbf{X}_{\text{his}} \in \mathbb{R}^{(T-H)\times F}$, trained model F_θ, starting step k_0, number of steps N, prediction length H_{pred}
Ensure: Predicted future $\hat{\mathbf{X}}_{\text{pred}} \in \mathbb{R}^{H\times F}$

// Step 1: Construct initial input
$\hat{\mathbf{X}}_{\text{guess}} \leftarrow \text{LinearExtrapolate}(\mathbf{X}_{\text{his}}, H_{\text{pred}})$
$\tilde{\mathbf{X}} \leftarrow \text{Concat}(\mathbf{X}_{\text{his}}, \hat{\mathbf{X}}_{\text{guess}})$
// Step 2: Transform to wavelet domain
$\mathbf{W}_{\text{init}} \leftarrow \text{WaveletScaler}(\text{DWT}(\text{OriginalScaler}(\tilde{\mathbf{X}})))$
// Step 3: Compute mask and conditioning
$\mathbf{m} \leftarrow \text{ComputeWaveletMask}(T, H_{\text{pred}}, \text{wavelet}, J)$
$\mathbf{c}_{W^{(0)}} \leftarrow \mathbf{W}_{\text{init}}[:, :, : L_0]$
// Step 4: Compute energy-adaptive sigma schedule
$\{k_i\}_{i=0}^{N} \leftarrow \text{Linspace}(k_0, 0, N+1)$
for $i = 0$ to N **do**
  $\boldsymbol{\sigma}_{k_i} \leftarrow \text{AdaptiveSigma}(k_i, \mathbf{W}_{\text{init}})$ $\{\boldsymbol{\sigma} \in \mathbb{R}^{B\times F\times L}\}$
end for
// Step 5: Initialize latents (SDEdit-style)
$\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
$\mathbf{W}_{k_0} \leftarrow (1-\mathbf{m}) \odot \mathbf{W}_{\text{init}} + \mathbf{m} \odot (\mathbf{W}_{\text{init}} + \boldsymbol{\sigma}_{k_0} \odot \boldsymbol{\epsilon})$
// Step 6: Heun solver with replacement
for $i = 0$ to $N-1$ **do**
  // Euler predictor step
  $\hat{\mathbf{W}}_0 \leftarrow F_\theta(\mathbf{W}_{k_i}, \boldsymbol{\sigma}_{k_i}, \mathbf{c}_{W^{(0)}})$
  $\mathbf{d}_i \leftarrow (\mathbf{W}_{k_i} - \hat{\mathbf{W}}_0)/\boldsymbol{\sigma}_{k_i}$
  $\Delta\sigma \leftarrow \boldsymbol{\sigma}_{k_{i+1}} - \boldsymbol{\sigma}_{k_i}$
  $\mathbf{W}'_{k_{i+1}} \leftarrow \mathbf{W}_{k_i} + \mathbf{d}_i \odot \Delta\sigma$
  if $i < N-1$ **then**
    // Heun corrector step
    $\hat{\mathbf{W}}'_0 \leftarrow F_\theta(\mathbf{W}'_{k_{i+1}}, \boldsymbol{\sigma}_{k_{i+1}}, \mathbf{c}_{W^{(0)}})$
    $\mathbf{d}'_{i+1} \leftarrow (\mathbf{W}'_{k_{i+1}} - \hat{\mathbf{W}}'_0)/\boldsymbol{\sigma}_{k_{i+1}}$
    $\bar{\mathbf{d}} \leftarrow (\mathbf{d}_i + \mathbf{d}'_{i+1})/2$
    $\mathbf{W}^{\text{raw}}_{k_{i+1}} \leftarrow \mathbf{W}_{k_i} + \bar{\mathbf{d}} \odot \Delta\sigma$
  else
    $\mathbf{W}^{\text{raw}}_{k_{i+1}} \leftarrow \mathbf{W}'_{k_{i+1}}$
  end if
  // Replacement: enforce history constraint
  $\mathbf{W}_{k_{i+1}} \leftarrow (1-\mathbf{m}) \odot \mathbf{W}_{\text{init}} + \mathbf{m} \odot \mathbf{W}^{\text{raw}}_{k_{i+1}}$
end for
// Step 7: Inverse transform
$\hat{\mathbf{X}} \leftarrow \text{OriginalScaler}^{-1}(\text{IDWT}(\text{WaveletScaler}^{-1}(\mathbf{W}_{k_N})))$
return $\hat{\mathbf{X}}_{\text{pred}} \leftarrow \hat{\mathbf{X}}[:, T-H:T, :]$