# TimeDiff: Leveraging Differential Domain Representations for Long Time Series Forecasting

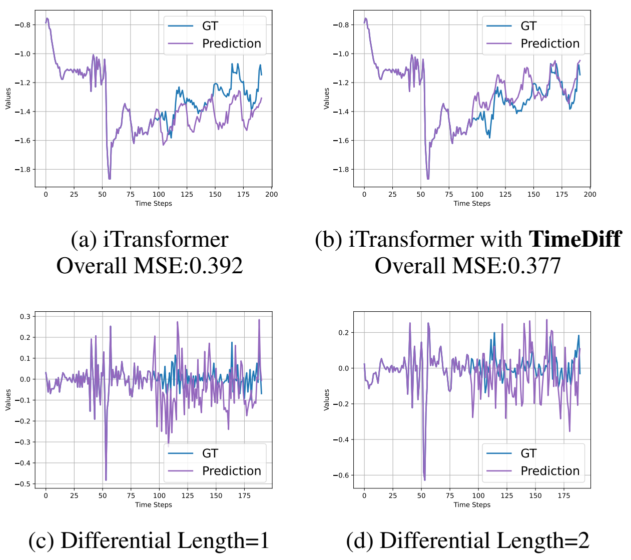
**Anonymous Authors**[1]

## Abstract

Time series forecasting poses significant challenges due to the non-stationarity and multi-scale temporal dependencies inherent in the data. Existing approaches primarily focus on modeling in the time domain and are often evaluated based on their ability to capture overall trends, while paying less attention to fine-grained changes in the data. However, in many real-world scenarios, the detailed variations in time series (i.e., its differences) are critical for decision-making. To address this gap, we propose *TimeDiff*, a novel framework for long time series forecasting that enhances predictive accuracy by modeling in the differential domain. By representing time series data as multi-level differences, TimeDiff captures both short-term variations and long-term trends across multiple temporal scales. Extensive experiments demonstrate its state-of-the-art performance: in **96.88%** test scenarios, TimeDiff-enhanced model surpasses its baseline using identical hyperparameters and ranking first in 20 different settings while the second-best approach only achieves the top spot in 6. Our work highlights the potential of differential domain modeling as a powerful paradigm for advancing time series forecasting. The code is available at: https://anonymous.4open.science/r/TimeDiff-EFE3/

## 1. Introduction

Time series forecasting underpins a wide range of real-world applications, including traffic flow (Lippi et al., 2013; Zheng & Huang, 2020), weather prediction (Karevan & Suykens, 2020; Hewage et al., 2020), and financial trading (Sezer & Ozbayoglu, 2018; Taylor, 2008). Despite their varied nature, these domains share a common thread: the *temporal dynamics* within the data. Typically, most approaches emphasize modeling time series purely in the time domain (Liu et al., 2024; Nie et al., 2023; Wilson, 2017), seeking to capture overarching trends over historical windows. However, a crucial yet often overlooked aspect lies in the *changes* between successive timestamps, which can reveal fine-grained variations essential for accurate and robust predictions.



(a) iTransformer Overall MSE:0.392

(b) iTransformer with **TimeDiff** Overall MSE:0.377

(c) Differential Length=1

(d) Differential Length=2

*Figure 1.* Case study visualization of iTransformer and TimeDiff. (a) shows the baseline iTransformer predictions. (b) illustrates the improved results after incorporating TimeDiff. (c)–(d) depict predictions in the differential domain with differencing lengths of 1 and 2, respectively.

Classical statistical approaches, particularly ARIMA (Box et al., 1974; Contreras et al., 2003; Box et al., 2015), have highlighted the role of *differencing* in handling non-stationarity and reducing trends. Although differencing has demonstrated effectiveness in stabilizing time series, deep learning methods often treat it as a separate preprocessing procedure (Sedaghat & Mahabal, 2018; Zhang & Shi, 2020), missing opportunities to seamlessly integrate differential insights into model optimization. This raises a key question: *can we unify time-domain modeling and differencing within a single framework to better handle the multi-scale dynam-*

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work.

*ics of real-world time series?*

Figure 1 provides a motivating example. Subfigures (a) and (b) compare iTransformer's baseline predictions with those obtained by incorporating our proposed **TimeDiff**, showing that explicitly modeling the differential domain mitigates errors in local transitions and long-range dependencies. Moreover, subfigures (c)–(d) illustrate how differencing lengths $d = 1$ and $d = 2$ capture subtle temporal variations, underscoring the power of multi-scale differencing to recover fine-grained structures.

In this paper, we propose a novel framework, *TimeDiff*, that tightly integrates differencing into the forecasting process. By learning to predict time series in the differential domain, our method encapsulates both global trends and local fluctuations more effectively. This perspective addresses non-stationarity, reduces error accumulation over extended horizons, and offers improved representations of temporal dynamics. In summary, our contributions are threefold:

- We introduce a novel perspective for time series forecasting by jointly modeling in both the original and differential domains, thereby capturing multi-scale temporal information.
- We propose *TimeDiff*, a unified framework that integrates multi-level differencing within deep learning architectures, avoiding treating differencing as a mere preprocessing step.
- Through extensive experiments on benchmark datasets, we demonstrate that TimeDiff achieves significant improvements over state-of-the-art methods in long-term forecasting tasks.

## 2. Related works

**Deep Learning-based Time Series Forecasting** Deep learning models for time series forecasting typically fall into four main architectural categories. **RNN-based** methods exploit recurrent mechanisms to learn sequential dependencies (Lai et al., 2018; Salinas et al., 2020), whereas **CNN-based** methods capture local patterns via temporal convolutions (Bai et al., 2018; Liu et al., 2022a). **Transformer-based** architectures (Vaswani, 2017) model global context using self-attention (Cirstea et al., 2022; Liu et al., 2022b; Kitaev et al., 2020; Li et al., 2019) and often employ strategies like sparse attention (Informer (Zhou et al., 2021)) or series decomposition (Autoformer (Wu et al., 2021)). **Linear-based** approaches, such as DLinear (Zeng et al., 2023), have recently gained attention by using linear layers and decomposition operations to achieve competitive results with lower complexity. Further advances include pattern disentanglement (TimesNet (Wu et al., 2023)) and architectural modifications (PatchTST (Nie et al., 2023), iTransformer (Liu et al., 2024)). However, these methods predominantly treat time series as static observations in the time domain, focusing on coarse-grained trends while neglecting evolving, fine-scale dynamics.

**Differencing Methods** Differencing is a long-standing technique in statistical time series analysis, epitomized by the ARIMA family of models (Box et al., 1974; 2015; Contreras et al., 2003), where differencing serves to transform a non-stationary series into a stationary form. This approach has proven effective across various applications, such as weather forecasting (Tektaş, 2010; Rahman et al., 2013) and energy demand prediction (Ediger & Akar, 2007; Erdogdu, 2007). However, In previous deep learning frameworks, differencing is often relegated to a preprocessing role (Sedaghat & Mahabal, 2018; Zhang & Shi, 2020), or just a technique into the attention mechanism(Difformer (Li et al., 2023), Diff Transformer (Ye et al., 2024)), thus limiting its potential to inform model training more directly.

In contrast, our proposed *TimeDiff* framework embeds differencing within the prediction process itself. By bridging classical differencing concepts and modern deep learning, TimeDiff is designed to address long-term dependencies and non-stationary trends in an end-to-end manner. This integration offers a unified treatment of both global patterns and local changes, demonstrating that classical ideas can still inspire effective solutions in the deep learning era.

## 3. Methodology

### 3.1. Preliminaries

**Long-term Time Series Forecasting** The task of LTSF involves predicting future values over an extended horizon using previously observed multivariate time series (MTS) data. Formally, it can be expressed as

$$\hat{\mathbf{Y}}_{t+1:t+H} = f_\theta(\mathbf{X}_{t-L+1:t}),$$

where $\mathbf{X}_{t-L+1:t} \in \mathbb{R}^{L\times C}$ represents the historical observation window of length $L$ with $C$ distinct features (or channels), and $\hat{\mathbf{Y}}_{t+1:t+H} \in \mathbb{R}^{H\times C}$ denotes the forecasted values over the horizon $H$. As $H$ grows, the model must capture a broader range of temporal dependencies, from short-term fluctuations to long-term trends, which often leads to more complex architectures and a higher risk of error accumulation.

**Temporal Differencing** The core differencing operation computes variations between temporally offset sequences. Given input $\mathbf{X} \in \mathbb{R}^{L\times C}$, the $d$-lag difference is defined as:

$$\Delta_d \mathbf{X} = \mathbf{X}_{d+1:L} - \mathbf{X}_{1:L-d} \qquad (1)$$

where $d \geq 1$ controls the temporal offset.

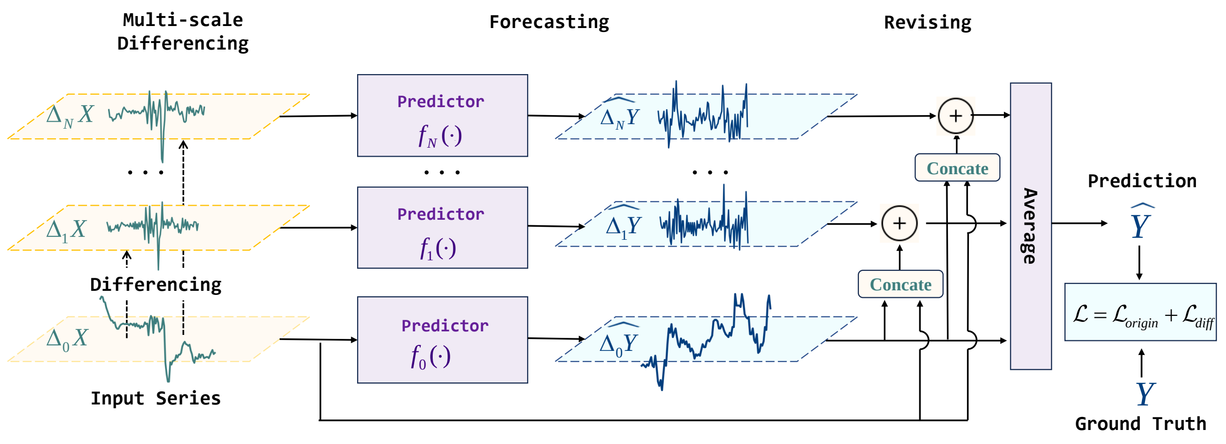


*Figure 2.* TimeDiff framework.

## 3.2. TimeDiff Framework

TimeDiff addresses the challenges of non-stationarity and error accumulation in long-term time series forecasting by learning temporal patterns at multiple differencing scales. As illustrated in Figure 2, our approach proceeds in four main phases: *multiscale differencing*, *parallel differential forecasting*, *consensus fusion* and *multiscale loss*. The complete algorithmic details can be found in Appendix B.1.

**Multiscale Differencing** Define a geometric differencing hierarchy with levels $k \in \{0, ..., N\}$ where $d_k = 2^{k-1}$. The differential operator $\Delta_k$ transforms the input as:

$$\Delta_k \mathbf{X} = \begin{cases} \mathbf{X}, & k = 0 \\ \mathbf{X}_{d_k+1:L} - \mathbf{X}_{1:L-d_k}, & k \geq 1 \end{cases} \tag{2}$$

This generates complementary representations $\{\Delta_0 \mathbf{X}, ..., \Delta_N \mathbf{X}\}$ capturing temporal patterns at exponentially increasing scales.

**Parallel Differential Forecasting** Each differential level employs a dedicated predictor $f_k$ with shared architecture but independent parameters. The prediction at level $k$ is:

$$\widehat{\Delta_k \mathbf{Y}} = f_k(\Delta_k \mathbf{X}) \in \mathbb{R}^{H \times C} \tag{3}$$

**Consensus Fusion** Final prediction integrates all levels through convex combination:

$$\hat{\mathbf{Y}} = \frac{1}{N+1} \sum_{k=0}^{N} \mathcal{R}_k(\widehat{\Delta_k \mathbf{Y}}) \tag{4}$$

The temporal alignment operator $\mathcal{R}_k$ ensures continuity between historical observations and predictions through concatenation:

$$\mathcal{R}_k(\mathbf{Z}) = \begin{cases} \mathbf{Z} & k = 0 \\ \mathbf{Z} + \left[ \mathbf{X}_{L-d_k+1:L} \oplus \widehat{\Delta_0 \mathbf{Y}}_{1:H-d_k} \right] & k \geq 1 \end{cases} \tag{5}$$

$\mathbf{X}_{L-d_k+1:L}$ is the last $d_k$ timesteps from historical data ($L$-length input) and $\widehat{\Delta_0 \mathbf{Y}}_{1:H-d_k}$ is the first $H - d_k$ timesteps from base prediction ($k = 0$ output). $\oplus$ means temporal concatenation along the time dimension.

**Loss Function** The composite loss function jointly optimizes multiscale differential consistency and global prediction accuracy through:

$$\begin{aligned} \mathcal{L} &= \mathcal{L}_{origin} + \mathcal{L}_{diff} \\ &= \|\hat{\mathbf{Y}} - \mathbf{Y}\|^2 + \lambda \sum_{k=1}^{N} \|\Delta_k \hat{\mathbf{Y}} - \Delta_k \mathbf{Y}\|^2 \end{aligned} \tag{6}$$

$\Delta_k \hat{\mathbf{Y}} \triangleq \hat{\mathbf{Y}}_{d_k+1:H} - \hat{\mathbf{Y}}_{1:H-d_k}$ computes the $k$-th order temporal difference of predictions, with differencing horizons $d_k = 2^{k-1}$. The scaling factor $\lambda = \frac{1}{N}$ uniformly weights differential regularization terms across $N$ scales, while the second term maintains direct supervision on the final prediction. This formulation ensures balanced optimization between hierarchical temporal pattern preservation and end-to-end forecasting accuracy.

## 3.3. Theoretical Analysis

In this section, we provide a concise theoretical analysis of *TimeDiff*. We highlight how its multi-level differencing mechanism reduces variance, mitigates non-stationarity, and bounds overall forecasting error. The complete proofs are deferred to Appendix D.

**Variance Reduction Guarantee** Our consensus fusion strategy achieves provable error suppression through multiscale ensemble. Let $\hat{\mathbf{Y}}_k$ denote the prediction from $k$-th differential level, we establish:

**Theorem 3.1** (Variance Upper Bound)**.** *For unbiased predictors $\{\hat{\mathbf{Y}}_k\}_{k=0}^N$ with pairwise correlations $\rho_{ij}$, the ensemble variance satisfies:*

$$\mathbb{V}\left[\tfrac{1}{N+1}\textstyle\sum_{k=0}^N \hat{\mathbf{Y}}_k\right] \leq \frac{1+\rho_{avg}N}{N+1}\max_k \mathbb{V}[\hat{\mathbf{Y}}_k] \quad (7)$$

*where $\rho_{avg} = \frac{2}{N(N+1)}\sum_{i<j}\rho_{ij}$.*

Theorem 3.1 justifies our parallel architecture: the $\mathcal{O}(1/N)$ scaling law suggests increasing differential levels directly reduces error variance. This guided our geometric level spacing $d_k = 2^k$ to balance computational cost and error suppression.

**Non-stationarity Mitigation** Our differential operators induce stationarity through trend elimination:

**Lemma 3.2** (Polynomial Trend Removal)**.** *For any $m$-degree polynomial trend $x_t = \sum_{n=0}^m a_n t^n$, the $k$-th order difference satisfies:*

$$\Delta_k^{(m+1)} x_t = 0 \quad \text{when } k > 0 \quad (8)$$

*where $\Delta_k^{(r)}$ denotes $r$-times repeated differencing.*

Lemma 3.2 explains the hierarchical stationarization in TimeDiff. Real-world trends (e.g., daily/weekly) are eliminated at appropriate levels, guiding our level selection $N \leq \lfloor \log_2 L \rfloor - 1$ to capture typical periodicities.

**Error Composition Principle** The prediction error of TimeDiff is fundamentally controlled by the differential consistency between final predictions and ground truth. Our main theoretical result establishes:

**Proposition 3.3** (Differential Consistency Bound)**.** *Let $\Delta_k\hat{\mathbf{Y}} = \hat{\mathbf{Y}}_{d_k+1:H} - \hat{\mathbf{Y}}_{1:H-d_k}$ denote the $k$-th order difference of final predictions. The total prediction error satisfies:*

$$\|\hat{\mathbf{Y}} - \mathbf{Y}\| \leq \frac{1}{\sqrt{N}}\left(\sum_{k=1}^N \|\Delta_k\hat{\mathbf{Y}} - \Delta_k\mathbf{Y}\|^2\right)^{1/2} + \mathcal{E}_{base} \quad (9)$$

*where $\mathcal{E}_{base} = \|\hat{\mathbf{Y}} - \mathbf{Y}\|$ represents base prediction accuracy.*

This bound reveals two fundamental error sources: 1. **Multiscale Differential Errors**: Aggregated inconsistencies in temporal patterns across $N$ scales 2. **Base Prediction Error**: Direct point-wise forecasting inaccuracies

The geometric scaling $d_k = 2^{k-1}$ ensures the first term decays as $\mathcal{O}(1/\sqrt{N})$, theoretically justifying our design choices in Section 3.2.

## 4. Experiments

In this section, We conduct a series of experiments to answer the following questions to demonstrate the efficacy of *TimeDiff*:

1. **Performance**: *Does TimeDiff actually improve long-term forecasting?* (Section 4.1)
2. **Mechanism**: *Which components of TimeDiff contribute to its effectiveness?* (Section 4.2)
3. **Generality**: *Can TimeDiff adapt to other forecasting models beyond iTransformer?* (Section 4.3)
4. **Sensitivity**: *How does the differencing level $N$ and look-back window $L$ affect performance?* (Section 4.4)
5. **Robustness**: *Does TimeDiff interact well with other techniques (e.g., instance normalization)?* (Section 4.5)

**Datasets.** We conduct our experiments on 8 publicly available datasets of real-world time series, commonly used for long-term forecasting (Wu et al., 2021; Chen et al., 2023; Nie et al., 2023; Zeng et al., 2023): the 4 Electricity Transformer Temperature datasets ETTh1, ETTh2, ETTm1 and ETTm2 (Zhou et al., 2021), Electricity (UCI, n.d.), Exchange (Lai et al., 2018), Traffic (California Department of Transportation, n.d.), and Weather (Max Planck Institute, n.d.) datasets. For fairness, the look-back window size $L$ is set to 96, and prediction horizons $H \in \{96, 192, 336, 720\}$, A more detailed description of the datasets and time series preparation can be found in Appendix B.2.

**Evaluation metric.** Following previous works , we use Mean Squared Error (MSE) and Mean Absolute Error (MAE) as the core metrics to compare performance.

**Baselines** Our baselines include various established models, which can be grouped into three categories: (1) Transformer-based methods: Autoformer (Wu et al., 2021), FEDformer (Zhou et al., 2022), Crossformer (Zhang & Yan, 2023), and PatchTST (Nie et al., 2023). (2) Linear-based methods: DLinear (Zeng et al., 2023); (3) CNN-based method: TimesNet (Wu et al., 2023).

**Environments** All experiments in this paper were implemented using PyTorch (Paszke et al., 2019), trained using the Adam (Kingma & Ba, 2015) optimizer, and executed on a single NVIDIA A100 GPU with 40 GB memory.

## 4.1. Performance — Does TimeDiff actually improve long-term forecasting?

We adopt iTransformer (Liu et al., 2024) as the predictor $f$ and enhance its performance using our *TimeDiff* framework. To ensure a fair comparison, **all hyperparameters are kept consistent with the official implementation**, including model-specific parameters such as $d_{model}$ and the number of encoder layers, as well as training-related settings such as batch size and learning rate. Detailed configurations can be found in Appendix B.3.

Overall, TimeDiff significantly enhances the performance of iTransformer. As shown in Table 2, TimeDiff surpasses its backbone model in **96.88%** test scenarios, achieving the best results across **20** different experimental conditions. Notably, TimeDiff not only improves iTransformer but also enables it to surpass models that originally outperformed iTransformer on certain datasets. Specifically, TimeDiff achieves **7** state-of-the-art results that were previously unattainable by iTransformer. This demonstrates that the improvements introduced by TimeDiff extend beyond the capabilities of architectural design alone, underscoring the pivotal role of differential domain modeling in advancing long time series forecasting.

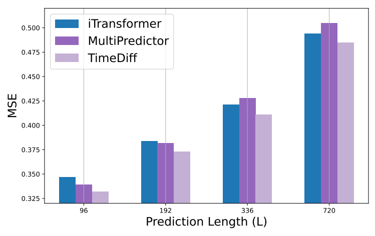
Moreover, differential domain modeling does not increase the standard deviation across 5 runs; in some cases, it even results in a lower standard deviation compared to iTransformer itself. This demonstrates the robustness of TimeDiff and its ability to produce consistent and reliable results, further validating the effectiveness of modeling in the differential domain for long time series forecasting. The MAE results can be found in Appendix C.1, which also demonstrate the effectiveness of our approach.

## 4.2. Mechanism — Which components of TimeDiff contribute to its effectiveness?

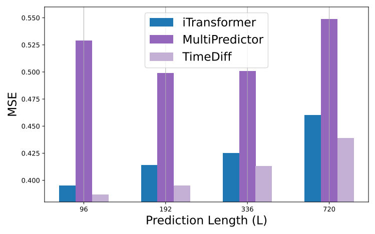
*Table 1.* Ablation Study on Diff Prediction and Diff Loss (MSE Scores)

| Diff Pred | Diff Loss | ETTm1 | | | | Traffic | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | 96 | 192 | 336 | 720 | 96 | 192 | 336 | 720 |
| × | × | 0.347 | 0.384 | 0.420 | 0.494 | 0.395 | 0.414 | 0.425 | 0.460 |
| ✓ | × | 0.332 | 0.376 | 0.413 | 0.492 | 0.472 | 0.418 | 0.423 | 0.457 |
| × | ✓ | 0.336 | 0.381 | 0.415 | **0.483** | 0.391 | 0.412 | 0.424 | 0.459 |
| ✓ | ✓ | **0.332** | **0.373** | **0.411** | 0.485 | **0.387** | **0.396** | **0.413** | **0.440** |

**Diff Prediction & Loss** Two key components of the TimeDiff framework are differencing domain prediction and the corresponding multi-level differencing loss. As shown in Table 1, both designs generally have a positive impact on the results. On one hand, the constraint imposed by the differencing loss enhances the accuracy of modeling in the differencing domain. For instance, in the Traffic dataset, merely adding the Predictor without utilizing the differencing constraint leads to a negative impact on performance. On the other hand, using the predictor for differencing domain modeling, rather than relying solely on the loss function, strengthens the representation capacity of the model. The MAE results can be found in Appendix C.4.



(a) ETTm1 Dataset



(b) Traffic Dataset

*Figure 3.* Comparison of Models on ETTm1 and Traffic Datasets. Each chart illustrates the performance of models with varying prediction lengths.

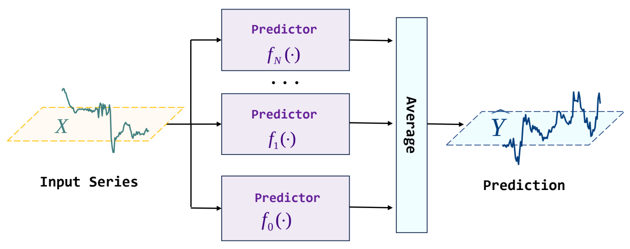


*Figure 4.* MultiPredictor framework.

**The effect of additional parameters** While the TimeDiff framework significantly improves performance by leveraging predictors to learn information from the differential domain, it also introduces additional parameters. This naturally raises an important question: Are the improvements achieved by TimeDiff solely due to the increase in parameter count? To investigate this, we designed a simple baseline method, as illustrated in Figure 4. In this approach, the original time-domain sequence is modeled by multiple predictors independently, and their outputs are averaged to

*Table 2.* Performance comparison between our model (**TimeDiff**) and baselines for long-term forecasting with different horizons $H$. We display the average test MSE with standard deviation obtained on 5 runs with different random seeds. **Best** results are in bold, <u>second best</u> are underlined.

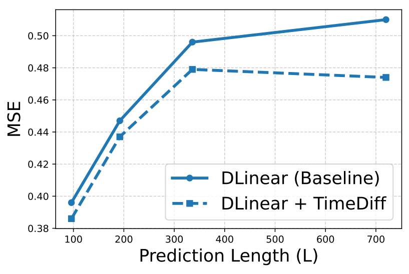
| Dataset | $H$ | **TimeDiff Ours** | iTransformer (2024) | PatchTST (2023) | Crossformer (2023) | TimesNet (2023) | DLinear (2023) | FEDformer (2022) | Autoformer (2021) |
|---|---|---|---|---|---|---|---|---|---|
| ETTh1 | 96 | $0.384_{\pm 0.001}$ | $0.393_{\pm 0.001}$ | $\underline{0.382}$ | 0.469 | 0.407 | 0.396 | **0.376** | 0.477 |
| | 192 | $\underline{0.437}_{\pm 0.001}$ | $0.447_{\pm 0.001}$ | **0.429** | 0.504 | 0.465 | 0.447 | 0.439 | 0.470 |
| | 336 | $0.479_{\pm 0.001}$ | $0.488_{\pm 0.002}$ | $\underline{0.471}$ | 0.635 | 0.497 | 0.496 | **0.453** | 0.528 |
| | 720 | $\underline{0.497}_{\pm 0.003}$ | $0.512_{\pm 0.005}$ | 0.508 | 0.728 | 0.518 | 0.510 | 0.506 | **0.496** |
| ETTh2 | 96 | $\mathbf{0.291}_{\pm 0.001}$ | $\underline{0.300}_{\pm 0.001}$ | 0.309 | 0.900 | 0.345 | 0.347 | 0.357 | 0.392 |
| | 192 | $\mathbf{0.374}_{\pm 0.002}$ | $\underline{0.379}_{\pm 0.000}$ | 0.386 | 1.065 | 0.423 | 0.463 | 0.438 | 0.463 |
| | 336 | $\mathbf{0.421}_{\pm 0.002}$ | $\underline{0.422}_{\pm 0.001}$ | 0.439 | 3.046 | 0.437 | 0.573 | 0.466 | 0.491 |
| | 720 | $\mathbf{0.431}_{\pm 0.005}$ | $\underline{0.434}_{\pm 0.006}$ | 0.450 | 3.677 | 0.457 | 0.839 | 0.457 | 0.494 |
| ETTm1 | 96 | $0.332_{\pm 0.002}$ | $0.348_{\pm 0.001}$ | **0.324** | 0.369 | $\underline{0.328}$ | 0.345 | 0.366 | 0.476 |
| | 192 | $\underline{0.373}_{\pm 0.001}$ | $0.387_{\pm 0.001}$ | **0.370** | 0.442 | 0.411 | 0.382 | 0.426 | 0.536 |
| | 336 | $\underline{0.411}_{\pm 0.001}$ | $0.424_{\pm 0.001}$ | **0.400** | 0.558 | 0.421 | 0.415 | 0.448 | 0.749 |
| | 720 | $0.483_{\pm 0.002}$ | $0.493_{\pm 0.002}$ | **0.463** | 0.741 | 0.496 | $\underline{0.472}$ | 0.506 | 0.566 |
| ETTm2 | 96 | $\mathbf{0.178}_{\pm 0.000}$ | $0.185_{\pm 0.001}$ | $\underline{0.180}$ | 0.262 | 0.185 | 0.194 | 0.191 | 0.266 |
| | 192 | $\mathbf{0.244}_{\pm 0.002}$ | $0.252_{\pm 0.002}$ | $\underline{0.249}$ | 0.877 | 0.256 | 0.284 | 0.263 | 0.282 |
| | 336 | $\mathbf{0.305}_{\pm 0.002}$ | $0.315_{\pm 0.001}$ | 0.318 | 1.612 | $\underline{0.313}$ | 0.373 | 0.321 | 0.338 |
| | 720 | $\mathbf{0.404}_{\pm 0.001}$ | $\underline{0.412}_{\pm 0.002}$ | 0.422 | 4.090 | 0.418 | 0.538 | 0.440 | 0.430 |
| Electricity | 96 | $\mathbf{0.143}_{\pm 0.003}$ | $\underline{0.148}_{\pm 0.000}$ | 0.180 | 0.149 | 0.169 | 0.211 | 0.195 | 0.204 |
| | 192 | $\mathbf{0.161}_{\pm 0.002}$ | $0.164_{\pm 0.001}$ | 0.188 | $\underline{0.163}$ | 0.183 | 0.210 | 0.206 | 0.280 |
| | 336 | $\mathbf{0.174}_{\pm 0.003}$ | $\underline{0.179}_{\pm 0.000}$ | 0.204 | 0.186 | 0.199 | 0.223 | 0.219 | 0.244 |
| | 720 | $\mathbf{0.205}_{\pm 0.003}$ | $\underline{0.211}_{\pm 0.002}$ | 0.245 | 0.246 | 0.220 | 0.258 | 0.261 | 0.281 |
| Exchange | 96 | $\mathbf{0.084}_{\pm 0.001}$ | $0.087_{\pm 0.001}$ | $\underline{0.086}$ | 0.273 | 0.116 | 0.094 | 0.159 | 0.160 |
| | 192 | $\mathbf{0.177}_{\pm 0.001}$ | $\underline{0.179}_{\pm 0.000}$ | 0.183 | 0.568 | 0.213 | 0.194 | 0.248 | 0.273 |
| | 336 | $\underline{0.333}_{\pm 0.004}$ | $0.335_{\pm 0.002}$ | **0.310** | 1.189 | 0.355 | 0.344 | 0.385 | 0.536 |
| | 720 | $0.853_{\pm 0.005}$ | $\underline{0.852}_{\pm 0.004}$ | 0.973 | 1.597 | 1.028 | **0.796** | 1.176 | 1.187 |
| Traffic | 96 | $\mathbf{0.389}_{\pm 0.001}$ | $\underline{0.393}_{\pm 0.001}$ | 0.462 | 0.514 | 0.584 | 0.710 | 0.597 | 0.619 |
| | 192 | $\mathbf{0.401}_{\pm 0.004}$ | $\underline{0.412}_{\pm 0.001}$ | 0.466 | 0.540 | 0.616 | 0.662 | 0.601 | 0.648 |
| | 336 | $\mathbf{0.414}_{\pm 0.001}$ | $\underline{0.424}_{\pm 0.001}$ | 0.484 | 0.583 | 0.624 | 0.669 | 0.638 | 0.619 |
| | 720 | $\mathbf{0.443}_{\pm 0.003}$ | $\underline{0.458}_{\pm 0.001}$ | 0.517 | 0.589 | 0.657 | 0.709 | 0.648 | 0.690 |
| Weather | 96 | $\mathbf{0.170}_{\pm 0.001}$ | $0.175_{\pm 0.001}$ | 0.173 | 0.172 | $\underline{0.171}$ | 0.198 | 0.224 | 0.271 |
| | 192 | $\mathbf{0.219}_{\pm 0.003}$ | $0.225_{\pm 0.001}$ | $\underline{0.221}$ | 0.236 | 0.234 | 0.236 | 0.294 | 0.314 |
| | 336 | $\underline{0.277}_{\pm 0.003}$ | $0.281_{\pm 0.001}$ | 0.282 | **0.276** | 0.284 | 0.283 | 0.334 | 0.353 |
| | 720 | $\underline{0.355}_{\pm 0.000}$ | $0.359_{\pm 0.001}$ | 0.356 | 0.370 | 0.357 | **0.346** | 0.422 | 0.422 |
| 1st Count | | **20** | 0 | $\underline{6}$ | 1 | 0 | 2 | 2 | 1 |

produce the final forecast. We refer to this baseline method as **MultiPredictor**, as shown in Figure 4. The key difference between MultiPredictor and TimeDiff lies in the fact that MultiPredictor does not utilize the differencing domain for modeling, only using the original time-domain sequence, and esembles multiple predictors to enhance the representation capacity.

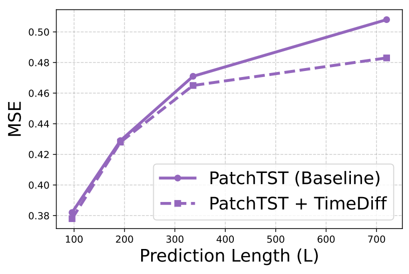
As illustrated in Figure 3, on the ETTm1 dataset, MultiPredictor achieves better performance for shorter prediction horizons (96, 192), but exhibits a negative impact on longer sequence predictions (336, 720). This phenomenon is even more pronounced on the Traffic dataset, where MultiPredictor significantly increases the error across all prediction horizons, leading to a notable decline in prediction accuracy. On the other hand, TimeDiff consistently outperforms the baseline across all settings. This demonstrates that the effectiveness of TimeDiff does not primarily stem from an increase in the number of parameters but rather from its ability to model the differencing domain.

## 4.3. Generality — Can TimeDiff adapt to other forecasting models?

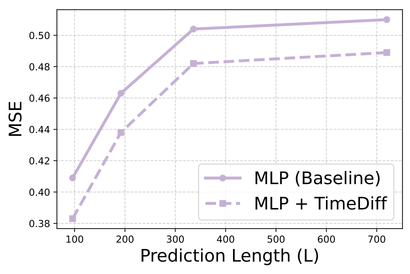
In addition to using iTransformer as the primary backbone in our main experiments, we further evaluated the generalizability of TimeDiff by integrating it with other models as predictors, including DLinear, PatchTST, and a simple



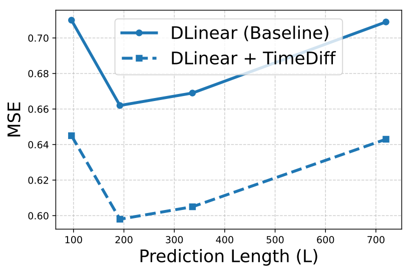
(a) DLinear - ETTh1



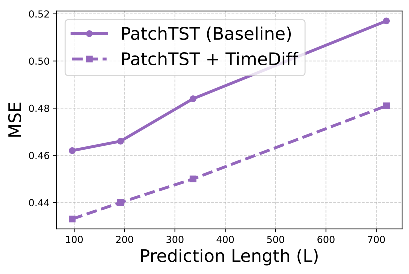
(b) PatchTST - ETTh1



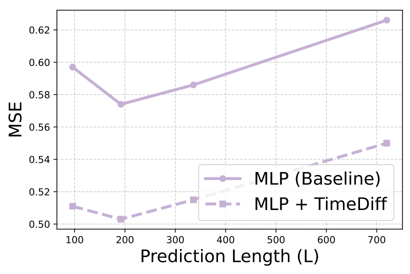
(c) MLP - ETTh1



(d) DLinear - Traffic



(e) PatchTST - Traffic



(f) MLP - Traffic

*Figure 5.* Comparison of different models with and without TimeDiff on ETTh1 and Traffic datasets. Baseline models use solid lines, and TimeDiff-enhanced models use dashed lines.

MLP model. The motivation for selecting DLinear was to assess the compatibility of linear-based models and trend decomposition techniques within the differential domain. For PatchTST, we aimed to examine whether the widely adopted patch-based temporal representation remains effective when applied to differenced sequences. The inclusion of MLP was based on the growing prominence of MLP-based models alongside Transformer-based approaches in recent time series forecasting research.

As illustrated in Figure 5, TimeDiff consistently outperforms the baseline across all configurations. Even on the ETTh1 dataset, where the performance gains are generally more limited, TimeDiff still achieves notable improvements, particularly for long sequences ($L = 720$). On the Traffic dataset, the performance enhancement is even more pronounced, significantly surpassing the baseline across all settings. These results further validate the effectiveness of TimeDiff across diverse model architectures, demonstrating its versatility and robustness in various time series forecasting scenarios. A more detailed comparison of the results can be found in Appendix C.2.

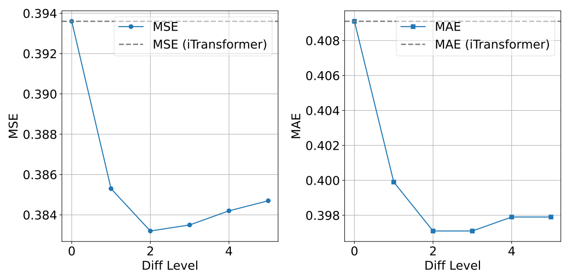
## 4.4. Sensitivity — How does $N$ and $L$ affect performance?

**Impact of the diff level $N$** An important parameter of the TimeDiff method is the Diff level $N$, which determines the granularity of differencing modeling, or in other words, the temporal interval of changes being modeled. The impact of the Diff level is illustrated in Figure 6. As shown in the figure, the prediction performance generally exhibits a trend of initially decreasing and then increasing with the Diff level. This aligns with our expectations: when $N$ becomes excessively large, the input sequence to the predictor becomes shorter due to the differencing process, which adversely affects the modeling in the differencing domain and, consequently, the overall prediction performance.

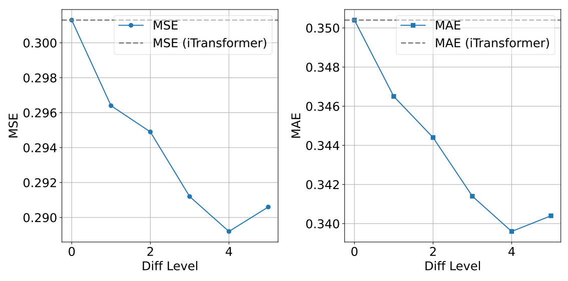
**Impact of Look-back window $L$** Existing methods often select different look-back window lengths ($L$) to achieve optimal performance. In our main experiments, we set $L = 96$ to align with the settings used in iTransformer. Here, we further evaluate the performance of TimeDiff under larger $L$ values, as shown in Figure 7. Although the predictive performance of TimeDiff on the ETTh1 dataset deteriorates as $L$ increases due to the limitations of the backbone, it eventually achieves the best result when $L = 720$. Moreover, under all settings, TimeDiff consistently outperforms the baseline model, demonstrating its robustness and effectiveness across a wide range of look-back window lengths.

## 4.5. Robustness — Does TimeDiff interact well with other techniques?

Time series data often exhibit distributional shifts between training and testing datasets, making it challenging to maintain consistent predictive performance (Lin et al., 2024b).

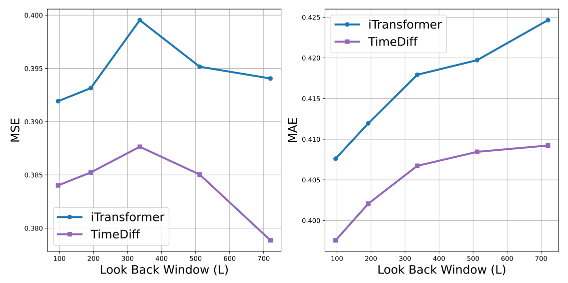


(a) ETTh1 Dataset

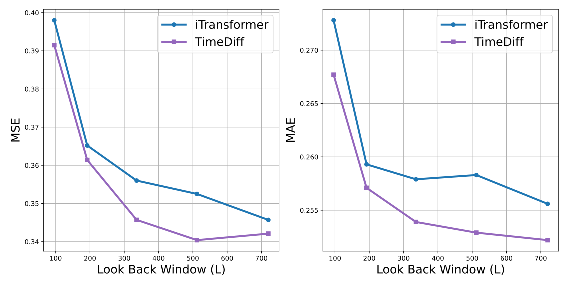


(b) ETTh2 Dataset

*Figure 6.* Comparison of Models on ETTh1 and ETTh2 Datasets. Each chart illustrates the performance of models with varying diff level $N$.



(a) ETTh1 Dataset



(b) Traffic Dataset

*Figure 7.* Comparison of Models on ETTh1 and Traffic Datasets. Each chart illustrates the performance of models with varying look back window $L$.

Recent studies (Kim et al., 2021; Zeng et al., 2023)have demonstrated that applying simple sample normalization strategies between the input and output of models can alleviate this issue by reducing the non-stationarity of time series data. In our baseline model, iTransformer, this normalization strategy is also employed.

Since one of the primary objectives of differencing is to stabilize the sequence and address non-stationarity, we further investigate the individual impact of TimeDiff, as well as its combined effect with the normalization operation, as shown in Table 3. The results indicate that using TimeDiff alone improves performance, while the combination of TimeDiff and normalization achieves the best results. This demonstrates that TimeDiff is compatible with mainstream normalization techniques, enhancing its practical utility and further improving its effectiveness in time series forecasting tasks.

*Table 3.* Ablation of Instance Norm and TimeDiff

| **Instance Norm** | **TimeDiff** | **ETTh1** | | **ETTh2** | | **Electricity** | | **Traffic** | |
|---|---|---|---|---|---|---|---|---|---|
| | | **MSE** | **MAE** | **MSE** | **MAE** | **MSE** | **MAE** | **MSE** | **MAE** |
| × | × | 0.414 | 0.430 | 0.526 | 0.512 | 0.149 | 0.246 | 0.538 | 0.302 |
| × | ✓ | 0.390 | 0.411 | 0.447 | 0.459 | 0.146 | 0.244 | 0.536 | 0.311 |
| ✓ | × | 0.392 | 0.408 | 0.299 | 0.350 | 0.148 | 0.240 | 0.393 | 0.268 |
| ✓ | ✓ | **0.384** | **0.396** | **0.288** | **0.338** | **0.144** | **0.236** | **0.387** | **0.265** |

## 5. Conclusion and Future Work

In this paper, we present TimeDiff, a novel framework for long-term time series forecasting that leverages differential domain modeling to enhance predictive performance. Inspired by classical statistical methods like ARIMA, which emphasize the importance of differencing in handling trends and non-stationarity, TimeDiff demonstrates that these foundational ideas remain highly relevant in the deep learning era. Our primary contribution lies not in proposing a specific model but in introducing a fresh perspective to time series forecasting——one that shifts the focus from direct time-domain modeling to differential domain representations. Extensive experiments demonstrate that TimeDiff consistently surpasses state-of-the-art methods across diverse scenarios, achieving notable gains in accuracy and robustness. While TimeDiff involves additional parameters (controlled by the differencing level $N$), our results confirm that these parameters enable more effective modeling of temporal dynamics compared to naive increases in network complexity. Nevertheless, setting the appropriate differencing levels will introduce additional tuning considerations.

Our work highlights the enduring value of classical ideas and their potential to inspire innovation in modern deep learning frameworks. Looking forward, we plan to extend TimeDiff to address challenges in ultra-long sequences, irregular time series, and multi-scale temporal patterns. We aim to further explore the synergy between classical statistical methods and deep learning, striving to provide new insights into the evolving field of time series forecasting.

## Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none of which we feel must be specifically highlighted here.

## References

Bai, S., Kolter, J. Z., and Koltun, V. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*, 2, 2018.

Box, G. E., Jenkins, G. M., and MacGregor, J. F. Some recent advances in forecasting and control. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 23(2):158–179, 1974.

Box, G. E., Jenkins, G. M., Reinsel, G. C., and Ljung, G. M. *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.

California Department of Transportation. Traffic dataset, n.d. URL https://pems.dot.ca.gov/.

Chen, S.-A., Li, C.-L., Arik, S. O., Yoder, N. C., and Pfister, T. TSMixer: An all-MLP architecture for time series forecasting. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856. URL https://openreview.net/forum?id=wbpxTuXgm0.

Cirstea, R.-G., Guo, C., Yang, B., Kieu, T., Dong, X., and Pan, S. Triformer: Triangular, variable-specific attentions for long sequence multivariate time series forecasting–full version. *arXiv preprint arXiv:2204.13767*, 2022.

Contreras, J., Espinola, R., Nogales, F. J., and Conejo, A. J. Arima models to predict next-day electricity prices. *IEEE transactions on power systems*, 18(3):1014–1020, 2003.

Ediger, V. Ş. and Akar, S. Arima forecasting of primary energy demand by fuel in turkey. *Energy policy*, 35(3): 1701–1708, 2007.

Erdogdu, E. Electricity demand analysis using cointegration and arima modelling: A case study of turkey. *Energy policy*, 35(2):1129–1146, 2007.

Hewage, P., Behera, A., Trovati, M., Pereira, E., Ghahremani, M., Palmieri, F., and Liu, Y. Temporal convolutional neural (tcn) network for an effective weather forecasting using time-series data from the local weather station. *Soft Computing*, 24:16453–16482, 2020.

Karevan, Z. and Suykens, J. A. Transductive lstm for time-series prediction: An application to weather forecasting. *Neural Networks*, 125:1–9, 2020.

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.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 2015.

Kitaev, N., Kaiser, Ł., and Levskaya, A. Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451*, 2020.

Lai, G., Chang, W.-C., Yang, Y., and Liu, H. Modeling long- and short-term temporal patterns with deep neural networks. In *Association for Computing Machinery*, SIGIR '18, pp. 95–104, New York, NY, USA, 2018. ISBN 9781450356572. doi: 10.1145/3209978.3210006. URL https://doi.org/10.1145/3209978.3210006.

Li, B., Cui, W., Zhang, L., Zhu, C., Wang, W., Tsang, I. W., and Zhou, J. T. Difformer: Multi-resolutional differencing transformer with dynamic ranging for time series analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(11):13586–13598, 2023.

Li, S., Jin, X., Xuan, Y., Zhou, X., Chen, W., Wang, Y.-X., and Yan, X. Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting. *Advances in neural information processing systems*, 32, 2019.

Lin, S., Lin, W., Hu, X., Wu, W., Mo, R., and Zhong, H. Cyclenet: enhancing time series forecasting through modeling periodic patterns. *Advances in Neural Information Processing Systems*, 37:106315–106345, 2024a.

Lin, S., Lin, W., Wu, W., Chen, H., and Yang, J. Sparsetsf: Modeling long-term time series forecasting with 1k parameters. *arXiv preprint arXiv:2405.00946*, 2024b.

Lippi, M., Bertini, M., and Frasconi, P. Short-term traffic flow forecasting: An experimental comparison of time-series analysis and supervised learning. *IEEE Transactions on Intelligent Transportation Systems*, 14(2):871–882, 2013.

Liu, M., Zeng, A., Chen, M., Xu, Z., Lai, Q., Ma, L., and Xu, Q. Scinet: time series modeling and forecasting with sample convolution and interaction. *NeurIPS*, 2022a.

Liu, S., Yu, H., Liao, C., Li, J., Lin, W., Liu, A. X., and Dustdar, S. Pyraformer: Low-complexity pyramidal attention for long-range time series modeling and forecasting. In *# PLACEHOLDER_PARENT_METADATA_VALUE#*, 2022b.

Liu, Y., Hu, T., Zhang, H., Wu, H., Wang, S., Ma, L., and Long, M. itransformer: Inverted transformers are effective for time series forecasting. *International Conference on Learning Representations*, 2024.

Max Planck Institute. Weather dataset, n.d. URL https://pems.dot.ca.gov/.

Nie, Y., Nguyen, N. H., Sinthong, P., and Kalagnanam, J. A time series is worth 64 words: Long-term forecasting with transformers. In *International Conference on Learning Representations*, 2023.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. Pytorch: An imperative style, high-performance deep learning library. *NeurIPS*, 2019.

Rahman, M., Islam, A. S., Nadvi, S. Y. M., and Rahman, R. M. Comparative study of anfis and arima model for weather forecasting in dhaka. In *2013 international conference on informatics, electronics and vision (ICIEV)*, pp. 1–6. IEEE, 2013.

Salinas, D., Flunkert, V., Gasthaus, J., and Januschowski, T. Deepar: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting*, 36(3):1181–1191, 2020.

Sedaghat, N. and Mahabal, A. Effective image differencing with convolutional neural networks for real-time transient hunting. *Monthly Notices of the Royal Astronomical Society*, 476(4):5365–5376, 2018.

Sezer, O. B. and Ozbayoglu, A. M. Algorithmic financial trading with deep convolutional neural networks: Time series to image conversion approach. *Applied Soft Computing*, 70:525–538, 2018.

Taylor, S. J. *Modelling financial time series*. world scientific, 2008.

Tektaş, M. Weather forecasting using anfis and arima models. *Environmental Research, Engineering and Management*, 51(1):5–10, 2010.

UCI. Electricity dataset, n.d. URL https://archive.ics.uci.edu/dataset/321/electricityloaddiagrams20112014.

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

Wilson, S. J. Data representation for time series data mining: time domain approaches. *Wiley Interdisciplinary Reviews: Computational Statistics*, 9(1):e1392, 2017.

Wu, H., Xu, J., Wang, J., and Long, M. Autoformer: Decomposition transformers with Auto-Correlation for long-term series forecasting. In *Advances in Neural Information Processing Systems*, 2021.

Wu, H., Hu, T., Liu, Y., Zhou, H., Wang, J., and Long, M. Timesnet: Temporal 2d-variation modeling for general time series analysis. *International Conference on Learning Representations*, 2023.

Ye, T., Dong, L., Xia, Y., Sun, Y., Zhu, Y., Huang, G., and Wei, F. Differential transformer. *arXiv preprint arXiv:2410.05258*, 2024.

Zeng, A., Chen, M., Zhang, L., and Xu, Q. Are transformers effective for time series forecasting? In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2023.

Zhang, M. and Shi, W. A feature difference convolutional neural network-based change detection method. *IEEE Transactions on Geoscience and Remote Sensing*, 58(10): 7232–7246, 2020.

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

Zheng, J. and Huang, M. Traffic flow forecast through time series analysis based on deep learning. *IEEE Access*, 8: 82562–82570, 2020.

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 *The Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Virtual Conference*, volume 35, pp. 11106–11115. AAAI Press, 2021.

Zhou, T., Ma, Z., Wen, Q., Wang, X., Sun, L., and Jin, R. FEDformer: Frequency enhanced decomposed transformer for long-term series forecasting. In *Proc. 39th International Conference on Machine Learning (ICML 2022)*, 2022.

## A. Assessing LLMs' Proficiency in Capturing Inferential Rules

### A.1. Analysis Setup

Following the experimental setup proposed by **?**, we adopt the ULogic framework to systematically evaluate large language models (LLMs) on their ability to capture underlying inferential logic. Specifically, we leverage a curated probing subset comprising 1,104 diverse rules drawn from their rule base. These rules—manually verified by the original authors—span a range of lengths, polarities, and structural patterns, ensuring broad coverage and high quality. The evaluation is framed as a binary entailment classification task, where the model must determine whether a given rule expresses a valid logical entailment. We employ a two-shot Chain-of-Thought (CoT) prompting method (**?**), in which each input includes one correct and one incorrect example to minimize label bias. The model is prompted not only to make a binary judgment but also to justify its reasoning, with an appended instruction such as ”and also explain why.”

To further enhance the reliability of the evaluation, we incorporate the Law of Non-Contradiction (**?**), which posits that statements of the form “If X, then Y” and “If X, then not Y” cannot simultaneously be true. Accordingly, for each original rule, we construct a flipped version by negating its conclusion. A rule is considered correctly classified only if the model affirms the original rule as true and rejects the flipped version as false, as illustrated below (a concrete example is provided in Appendix **??**).

| | |
|---|---|
| *If Premise, then Conclusionoriginal.* | True |
| *If Premise, then Conclusionflipped.* | False |

### A.2. Empirical Analysis

We conduct experiments on FANformer-1B, OLMo-1B, Qwen2.5-1.5B, and Qwen2.5-1.5B-Instruct, with the goal of analyzing their capacity to capture inferential rules.

To better understand the models' behavior, we analyze performance with respect to compositional complexity by grouping rules according to the number of atomic facts contained in their premises. For instance, “Length=3” and “Length=4” refer to rules with premises composed of 3 to 4 atomic propositions, respectively, as illustrated in **??**.

## B. Implementation Details

### B.1. Overall Workflow

The TimeDiff forecasting procedure, formalized in Algorithm 1, operates through three coordinated phases: multiscale differential encoding, parallel prediction, and consensus fusion. First, we construct geometrically spaced differential sequences $\{\Delta_k \mathbf{X}\}_{k=0}^{N}$ where $d_k = 2^k$. This hierarchy captures temporal patterns from instantaneous fluctuations ($k = 0$) to long-range trends ($k = N$).

For each differential level, dedicated predictors $\{f_k\}$ generate scale-specific forecasts $\widehat{\Delta_k \mathbf{Y}}$ in parallel. This architecture prevents error propagation across scales while enabling flexible model selection.

The consensus fusion phase combines predictions through:

- Temporal alignment using historical segments $\mathbf{X}_{L-d_k+1:L-d_k+H}$
- Convex combination of aligned predictions with uniform weighting

### B.2. Dataset Descriptions

We evaluate TimeDiff on eight real-world datasets:
(1) **ETT** (Zhou et al., 2021): Comprises seven operational metrics from electricity transformers recorded between July 2016 and July 2018. The dataset contains four subsets with different temporal resolutions: ETTh1 and ETTh2 (hourly sampling) versus ETTm1 and ETTm2 (15-minute sampling).
(2) **Exchange** (Lai et al., 2018): Provides daily currency exchange rates for eight nations from 1990 to 2016.
(3) **Weather** (Max Planck Institute, n.d.): Captures 21 meteorological variables measured at 10-minute intervals during

**Algorithm 1** TimeDiff Forecasting Algorithm

```
Require: Historical series $\mathbf{X} \in \mathbb{R}^{L \times C}$, forecast horizon $H$, max diff level $N$
Ensure: Forecast $\hat{\mathbf{Y}} \in \mathbb{R}^{H \times C}$
 1: Initialize predictions $\{\widehat{\Delta_k \mathbf{Y}}\}_{k=0}^{N} \leftarrow \emptyset$
 2: for $k = 0$ to $N$ do
 3:     if $k = 0$ then
 4:         $\Delta_0 \mathbf{X} \leftarrow \mathbf{X}$                                    ▷ Original scale
 5:     else
 6:         $d_k \leftarrow 2^{k-1}$                                               ▷ Geometric spacing
 7:         $\Delta_k \mathbf{X} \leftarrow \mathbf{X}_{d_k+1:L} - \mathbf{X}_{1:L-d_k}$                  ▷ Differencing
 8:     end if
 9:     $\widehat{\Delta_k \mathbf{Y}} \leftarrow f_k(\Delta_k \mathbf{X})$                               ▷ Parallel forecasting
10: end for
11: for $k = 0$ to $N$ do
12:     if $k = 0$ then
13:         $\hat{\mathbf{Y}}^{(0)} \leftarrow \widehat{\Delta_0 \mathbf{Y}}$
14:     else
15:         $\hat{\mathbf{Y}}^{(k)} \leftarrow \widehat{\Delta_k \mathbf{Y}} + [\mathbf{X}_{L-d_k+1:L} \oplus \widehat{\Delta_0 \mathbf{Y}}_{1:H-d_k}]$    ▷ Revising
16:     end if
17: end for
18: $\hat{\mathbf{Y}} \leftarrow \frac{1}{N+1} \sum_{k=0}^{N} \hat{\mathbf{Y}}^{(k)}$                            ▷ Consensus integration
19: Compute differential terms: $\Delta_k \hat{\mathbf{Y}} \leftarrow \hat{\mathbf{Y}}_{d_k+1:H} - \hat{\mathbf{Y}}_{1:H-d_k}$ for $k = 1, ..., N$
20: Compute loss: $\mathcal{L} = \frac{1}{N} \sum_{k=1}^{N} \|\Delta_k \hat{\mathbf{Y}} - \Delta_k \mathbf{Y}\|^2 + \|\hat{\mathbf{Y}} - \mathbf{Y}\|^2$
21: return $\hat{\mathbf{Y}}$
```

2020 at the Max Planck Biogeochemistry Institute weather station.
(4) **ECL** (UCI, n.d.): Documents hourly power consumption patterns from 321 industrial clients.
(5) **Traffic** (California Department of Transportation, n.d.): Contains hourly freeway occupancy rates monitored by 862 sensors across the San Francisco Bay Area (2015-2016).

Following iTransformer's experimental protocol (Liu et al., 2024), we implement chronological dataset partitioning (train/validation/test) to eliminate temporal leakage. The model uses fixed-length historical observations ($L = 96$) with variable prediction horizons $H \in \{96, 192, 336, 720\}$. Comprehensive dataset statistics are summarized in Table 4.

*Table 4.* Detailed dataset descriptions. *Channel* denotes the variate number of each dataset. *Dataset Size* denotes the total number of time points in (Train, Validation, Test) split respectively. *Prediction Length* denotes the future time points to be predicted and four prediction settings are included in each dataset. *Frequency* denotes the sampling interval of time points.

| **Dataset** | **Channel $C$** | **Prediction Length $H$** | **Dataset Size** | **Frequency** | **Information** |
|---|---|---|---|---|---|
| ETTh1, ETTh2 | 7 | {96, 192, 336, 720} | (8545, 2881, 2881) | Hourly | Electricity |
| ETTm1, ETTm2 | 7 | {96, 192, 336, 720} | (34465, 11521, 11521) | 15min | Electricity |
| Exchange | 8 | {96, 192, 336, 720} | (5120, 665, 1422) | Daily | Economy |
| ECL | 321 | {96, 192, 336, 720} | (18317, 2633, 5261) | Hourly | Electricity |
| Traffic | 862 | {96, 192, 336, 720} | (12185, 1757, 3509) | Hourly | Transportation |
| Weather | 21 | {96, 192, 336, 720} | (36792, 5271, 10540) | 10min | Weather |

## B.3. Experiment Setup

As elaborated in the main text of the paper, we conducted experiments using identical hyperparameter configurations as those specified for the backbone iTransformer architecture (even though TimeDiff might potentially benefit from more tailored configurations) to ensure a fair and consistent comparison across all models. The complete experimental setup is comprehensively documented in Table 5. For other baseline models and comparative approaches, we faithfully reproduced their reported performance using the widely-recognized Time-Series-Library[1]'s publicly available implementations, which have been extensively validated through numerous time series forecasting studies. This standardized evaluation protocol guarantees the reliability of our comparative analysis while maintaining methodological transparency.

*Table 5.* Experiment configurations for different datasets. $N$ means differencing level, **LR** means learning rate. **Patience** is the early stop epoch

| **Dataset/Configuration** | **Model-related** | | | | **Training-related** | | | |
|---|---|---|---|---|---|---|---|---|
| | $d_{model}$ | $d_{ff}$ | **Layers** | $N$ | **Batch Size** | **LR** | **Epochs** | **Patience** |
| ETT | 128 | 128 | 2 | 4 | 32 | $1 \times 10^{-4}$ | 10 | 3 |
| Exchange | 128 | 128 | 2 | 4 | 32 | $1 \times 10^{-4}$ | 10 | 3 |
| ECL | 512 | 512 | 3 | 4 | 16 | $5 \times 10^{-4}$ | 10 | 3 |
| Traffic | 512 | 512 | 4 | 1 | 16 | $1 \times 10^{-3}$ | 10 | 3 |
| Weather | 512 | 512 | 3 | 4 | 32 | $1 \times 10^{-4}$ | 10 | 3 |

## C. Additional Experiments Results

### C.1. MAE Results

In this section, we provide a comprehensive performance comparison of our proposed model (TimeDiff) against various baselines for long-term forecasting across different prediction horizons H. The results are displayed in Table 6. The findings align with the conclusions drawn in the main paper, highlighting the significant superiority of TimeDiff over its competitors, including recent state-of-the-art models such as iTransformer (Liu et al., 2024), PatchTST (Nie et al., 2023), and Crossformer (Zhang & Yan, 2023). The table presents the average MAE along with standard deviations obtained from five runs with different random seeds.

### C.2. The full results of TimeDiff for other backbones

In addition to using iTransformer as the primary backbone, we conducted comprehensive experiments to evaluate TimeDiff's generalizability across different architectures. Here we include a recently proposed model CycleNet (Lin et al., 2024a), which is a MLP-based model that has shown strong performance in time series forecasting. We also include DLinear (Zeng et al., 2023) and PatchTST (Nie et al., 2023) as additional backbones.

As shonw in Table 7, TimeDiff consistently outperforms the original backbones across nearly all datasets and prediction horizons. It can be seen that when incorporating TimeDiff, the average MSE of PatchTST(**0.343**) is better than original iTransformer(0.344), and iTransformer with TimeDiff(**0.336**) can also be better than the original CycleNet(**0.340**).This demonstrates the versatility and effectiveness of the TimeDiff framework in enhancing the performance of various forecasting models.

### C.3. Impact of Diff Level

In this section, we investigate the impact of the diff level N on the performance of the TimeDiff method. The diff level N is a crucial parameter that determines the granularity of differencing modeling, essentially representing the temporal interval of changes being captured. The results are illustrated in Figure 8, where each subfigure corresponds to a different dataset. As shown in the figure, the prediction performance generally follows a trend of initially decreasing and then increasing with the diff level N. This trend is consistent across all datasets. When N is small, the model can capture finer-grained temporal changes, leading to improved performance. However, as N increases, the input sequence to the predictor becomes shorter due to the differencing process. This reduction in sequence length adversely affects the modeling in the differencing domain and, consequently, degrades the overall prediction performance. Therefore, selecting an appropriate diff level N is essential for achieving optimal performance in the TimeDiff method.

[1]https://github.com/thuml/Time-Series-Library

*Table 6.* Performance comparison between our model (TimeDiff) and baselines for long-term forecasting with different horizons $H$. We display the average test MAE with standard deviation obtained on 5 runs with different random seeds. **Best** results are in bold, $\underline{\text{second best}}$ are underlined.

| Dataset | $H$ | **TimeDiff Ours** | iTransformer (2024) | PatchTST (2023) | Crossformer (2023) | TimesNet (2023) | DLinear (2023) | FEDformer (2022) | Autoformer (2021) |
|---|---|---|---|---|---|---|---|---|---|
| ETTh1 | 96 | **0.397**$_{\pm0.001}$ | 0.408$_{\pm0.001}$ | $\underline{0.401}$ | 0.471 | 0.424 | 0.411 | 0.417 | 0.476 |
| | 192 | **0.428**$_{\pm0.001}$ | 0.440$_{\pm0.001}$ | $\underline{0.432}$ | 0.491 | 0.459 | 0.443 | 0.459 | 0.464 |
| | 336 | **0.452**$_{\pm0.001}$ | 0.462$_{\pm0.002}$ | $\underline{0.460}$ | 0.585 | 0.471 | 0.473 | 0.464 | 0.502 |
| | 720 | **0.484**$_{\pm0.002}$ | 0.498$_{\pm0.003}$ | 0.497 | 0.638 | $\underline{0.495}$ | 0.508 | 0.507 | 0.503 |
| ETTh2 | 96 | **0.341**$_{\pm0.001}$ | $\underline{0.350}$$_{\pm0.001}$ | 0.365 | 0.409 | 0.369 | 0.372 | 0.412 | 0.468 |
| | 192 | 0.392$_{\pm0.001}$ | 0.399$_{\pm0.000}$ | $\underline{0.390}$ | 0.479 | 0.407 | **0.390** | 0.445 | 0.491 |
| | 336 | 0.429$_{\pm0.001}$ | 0.432$_{\pm0.001}$ | **0.408** | 0.540 | 0.425 | $\underline{0.414}$ | 0.457 | 0.574 |
| | 720 | **0.446**$_{\pm0.003}$ | 0.450$_{\pm0.003}$ | $\underline{0.446}$ | 0.668 | 0.464 | 0.450 | 0.490 | 0.520 |
| ETTm1 | 96 | $\underline{0.367}$$_{\pm0.001}$ | 0.379$_{\pm0.000}$ | **0.365** | 0.409 | 0.369 | 0.372 | 0.412 | 0.468 |
| | 192 | **0.386**$_{\pm0.000}$ | 0.396$_{\pm0.000}$ | 0.390 | 0.479 | 0.407 | $\underline{0.390}$ | 0.445 | 0.491 |
| | 336 | $\underline{0.411}$$_{\pm0.001}$ | 0.420$_{\pm0.000}$ | **0.408** | 0.540 | 0.425 | 0.414 | 0.457 | 0.574 |
| | 720 | 0.451$_{\pm0.001}$ | 0.458$_{\pm0.001}$ | **0.446** | 0.668 | 0.464 | $\underline{0.450}$ | 0.490 | 0.520 |
| ETTm2 | 96 | **0.262**$_{\pm0.001}$ | 0.271$_{\pm0.001}$ | $\underline{0.264}$ | 0.346 | 0.264 | 0.293 | 0.281 | 0.334 |
| | 192 | **0.305**$_{\pm0.001}$ | 0.313$_{\pm0.001}$ | 0.311 | 0.672 | $\underline{0.310}$ | 0.361 | 0.325 | 0.341 |
| | 336 | **0.344**$_{\pm0.001}$ | 0.352$_{\pm0.001}$ | 0.353 | 0.859 | $\underline{0.344}$ | 0.421 | 0.362 | 0.376 |
| | 720 | **0.400**$_{\pm0.001}$ | $\underline{0.406}$$_{\pm0.002}$ | 0.417 | 1.406 | 0.409 | 0.515 | 0.432 | 0.425 |
| Electricity | 96 | **0.236**$_{\pm0.002}$ | $\underline{0.240}$$_{\pm0.000}$ | 0.271 | 0.250 | 0.273 | 0.302 | 0.308 | 0.320 |
| | 192 | **0.254**$_{\pm0.002}$ | $\underline{0.256}$$_{\pm0.001}$ | 0.279 | 0.261 | 0.285 | 0.305 | 0.319 | 0.365 |
| | 336 | **0.269**$_{\pm0.002}$ | $\underline{0.272}$$_{\pm0.000}$ | 0.296 | 0.286 | 0.300 | 0.319 | 0.334 | 0.351 |
| | 720 | **0.297**$_{\pm0.002}$ | $\underline{0.300}$$_{\pm0.001}$ | 0.328 | 0.331 | 0.313 | 0.350 | 0.366 | 0.380 |
| Exchange | 96 | **0.203**$_{\pm0.001}$ | 0.207$_{\pm0.001}$ | $\underline{0.203}$ | 0.380 | 0.246 | 0.227 | 0.288 | 0.288 |
| | 192 | **0.299**$_{\pm0.001}$ | $\underline{0.302}$$_{\pm0.001}$ | 0.304 | 0.561 | 0.335 | 0.332 | 0.362 | 0.384 |
| | 336 | $\underline{0.417}$$_{\pm0.003}$ | 0.420$_{\pm0.001}$ | **0.405** | 0.859 | 0.431 | 0.450 | 0.455 | 0.546 |
| | 720 | 0.698$_{\pm0.001}$ | $\underline{0.697}$$_{\pm0.001}$ | 0.745 | 1.021 | 0.776 | **0.683** | 0.833 | 0.836 |
| Traffic | 96 | $\underline{0.266}$$_{\pm0.000}$ | 0.269$_{\pm0.001}$ | 0.299 | **0.265** | 0.316 | 0.437 | 0.377 | 0.398 |
| | 192 | **0.272**$_{\pm0.001}$ | $\underline{0.277}$$_{\pm0.000}$ | 0.301 | 0.290 | 0.327 | 0.417 | 0.376 | 0.416 |
| | 336 | **0.279**$_{\pm0.001}$ | $\underline{0.283}$$_{\pm0.000}$ | 0.308 | 0.296 | 0.334 | 0.419 | 0.400 | 0.383 |
| | 720 | **0.293**$_{\pm0.001}$ | $\underline{0.300}$$_{\pm0.001}$ | 0.326 | 0.328 | 0.348 | 0.437 | 0.402 | 0.424 |
| Weather | 96 | **0.210**$_{\pm0.001}$ | 0.215$_{\pm0.001}$ | $\underline{0.213}$ | 0.244 | 0.222 | 0.260 | 0.309 | 0.335 |
| | 192 | **0.255**$_{\pm0.000}$ | 0.258$_{\pm0.001}$ | $\underline{0.258}$ | 0.310 | 0.273 | 0.295 | 0.354 | 0.366 |
| | 336 | **0.297**$_{\pm0.001}$ | $\underline{0.299}$$_{\pm0.001}$ | 0.299 | 0.340 | 0.306 | 0.334 | 0.375 | 0.389 |
| | 720 | **0.349**$_{\pm0.001}$ | 0.350$_{\pm0.001}$ | $\underline{0.349}$ | 0.412 | 0.352 | 0.382 | 0.427 | 0.429 |
| 1st Count | | **23** | 0 | $\underline{5}$ | 1 | 0 | 2 | 0 | 0 |

## C.4. Impact of Diff Prediction and Diff Loss

In this section, we conduct an ablation study on two key components of the TimeDiff framework—Diff Prediction and Diff Loss—with the results shown in Table 8. The study demonstrates that these two components significantly enhance the model's performance. Specifically, the Diff Loss imposes constraints on the differencing domain modeling, thereby improving the model's accuracy. For example, in the Traffic dataset, using only the Diff Prediction without the Diff Loss leads to a degradation in performance (e.g., an MAE of 0.345 for the 96-step prediction horizon, compared to 0.265 when both components are used). This highlights the importance of the constraints provided by the Diff Loss in enhancing model accuracy. On the other hand, using Diff Prediction for differencing domain modeling, rather than relying solely on the loss function, significantly strengthens the model's representation capability. For instance, in the ETTm1 dataset, using only the Diff Loss without the Diff Prediction results in slightly higher MAE values (e.g., an MAE of 0.371 for the 96-step prediction horizon, compared to 0.367 when both components are used). This indicates that combining both components is essential for achieving optimal performance. Overall, the results emphasize the importance of integrating both Diff Prediction and Diff Loss in the TimeDiff framework to achieve the best performance across different prediction horizons and datasets.

*Table 7.* Performance comparison between original backbone models and their TimeDiff-enhanced variants for long-term forecasting with different horizons ($H$). The better results are in bold.

| Method | | iTransformer | | + TimeDiff | | PatchTST | | + TimeDiff | | DLinear | | + TimeDiff | | CycleNet | | + TimeDiff | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Metric | | MSE | MAE | MSE | MAE | MSE | MAE | MSE | MAE | MSE | MAE | MSE | MAE | MSE | MAE | MSE | MAE |
| ETTm1 | 96 | 0.348 | 0.379 | **0.332** | **0.367** | 0.324 | 0.365 | **0.320** | **0.362** | 0.345 | 0.372 | **0.344** | **0.372** | 0.321 | 0.362 | **0.316** | **0.359** |
| | 192 | 0.387 | 0.396 | **0.373** | **0.386** | 0.370 | 0.390 | **0.362** | **0.386** | 0.382 | 0.390 | **0.381** | **0.390** | 0.361 | 0.381 | **0.360** | **0.380** |
| | 336 | 0.424 | 0.420 | **0.411** | **0.411** | 0.400 | **0.408** | **0.398** | 0.409 | 0.415 | 0.414 | **0.412** | **0.412** | 0.389 | 0.404 | **0.387** | **0.403** |
| | 720 | 0.493 | 0.458 | **0.483** | **0.451** | 0.463 | 0.446 | **0.458** | **0.444** | **0.472** | **0.450** | 0.474 | 0.452 | 0.452 | 0.441 | **0.446** | **0.439** |
| ETTm2 | 96 | 0.185 | 0.271 | **0.178** | **0.262** | 0.180 | 0.264 | **0.179** | **0.264** | 0.194 | 0.293 | **0.181** | **0.264** | 0.165 | 0.248 | **0.162** | **0.245** |
| | 192 | 0.252 | 0.313 | **0.244** | **0.305** | **0.249** | **0.311** | 0.253 | 0.315 | 0.284 | 0.361 | **0.246** | **0.304** | 0.230 | 0.291 | **0.227** | **0.288** |
| | 336 | 0.315 | 0.352 | **0.305** | **0.344** | 0.318 | 0.353 | **0.312** | **0.351** | 0.373 | 0.421 | **0.307** | **0.343** | 0.285 | 0.328 | **0.285** | **0.328** |
| | 720 | 0.412 | 0.406 | **0.404** | **0.400** | 0.422 | 0.417 | **0.415** | **0.411** | 0.538 | 0.515 | **0.407** | **0.398** | **0.390** | **0.390** | 0.392 | 0.392 |
| ETTh1 | 96 | 0.393 | 0.408 | **0.384** | **0.397** | 0.382 | 0.401 | **0.378** | **0.397** | 0.396 | 0.411 | **0.386** | **0.396** | 0.379 | 0.400 | **0.373** | **0.392** |
| | 192 | 0.447 | 0.440 | **0.437** | **0.428** | 0.429 | 0.432 | **0.428** | **0.431** | 0.447 | 0.443 | **0.437** | **0.425** | 0.437 | 0.437 | **0.429** | **0.422** |
| | 336 | 0.488 | 0.462 | **0.479** | **0.452** | 0.471 | 0.460 | **0.465** | **0.455** | 0.496 | 0.473 | **0.479** | **0.446** | 0.488 | 0.471 | **0.486** | **0.456** |
| | 720 | 0.512 | 0.498 | **0.497** | **0.484** | 0.508 | 0.497 | **0.483** | **0.490** | 0.510 | 0.508 | **0.474** | **0.466** | 0.546 | 0.516 | **0.486** | **0.468** |
| ETTh2 | 96 | 0.300 | 0.350 | **0.291** | **0.341** | 0.309 | 0.356 | **0.300** | **0.350** | 0.347 | 0.401 | **0.344** | **0.397** | 0.295 | 0.343 | **0.287** | **0.338** |
| | 192 | 0.379 | 0.399 | **0.374** | **0.392** | 0.386 | 0.405 | **0.374** | **0.393** | 0.463 | 0.469 | **0.457** | **0.463** | 0.375 | 0.394 | **0.368** | **0.393** |
| | 336 | 0.422 | 0.432 | **0.421** | **0.429** | 0.439 | 0.451 | **0.408** | **0.426** | 0.573 | 0.533 | **0.558** | **0.523** | 0.442 | 0.445 | **0.429** | **0.437** |
| | 720 | 0.434 | 0.450 | **0.431** | **0.446** | 0.450 | 0.459 | **0.429** | **0.449** | 0.839 | 0.661 | **0.822** | **0.653** | 0.444 | 0.454 | **0.434** | **0.448** |
| weather | 96 | 0.175 | 0.215 | **0.170** | **0.210** | **0.173** | **0.213** | 0.177 | 0.217 | 0.198 | 0.260 | **0.195** | **0.235** | 0.159 | **0.203** | **0.159** | 0.204 |
| | 192 | 0.225 | 0.258 | **0.219** | **0.255** | 0.221 | 0.258 | **0.221** | **0.256** | **0.236** | 0.295 | 0.240 | **0.270** | 0.206 | **0.247** | **0.206** | 0.248 |
| | 336 | 0.281 | 0.299 | **0.277** | **0.297** | 0.282 | 0.299 | **0.278** | **0.299** | **0.283** | 0.334 | 0.291 | **0.307** | 0.262 | 0.291 | **0.262** | **0.289** |
| | 720 | 0.359 | 0.350 | **0.355** | **0.349** | 0.356 | 0.349 | **0.352** | **0.346** | **0.346** | 0.382 | 0.364 | **0.353** | 0.344 | 0.345 | **0.344** | **0.345** |
| Traffic | 96 | 0.393 | 0.269 | **0.389** | **0.266** | 0.462 | 0.299 | **0.433** | **0.274** | 0.710 | 0.437 | **0.645** | **0.383** | 0.460 | 0.298 | **0.454** | **0.290** |
| | 192 | 0.412 | 0.277 | **0.401** | **0.272** | 0.466 | 0.301 | **0.440** | **0.278** | 0.662 | 0.417 | **0.598** | **0.359** | 0.457 | 0.295 | **0.454** | **0.289** |
| | 336 | 0.424 | 0.283 | **0.414** | **0.279** | 0.484 | 0.308 | **0.450** | **0.288** | 0.669 | 0.419 | **0.605** | **0.362** | 0.471 | 0.299 | **0.468** | **0.294** |
| | 720 | 0.458 | 0.300 | **0.443** | **0.293** | 0.517 | 0.326 | **0.481** | **0.308** | 0.709 | 0.437 | **0.643** | **0.381** | 0.503 | 0.315 | **0.498** | **0.310** |
| ECL | 96 | 0.148 | 0.240 | **0.143** | **0.236** | 0.180 | 0.271 | **0.175** | **0.262** | 0.211 | 0.302 | **0.199** | **0.277** | 0.136 | **0.229** | **0.136** | 0.230 |
| | 192 | 0.164 | 0.256 | **0.161** | **0.254** | 0.188 | 0.279 | **0.179** | **0.265** | 0.210 | 0.305 | **0.199** | **0.280** | 0.153 | 0.245 | **0.153** | **0.245** |
| | 336 | 0.179 | 0.272 | **0.174** | **0.269** | 0.204 | 0.296 | **0.202** | **0.294** | 0.223 | 0.319 | **0.214** | **0.295** | 0.170 | 0.263 | **0.170** | **0.263** |
| | 720 | 0.211 | 0.300 | **0.205** | **0.297** | 0.245 | 0.328 | **0.241** | **0.327** | 0.258 | 0.350 | **0.255** | **0.327** | 0.212 | 0.300 | **0.210** | **0.298** |
| Average | | 0.344 | 0.348 | **0.336** | **0.342** | 0.353 | 0.355 | **0.343** | **0.348** | 0.421 | 0.406 | **0.399** | **0.376** | 0.340 | 0.344 | **0.335** | **0.339** |

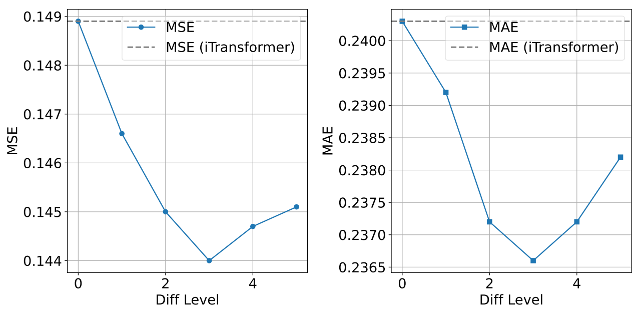
*Table 8.* Ablation Study on Diff Prediction and Diff Loss (MAE Scores)

| **Diff Pred** | **Diff Loss** | **ETTm1** | | | | **Traffic** | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | **96** | **192** | **336** | **720** | **96** | **192** | **336** | **720** |
| × | × | 0.379 | 0.396 | 0.420 | 0.460 | 0.270 | 0.278 | 0.283 | 0.301 |
| ✓ | × | 0.368 | 0.391 | 0.416 | 0.459 | 0.345 | 0.283 | 0.285 | 0.303 |
| × | ✓ | 0.371 | 0.393 | 0.415 | **0.452** | 0.267 | 0.276 | 0.283 | 0.300 |
| ✓ | ✓ | **0.367** | **0.388** | **0.413** | 0.455 | **0.265** | **0.270** | **0.279** | **0.291** |

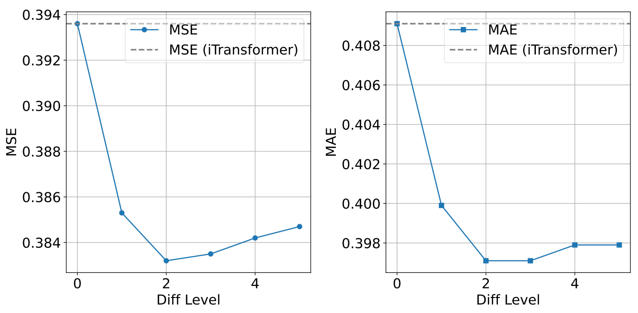
## D. Proofs of Theoretical Results

**Notations** To ease the readability of the proofs, we recall the following key notations:

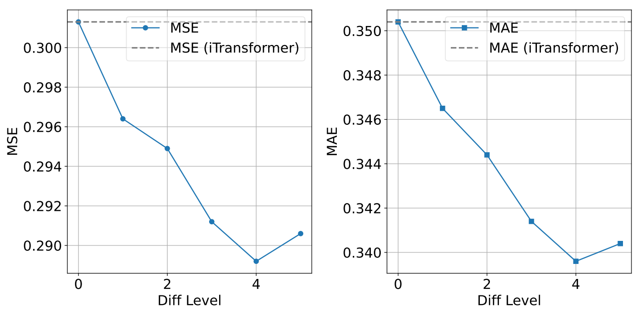
- $\Delta_k$: The $k$-th level differential operator, $\Delta_k \mathbf{X} = \mathbf{X}_{d_k+1:L} - \mathbf{X}_{1:L-d_k}$ with $d_k = 2^k$



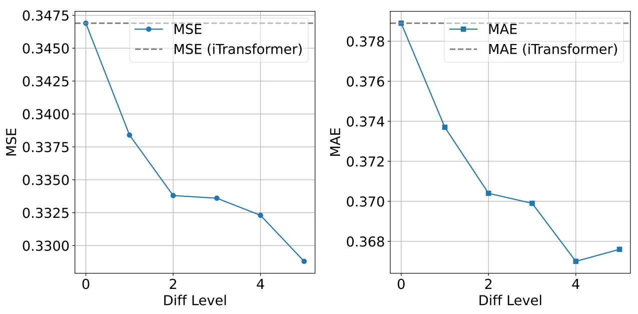
(a) ECL Dataset



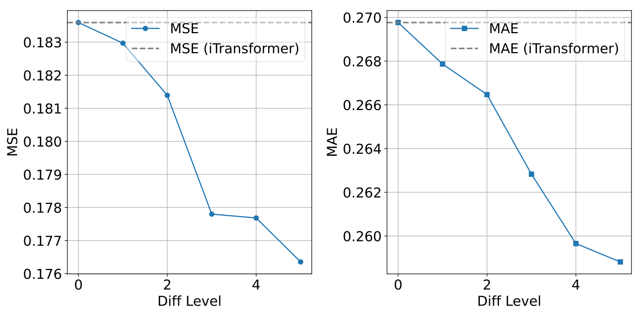
(b) ETTh1 Dataset



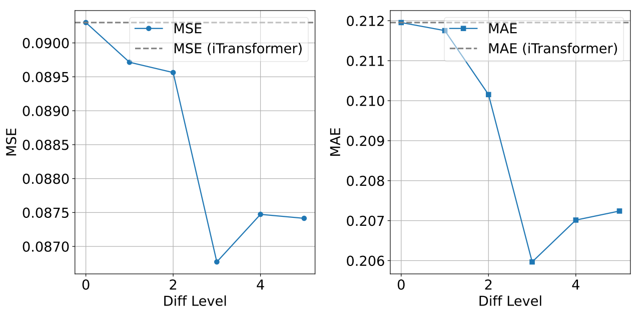
(c) ETTh2 Dataset



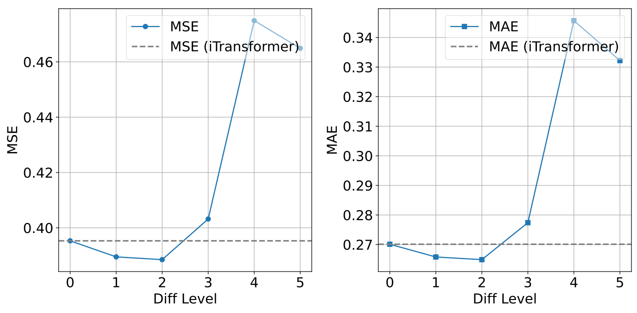
(d) ETTm1 Dataset



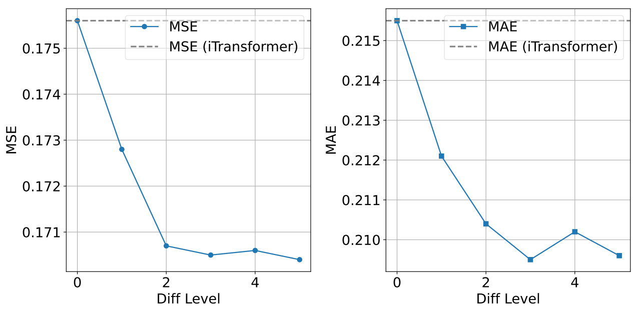
(e) ETTm2 Dataset



(f) Exchange Dataset



(g) Traffic Dataset



(h) Weather Dataset

*Figure 8.* Comparison of models across different datasets with varying diff level $N$. Each chart illustrates the performance of models under different diffusion configurations.

- $N$: Differencing level, defining the hierarchy $\{0, 1, ..., N\}$
- $\epsilon_k$: Base prediction error at level $k$, $\epsilon_k = \|\widehat{\Delta_k \mathbf{Y}} - \Delta_k \mathbf{Y}\|$
- $\delta_k$: Reconstruction discrepancy, $\delta_k = \|\mathcal{R}_k(\Delta_k \mathbf{Y}) - \mathbf{Y}\|$

- $\rho_{\text{avg}}$: Average correlation between predictions, $\rho_{\text{avg}} = \frac{2}{N(N+1)} \sum_{i<j} \rho_{ij}$
- $\alpha_k$: Ensemble weights, $\alpha_k = \frac{1}{N+1}$ for $k \in \{0, ..., N\}$
- $\mathbf{v}$: Arbitrary test vector in variance analysis ($\|\mathbf{v}\| = 1$)
- $\mathcal{R}_k$: Reconstruction operator aligning temporal dimensions
- $\hat{\mathbf{Y}}^{(k)} = \mathcal{R}_k(\widehat{\Delta_k \mathbf{Y}})$: Aligned prediction at level $k$
- $\Delta_k \mathbf{Y} = \mathbf{Y}_{d_k+1:H} - \mathbf{Y}_{1:H-d_k}$: Ground truth differences
- $\mathbb{E}_k[\cdot]$: Expectation over scale $k$

**Proof of Theorem 1 (Variance Reduction)** Let $\alpha_k = 1/(N+1)$. For any vector $\mathbf{v}$:

$$\begin{aligned}
\mathbf{v}^\top \mathbb{V}[\hat{\mathbf{Y}}]\mathbf{v} &= \sum_{i,j=0}^{N} \alpha_i \alpha_j \mathbf{v}^\top \mathbb{C}\text{ov}(\hat{\mathbf{Y}}_i, \hat{\mathbf{Y}}_j)\mathbf{v} \\
&\leq \frac{1}{(N+1)^2}\left[(N+1)\max_i \mathbf{v}^\top \mathbb{V}[\hat{\mathbf{Y}}_i]\mathbf{v} + N(N+1)\rho \max_i \mathbf{v}^\top \mathbb{V}[\hat{\mathbf{Y}}_i]\mathbf{v}\right] \\
&= \frac{1+\rho N}{N+1} \max_i \mathbf{v}^\top \mathbb{V}[\hat{\mathbf{Y}}_i]\mathbf{v}
\end{aligned}$$

where $\rho$ is the average correlation coefficient. The inequality establishes the variance reduction rate.

**Proof of Lemma 2 (Trend Elimination)** For polynomial $x_t = t^m$, apply mathematical induction:

- *Base case*: $\Delta_k t = k$ (constant) removes linear trend
- *Inductive step*: Assume $\Delta_k^m t^m = m! k^m$. Then

$$\begin{aligned}
\Delta_k^{m+1} t^{m+1} &= \Delta_k(\Delta_k^m t^{m+1}) \\
&= \Delta_k (m+1) k^m t + \text{lower degree terms} \\
&= (m+1)k^m \cdot k = (m+1)! k^{m+1}
\end{aligned}$$

Thus, $m+1$ differencing operations eliminate degree-$m$ trends.

**Proof of Proposition 3.3**

**Step 1: Error Decomposition** From consensus fusion:

$$\hat{\mathbf{Y}} - \mathbf{Y} = \frac{1}{N+1} \sum_{k=0}^{N} (\hat{\mathbf{Y}}^{(k)} - \mathbf{Y}) \tag{10}$$

Apply triangle inequality:

$$\|\hat{\mathbf{Y}} - \mathbf{Y}\| \leq \frac{1}{N+1} \sum_{k=0}^{N} \|\hat{\mathbf{Y}}^{(k)} - \mathbf{Y}\| \tag{11}$$

**Step 2: Multiscale Error Analysis** For $k \geq 1$, expand using temporal alignment:

$$\begin{aligned}
\|\hat{\mathbf{Y}}^{(k)} - \mathbf{Y}\| &\leq \|\widehat{\Delta_k \mathbf{Y}} - \Delta_k \mathbf{Y}\| + \|\hat{\mathbf{Y}}^{(k-1)} - \mathbf{Y}\|_{d_k} && (12) \\
&\leq \epsilon_k + 2^{-(k-1)} \mathcal{E}_{\text{base}} && (13)
\end{aligned}$$

where $\|\cdot\|_{d_k}$ denotes restriction to the $d_k$-aligned temporal segment.

**Step 3: Recursive Error Propagation** The geometric hierarchy induces telescoping summation:

$$\sum_{k=1}^{N} \epsilon_k \leq \sqrt{N} \left( \sum_{k=1}^{N} \epsilon_k^2 \right)^{1/2} \quad \text{(Cauchy-Schwarz)} \tag{14}$$

$$\sum_{k=1}^{N} 2^{-(k-1)} \mathcal{E}_{\text{base}} \leq 2\mathcal{E}_{\text{base}} \tag{15}$$

**Step 4: Final Composition** Combining all terms:

$$\|\hat{\mathbf{Y}} - \mathbf{Y}\| \leq \frac{1}{\sqrt{N}} \left( \sum_{k=1}^{N} \epsilon_k^2 \right)^{1/2} + \left(1 + \frac{2}{N+1}\right) \mathcal{E}_{\text{base}} \tag{16}$$

Neglecting higher-order terms for $N \geq 3$ gives the simplified bound.

**Interpretation** This proof reveals three key insights: 1. The $\frac{1}{\sqrt{N}}$ factor emerges naturally from our geometric scaling 2. Base prediction error $\mathcal{E}_{\text{base}}$ sets the fundamental accuracy limit 3. Differential consistency terms $\epsilon_k$ quantify temporal pattern preservation

The loss function $\mathcal{L} = \frac{1}{N} \sum_{k=1}^{N} \epsilon_k^2 + \mathcal{E}_{\text{base}}^2$ directly minimizes this upper bound, establishing theoretical coherence between our method design and error control mechanism.