Are Data Embeddings effective in time series forecasting?

Anonymous authors
Paper under double-blind review

Abstract

Time series forecasting plays a crucial role in many real-world applications, and numerous complex forecasting models have been proposed in recent years. Despite their architectural innovations, most state-of-the-art models report only marginal improvements—typically just a few thousandths in standard error metrics. These models often incorporate complex data embedding layers, which typically transform raw inputs into higher-dimensional representations to enhance accuracy. But are data embedding techniques actually effective in time series forecasting? Through extensive ablation studies across fifteen state-of-the-art models and four benchmark datasets, we find that removing data embedding layers from many state-of-the-art models does not degrade forecasting performance—in many cases, it improves both accuracy and computational efficiency. The gains from removing embedding layers often exceed the performance differences typically reported between competing state-of-the-art models. The code is available at `https://github.com/Data-Embedding-Study/data-embedding-study`.

1 Introduction

Time series forecasting is a fundamental task in machine learning with broad applications, including energy systems, traffic management, healthcare, finance, and weather prediction (Wang et al., 2024b). In recent years, numerous deep learning frameworks have been proposed to improve forecasting performance. These models often employ complex architectures such as statistical components, Transformers, Multilayer Perceptrons (MLPs), and Convolutional neural networks (CNNs). Despite architectural diversity, recent state-of-the-art models achieve gains of only a few thousandths of a point in common metrics such as mean squared error (MSE) or mean absolute error (MAE). Table 1 presents the performance of several time series forecasting models (Nematirad et al., 2025; Yu et al., 2025; Liu et al., 2023; Li et al., 2024b; Wang et al., 2023; 2024a; Han et al., 2024; Dai et al., 2024b) that report state-of-the-art results. Despite substantial architectural differences, the actual improvements in MSE and MAE are often minimal—frequently less than a thousandth compared to competing models. For instance, Times2D outperforms LiNo on the ETTh1 dataset at a prediction horizon of 96 by only 0.001 in both MSE and MAE.

Furthermore, time series forecasting algorithms consist of complex and advanced components. However, their effectiveness and their individual contributions to overall forecasting performance are not adequately investigated. One prominent component is data embedding, which typically transforms raw input data into higher-dimensional representations. For instance, PDF Dai et al. (2024b) and Times2D Nematirad et al. (2025) apply various data embedding techniques without sufficiently justifying the rationale behind using them. On the other hand, models such as PatchTST Nie et al. (2023), SOFTS Han et al. (2024), MICN Wang et al. (2023), and ETSFormer Woo et al. (2023) provide specific justifications for incorporating particular embedding techniques into their architecture. However, the effectiveness of the utilized embedding techniques is not adequately discussed. Consequently, it is unclear whether data embedding techniques truly improve forecasting performance.

Motivated by the minimal improvements achieved through increasingly complex models (Table 1) and insufficient evaluation of core components, we revisit the effectiveness of data embedding layers in time series forecasting. We directly investigate a simple yet important question: **are data embeddings actually effective in time series forecasting?**

Table 1: Performance of recent multivariate forecasting models on the ETTh1 and ETTm1 datasets for prediction horizon $H \in \{96, 192, 336, 720\}$ and input length $L = 96$. red and blue denote best and second-best results.

Models		Times2D (2025)		LiNo (2025)		iTransformer (2024)		RLinear (2024)		MICN (2024)		TimeMixer (2024)		SOFTS (2024)		PDF (2024)	
Data	H	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	96	**0.378**	**0.394**	0.379	0.395	0.386	0.405	0.395	0.419	0.404	0.428	0.375	0.400	0.381	0.399	0.387	0.405
	192	0.431	0.422	0.423	0.423	0.441	0.436	0.424	0.445	0.471	0.471	0.429	**0.421**	0.435	**0.411**	0.439	0.438
	336	0.463	**0.436**	0.455	0.438	0.487	0.458	**0.446**	0.466	0.571	0.538	0.484	0.458	0.480	0.452	0.494	0.464
	720	0.473	0.464	**0.459**	0.456	0.503	0.491	0.470	0.488	0.651	0.622	0.498	0.482	0.499	**0.448**	0.491	0.484
ETTm1	96	0.324	0.363	0.322	0.361	0.334	0.368	0.329	0.367	**0.320**	0.374	**0.320**	**0.357**	0.325	0.361	0.335	0.367
	192	0.370	0.386	0.365	0.383	0.377	0.391	0.367	0.385	0.378	0.414	**0.361**	**0.381**	0.375	0.389	0.377	0.393
	336	0.402	0.406	0.401	0.408	0.426	0.420	0.399	0.410	0.428	0.452	**0.390**	**0.404**	0.405	0.412	0.408	0.415
	720	0.459	**0.439**	0.469	0.447	0.491	0.459	0.454	0.483	0.482	**0.441**	0.454	**0.441**	0.466	0.447	0.457	0.448

Our claim is simple but promising: **removing the data embedding layers from many state-of-the-art forecasting models does not degrade forecasting performance—in many cases, it enhances both forecasting accuracy and computational efficiency. Interestingly, the gains from removing embedding layers often exceed the performance differences typically reported between competing state-of-the-art models**. Our goal is not to imply that data embedding will never be effective in time series forecasting. Instead, we aim to highlight our promising findings and suggest that the community should devote greater attention to critically assessing the actual impact of embedding layers in existing models.

We substantiate our claims by conducting extensive experiments using fifteen time series forecasting models on four standard benchmark datasets originally reported in their studies. Each selected model explicitly utilizes data embedding as a core architectural component. First, we make a great effort to reproduce the results of the standard time series models, using their publicly available publications and the code provided in their official repositories. Next, we identify the data embedding components in each model and rerun the models with these embedding layers bypassed. It should be noted that, in the absence of embedding layers, some preprocessing steps, such as permutation and concatenation are performed to reconcile the input with the model expected dimensions. We further clarify our definition of data embedding and detail the specific steps taken to bypass embedding components in the following sections. Then, we evaluate forecasting performance and computational efficiency in both settings, with and without embedding layers, by measuring epoch runtime and memory usage. The contributions of this study are summarized as follows:

- To our knowledge, this is the first systematic study to rigorously evaluate the effectiveness of data embedding layers in time series forecasting models.
- We conduct comprehensive ablation studies on fifteen high-performing forecasting algorithms across standard benchmark datasets. We show that removing embedding layers generally does not degrade forecasting performance—in many cases, it enhances both forecasting accuracy and computational efficiency.
- We highlight that the gains from removing embedding layers often exceed the performance differences typically reported between competing state-of-the-art models. This finding emphasizes the importance of carefully evaluating the model components before adding further complexity.

2 Data embedding and their removal

Data embedding refers to techniques that transform raw input data into structured representations (Koshil et al., 2024). These transformations are widely used in modern time series forecasting models to prepare data for downstream neural network components. Data embedding strategies can be categorized as follows.

2.1 Value embedding

Value embedding refers to the transformation of raw input time series into a latent feature space, typically of higher dimension. Formally, given a multivariate input sequence $\mathbf{X} \in \mathbb{R}^{B \times L \times N}$, where B is the batch size, L is the sequence length, and N is the number of input variables (features), a value embedding module projects $\mathbf{X}$ into an embedding space of dimension d by mapping the variable dimension $N \to d$. Two value embedding methods are commonly used (Li et al., 2024a):

- **Token-based convolutional embedding**: Applies a 1D convolution along the temporal axis to project input features into a higher-dimensional space. This operation can be expressed as

$$\mathbf{U} = \text{Conv1D}(\mathbf{X})$$

  where $\mathbf{U} \in \mathbb{R}^{B \times L \times d}$.

- **Linear projection**: Applies a linear transformation independently at each time step and maps each input feature vector $\mathbf{x}_t \in \mathbb{R}^N$ to an embedding vector in $\mathbb{R}^d$.

In scenarios where embedding is not used, the input tensor $\mathbf{X}$ is passed directly to the model without projection:

$$\mathbf{U} = \mathbf{X} \in \mathbb{R}^{B \times L \times N}$$

To ensure shape consistency with the embedding scenario, we configure the model such that $d = N$ whenever required across different components of the forecasting architecture.

2.2 Temporal embedding

Temporal embedding encodes time-related features such as minute, hour, day, or month into continuous vectors. Two main types of temporal embedding are used in forecasting models (Li et al., 2021):

- **Discrete temporal embedding**: Embeds categorical time fields (e.g., hour of day, day of week, month) using one of the following techniques:
  - **Fixed embedding**: Uses non-trainable sinusoidal vectors to map each discrete time index to a fixed vector based on sine and cosine functions.
  - **Learnable embedding**: Implements trainable lookup tables (via `nn.Embedding`) that map each discrete temporal category to a vector learned during training.
- **Continuous time feature embedding**: Encodes normalized continuous features (e.g., scaled hour or day values) using a linear projection. Time features are fed into a fully connected layer that maps them to the embedding space $\mathbb{R}^d$.

Given the time covariates $\mathbf{X}_{\text{mark}} \in \mathbb{R}^{B \times L \times N_{\text{time}}}$, where N_{time} is the number of temporal features (e.g., hour-of-day, day-of-week, month, etc.), a temporal embedding layer projects each temporal component into a shared latent space of dimension d, resulting in:

$$\mathbf{U} = \text{TemporalEmbedding}(\mathbf{X}_{\text{mark}}) \in \mathbb{R}^{B \times L \times d}$$

In scenarios where embedding is not used, the temporal features $\mathbf{X}_{\text{mark}}$ are used directly, and we set $d = N_{\text{time}}$.

2.3 Positional embedding

Positional embedding injects information about the position of each time step in the sequence, which is not inherently modeled by components like attention or MLPs. Unlike value or temporal embeddings, which depend on the content of the input features or time-related fields, positional embeddings are purely based on the position index in the sequence.

Given an embedding dimension d and sequence length L, a deterministic positional matrix $\mathbf{P} \in \mathbb{R}^{1\times L\times d}$ is constructed using sine and cosine functions at varying frequencies:

$$\mathbf{P}_{t,2i} = \sin\left(\frac{t}{10000^{2i/d}}\right), \quad \mathbf{P}_{t,2i+1} = \cos\left(\frac{t}{10000^{2i/d}}\right)$$

Here, $i \in \left[0, \left\lfloor \frac{d}{2} \right\rfloor\right)$ denotes the embedding dimension index. The constant 10000 is an empirically chosen scaling factor that ensures smooth variation across dimensions (Chen et al., 2023). Given the input $\mathbf{X} \in \mathbb{R}^{B\times N\times L}$, the positional matrix $\mathbf{P}$ is broadcast across the batch dimension and typically added to the value and/or temporal embeddings before being passed to the model. The output after incorporating positional information is:

$$\mathbf{U} = \text{PositionalEmbedding}(\mathbf{X}) \in \mathbb{R}^{B\times N\times d}$$

In scenarios where embedding is not used, positional encoding is omitted, and the input is passed directly to the model:

$$\mathbf{U} = \mathbf{X} \in \mathbb{R}^{B\times N\times L}$$

To preserve shape compatibility, we set $d = L$ when embedding is removed.

2.4 Inverted embedding

Inverted embedding refers to a design where both the raw input features and the associated time-based features (e.g., hour, day, month) are concatenated along the feature dimension. Given an input sequence $\mathbf{X} \in \mathbb{R}^{B\times N\times L}$ and temporal covariates $\mathbf{X}_{\text{mark}} \in \mathbb{R}^{B\times N_{\text{time}}\times L}$, the two are combined as

$$\mathbf{X}_{\text{concat}} = \text{Concat}(\mathbf{X}, \mathbf{X}_{\text{mark}}) \in \mathbb{R}^{B\times (N+N_{\text{time}})\times L}$$

Unlike traditional embeddings (value, temporal, positional) that enrich or project the feature dimension while preserving the sequence length, inverted embedding treats each *variable* as a token and applies the projection along the temporal dimension. A linear mapping $\mathbf{W} \in \mathbb{R}^{L\times d}$ transforms the sequence dimension into the embedding space:

$$\mathbf{U} = \mathbf{X}_{\text{concat}} \cdot \mathbf{W} \in \mathbb{R}^{B\times (N+N_{\text{time}})\times d}$$

This design emphasizes temporal patterns of individual variables rather than stepwise tokens, enabling the model to capture variable-level dynamics across the entire horizon (Han et al., 2024; Wan et al., 2025).

In scenarios without embedding, the projection step is skipped and the concatenated tensor is used directly:

$$\mathbf{U} = \mathbf{X}_{\text{concat}} \in \mathbb{R}^{B\times (N+N_{\text{time}})\times L}$$

with $d = L$ set to maintain compatibility.

2.5 Patch embedding

Patch embedding segments the input time series into overlapping or non-overlapping temporal patches. Each patch is then projected into a latent embedding space. This reduces input length for long series while preserving fine-grained patterns. Given a multivariate time series input $\mathbf{X} \in \mathbb{R}^{B\times N\times L}$, patching is applied along the temporal axis. The sequence for each variable is divided into fixed-length patches of size P, using a sliding window with stride S, producing:

$$\mathbf{X}_{\text{patch}} \in \mathbb{R}^{B\times N\times L_p\times P}, \quad L_p = \left\lfloor \frac{L-P}{S} \right\rfloor + 1.$$

When embedding is applied, each patch is linearly projected into a latent space of dimension d:

$$\mathbf{Z} = \mathbf{X}_{\text{patch}} \cdot \mathbf{W}_P, \quad \mathbf{W}_P \in \mathbb{R}^{P\times d}, \quad \mathbf{Z} \in \mathbb{R}^{B\times N\times L_p\times d}.$$

The tensor is then reshaped for downstream processing:

$$\mathbf{U} = \text{Reshape}(\mathbf{Z}) \in \mathbb{R}^{(B \cdot N) \times L_p \times d}.$$

In scenarios without embedding, the projection step is omitted and the reshaped patches are used directly:

$$\mathbf{U} = \text{Reshape}(\mathbf{X}_{\text{patch}}) \in \mathbb{R}^{(B \cdot N) \times L_p \times P},$$

with $d = P$ to preserve dimensional compatibility (Nie et al., 2023; Dai et al., 2024b).

A complete summary of the embedding categories and techniques used in this study is provided in Appendix A.1.

3 Related works

The Multi-scale Isometric Convolution Network (MICN) Wang et al. (2023) decomposes the input into seasonal and trend components. A value embedding via 1D convolution (token embedding) is applied to the seasonal sequence, combined with a time-based embedding using fixed values and a sinusoidal positional embedding. These three embedded components are summed and passed through a dropout layer before being fed into the model. ETSformer Woo et al. (2023) proposes a Transformer architecture inspired by exponential smoothing, using decomposed components for level, growth, and seasonality. It employs a value embedding module implemented via 1D convolution to map input features into a latent space. WITRAN Jia et al. (2023) introduces a bi-granular recurrent framework for time series forecasting that models short- and long-term repetitive patterns through 2D information flows. The model concatenates raw input features and time-based covariates along the feature dimension. Then, a fixed temporal embedding is applied to the feature dimension.

The Series-cOre Fused Time Series forecaster (SOFTS) Han et al. (2024) is an efficient MLP-based framework that introduces the STar Aggregate-Redistribute (STAR) module, which employs a centralized strategy to aggregate all series into a global core representation. SOFTS employs an inverted embedding mechanism. It uses linear projection to combine multivariate feature values and time-related metadata. Then, the combined representation is mapped into a high-dimensional space over the sequence dimension. EDformer Chakraborty et al. (2024) introduces a decomposition-based Transformer that separates multivariate time series into trend and seasonal components. It adopts an inverted embedding strategy by concatenating the seasonal component with time-based features and projecting the sequence dimension into a latent space using a linear layer. PPDformer Wan et al. (2025) also adopts an inverted embedding design. It concatenates denoised multivariate feature values with time-based features across the feature axis, and maps the sequence into the embedding space using a linear projection.

Times2D Nematirad et al. (2025) proposes a multi-block decomposition that transforms raw multivariate time series into 2D periodic segments using the Fast Fourier Transform. These segments are passed through 2D convolutional layers, followed by flattening to produce embeddings. This patch-style embedding does not rely on common value, temporal, or positional embeddings.

Crossformer Zhang & Yan (2023) introduces a hierarchical Transformer architecture that models both temporal and cross-variable dependencies for multivariate forecasting. It employs a dual-embedding mechanism. First, it employs a patch-based embedding strategy where the multivariate time series is first segmented into fixed-length patches using a sliding window. Then, each patch is projected into a latent space using the summation of value embedding through linear layers and sinusoidal positional embeddings within the patches.

PatchTST Nie et al. (2023) proposes a channel-independent Transformer for time series forecasting, where each univariate time series is processed separately. It applies the same patch-based embedding mechanism. Unlike Crossformer, PatchTST focuses solely on modeling temporal dependencies within each variable and does not capture cross-variable interactions.

4 Experimental setup

Baselines and datasets. We evaluate fifteen high-performing time series forecasting models. These models have been introduced in top-tier venues in artificial intelligence and machine learning. Models selected in this study cover a broad spectrum of architectural paradigms. Transformer-based architectures include Crossformer (Zhang & Yan, 2023), PatchTST (Nie et al., 2023), ETSformer (Woo et al., 2023), iFlowformer (Kang et al., 2025) and iFlashAttention (Kang et al., 2025). MLP-based approaches comprise MICN (Wang et al., 2023), SOFTS (Han et al., 2024), EDformer (Chakraborty et al., 2024), LiNo (Yu et al., 2025), Minusformer (Liang et al., 2024), and VarDrop (Kang et al., 2025). Finally, hybrid and decomposition-based frameworks are represented by Times2D (Nematirad et al., 2025), PDF (Dai et al., 2024a), PPDformer (Wan et al., 2025), and WITRAN (Jia et al., 2023). We evaluate all models on four widely-used benchmark datasets: ETTh1, ETTh2, ETTm1, and ETTm2. These datasets consist of multivariate time series that capture diverse temporal patterns across various operational domains. ETTh1 and ETTh2 are recorded at hourly intervals, while ETTm1 and ETTm2 are sampled every 15 minutes (Jin et al., 2024). Additional details on the datasets are provided in Appendix A.2.

Setup and Evaluation Metric. All input time series are normalized using the mean and standard deviation from the training set. The input sequence length is fixed at 96, and prediction horizons are set to $H \in \{96, 192, 336, 720\}$, consistent across all models and datasets. Forecasting accuracy is evaluated using MSE and MAE, while computational efficiency is measured by the average training time per epoch (in seconds) and memory usage (in MB).

Infrastructure. All simulations are conducted on a high-performance Linux workstation equipped with an NVIDIA L4 GPU (23 GB memory), CUDA version 12.4, and dual AMD EPYC 7713 64-core processors (128 threads in total). The system has 1 TB of RAM and runs on Ubuntu with Python 3.10 and PyTorch 2.2.1.

5 Results

We present a comprehensive comparison of model performance with and without data embedding layers. Results for the ETTh1 and ETTm1 datasets are shown in Tables 2 and 3, respectively. Additional results for ETTh2 and ETTm2 are reported in Appendix A.3. Below, we summarize key trends observed across models and datasets.

Accuracy typically improves without embeddings. In over 95% of the evaluated configurations, removing data embedding layers improves forecasting accuracy across both MSE and MAE. On the ETTh1 dataset, removing the embedding layer yields an average reduction of 0.0296 in MSE and 0.0193 in MAE (Table 2). ETTh2 exhibits similar behavior, with MSE and MAE decreasing by 0.0208 and 0.0096, respectively (Appendix A.3, Table 8). For the higher-resolution ETTm1 and ETTm2 datasets, the improvements are also evident, with average reductions of 0.0282 and 0.0080 in MSE, and 0.0203 and 0.0091 in MAE, respectively (Table 3; Appendix A.3, Table 9).

Notably, in some cases, the observed gains are remarkably large. For instance, removing the embedding layer from ETSformer on the ETTh1 dataset at horizon 720 reduces MSE by 0.360 and MAE by 0.248. Similarly, Crossformer on ETTm1 at the same horizon achieves a 0.356 drop in MSE, while ETSformer again yields a 0.246 reduction in MAE. Even on shorter horizons and across other datasets such as ETTh2 and ETTm2, we observe improvements exceeding 0.2 in key metrics (Appendix A.3). These results highlight that removing embedding layers can lead to dramatic performance gains.

Furthermore, these accuracy gains are meaningful in practice. As shown in Table 1, recent state-of-the-art forecasting models surpass the second-best models by only 0.001 to 0.009 in evaluation metrics. In contrast, our results show that simply removing the data embedding layers leads to much larger improvements. For example, on the ETTh1 dataset with horizon 96, Times2D and LiNo report MSEs of 0.378 and 0.379, and MAEs of 0.394 and 0.395. These differences are minimal. However, removing the embedding layer from Times2D improves its MSE by 0.019 and MAE by 0.011. For LiNo, the improvements are also notable,

Table 2: ETTh1 forecasting results with and without embeddings for input length $L = 96$ and prediction horizons $H \in \{96, 192, 336, 720\}$. Bold values indicate better performance.

Model	Metric	With Embedding						Without Embedding					
		H				Time	Mem	H				Time	Mem
		96	192	336	720			96	192	336	720		
PDF	MSE	0.387	0.439	0.494	0.491	48.01	2857	**0.377**	**0.430**	**0.484**	**0.503**	**16.50**	**2760**
	MAE	0.405	0.438	0.464	0.484			**0.401**	**0.429**	**0.453**	**0.481**		
ETSformer	MSE	0.564	0.747	0.987	0.987	24.61	4506	**0.563**	**0.611**	**0.643**	**0.627**	**10.95**	**4496**
	MAE	0.536	0.651	0.788	0.806			**0.505**	**0.528**	**0.545**	**0.558**		
PatchTST	MSE	0.389	0.449	0.498	0.544	10.90	**4351**	**0.385**	**0.438**	**0.488**	**0.541**	**8.41**	4353
	MAE	0.409	0.445	0.474	0.517			**0.404**	**0.433**	**0.459**	**0.511**		
MICN	MSE	0.404	0.471	0.576	0.651	19.75	2739	**0.402**	**0.450**	**0.475**	**0.531**	**5.52**	**2709**
	MAE	0.428	0.471	0.538	0.622			**0.421**	**0.448**	**0.473**	**0.527**		
SOFTS	MSE	0.385	0.445	0.501	0.565	9.93	2706	**0.383**	**0.444**	**0.486**	**0.519**	**7.97**	**2223**
	MAE	0.405	0.441	0.469	0.529			**0.401**	**0.439**	**0.462**	**0.502**		
VarDrop	MSE	0.416	0.447	0.490	0.537	9.94	497	**0.386**	**0.442**	**0.491**	**0.495**	**8.37**	**395**
	MAE	0.425	0.445	0.466	0.504			**0.408**	**0.439**	**0.467**	**0.488**		
Crossformer	MSE	**0.390**	0.561	0.639	0.921	40.48	4396	0.404	**0.501**	**0.634**	**0.871**	**40.32**	**4383**
	MAE	**0.421**	0.543	0.588	0.755			0.427	**0.493**	**0.581**	**0.739**		
iFlashAttention	MSE	0.407	0.456	0.487	0.5532	10.80	2293	**0.387**	**0.443**	0.490	**0.492**	**9.75**	**2292**
	MAE	0.420	0.451	0.467	0.5131			**0.407**	**0.439**	**0.467**	**0.486**		
iFlowformer	MSE	0.394	0.459	0.493	0.545	12.40	2297	**0.391**	**0.441**	**0.479**	**0.499**	**7.69**	**2281**
	MAE	0.408	0.450	0.466	0.508			**0.409**	**0.440**	**0.458**	**0.490**		
PPDformer	MSE	0.415	0.460	0.496	0.506	36.08	2738	**0.398**	0.470	**0.473**	**0.487**	**17.92**	**2709**
	MAE	0.424	0.451	0.468	0.492			**0.419**	0.455	**0.461**	**0.486**		
LiNo	MSE	0.379	0.443	0.476	0.496	3.83	2036	**0.372**	**0.429**	**0.454**	**0.460**	**3.61**	**2026**
	MAE	0.395	0.432	0.446	0.474			**0.389**	**0.427**	**0.436**	**0.458**		
EDformer	MSE	0.433	0.520	0.582	**0.661**	**3.52**	**2260**	**0.420**	**0.493**	**0.546**	0.666	3.64	2661
	MAE	0.449	0.504	0.537	**0.608**			**0.441**	**0.488**	**0.519**	0.618		
Minusformer	MSE	0.382	0.431	0.481	0.522	8.03	**2395**	**0.374**	**0.425**	**0.477**	**0.520**	**7.44**	2693
	MAE	0.398	0.430	0.454	**0.492**			**0.395**	**0.429**	**0.450**	0.493		
WITRAN	MSE	0.552	0.646	0.757	0.899	**16.34**	2050	**0.545**	**0.634**	**0.764**	**0.895**	16.38	**2036**
	MAE	0.548	0.608	0.676	0.746			**0.541**	**0.599**	**0.659**	**0.746**		
Times2D	MSE	0.378	0.431	0.463	0.473	**5.58**	778	**0.359**	**0.427**	**0.461**	**0.472**	6.48	**758**
	MAE	0.394	0.422	0.436	0.464			**0.383**	**0.421**	**0.435**	**0.463**		

with reductions of 0.007 in MSE and 0.006 in MAE. These findings suggest that simplifying model architectures by eliminating embedding layers can yield benefits that exceed those obtained by designing entirely new forecasting models. These findings suggest that raw input features in multivariate time series often contain sufficient representational richness for forecasting tasks without the need for additional embedding transformations.

Significant computational savings. Removing data embedding layers consistently reduces computational overhead. The average training time per epoch (in seconds) is reduced across all datasets, and on ETTm1 and ETTh1 we find savings of up to 25 seconds. Memory usage also decreases notably. For example, the average memory savings are 84 MB on ETTh1 and 45 MB on ETTm2 (Appendix A.3). In some configurations, the improvements are substantial. On ETTm2 (Appendix A.3), ETSformer shows a 597 MB

Table 3: ETTm1 forecasting results with and without embeddings for input length $L = 96$ and prediction horizons $H \in \{96, 192, 336, 720\}$. Bold values indicate better performance.

Model	Metric	With Embedding						Without Embedding					
		H				Time	Mem	H				Time	Mem
		96	192	336	720			96	192	336	720		
PDF	MSE	0.335	0.377	0.408	0.457	194.3	2961	**0.321**	**0.365**	**0.392**	**0.451**	**64.44**	**2880**
	MAE	0.367	0.393	0.415	0.448			**0.359**	**0.386**	**0.405**	**0.442**		
ETSformer	MSE	0.526	0.577	0.677	0.802	94.28	2803	**0.373**	**0.408**	**0.441**	**0.499**	**33.18**	**2200**
	MAE	0.515	0.553	0.620	0.708			**0.397**	**0.410**	**0.429**	**0.462**		
PatchTST	MSE	**0.344**	0.375	0.407	0.473	58.62	**2476**	0.348	**0.370**	**0.393**	**0.459**	**23.48**	2893
	MAE	**0.367**	0.395	0.415	0.453			0.371	**0.387**	**0.406**	**0.440**		
MICN	MSE	**0.320**	0.378	0.428	0.483	70.17	2850	0.354	**0.363**	**0.416**	**0.478**	**17.05**	**2805**
	MAE	**0.374**	0.414	0.452	0.482			0.380	**0.395**	**0.416**	**0.455**		
SOFTS	MSE	0.325	0.384	0.429	0.477	33.51	**2403**	**0.323**	**0.367**	**0.407**	**0.475**	**27.16**	2419
	MAE	0.361	0.397	0.423	0.455			**0.341**	**0.386**	**0.411**	**0.452**		
VarDrop	MSE	0.340	0.398	0.439	**0.490**	36.26	504	0.344	**0.382**	**0.428**	0.505	**27.86**	**393**
	MAE	0.375	0.403	0.427	**0.457**			0.378	**0.397**	**0.425**	0.467		
Crossformer	MSE	0.366	0.413	0.453	0.867	233.8	2237	**0.363**	**0.406**	**0.447**	**0.511**	**230.4**	**2228**
	MAE	0.406	0.427	0.454	0.711			**0.404**	**0.418**	**0.446**	**0.481**		
iFlashAttention	MSE	0.350	0.402	0.442	0.500	45.11	2438	**0.344**	**0.382**	**0.434**	**0.527**	**34.28**	**2421**
	MAE	0.381	0.405	0.428	0.464			**0.378**	**0.397**	**0.428**	**0.476**		
iFlowformer	MSE	0.340	0.418	**0.420**	**0.492**	45.66	2409	**0.339**	**0.388**	0.448	0.501	**34.97**	**2331**
	MAE	0.373	0.412	**0.424**	**0.461**			**0.372**	**0.399**	0.434	0.470		
PPDformer	MSE	0.356	0.411	0.440	0.503	149.5	2840	**0.339**	**0.389**	**0.432**	**0.493**	**86.75**	**2833**
	MAE	0.392	0.420	0.438	0.471			**0.376**	**0.400**	**0.429**	**0.467**		
LiNo	MSE	0.331	0.400	0.435	0.503	13.50	2146	**0.323**	**0.375**	**0.418**	**0.497**	**12.47**	**2145**
	MAE	0.365	0.404	0.423	0.463			**0.361**	**0.390**	**0.416**	**0.462**		
EDformer	MSE	0.395	0.432	0.486	**0.544**	**10.87**	**2404**	**0.378**	**0.426**	**0.463**	0.551	11.36	2805
	MAE	0.427	0.448	0.478	**0.509**			**0.413**	**0.443**	**0.466**	0.520		
Minusformer	MSE	0.351	0.384	**0.451**	0.491	40.98	**2528**	**0.330**	**0.381**	0.449	**0.490**	**32.57**	2847
	MAE	0.377	0.394	0.432	0.459			**0.367**	**0.392**	**0.428**	**0.457**		
WITRAN	MSE	0.640	0.769	0.827	0.951	107.8	937	**0.637**	**0.688**	**0.803**	**0.860**	**107.6**	**914**
	MAE	0.590	0.668	0.714	0.766			**0.585**	**0.622**	**0.699**	**0.713**		
Times2D	MSE	0.325	0.370	0.402	0.459	21.30	783	**0.324**	**0.368**	**0.397**	**0.458**	**21.09**	**761**
	MAE	0.363	0.386	0.406	0.439			**0.361**	**0.383**	**0.403**	**0.438**		

memory reduction. These findings suggest that eliminating embedding layers improves training efficiency, particularly in memory-constrained environments.

Performance gains increase with horizon length. The effect of removing the embedding layer increases as the forecasting horizon increases across all four benchmarks. While short-term configurations (e.g., $H = 96$) show limited changes, longer horizons often yield substantial improvements. For instance, on ETTm1, Crossformer shows an MSE reduction of 0.003 at $H = 96$, which increases significantly to 0.365 at $H = 720$. This finding indicates that embedding-free designs may be advantageous for long-term forecasting tasks.

Architectural sensitivity to embedding layers. Embedding removal impacts architectural families differently. Transformer-based models rely on self-attention mechanisms to capture long-range dependencies,

but they do not include any inherent structure to model sequential order. Unlike recurrent architectures, Transformers require explicit positional and token embeddings to encode temporal progression. In theory, these embeddings are intended to compensate for the lack of built-in sequence modeling. However, our empirical results reveal that removing these embeddings often leads to better performance. MLP-based architectures employ purely feedforward pathways and rely on dense transformations to model dependencies across time and variables. Since MLPs do not explicitly model sequence order, embedding layers might be expected to play a more important role. Yet, our results show that embeddings are often redundant in MLPs. Hybrid and decomposition-based models incorporate preprocessing such as seasonal-trend decomposition, filtering, or statistical projections. These models are less sensitive to the presence of embedding layers. For example, PDF shows a modest gain. Since these architectures already extract and isolate key patterns before learning begins, embedding layers often duplicate or disrupt this structure, resulting in minimal or inconsistent effects.

Confidence intervals. Since deep learning models are inherently stochastic and sensitive to random initialization, we compute 95% confidence intervals (CIs) to assess the statistical reliability of our findings. Tables 4 and 5 present the MSE and MAE, respectively, along with corresponding 95% confidence intervals for selected high-performing models on ETTh1 and ETTm1. In all cases—except for LiNo on ETTm1 with $H = 720$ in both MAE and MSE—removing the embedding layers improves performance. Additionally, the corresponding confidence intervals for the models with and without embedding layers do not overlap. These results highlight that the performance improvements are consistent and statistically significant.

Table 4: Confidence intervals for MSE of selected models on ETTh1 and ETTm1 with input length $L = 96$ and prediction horizons $H \in \{96, 192, 336, 720\}$. **Bold** values indicate better performance.

Models	H	Times2D		PDF		LiNo		SOFTS	
		With Embedding							
		MSE	CI	MSE	CI	MSE	CI	MSE	CI
ETTh1	96	0.379	(0.378, 0.380)	0.385	(0.382, 0.388)	0.385	(0.384, 0.386)	0.384	(0.383, 0.386)
	192	0.431	(0.429, 0.433)	0.439	(0.436, 0.442)	0.442	(0.438, 0.446)	0.448	(0.445, 0.450)
	336	0.473	(0.469, 0.476)	0.492	(0.486, 0.498)	0.476	(0.471, 0.480)	0.501	(0.494, 0.509)
	720	0.473	(0.469, 0.477)	0.521	(0.501, 0.544)	0.482	(0.473, 0.491)	0.538	(0.524, 0.552)
ETTm1	96	0.326	(0.323, 0.328)	0.335	(0.334, 0.336)	0.332	(0.331, 0.333)	0.328	(0.325, 0.330)
	192	0.371	(0.370, 0.372)	0.374	(0.372, 0.376)	0.383	(0.375, 0.392)	0.386	(0.381, 0.391)
	336	0.407	(0.402, 0.411)	0.403	(0.401, 0.405)	0.438	(0.432, 0.444)	0.438	(0.428, 0.447)
	720	0.459	(0.455, 0.463)	0.457	(0.455, 0.459)	**0.496**	(**0.485**, 0.506)	0.480	(0.477, 0.482)
		Without Embedding							
		MSE	CI	MSE	CI	MSE	CI	MSE	CI
ETTh1	96	**0.361**	(**0.359**, **0.364**)	**0.378**	(**0.376**, **0.380**)	**0.377**	(**0.375**, **0.379**)	**0.383**	(**0.382**, **0.384**)
	192	**0.428**	(**0.427**, **0.429**)	**0.432**	(**0.428**, **0.436**)	**0.428**	(**0.426**, **0.429**)	**0.441**	(**0.438**, **0.444**)
	336	**0.466**	(**0.463**, **0.469**)	**0.479**	(**0.476**, **0.482**)	**0.463**	(**0.460**, **0.466**)	**0.487**	(**0.483**, **0.490**)
	720	**0.472**	(**0.468**, **0.476**)	**0.518**	(**0.499**, **0.537**)	**0.470**	(**0.464**, **0.476**)	**0.526**	(**0.514**, **0.537**)
ETTm1	96	**0.325**	(**0.322**, **0.327**)	**0.324**	(**0.322**, **0.326**)	**0.326**	(**0.323**, **0.329**)	**0.322**	(**0.321**, **0.323**)
	192	**0.369**	(**0.368**, **0.370**)	**0.368**	(**0.365**, **0.370**)	**0.373**	(**0.370**, **0.376**)	**0.368**	(**0.367**, **0.369**)
	336	**0.401**	(**0.397**, **0.406**)	**0.395**	(**0.394**, **0.397**)	**0.421**	(**0.415**, **0.426**)	**0.406**	(**0.405**, **0.407**)
	720	**0.455**	(**0.451**, **0.459**)	**0.454**	(**0.452**, **0.456**)	0.497	(0.492, **0.502**)	**0.476**	(**0.474**, **0.477**)

Configurations with degraded performance. While the majority of models benefit from removing embedding layers, a few configurations exhibit performance degradation. This outcome can be attributed to several architectural and/or hardware-related factors. First, in the absence of the embedding layer, the model manually permutes, concatenates, and processes the raw input data to reconcile it with the model expected dimensions. These operations introduce additional intermediate tensors and temporary memory allocations, which increase the average memory usage during training. Second, the lower dimensionality resulting from the

Table 5: Confidence intervals for MAE of selected models on ETTh1 and ETTm1 with input length $L = 96$ and prediction horizons $H \in \{96, 192, 336, 720\}$. **Bold** values indicate better performance.

Models	H	Times2D		PDF		LiNo		SOFTS	
		With Embedding							
		MAE	CI	MAE	CI	MAE	CI	MAE	CI
ETTh1	96	0.402	(0.400, 0.405)	0.405	(0.403, 0.407)	0.403	(0.402, 0.404)	0.404	(0.403, 0.405)
	192	0.432	(0.431, 0.433)	0.438	(0.436, 0.440)	0.433	(0.431, 0.436)	0.442	(0.440, 0.444)
	336	0.442	(0.440, 0.444)	0.466	(0.463, 0.470)	0.446	(0.444, 0.448)	0.469	(0.464, 0.475)
	720	0.465	(**0.462**, 0.468)	0.497	(0.484, 0.509)	0.468	(0.463, 0.473)	0.513	(0.504, 0.521)
ETTm1	96	0.364	(0.362, 0.366)	0.368	(0.367, 0.369)	0.367	(0.366, 0.368)	0.365	(0.363, 0.366)
	192	0.390	(0.385, 0.395)	0.392	(0.390, 0.393)	0.395	(0.390, 0.399)	0.397	(0.394, 0.400)
	336	0.410	(0.405, 0.415)	0.414	(0.412, 0.416)	0.426	(0.423, 0.429)	0.429	(0.424, 0.433)
	720	0.441	(0.439, 0.443)	0.465	(0.457, 0.472)	**0.460**	(**0.455**, 0.465)	0.455	(0.454, 0.457)
		Without Embedding							
		MAE	CI	MAE	CI	MAE	CI	MAE	CI
ETTh1	96	**0.392**	(**0.391**, **0.393**)	**0.399**	(**0.398**, **0.400**)	**0.399**	(**0.397**, **0.401**)	**0.402**	(**0.401**, **0.403**)
	192	**0.422**	(**0.421**, **0.423**)	**0.431**	(**0.429**, **0.433**)	**0.425**	(**0.424**, **0.426**)	**0.437**	(**0.435**, **0.439**)
	336	**0.439**	(**0.438**, **0.440**)	**0.453**	(**0.450**, **0.455**)	**0.440**	(**0.438**, **0.441**)	**0.463**	(**0.460**, **0.466**)
	720	0.465	(0.463, **0.467**)	**0.491**	(**0.481**, **0.500**)	**0.463**	(**0.459**, **0.466**)	**0.506**	(**0.500**, **0.513**)
ETTm1	96	**0.363**	(**0.361**, **0.365**)	**0.362**	(**0.360**, **0.364**)	**0.363**	(**0.361**, **0.365**)	**0.361**	(**0.360**, **0.362**)
	192	**0.387**	(**0.385**, **0.389**)	**0.387**	(**0.385**, **0.388**)	**0.389**	(**0.387**, **0.390**)	**0.386**	(**0.385**, **0.387**)
	336	**0.406**	(**0.403**, **0.408**)	**0.408**	(**0.407**, **0.409**)	**0.419**	(**0.416**, **0.423**)	**0.411**	(**0.410**, **0.412**)
	720	**0.439**	(**0.438**, **0.441**)	**0.455**	(**0.451**, **0.460**)	0.461	(0.458, **0.464**)	**0.451**	(**0.451**, **0.452**)

removal of embedding layers does not align well with the tile sizes optimized in GPU libraries such as cuBLAS, leading to less efficient matrix multiplications and increased computational time. In particular, EDformer originally uses an inverted embedding that transforms the sequence length (e.g., 96) into a typically higher dimension (e.g., 512). EDformer trains approximately 0.3 to 0.5 seconds slower per epoch and consumes an additional 400 MB of memory on average when the embedding layers are removed. The extra permutation, concatenation, and duplication required to adapt the raw inputs to the expected format increases memory usage. Furthermore, the encoder operates on input tensors with a sequence dimension of 96 instead of 512, which reduces computational throughput due to suboptimal memory access patterns and kernel launch configurations in GPU backends.

6 Conclusion

In this paper, we presented a large-scale study assessing the effectiveness of embedding layers in modern time series forecasting models. Our results show that, despite their widespread use, removing data embedding layers from many state-of-the-art forecasting models does not degrade forecasting performance—in many cases, it enhances both forecasting accuracy and computational efficiency. These findings suggest that raw multivariate inputs are often sufficiently informative without the need for additional embedding transformations. Our goal is not to imply that data embedding will never be effective in time series forecasting. Instead, we aim to highlight our promising findings and suggest that the community devote greater attention to critically assessing the actual impact of embedding layers in existing models. For future studies, the effectiveness of embedding layers can be explored on other tasks (e.g., classification, clustering, and imputation) and datasets. Additionally, the effectiveness of other overlooked architectural components—such as normalization strategies, including RevIN—can be investigated.

Limitations

Here, we outline the limitations of our study:

- We evaluate the impact of data embedding layers specifically for time-series forecasting. However, embedding layers may play different roles in other downstream tasks such as classification, clustering, or imputation, which are not explored in this work.
- Our evaluation is restricted to four benchmark datasets with regular sampling intervals. The effectiveness of embedding layers can be investigated on other real-world datasets, including those with irregularly sampled time series.
- The analysis focuses solely on the effect of embedding layers and does not account for potential interactions with other architectural components such as normalization strategies or residual connections.

Broader impact statement

This work makes a fundamental contribution to time series analysis, particularly in the context of forecasting. It encourages researchers to move beyond default assumptions and critically assess whether each architectural component, such as data embedding layers, meaningfully contributes to performance. Our findings promote a shift in focus: rather than continually developing more complex models, researchers across domains are encouraged to revisit and analyze existing architectures. This approach can lead to significant savings in time, resources, and energy. While our results are limited to forecasting tasks on regularly sampled datasets, the broader methodology—systematic ablation testing of architectural components—can inspire more rigorous empirical validation in other areas of machine learning. We hope this work supports the community in understanding the role and effectiveness of foundational model elements before advancing to further architectural complexity and innovation.

References

Sanjay Chakraborty, Ibrahim Delibasoglu, and Fredrik Heintz. Edformer: Embedded decomposition transformer for interpretable multivariate time series predictions, 2024. URL https://arxiv.org/abs/2412.12227.

Yuqi Chen, Kan Ren, Yansen Wang, Yuchen Fang, Weiwei Sun, and Dongsheng Li. Contiformer: Continuous-time transformer for irregular time series modeling. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=YJDz4F2AZu.

Tao Dai, Beiliang Wu, Peiyuan Liu, Naiqi Li, Jigang Bao, Yong Jiang, and Shu-Tao Xia. Periodicity decoupling framework for long-term series forecasting. *International Conference on Learning Representations*, 2024a.

Tao Dai, Beiliang Wu, Peiyuan Liu, Naiqi Li, Jigang Bao, Yong Jiang, and Shu-Tao Xia. Periodicity decoupling framework for long-term series forecasting. In *The Twelfth International Conference on Learning Representations*, 2024b. URL https://openreview.net/forum?id=dp27P5HBBt.

Lu Han, Xu-Yang Chen, Han-Jia Ye, and De-Chuan Zhan. SOFTS: Efficient multivariate time series forecasting with series-core fusion. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=89AUi5L1uA.

Yuxin Jia, Youfang Lin, Xinyan Hao, Yan Lin, Shengnan Guo, and Huaiyu Wan. WITRAN: Water-wave information transmission and recurrent acceleration network for long-range time series forecasting. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=y08bkEtNBK.

Ming Jin, Shiyu Wang, Lintao Ma, Zhixuan Chu, James Y. Zhang, Xiaoming Shi, Pin-Yu Chen, Yuxuan Liang, Yuan-Fang Li, Shirui Pan, and Qingsong Wen. Time-LLM: Time series forecasting by reprogramming large language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=Unb5CVPtae.

Junhyeok Kang, Yooju Shin, and Jae-Gil Lee. Vardrop: Enhancing training efficiency by reducing variate redundancy in periodic time series forecasting. In *AAAI*, 2025.

Mykhailo Koshil, Thomas Nagler, Matthias Feurer, and Katharina Eggensperger. Towards localization via data embedding for tabPFN. In *NeurIPS 2024 Third Table Representation Learning Workshop*, 2024. URL https://openreview.net/forum?id=LFyQyV5HxQ.

Linsen Li, Chunfei Jian, Feng Wan, Dongdong Geng, Ziquan Fang, Lu Chen, Yunjun Gao, Weihao Jiang, and Jiang Zhu. Lagcnn: A fast yet effective model for multivariate long-term time series forecasting. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, pp. 1235–1244, 2024a.

Zhe Li, Shiyi Qi, Yiduo Li, and Zenglin Xu. Revisiting long-term time series forecasting: An investigation on affine mapping, 2024b. URL https://openreview.net/forum?id=T97kxctihq.

Zhihan Li, Youjian Zhao, Jiaqi Han, Ya Su, Rui Jiao, Xidao Wen, and Dan Pei. Multivariate time series anomaly detection and interpretation using hierarchical inter-metric and temporal embedding. In *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*, pp. 3220–3230, 2021.

Daojun Liang, Haixia Zhang, Dongfeng Yuan, Bingzheng Zhang, and Minggao Zhang. Minusformer: Improving time series forecasting by progressively learning residuals. *arXiv preprint arXiv:2402.02332*, 2024.

Yong Liu, Tengge Hu, Haoran Zhang, Haixu Wu, Shiyu Wang, Lintao Ma, and Mingsheng Long. itransformer: Inverted transformers are effective for time series forecasting. *arXiv preprint arXiv:2310.06625*, 2023.

Reza Nematirad, Anil Pahwa, and Balasubramaniam Natarajan. Times2d: Multi-period decomposition and derivative mapping for general time series forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 19651–19658, 2025.

Yuqi Nie, Nam H Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. A time series is worth 64 words: Long-term forecasting with transformers. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=Jbdc0vTOcol.

Meng Wan, Qi Su, Huan Hao, Jue Wang, Yuexiu Cui, Yuxuan Bi, Rongqiang Cao, Peng Shi, Yangang Wang, Zonghua Qiu, et al. Ppdformer: Channel-specific periodic patch division for time series forecasting. In *ICASSP 2025-2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1–5. IEEE, 2025.

Huiqiang Wang, Jian Peng, Feihu Huang, Jince Wang, Junhui Chen, and Yifei Xiao. MICN: Multi-scale local and global context modeling for long-term series forecasting. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=zt53IDUR1U.

Shiyu Wang, Haixu Wu, Xiaoming Shi, Tengge Hu, Huakun Luo, Lintao Ma, James Y. Zhang, and JUN ZHOU. Timemixer: Decomposable multiscale mixing for time series forecasting. In *The Twelfth International Conference on Learning Representations*, 2024a. URL https://openreview.net/forum?id=7oLshfEIC2.

Yuxuan Wang, Haixu Wu, Jiaxiang Dong, Guo Qin, Haoran Zhang, Yong Liu, Yun-Zhong Qiu, Jianmin Wang, and Mingsheng Long. Timexer: Empowering transformers for time series forecasting with exogenous variables. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024b. URL https://openreview.net/forum?id=INAeUQO4lT.

Gerald Woo, Chenghao Liu, Doyen Sahoo, Akshat Kumar, and Steven Hoi. ETSformer: Exponential smoothing transformers for time-series forecasting, 2023. URL https://openreview.net/forum?id=5m_3whfo483.

Guoqi Yu, Yaoming Li, Xiaoyu Guo, Shujun Wang, Zirui Liu, Dayu Wang, and Tong Yang. Lino: Advancing recursive residual decomposition of linear and nonlinear patterns for robust time series forecasting, 2025. URL https://openreview.net/forum?id=JeZS4jQF77.

Yunhao Zhang and Junchi Yan. Crossformer: Transformer utilizing cross-dimension dependency for multivariate time series forecasting. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=vSVLM2j9eie.

A Appendix

A.1 Embedding layer categories

This section provides a summary of the embedding categories and techniques used in our study.

Table 6: Summary of embedding layer categories and techniques used in this study.

Category	Technique	Description
Temporal	Fixed embedding	Maps discrete temporal indices to non-trainable, sinusoidal embeddings.
	Learnable embedding	Maps discrete temporal indices (e.g., hour, day) to trainable embeddings.
	TimeFeature	Projects numeric time features into high-dimensional space via linear projection.
Value	Token embedding	Projects multivariate features to higher dimensions using 1D convolution.
	Linear projection	Maps input features directly to embedding space using linear layers.
Positional	Sinusoidal positional	Encodes sequence positions using fixed sine and cosine functions.
	Learnable positional	Learns embeddings for positional indices in sequences.
Combined	Inverted	Fuses variables and time features using linear transformations.
Patching	Patchwise encoding	Divides input sequences into patches, encodes each patch, and adds positional info.

A.2 Benchmark datasets

This section summarizes the benchmark datasets used in our experiments.

Table 7: Summary of benchmark datasets used in this study.

Dataset	Dimension	Train / Val / Test	Frequency	Duration
ETTm1	7	(34465 / 11521 / 11521)	15 minutes	Jul 2016 – Jul 2018
ETTm2	7	(34465 / 11521 / 11521)	15 minutes	Jul 2016 – Jul 2018
ETTh1	7	(8545 / 2881 / 2881)	1 hour	Jul 2016 – Jul 2018
ETTh2	7	(8545 / 2881 / 2881)	1 hour	Jul 2016 – Jul 2018

A.3 Forecasting results on ETTh2 and ETTm2

This section reports additional forecasting results on the ETTh2 and ETTm2 benchmark datasets, complementing the ETTh1 and ETTm1 results presented in the main text (Tables 2 and 3). All experiments are conducted with input length $L = 96$ and prediction horizons $H \in \{96, 192, 336, 720\}$. We compare model performance with and without embedding layers. As in the main tables, bold values indicate better performance.

Table 8: ETTh2 forecasting results with and without embeddings for input length $L = 96$ and prediction horizons $H \in \{96, 192, 336, 720\}$. Bold values indicate better performance.

Model	Metric	With Embedding						Without Embedding					
		H				Time	Mem	H				Time	Mem
		96	192	336	720			96	192	336	720		
PDF	MSE	0.307	0.376	0.414	0.437	47.82	2838	**0.300**	**0.375**	**0.412**	**0.431**	**16.45**	**2752**
	MAE	0.353	0.401	0.426	0.452			**0.348**	**0.397**	**0.426**	**0.447**		
ETSformer	MSE	0.399	0.521	0.615	0.692	23.50	2721	**0.345**	**0.436**	**0.487**	**0.494**	**8.80**	**2080**
	MAE	0.435	0.505	0.569	0.616			**0.399**	**0.446**	**0.483**	**0.497**		
PatchTST	MSE	0.300	0.382	0.435	0.447	14.96	**2351**	**0.297**	**0.375**	**0.413**	**0.418**	**6.46**	2758
	MAE	**0.350**	0.404	0.441	0.463			0.384	**0.399**	**0.428**	**0.440**		
MICN	MSE	0.354	0.475	0.602	0.829	17.95	2761	**0.333**	**0.463**	**0.567**	**0.804**	**5.24**	**2708**
	MAE	0.400	0.477	0.540	0.654			**0.389**	**0.472**	**0.527**	**0.646**		
SOFTS	MSE	0.305	0.375	0.437	0.439	10.06	634	**0.295**	**0.374**	**0.416**	**0.434**	**8.09**	**473**
	MAE	0.350	0.396	0.437	**0.447**			**0.347**	**0.395**	**0.430**	0.450		
VarDrop	MSE	0.306	0.393	0.423	0.436	9.64	2288	**0.303**	**0.383**	**0.423**	**0.418**	**8.39**	**2274**
	MAE	0.355	0.406	0.438	0.452			**0.353**	**0.400**	**0.436**	**0.442**		
Crossformer	MSE	0.588	0.978	0.996	1.161	56.51	2243	**0.573**	**0.757**	**0.796**	**0.945**	**56.41**	**2223**
	MAE	0.576	0.698	0.709	0.787			**0.537**	**0.628**	**0.643**	**0.803**		
iFlashAttention	MSE	0.306	0.392	0.428	0.442	11.43	2303	**0.305**	**0.382**	**0.426**	**0.417**	**10.00**	**2294**
	MAE	0.355	0.406	0.438	0.455			**0.353**	**0.400**	**0.430**	**0.440**		
iFlowformer	MSE	0.308	0.389	0.431	0.440	13.02	2293	0.308	**0.382**	**0.422**	**0.432**	**10.70**	**2281**
	MAE	0.357	0.409	0.440	0.453			**0.355**	**0.402**	**0.433**	**0.447**		
PPDformer	MSE	0.321	0.405	0.439	0.461	36.94	2727	**0.320**	**0.401**	**0.436**	**0.438**	**22.24**	**2714**
	MAE	0.365	0.415	0.445	0.465			**0.361**	**0.406**	**0.439**	**0.454**		
LiNo	MSE	0.305	0.384	0.389	0.417	3.96	2035	**0.296**	**0.378**	**0.385**	**0.412**	**3.83**	**2021**
	MAE	0.352	0.398	0.413	0.436			**0.345**	**0.395**	**0.411**	**0.432**		
EDformer	MSE	0.422	0.485	0.549	0.799	**3.66**	**2288**	**0.404**	**0.484**	**0.504**	**0.688**	3.98	2661
	MAE	0.429	0.464	0.500	0.621			**0.423**	**0.482**	**0.490**	**0.593**		
Minusformer	MSE	0.304	0.379	0.430	0.427	10.94	2420	**0.293**	**0.373**	**0.421**	**0.422**	**8.78**	**2699**
	MAE	0.349	0.396	0.435	0.441			**0.344**	**0.393**	**0.429**	**0.439**		
WITRAN	MSE	1.659	2.810	2.641	3.488	27.12	804	1.771	2.752	2.632	3.752	25.12	803
	MAE	1.055	1.464	1.415	1.663			1.120	1.439	1.403	1.682		
Times2D	MSE	**0.292**	0.376	0.379	0.413	5.52	775	0.294	**0.371**	**0.376**	**0.406**	**5.19**	**740**
	MAE	**0.340**	0.391	0.407	0.434			0.342	**0.390**	**0.405**	**0.429**		

Table 9: ETTm2 forecasting results with and without embeddings for input length $L = 96$ and prediction horizons $H \in \{96, 192, 336, 720\}$. Bold values indicate better performance.

Model	Metric	With Embedding						Without Embedding					
		H				Time	Mem	H				Time	Mem
		96	192	336	720			96	192	336	720		
PDF	MSE	0.183	0.246	0.300	0.403	193.9	2975	**0.176**	**0.241**	0.304	**0.402**	**66.29**	**2877**
	MAE	0.265	0.307	0.342	0.401			**0.258**	**0.301**	0.344	0.403		
ETSformer	MSE	0.267	0.333	0.398	0.501	92.92	2806	**0.189**	**0.255**	**0.318**	**0.432**	**32.23**	**2209**
	MAE	0.372	0.409	0.444	0.495			**0.284**	**0.323**	**0.361**	**0.427**		
PatchTST	MSE	0.183	0.248	**0.311**	**0.407**	58.71	2472	**0.177**	**0.244**	0.314	0.410	**24.58**	**2458**
	MAE	0.263	0.309	0.351	**0.402**			**0.261**	**0.305**	**0.350**	0.408		
MICN	MSE	**0.183**	**0.272**	0.396	0.579	70.61	2845	0.193	0.280	**0.314**	**0.508**	**17.24**	**2836**
	MAE	**0.280**	**0.346**	0.429	0.532			0.292	0.358	**0.350**	**0.498**		
SOFTS	MSE	0.180	0.251	0.315	0.417	30.73	638	**0.179**	**0.247**	**0.309**	**0.411**	**28.43**	**481**
	MAE	0.262	0.309	0.349	0.407			**0.263**	**0.308**	**0.346**	**0.405**		
VarDrop	MSE	0.182	0.250	**0.312**	**0.410**	36.48	499	**0.181**	**0.250**	0.322	0.421	**28.34**	**391**
	MAE	0.266	0.311	**0.350**	**0.405**			**0.265**	**0.309**	0.357	0.410		
Crossformer	MSE	**0.237**	0.450	**0.640**	1.660	232.8	2145	0.257	**0.437**	0.701	**1.520**	**232.7**	**2131**
	MAE	0.342	0.462	**0.548**	0.914			**0.336**	**0.442**	0.602	**0.886**		
iFlashAttention	MSE	**0.182**	0.250	**0.312**	**0.411**	39.93	2389	0.194	**0.249**	0.322	0.416	**33.69**	**2375**
	MAE	**0.266**	0.311	**0.349**	**0.405**			0.279	**0.309**	0.357	0.408		
iFlowformer	MSE	0.183	0.249	0.311	**0.409**	32.70	2391	**0.181**	**0.248**	**0.312**	0.420	**27.12**	**2391**
	MAE	0.269	0.310	0.349	**0.404**			**0.267**	**0.307**	**0.348**	0.410		
PPDformer	MSE	0.188	0.269	0.322	0.417	148.7	2870	**0.180**	**0.254**	**0.308**	**0.407**	**86.57**	**2802**
	MAE	0.276	0.329	0.357	0.411			**0.260**	**0.307**	**0.345**	**0.403**		
LiNo	MSE	0.177	0.244	0.309	0.404	13.98	**2147**	**0.173**	**0.241**	**0.304**	**0.403**	**11.19**	2171
	MAE	0.260	0.304	0.346	**0.398**			**0.256**	**0.301**	**0.342**	0.399		
EDformer	MSE	0.310	0.500	**0.647**	**0.755**	12.34	**2397**	**0.262**	**0.449**	0.668	0.776	**10.93**	2804
	MAE	0.388	0.492	**0.590**	0.637			**0.353**	**0.474**	0.607	**0.617**		
Minusformer	MSE	0.183	0.248	0.309	0.409	41.26	2543	**0.176**	**0.246**	**0.308**	**0.401**	**33.69**	**2848**
	MAE	0.268	0.308	0.347	0.402			**0.260**	**0.304**	**0.345**	**0.400**		
WITRAN	MSE	0.807	1.136	**1.293**	4.448	109.5	943	**0.795**	**1.092**	1.313	**4.439**	107.5	**925**
	MAE	0.722	0.903	**0.916**	1.793			**0.709**	**0.886**	0.965	**1.628**		
Times2D	MSE	0.179	0.241	0.301	0.397	**20.72**	781	**0.175**	**0.240**	**0.300**	**0.394**	21.53	**764**
	MAE	0.263	0.301	0.339	0.394			**0.256**	**0.299**	**0.338**	**0.392**		