

EMAformer: Enhancing Transformer through Embedding Armor for Time Series Forecasting

Zhiwei Zhang¹, Xinyi Du², Xuanchi Guo¹, Weihao Wang¹, Wenjuan Han^{1*},

¹School of Computer Science and Technology, Beijing Jiaotong University, Beijing, China

²Beijing Normal University, Beijing, China

{zhiweizhang, xuanchiguo, weihao, wjhan}@bjtu.edu.cn
{xinyidu}@mail.bnu.edu.cn

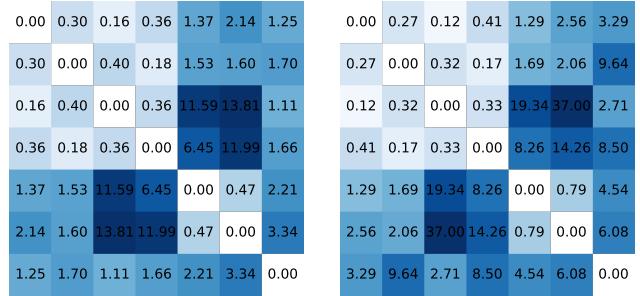
Abstract

Multivariate time series forecasting is crucial across a wide range of domains. While presenting notable progress for the Transformer architecture, iTransformer still lags behind the latest MLP-based models. We attribute this performance gap to unstable inter-channel relationships. To bridge this gap, we propose EMAformer, a simple yet effective model that enhances the Transformer with an auxiliary embedding suite, akin to armor that reinforces its ability. By introducing three key inductive biases, i.e., *global stability*, *phase sensitivity*, and *cross-axis specificity*, EMAformer unlocks the further potential of the Transformer architecture, achieving state-of-the-art performance on 12 real-world benchmarks and reducing forecasting errors by an average of 2.73% in MSE and 5.15% in MAE. This significantly advances the practical applicability of Transformer-based approaches for multivariate time series forecasting. The code is available on <https://github.com/PlanckChang/EMAformer>.

1 Introduction

Multivariate time series forecasting (MTSF) has attracted close attention in both industry and academia, owing to its wide-ranging applications (Qiu et al. 2024; Wang et al. 2024b; Wen et al. 2023; Luo et al. 2024). The core challenge of MTSF lies in the complex inter-channel dependencies and temporal dynamics (Lan et al. 2022; Zhang and Yan 2023; Yu et al. 2024a; Lu et al. 2024). Over a period of time, the Transformer architecture (Vaswani et al. 2017), which has become the de facto standard in many domains (Achiam et al. 2023; Devlin et al. 2019; Dosovitskiy et al. 2021), was unexpectedly outperformed by straightforward Linear/MLP models (Zeng et al. 2023). iTransformer (Liu et al. 2024c) introduces variate tokenization, which adopts an inverted perspective by encoding each channel’s temporal sequence into variate tokens, and effectively addresses concerns about the suitability of the Transformer architecture for MTSF (Zeng et al. 2023; Kim et al. 2024).

However, more recently, MLP-based methods, such as CycleNet (Lin et al. 2024b) and TQNet (Lin et al. 2025), have once again surpassed iTransformer. Notably, all these



(a) CoV of ETTh2.

(b) CoV of ETThm2.

Figure 1: Coefficients of variation (CoV) of the inter-channel correlations. We compute CoV among channels by first measuring the correlations within each day and then computing the mean and standard deviation across days. A CoV value greater than 1 signals substantial variability, indicating that local inter-channel relationships are unstable. These findings imply that vanilla self-attention mechanisms struggle with such rapidly changing dynamics.

models employ variate tokenization, and iTransformer couples self-attention with an MLP. This performance gap naturally raises questions: *why does self-attention struggle in MTSF and why does explicitly modeling inter-channel dependencies bring no forecasting gain?* To explore these, we analyzed inter-channel correlations using the coefficient of variation (CoV) (Everitt and Skrondal 2010), which measures relative fluctuation around the mean of inter-channel correlations. On standard benchmarks, these correlations swing wildly over time, yielding the extremely high CoV values shown in Figure 1. Such volatility implies that local inter-channel relationships are inherently unstable. Specifically, the relationship between two channels can be highly correlated at one moment and nearly uncorrelated the next. Such fluctuations can mislead self-attention mechanisms, resulting in suboptimal performance. In contrast, token interactions in domains where Transformers excel are markedly more stable. In natural language processing, a word token retains its semantic role throughout a sentence (Achiam et al. 2023; Devlin et al. 2019); in computer vision, an image patch preserves a fixed spatial context (Dosovitskiy et al.

*Corresponding author.

Copyright © 2026, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

2021). We therefore contend that the pronounced instability of inter-channel correlations in MTSF causes vanilla self-attention and inter-channel modeling not merely to falter but to backfire. The detailed analysis of CoV is in Appendix A. To address this challenge, we introduce a set of embeddings into the Transformer that embody three key inductive biases.

To counteract unstable local inter-channel correlations, we introduce a *channel embedding* mechanism. Reusing a fixed embedding for each channel and sharing it across all time steps builds a stable, global representation that smooths temporal fluctuations. Equipped with this global context, self-attention can focus on genuinely relevant tokens while suppressing noise and spurious correlations.

To recover temporal details that channel embeddings may blur, we introduce *phase embeddings*, which encode the position of each variate token within its underlying periodic cycle. While inter-channel correlations tend to fluctuate over time, they often align at specific phases, which is a manifestation of periodicity (Lin et al. 2024b; Zhou et al. 2021). By incorporating the phase-sensitive inductive bias, we encourage the model to be time-aware and recognize phase-dependent patterns within these cycles.

To capture this cross-axis specificity, we introduce *joint channel-phase embedding*. Channel embeddings are temporally invariant, which gives every time step in a channel the same global descriptor; meanwhile, phase embeddings are channel-invariant, supplying identical temporal cues across all channels at the same moment. Yet the dependencies in MTSF do not respect such a clean separation; channel and temporal dynamics are tightly entangled. By binding a channel’s global signature to its specific phase, these joint embeddings inject finer-grained information into the model, enabling it to disentangle and effectively capture channel-specific temporal patterns.

Building on the above three insights, we integrate the three embeddings, akin to an **EM**bedding **A**rmor, into **T**ransformer, referred to as **EMAformer**. Concretely, we fuse the embeddings with standard variate token embeddings and feed them into the Transformer encoder backbone. In this way, EMAformer enriches the Transformer’s representational capacity while leaving the underlying architecture untouched. Our main contributions are as follows:

- We revisit the inter-channel correlations in MTSF and trace the Transformer’s performance shortfall to the volatility of channel interactions.
- We propose EMAformer, which introduces three inductive biases, global stability, phase sensitivity, and cross-axis specificity, to expand the Transformer’s capacity without altering its architecture. EMAformer stabilizes inter-channel relations while preserving temporal dynamics.
- Empirically, EMAformer establishes state-of-the-art results on 12 real-world benchmarks, yielding an average accuracy improvement of 2.73% in MSE and 5.15% in MAE. Extensive analyses demonstrate the effectiveness and robustness of our model, further validating the promise of Transformer-based methods for MTSF.

2 Related Work

Given the wealth of excellent work in MTSF (Wen et al. 2022; Wang et al. 2024c; Jin et al. 2024), we adopt the perspective of tokenization to organize related work and position our contributions. Although the term *tokenization* is often tied to Transformer architectures, we deliberately decouple it from a specific backbone in the MTSF context. We focus on the choice of representation units and data processing strategies in Section 2.1. Additionally, we discuss embedding techniques in MTSF in Section 2.2.

2.1 Tokenization Strategies for MTSF

Temporal tokenization. The temporal tokenization manner has a long-standing tradition, tracing back to the application of statistical methods to time series, such as ARIMA (Box et al. 2015). Under this paradigm, all variates are embedded into temporal tokens, and models typically rely on auto-regressive mechanisms to forecast a fixed number of future steps. Early Transformer variants in time series also follow this, employing attention across time steps to model temporal dependencies (Li et al. 2019; Zhou et al. 2021; Wu et al. 2021; Zhou et al. 2022; Woo et al. 2022; Liu et al. 2021).

Variate tokenization. Although the temporal tokenization approach is intuitive, the underwhelming performance of various Transformer derivatives has raised questions about the suitability of the Transformer architecture for MTSF. DLinear (Zeng et al. 2023) was the first to challenge this trend by implicitly adopting a variate tokenization strategy, applying linear layers across the time steps of each channel. This notion of variate tokenization was later explicitly formalized by iTransformer (Liu et al. 2024c), which embeds the entire sequence of time steps of a single variate (or channel) into a variate token. From this perspective, while the debate between Transformer and linear models centers on architectural differences (Zeng et al. 2023), it also reflects choices in tokenization and representation units.

As the focus of representation shifts from the temporal dimension to the channel dimension, methods under variate tokenization can be broadly categorized into *channel-independent* and *channel-dependent* paradigms. Considering the page limit, we have included the detailed literature review of these two paradigms in the Appendix C.

With the basic inter-channel modeling advantage assumption, EMAformer falls within the scope of channel-dependent mechanisms, and our direct predecessor is iTransformer (Liu et al. 2024c). To keep the trend of changing the input representation of Transformer without modifying the architecture on a broader field (Achiam et al. 2023; Devlin et al. 2019; Dosovitskiy et al. 2021), we tailor a suite of auxiliary embeddings to introduce inductive biases for the Transformer within the variate tokenization framework.

2.2 Embedding Techniques for MTSF

In the broader community, embeddings are widely used to help Transformer-based models capture semantic distinctions, such as sentence embeddings in BERT (Devlin et al.

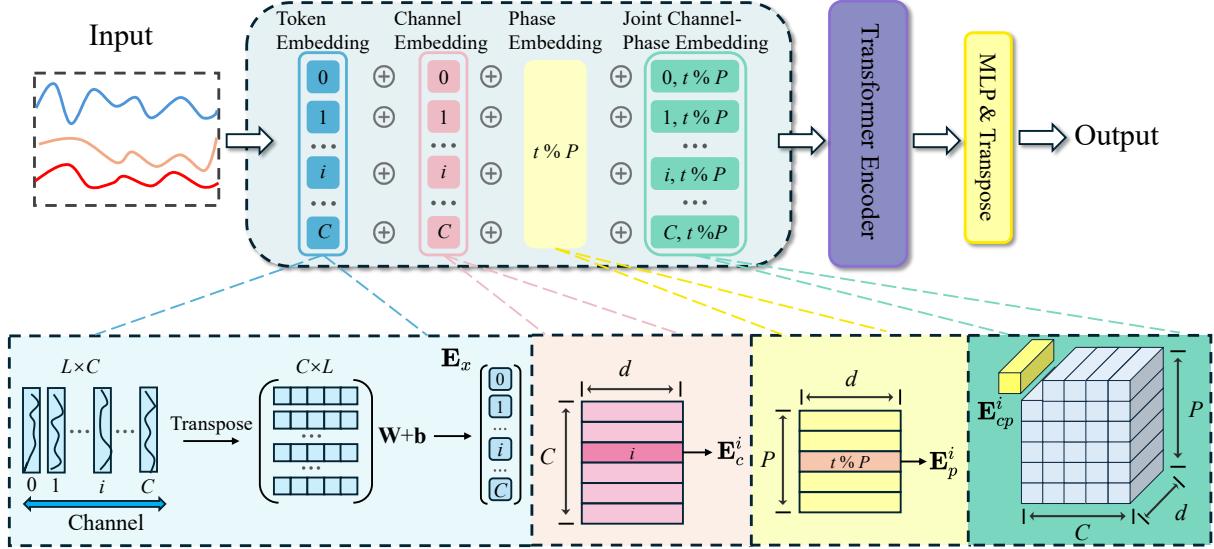


Figure 2: Overview of EMAformer. We enhance a Transformer within variate tokenization framework by integrating three types of auxiliary embeddings: (1) channel embeddings to capture the global representation and stabilize local inter-channel relations; (2) phase embeddings to restore the temporal detail and enhance phase sensitivity; and (3) joint channel-phase embeddings to capture intricate dependencies across channel and temporal dimensions. These are combined with variate token embeddings and processed by the Transformer encoder, effectively augmenting its capacity without modifying its core architecture. The notation % denotes the modulo operator.

2019) and patch position embeddings in ViT (Dosovitskiy et al. 2021). In MTSF, token embeddings are typically the projections of raw data (Zhang and Yan 2023; Liu et al. 2024c). Auxiliary embeddings are commonly used in temporal tokenization; for instance, Informer (Zhou et al. 2021) incorporates timestamp embeddings for week, month, and holiday information, and positional embeddings are more common for encoding the temporal token sequence order (Zhou et al. 2021; Wu et al. 2021).

Within the variate tokenization framework, CycleNet (Lin et al. 2024b) introduces a residual cycle forecasting technique as external parameters to inject cycle information into the MLP. However, according to the original paper’s analysis, this technique does not consistently improve Transformer performance. To locate the position in the cycle and supplement the temporal details, we propose phase embeddings to encode the sequence-level periodic patterns. As demonstrated in our ablation study in Section 4.3, this approach consistently yields performance gains. We find no similar design for the other novel embeddings in MTSF.

3 Method

3.1 Problem Formulation

Given the observation of C variates/channels in the past L time steps, the multivariate time series forecasting task learns a function $f : \mathbb{R}^{L \times C} \rightarrow \mathbb{R}^{H \times C}$ to predict the future

H steps, such that,

$$\hat{\mathbf{Y}}_{t+1:t+H} = f(\mathbf{X}_{t-L+1:t}) \quad (1)$$

$$\mathbf{X}_{t-L+1:t} = [\mathbf{x}_{t-L+1}, \mathbf{x}_{t-L+2}, \dots, \mathbf{x}_t] \quad (2)$$

$$\hat{\mathbf{Y}}_{t+1:t+H} = [\hat{\mathbf{x}}_{t+1}, \hat{\mathbf{x}}_{t+2}, \dots, \hat{\mathbf{x}}_{t+H}] \quad (3)$$

where each $\mathbf{x}_t \in \mathbb{R}^C$. For simplicity, we denote $\mathbf{X}_{t-L+1:t}$ as \mathbf{X} and $\hat{\mathbf{Y}}_{t+1:t+H}$ as $\hat{\mathbf{Y}}$ herein.

3.2 Overview of EMAformer

EMAformer’s three auxiliary embeddings are designed to explicitly capture the key inductive biases. (1) Channel embeddings encode global inter-channel relationships, helping to stabilize local fluctuations; (2) phase embeddings encode phase location and help to capture temporal details; and (3) joint channel-phase embeddings capture the intricate interactions across both channel and temporal dimensions. An overview of our approach is illustrated in Figure 2.

3.3 Variate Token Embedding

In the era of temporal tokenization (Zhou et al. 2021, 2022; Wu et al. 2021; Liu et al. 2021), the multivariate time series models typically treat all variates observed at a single time step as a temporal token. In contrast, iTransformer (Liu et al. 2024c) adopts a different perspective by transposing the input matrix and embedding the time series of each channel as a variate token. This process, known as variate tokenization, is formulated as:

$$\mathbf{E}_x = \mathbf{X}^\top \mathbf{W} + \mathbf{b} \in \mathbb{R}^{C \times d}, \quad (4)$$

where \mathbf{E}_x denotes the resulting variate token embeddings and d is the embedding dimension. Here, $\mathbf{W} \in \mathbb{R}^{L \times d}$ and $\mathbf{b} \in \mathbb{R}^d$ are transformation parameters.

3.4 Channel Embedding

To stabilize each channel’s local representation, we introduce a channel embedding mechanism that explicitly encodes each channel’s identity and global semantic attributes. We define a learnable embedding matrix $\Omega_c \in \mathbb{R}^{C \times d}$, where each row corresponds to a specific channel. The embedding for the i -th channel is retrieved via a lookup operation:

$$\mathbf{E}_c^i = \text{Lookup}(\Omega_c, i) \in \mathbb{R}^{1 \times d}. \quad (5)$$

3.5 Phase Embedding

To restore the temporal details and encode the phase location into the Transformer within variate tokenization, we propose phase embedding. Given a predefined period length P , we construct a learnable embedding matrix $\Omega_p \in \mathbb{R}^{P \times d}$. Unlike channel embeddings, which establish a one-to-one correspondence between tokens and channels, each variate token spans multiple time steps, forming a one-to-many relationship. This introduces ambiguity in determining their exact phase position within the cycle. Since the history window has a fixed length and the sequence is chronologically ordered, every moment within the window can represent the sequence of the channel. Leveraging this property, we assign a phase embedding to each token according to the absolute time t of the last observed step:

$$\mathbf{E}_p^i = \text{Lookup}(\Omega_p, t \bmod P) \in \mathbb{R}^{1 \times d}, \quad (6)$$

where $t \bmod P$ computes the phase within the cycle. Thus, the phase embedding incorporates the sequence-level periodic patterns.

3.6 Joint Channel-Phase Embedding

Channel embeddings are invariant across all time steps, while phase embeddings are invariant across all channels. Since each independently captures channel-wise characteristics and temporal periodicity, they fail to model the intertwined dependencies across these two dimensions.

To address this limitation and capture more fine-grained interactions, we introduce the joint channel-phase embedding. Concretely, we define an embedding tensor $\Omega_{cp} \in \mathbb{R}^{C \times P \times d}$, where each embedding is indexed by the channel i and the phase $t \bmod P$:

$$\mathbf{E}_{cp}^i = \text{Lookup}(\Omega_{cp}, i, t \bmod P) \in \mathbb{R}^{1 \times d}. \quad (7)$$

This joint embedding allows the model to learn channel-specific phase-related patterns that may be missed when channel and phase embeddings are treated independently. Disentangling how different channels respond to temporal cycles provides richer and more expressive representations, particularly beneficial in real-world multivariate time series, where sensors often exhibit distinct periodic behaviors or temporal misalignment (Liu et al. 2024c).

3.7 Network Architecture

Our core contribution lies in the multifaceted embedding strategy designed for the Transformer within variate tokenization, as described above. This subsection details how these enriched embeddings are processed through a transformer-based architecture. As is common practice (Liu et al. 2024c; Lin et al. 2024b, 2025), the network uses an optional normalization-denormalization sandwich structure (Kim et al. 2021).

Given the variate token embedding \mathbf{E}_x , channel embedding \mathbf{E}_c , phase embedding \mathbf{E}_p , and joint channel-phase embedding \mathbf{E}_{cp} , we first construct a comprehensive token representation by element-wise summation:

$$\mathbf{Z}_0 = \mathbf{E}_x + \mathbf{E}_c + \mathbf{E}_p + \mathbf{E}_{cp} \in \mathbb{R}^{C \times d}, \quad (8)$$

where \mathbf{Z}_0 serves as input to the Transformer encoder.

The Transformer encoder consists of N stacked layers, each comprising a multi-head self-attention (MSA) module followed by a position-wise feed-forward network (FFN). Layer normalization (LN) is applied after each module, and residual connections are employed throughout. Formally, the computation at the l -th layer is defined as follows:

$$\mathbf{Z}'_l = \text{LN}(\text{MSA}(\mathbf{Z}_{l-1}) + \mathbf{Z}_{l-1}), \quad (9)$$

$$\mathbf{Z}_l = \text{LN}(\text{FFN}(\mathbf{Z}'_l) + \mathbf{Z}'_l), \quad (10)$$

where $\mathbf{Z}_l \in \mathbb{R}^{C \times d}$ denotes the output of the l -th Transformer layer. More specifically, the MSA operation is formulated as:

$$\text{MSA} = \text{Concat}(\text{head}_1, \dots, \text{head}_h) \mathbf{W}^O, \quad (11)$$

$$\text{head}_k = \text{Attention}(\mathbf{Z} \mathbf{W}_k^Q, \mathbf{Z} \mathbf{W}_k^K, \mathbf{Z} \mathbf{W}_k^V), \quad (12)$$

with $\mathbf{W}_k^Q, \mathbf{W}_k^K, \mathbf{W}_k^V \in \mathbb{R}^{d \times d_h}$ and $\mathbf{W}^O \in \mathbb{R}^{h d_h \times d}$ being projection parameters, where h is the number of attention heads and $d_h = d/h$ is the dimension of each head.

After processing through all N Transformer layers, the final representation \mathbf{Z}_N is then used to generate the forecasting output $\hat{\mathbf{Y}}$ via an MLP and matrix transpose operation. We provide a concise pseudocode in Appendix E.

4 Experiment

4.1 Setup

Dataset. We conducted extensive experiments on 12 benchmarks: ECL, ETT series, Traffic, Weather, Solar-Energy, and PEMS series. The data preprocessing follows the mainstream setting (Liu et al. 2022, 2024c; Lin et al. 2024b, 2025), including the train-valid-test splitting and z-score transformation. We employed the Mean Squared Error (MSE) and the Mean Absolute Error (MAE) as metrics.

Baselines. The experiments compared with the recent state-of-the-art methods, which are TQNet (Lin et al. 2025), TimeXer (Wang et al. 2024d), CycleNet (Lin et al. 2024b), iTransformer (Liu et al. 2024c), TimesNet (Wu et al. 2023), PatchTST (Nie et al. 2023), Crossformer (Zhang and Yan 2023), DLinear (Zeng et al. 2023) and SCINet (Liu et al. 2022).

Model	EMAformer (Ours)		TQNet (2025)		TimeXer (2024d)		CycleNet (2024b)		iTransformer (2024c)		TimesNet (2023)		PatchTST (2023)		Crossformer (2023)		DLinear (2023)		SCINet (2022)	
Metric	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	0.432	0.424	0.441	<u>0.434</u>	<u>0.437</u>	0.437	0.457	0.441	0.454	0.448	0.458	0.450	0.469	0.455	0.529	0.522	0.456	0.452	0.747	0.647
ETTh2	<u>0.373</u>	0.395	0.378	0.402	0.368	<u>0.396</u>	0.388	0.409	0.383	0.407	0.414	0.427	0.387	0.407	0.942	0.684	0.559	0.515	0.954	0.723
ETTm1	0.381	0.380	0.377	<u>0.393</u>	0.382	0.397	<u>0.379</u>	0.396	0.407	0.410	0.400	0.406	0.387	0.400	0.513	0.495	0.403	0.407	0.486	0.481
ETTm2	<u>0.271</u>	0.313	0.277	0.323	0.274	0.322	0.266	<u>0.314</u>	0.288	0.332	0.291	0.333	0.281	0.326	0.757	0.611	0.350	0.401	0.571	0.537
ECL	0.158	0.247	0.164	<u>0.259</u>	0.171	0.270	0.168	<u>0.259</u>	0.178	0.270	0.193	0.295	0.205	0.290	0.244	0.334	0.212	0.300	0.571	0.537
Solar	<u>0.197</u>	0.224	0.198	<u>0.256</u>	0.237	0.302	0.210	0.261	0.233	0.262	0.301	0.319	0.270	0.307	0.641	0.639	0.330	0.401	0.282	0.375
Traffic	<u>0.430</u>	0.255	0.445	<u>0.276</u>	0.466	0.287	0.472	0.301	0.428	0.282	0.620	0.336	0.481	0.300	0.550	0.304	0.625	0.383	0.804	0.509
Weather	0.240	0.262	0.242	<u>0.269</u>	<u>0.241</u>	0.271	0.243	0.271	0.258	0.278	0.259	0.287	0.259	0.273	0.259	0.315	0.265	0.317	0.292	0.363
PEMS03	0.089	0.190	0.097	<u>0.203</u>	0.112	0.214	0.118	0.226	0.113	0.222	0.147	0.248	0.180	0.291	0.169	0.282	0.278	0.375	0.114	0.224
PEMS04	0.081	0.180	0.091	<u>0.197</u>	0.105	0.209	0.119	0.232	0.111	0.221	0.129	0.241	0.195	0.307	0.209	0.314	0.295	0.388	0.093	0.202
PEMS07	0.073	0.162	0.075	<u>0.171</u>	0.085	0.182	0.113	0.214	0.101	0.204	0.125	0.226	0.211	0.303	0.235	0.315	0.329	0.396	0.119	0.217
PEMS08	0.128	0.208	0.142	0.229	0.175	0.250	0.150	0.246	0.150	<u>0.226</u>	0.193	0.271	0.280	0.321	0.268	0.307	0.379	0.416	0.159	0.244
Count	20		1		1		1		1		0		0		0		0		0	

Table 1: Performance comparison of multivariate long-term time series forecasting. Our model achieves the best results in 20 cases and the second-best in 3 cases. Experiments are conducted with a fixed historical window length $L = 96$, and results are averaged over prediction horizons $H \in \{12, 24, 48, 96\}$ for PEMS series or $\{96, 192, 336, 720\}$ for the rest of datasets. The best outcomes are highlighted in **bold**, while the second-best are underlined. Lower values indicate better performance.

Implementation details. All the experiments were conducted on a single GeForce RTX 4090, implemented using PyTorch. We use Adam (Kingma and Ba 2014) and L1 loss for model optimization.

4.2 Main Results

Table 1 compares our model’s multivariate long-term forecasting performance against 9 strong baselines across 12 real-world benchmarks. The complete results with standard deviation are provided in Appendix D. Our model demonstrates clear superiority, achieving the top rank in 20 out of 24 scenarios and placing second in the remaining 3. Compared to the second-best results (or the best results when our model does not rank first), our model yields an average improvement of 2.73% in MSE and 5.15% in MAE. Notably, our model performs exceptionally well on datasets with more than 100 channels, achieving a reduction of 5.07% in MSE and a 7.57% reduction in MAE. The improvements are particularly striking on the PEMS04 and PEMS08 datasets, where our model achieves approximately a 10% performance gain.

4.3 Ablation and Analysis Study

Ablation of the three embeddings. We conducted a comprehensive ablation study to evaluate the efficacy of our proposed embedding strategy. The results, presented as MAE on the radar chart in Figure 3, highlight the comparative performance of different ablation levels. We rescaled the outcomes using max–min normalization (Patro and Sahu 2015), positioning the variant that uses only variate token embeddings at the center (the maximum values), so that points farther from this center represent progressively better performance. The reported ETT and PEMS results represent averages across their respective entire series.

Figure 3 shows that our complete model consistently achieves the best performance. The ablated variants reveal varying degrees of sensitivity, which we attribute to the inherent heterogeneity across these datasets. Overall, combining the three embeddings enables our model to better capture the underlying inductive biases and diverse characteristics.

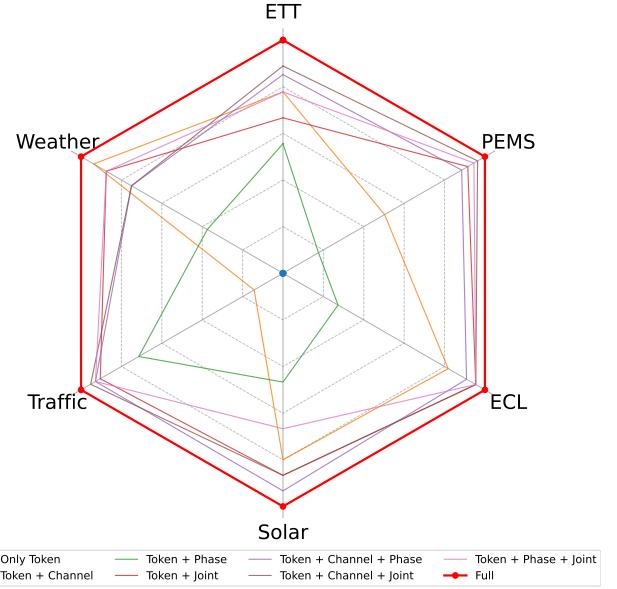


Figure 3: Ablation study. The legends omit the word *embedding* for brevity. Our complete model achieves the best overall performance.

Entropy of attention score. To evaluate the guidance for attention of our embedding strategy, we analyze the attention distribution by calculating the entropy of the attention scores from the last Transformer layer. Specifically, we compute the entropy for each row and report the average entropy across all rows. To isolate temporal effects, we perform this analysis at phase 0 of the cycle. The calculation details are in Appendix B.

As shown in Table 2, the original entropy aligns with the CoV-based analysis of inter-channel correlations in Figure 1. For example, the ETT series exhibits a nearly uniform distribution, as indicated by its entropy approaching the maximum value for 7 variates (~ 2.81).

The addition of channel embeddings reduces the entropy,

Dataset	Only Token	Token+Channel	Full
ETTh1	2.360	1.902	1.910
ETTh2	2.358	1.867	1.877
ETTm1	2.367	1.897	1.901
ETTm2	2.290	1.716	1.742
ECL	5.690	5.494	5.568
Solar	4.763	4.664	4.668
Traffic	6.517	4.766	5.567
Weather	3.076	2.782	2.852

Table 2: Entropy of the attention score. According to entropy changes, it turns out that our embedding scheme preserves periodicity while enabling the channel to focus more on information that is truly beneficial for prediction.

indicating a more concentrated attention distribution. This means the model can focus more effectively on the truly relevant channels, which aligns with the performance improvements observed in our ablation study.

Interestingly, when we incorporate the full embedding (including phase information), the entropy increases slightly compared to using channel embeddings alone but still remains lower than the original model without these embeddings. This is reasonable, as correlations between channels fluctuate throughout the day, so at certain phases of the cycle, increased entropy may reflect the model capturing additional informative variations. This explains why introducing phase sensitivity continues to boost performance.

Improvement of other Transformer variants. Table 3 presents an analysis of three Transformer variants: Reformer (Kitaev, Kaiser, and Levskaya 2020), Informer (Zhou et al. 2021), and Flowformer (Wu et al. 2022), and all augmented with our proposed embedding strategy. The experiments follow the same settings as our main experiments, averaging performance across four forecasting horizons. Table 3 reports the baseline results (Original), improvements achieved by applying the inverted mechanism (+Inverted), and the additional gains from integrating our embedding strategy (+Embedding). We also include the relative improvement over the +Inverted variant to quantify the incremental benefit of our method. These promotion percentages clearly demonstrate the generalization ability and effectiveness of the proposed embeddings.

Performance of replacing Transformer with MLP. We evaluate our embedding strategy by isolating its effect from the backbone architecture. Specifically, we remove the Transformer backbone and retain only the prediction head, while keeping all other hyperparameters fixed. This allows us to compare our MLP-based version directly with the strongest existing MLP baselines, as shown in Figure 4.

The results clearly demonstrate that the Transformer-based version consistently outperforms its MLP counterpart, highlighting the necessity of the Transformer backbone. Furthermore, when using the same backbone, our MLP model equipped with the proposed embeddings surpasses the strongest MLP baselines in almost all cases, except on the PEMS07 dataset, where TQNet slightly outperforms our

	Models Metric	Reformer		Informer		Flowformer	
		MSE	MAE	MSE	MAE	MSE	MAE
ECL	Original	0.338	0.422	0.311	0.397	0.267	0.359
	+Inverted	0.208	0.301	0.216	0.311	0.21	0.293
	+Embedding	0.186	0.269	0.186	0.268	0.187	0.269
	Promotion	10.58%	10.63%	13.89%	13.83%	10.95%	8.19%
Traffic	Original	0.741	0.422	0.764	0.416	0.75	0.421
	+Inverted	0.647	0.37	0.662	0.38	0.524	0.355
	+Embedding	0.494	0.311	0.496	0.312	0.49	0.31
	Promotion	23.65%	15.95%	25.08%	17.89%	6.49%	12.68%
Weather	Original	0.803	0.656	0.634	0.548	0.286	0.308
	+Inverted	0.248	0.292	0.271	0.33	0.266	0.285
	+Embedding	0.245	0.265	0.241	0.263	0.244	0.265
	Promotion	1.21%	9.25%	11.07%	20.30%	8.27%	7.02%

Table 3: Performance of Transformer variants with our embeddings. The proposed embedding consistently enhances performance beyond the inverted mechanism, demonstrating its effectiveness in further reducing forecasting errors.

method. We attribute this to hyperparameter sensitivity: the MLP and Transformer backbones generally require distinct hyperparameter settings, but to ensure a fair comparison, we replaced the Transformer with an MLP without retuning hyperparameters. Overall, these experiments highlight the robustness and effectiveness of our embedding strategy.

Performance of removing local historical information. To investigate the global patterns captured by our embedding, we conducted an experiment in which we replaced the input series with its mean (effectively zero, due to the pre-normalization (Kim et al. 2021)), as illustrated in Figure 5. Naturally, significant performance degradation was anticipated in this extreme scenario, owing to the complete absence of historical information. Surprisingly, across four datasets, although prediction accuracy declined to some extent, performance still matched or even surpassed some baselines proposed in 2022 and 2023. This experiment highlights the robustness of our embeddings in capturing dataset-level properties and global relationships. Remarkably, it suggests that even with minimal historical context, forecasting the future using additional embeddings alone seems viable.

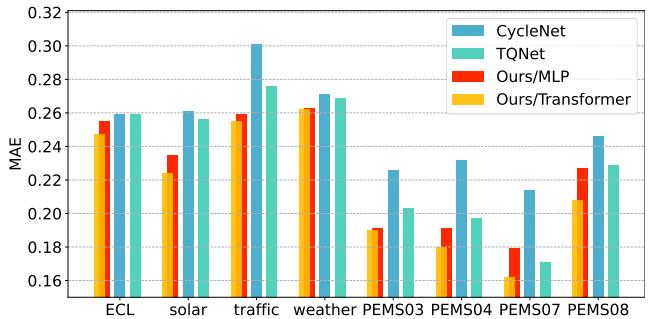


Figure 4: Comparison of our embedding strategy by replacing the Transformer backbone with MLP, against the strongest existing MLP baselines. Our MLP variant outperforms other MLP baselines across almost all datasets, underscoring the effectiveness of our embedding design.

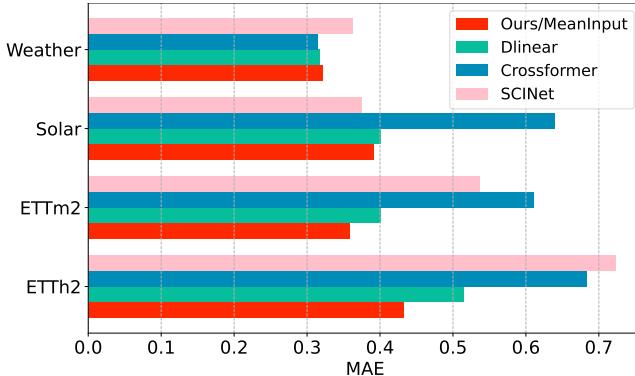


Figure 5: Replacing the input series with its mean values to eliminate the history information. Our model still matches or outperforms some baselines.

Visualization of learned representation. To investigate the representations learned by the auxiliary embeddings, we employ T-SNE (Van der Maaten and Hinton 2008) to visualize them in two-dimensional space. Figure 6a illustrates how the channel representations are distributed, which clearly reaches the modeling goal of clustering the relevant channels and separating the discrepant ones. These representations are reused across channels over time, leveraging the global, stable inter-channel relationship to guide the backbone in attending to stably relevant channels.

Figure 6b presents the phase embeddings (LEFT) and the joint channel-phase embeddings (RIGHT, for Channel 0) on the Traffic dataset. The color intensity of the points indicates their distance from the center of the predefined cycle. Both visualizations reveal the periodicity along the temporal dimension. Notably, the joint embeddings (RIGHT) uncover channel-specific, fine-grained patterns that reveal a multi-periodic cycle in which the daily cycle is nested within the weekly one. While the overall patterns are clear, there are also intermittent scattered points, reflecting short-term volatility and local noise.

In summary, these visualizations demonstrate that our model effectively captures both channel and temporal structures, which explains the model’s robust performance across different scenarios.

Period length P . We investigate the selection of the hyperparameter period length P , with daily and weekly cycles chosen as candidate values based on common temporal patterns. Although a longer cycle can implicitly capture shorter cycles, i.e., the weekly period encompassing the daily patterns, using a weekly cycle reduces the number of samples available per phase, which can impact learning. Therefore, we select the period length based on which option yields better empirical performance. The results are summarized in Table 4, with detailed period statistics provided in Table 6.

5 Conclusion and Future Work

In this paper, we extend the prevailing trend of enhancing Transformer-based models by modifying the input represen-

Dataset	Daily Period		Weekly Period	
	MSE	MAE	MSE	MAE
ETTh1	0.432	0.424	0.434	0.426
ETTh2	0.373	0.395	0.374	0.395
ETTm1	0.381	0.380	0.384	0.382
ETTm2	0.271	0.313	0.273	0.314
ECL	0.160	0.248	0.158	0.247
Solar	0.197	0.224	0.215	0.234
Traffic	0.440	0.260	0.430	0.255
Weather	0.240	0.262	0.243	0.265

Table 4: Comparison of daily vs. weekly period lengths.

tations rather than altering the powerful network architecture itself. Specifically, we introduce a suite of embedding armor within the variate tokenization framework to incorporate critical inductive biases into the Transformer backbone. Our proposed EMAformer achieves state-of-the-art performance across multiple benchmarks, while deliberately preserving flexibility, thereby paving the way for future research on complementary architectural innovations. We hope our work inspires the development of Transformer-based and LLM-based approaches in time series analysis.

In the future, we plan to explore the multi-period patterns illustrated in Figure 6b, which exhibit more complex nested structures and offer promising opportunities for deeper investigation. Additionally, given the fixed period length assumed in this paper, we plan to incorporate adaptive cycles to better capture varying temporal dynamics. Finally, regarding the channel dimension, we intend to investigate explicit topological structures across channels to better guide the attention mechanism.

Acknowledgment

The work described in this paper has been supported by the National Nature Science Foundation of China (No. 62406020), CHN RAILWAY GRANT Number # L2023G013, and Fundamental Research Funds for the Central Universities under Grant 2025JBZX058.

References

- Achiam, J.; Adler, S.; Agarwal, S.; Ahmad, L.; Akkaya, I.; Aleman, F. L.; Almeida, D.; Altenschmidt, J.; Altman, S.; Anadkat, S.; et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.
- Box, G. E.; Jenkins, G. M.; Reinsel, G. C.; and Ljung, G. M. 2015. *Time series analysis: forecasting and control*. John Wiley & Sons.
- Challu, C.; Olivares, K. G.; Oreshkin, B. N.; Ramirez, F. G.; Canseco, M. M.; and Dubrawski, A. 2023. Nhits: Neural hierarchical interpolation for time series forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, 6989–6997.
- Chen, S.-A.; Li, C.-L.; Yoder, N.; Arik, S. O.; and Pfister, T. 2023. Tsmixer: An all-mlp architecture for time series forecasting. *arXiv preprint arXiv:2303.06053*.

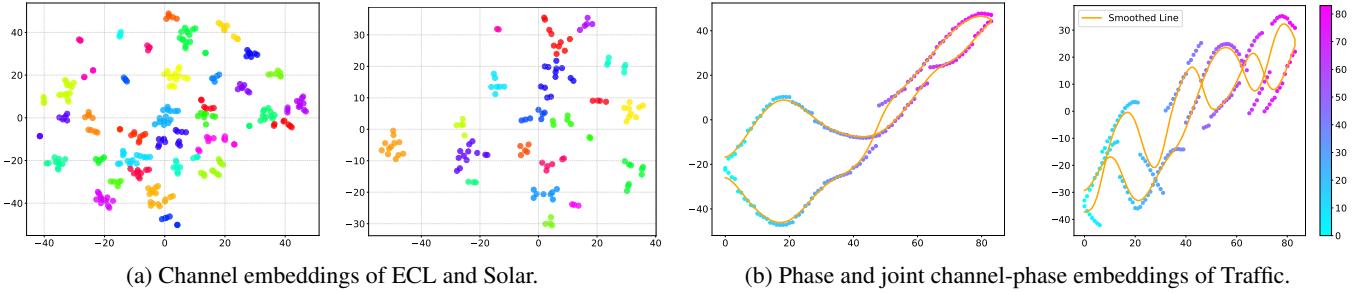


Figure 6: Visualization of learned representation.

Dai, T.; Wu, B.; Liu, P.; Li, N.; Bao, J.; Jiang, Y.; and Xia, S.-T. 2024. Periodicity decoupling framework for long-term series forecasting. In *The Twelfth International Conference on Learning Representations*.

Das, A.; Kong, W.; Leach, A.; Mathur, S. K.; Sen, R.; and Yu, R. 2023. Long-term Forecasting with TIDE: Time-series Dense Encoder. *Transactions on Machine Learning Research*.

Deng, J.; Ye, F.; Yin, D.; Song, X.; Tsang, I.; and Xiong, H. 2024. Parsimony or Capability? Decomposition Delivers Both in Long-term Time Series Forecasting. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, 4171–4186.

Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; Uszkoreit, J.; and Houlsby, N. 2021. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. *ICLR*.

Ekambaram, V.; Jati, A.; Nguyen, N.; Sinthong, P.; and Kalagnanam, J. 2023. Tsmixer: Lightweight mlp-mixer model for multivariate time series forecasting. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 459–469.

Everitt, B. S.; and Skrondal, A. 2010. *The Cambridge dictionary of statistics*, volume 4. Cambridge university press Cambridge, UK.

Han, L.; Chen, X.-Y.; Ye, H.-J.; and Zhan, D.-C. 2024. SOFTS: Efficient Multivariate Time Series Forecasting with Series-Core Fusion. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Huang, Q.; Shen, L.; Zhang, R.; Cheng, J.; Ding, S.; Zhou, Z.; and Wang, Y. 2024. HDMixer: hierarchical dependency with extendable patch for multivariate time series forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, 12608–12616.

Huang, Q.; Shen, L.; Zhang, R.; Ding, S.; Wang, B.; Zhou, Z.; and Wang, Y. 2023. Crossgnn: Confronting noisy mul-

tivariate time series via cross interaction refinement. *Advances in Neural Information Processing Systems*, 36.

Ilbert, R.; Odonnat, A.; Feofanov, V.; Virmaux, A.; Paolo, G.; Palpanas, T.; and Redko, I. 2024. SAMformer: Unlocking the Potential of Transformers in Time Series Forecasting with Sharpness-Aware Minimization and Channel-Wise Attention. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235. PMLR.

Jin, M.; Koh, H. Y.; Wen, Q.; Zambon, D.; Alippi, C.; Webb, G. I.; King, I.; and Pan, S. 2024. A survey on graph neural networks for time series: Forecasting, classification, imputation, and anomaly detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Kim, D.; Park, J.; Lee, J.; and Kim, H. 2024. Are Self-Attentions Effective for Time Series Forecasting? In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

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

Kingma, D. P.; and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

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

Lan, X.; Ng, D.; Hong, S.; and Feng, M. 2022. Intra-Inter Subject Self-Supervised Learning for Multivariate Cardiac Signals. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(4): 4532–4540.

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

Lin, S.; Chen, H.; Wu, H.; Qiu, C.; and Lin, W. 2025. Temporal Query Network for Efficient Multivariate Time Series Forecasting. In *Forty-second International Conference on Machine Learning*.

Lin, S.; Lin, W.; Wu, W.; Chen, H.; and Yang, J. 2024a. SparseTSF: Modeling Long-term Time Series Forecasting with 1k Parameters. In *Forty-first International Conference on Machine Learning*.

Lin, S.; Lin, W.; Xinyi, H.; Wu, W.; Mo, R.; and Zhong, H. 2024b. CycleNet: Enhancing Time Series Forecasting

- through Modeling Periodic Patterns. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.
- Liu, J.; Liu, C.; Woo, G.; Wang, Y.; Hooi, B.; Xiong, C.; and Sahoo, D. 2024a. UniTST: Effectively Modeling Inter-Series and Intra-Series Dependencies for Multivariate Time Series Forecasting. *arXiv preprint arXiv:2406.04975*.
- Liu, M.; Zeng, A.; Chen, M.; Xu, Z.; Lai, Q.; Ma, L.; and Xu, Q. 2022. Scinet: Time series modeling and forecasting with sample convolution and interaction. *Advances in Neural Information Processing Systems*, 35: 5816–5828.
- Liu, P.; Wu, B.; Hu, Y.; Li, N.; Dai, T.; Bao, J.; and Xia, S.-t. 2024b. TimeBridge: Non-Stationarity Matters for Long-term Time Series Forecasting. *arXiv preprint arXiv:2410.04442*.
- Liu, S.; Yu, H.; Liao, C.; Li, J.; Lin, W.; Liu, A. X.; and Dustdar, S. 2021. Pyraformer: Low-complexity pyramidal attention for long-range time series modeling and forecasting. In *International conference on learning representations*.
- Liu, Y.; Hu, T.; Zhang, H.; Wu, H.; Wang, S.; Ma, L.; and Long, M. 2024c. iTransformer: Inverted Transformers Are Effective for Time Series Forecasting. In *The Twelfth International Conference on Learning Representations*.
- Lu, J.; Han, X.; Sun, Y.; and Yang, S. 2024. CATS: Enhancing Multivariate Time Series Forecasting by Constructing Auxiliary Time Series as Exogenous Variables. In Salakhutdinov, R.; Kolter, Z.; Heller, K.; Weller, A.; Oliver, N.; Scarlett, J.; and Berkenkamp, F., eds., *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, 32990–33006. PMLR.
- Luo, D.; and Wang, X. 2024. ModernTCN: A modern pure convolution structure for general time series analysis. In *The Twelfth International Conference on Learning Representations*.
- Luo, Y.; Liu, Z.; Wang, L.; Wu, B.; Zheng, J.; and Ma, Q. 2024. Knowledge-Empowered Dynamic Graph Network for Irregularly Sampled Medical Time Series. *Advances in Neural Information Processing Systems*, 37: 67172–67199.
- Nie, Y.; H. Nguyen, N.; Sinthong, P.; and Kalagnanam, J. 2023. A Time Series is Worth 64 Words: Long-term Forecasting with Transformers. In *International Conference on Learning Representations*.
- Patro, S.; and Sahu, K. K. 2015. Normalization: A preprocessing stage. *arXiv preprint arXiv:1503.06462*.
- Qiu, X.; Hu, J.; Zhou, L.; Wu, X.; Du, J.; Zhang, B.; Guo, C.; Zhou, A.; Jensen, C. S.; Sheng, Z.; and Yang, B. 2024. TFB: Towards Comprehensive and Fair Benchmarking of Time Series Forecasting Methods. *Proc. VLDB Endow.*, 17(9): 2363–2377.
- Qiu, X.; Wu, X.; Lin, Y.; Guo, C.; Hu, J.; and Yang, B. 2025. DUET: Dual Clustering Enhanced Multivariate Time Series Forecasting. In *SIGKDD*.
- Van der Maaten, L.; and Hinton, G. 2008. Visualizing data using t-SNE. *Journal of machine learning research*, 9(11).
- Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.
- Wang, H.; Peng, J.; Huang, F.; Wang, J.; Chen, J.; and Xiao, Y. 2023. Micn: Multi-scale local and global context modeling for long-term series forecasting. In *International Conference on Learning Representations*.
- Wang, S.; Wu, H.; Shi, X.; Hu, T.; Luo, H.; Ma, L.; Zhang, J. Y.; and ZHOU, J. 2024a. TimeMixer: Decomposable Multiscale Mixing for Time Series Forecasting. In *The Twelfth International Conference on Learning Representations*.
- Wang, Y.; Wu, H.; Dong, J.; Liu, Y.; Long, M.; and Wang, J. 2024b. Deep time series models: A comprehensive survey and benchmark. *arXiv preprint arXiv:2407.13278*.
- Wang, Y.; Wu, H.; Dong, J.; Liu, Y.; Long, M.; and Wang, J. 2024c. Deep time series models: A comprehensive survey and benchmark. *arXiv preprint arXiv:2407.13278*.
- Wang, Y.; Wu, H.; Dong, J.; Qin, G.; Zhang, H.; Liu, Y.; Qiu, Y.; Wang, J.; and Long, M. 2024d. Timexer: Empowering transformers for time series forecasting with exogenous variables. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.
- Wen, Q.; Zhou, T.; Zhang, C.; Chen, W.; Ma, Z.; Yan, J.; and Sun, L. 2022. Transformers in time series: A survey. *arXiv preprint arXiv:2202.07125*.
- Wen, Q.; Zhou, T.; Zhang, C.; Chen, W.; Ma, Z.; Yan, J.; and Sun, L. 2023. Transformers in time series: a survey. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence*, 6778–6786.
- Woo, G.; Liu, C.; Sahoo, D.; Kumar, A.; and Hoi, S. 2022. Etsformer: Exponential smoothing transformers for time-series forecasting. *arXiv preprint arXiv:2202.01381*.
- Wu, H.; Hu, T.; Liu, Y.; Zhou, H.; Wang, J.; and Long, M. 2023. Timesnet: Temporal 2d-variation modeling for general time series analysis. In *International Conference on Learning Representations*.
- Wu, H.; Wu, J.; Xu, J.; Wang, J.; and Long, M. 2022. Flowformer: Linearizing Transformers with Conservation Flows. In Chaudhuri, K.; Jegelka, S.; Song, L.; Szepesvari, C.; Niu, G.; and Sabato, S., eds., *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, 24226–24242. PMLR.
- Wu, H.; Xu, J.; Wang, J.; and Long, M. 2021. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. *Advances in neural information processing systems*, 34: 22419–22430.
- Xu, Z.; Zeng, A.; and Xu, Q. 2024. FITS: Modeling Time Series with 10k Parameters. In *The Twelfth International Conference on Learning Representations*.
- Xue, W.; Zhou, T.; Wen, Q.; Gao, J.; Ding, B.; and Jin, R. 2024. CARD: Channel Aligned Robust Blend Transformer for Time Series Forecasting. In *International Conference on Learning Representations (ICLR)*.

Yi, K.; Zhang, Q.; Fan, W.; He, H.; Hu, L.; Wang, P.; An, N.; Cao, L.; and Niu, Z. 2023. FourierGNN: Rethinking multivariate time series forecasting from a pure graph perspective. *Advances in Neural Information Processing Systems*, 36.

Yu, G.; Zou, J.; Hu, X.; Aviles-Rivero, A. I.; Qin, J.; and Wang, S. 2024a. Revitalizing multivariate time series forecasting: Learnable decomposition with inter-series dependencies and intra-series variations modeling. *arXiv preprint arXiv:2402.12694*.

Yu, G.; Zou, J.; Hu, X.; Aviles-Rivero, A. I.; Qin, J.; and Wang, S. 2024b. Revitalizing Multivariate Time Series Forecasting: Learnable Decomposition with Inter-Series Dependencies and Intra-Series Variations Modeling. In *Forty-first International Conference on Machine Learning*.

Zeng, A.; Chen, M.; Zhang, L.; and Xu, Q. 2023. Are transformers effective for time series forecasting? In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, 11121–11128.

Zhang, T.; Zhang, Y.; Cao, W.; Bian, J.; Yi, X.; Zheng, S.; and Li, J. 2022. Less is more: Fast multivariate time series forecasting with light sampling-oriented mlp structures. *arXiv preprint arXiv:2207.01186*.

Zhang, Y.; and Yan, J. 2023. Crossformer: Transformer Utilizing Cross-Dimension Dependency for Multivariate Time Series Forecasting. In *International Conference on Learning Representations*.

Zhou, H.; Zhang, S.; Peng, J.; Zhang, S.; Li, J.; Xiong, H.; and Zhang, W. 2021. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, 11106–11115.

Zhou, T.; Ma, Z.; Wen, Q.; Wang, X.; Sun, L.; and Jin, R. 2022. Fedformer: Frequency enhanced decomposed transformer for long-term series forecasting. In *International conference on machine learning*, 27268–27286. PMLR.

Appendix for “EMAformer: Enhancing Transformer through Embedding Armor for Time Series Forecasting”

A Details of Coefficients of Variation

In the main body, we describe our motivation by quantifying the variability of inter-channel relationships. Specifically, we calculate the coefficient of variation (CoV) on the correlation matrix. This section provides a detailed explanation of the CoV calculation process used in Figure 1.

Formally, given raw data with C channels for day n , we compute the pairwise Pearson correlation coefficients to obtain a correlation matrix $R^{(n)} \in \mathbb{R}^{C \times C}$, where

$$R_{i,j}^{(n)} = \frac{\mathbb{E}[(X_i^{(n)} - \mu_{X_i^{(n)}})(X_j^{(n)} - \mu_{X_j^{(n)}})]}{\sigma_{X_i^{(n)}} \sigma_{X_j^{(n)}}}. \quad (13)$$

We then focus on the correlation pairs across all days. For each pair (i, j) , we collect the sequence $\{R_{i,j}^{(n)}\}_{n=1}^N$ over N days (according to the dataset size). The mean and standard deviation across days for each pair are computed as

$$\mu_{i,j} = \frac{1}{N} \sum_{d=1}^N R_{i,j}^{(n)}, \quad (14)$$

$$\sigma_{i,j} = \sqrt{\frac{1}{N} \sum_{d=1}^N (R_{i,j}^{(n)} - \mu_{i,j})^2}. \quad (15)$$

Finally, we calculate the overall CoV across all channel pairs by

$$\text{CoV} = \frac{\sigma}{\mu} \in \mathbb{R}^{C \times C}. \quad (16)$$

We give the CoV visualization for the ETTh2 and ETTm2 datasets in Figure 1. We provide the intermediate result in Figure 7. This indicates that local correlations are unstable. Specifically, the relationship between two channels may be strongly correlated at one moment and barely correlated at another. This can be highly misleading for self-attention mechanisms, ultimately leading to suboptimal performance. To address this issue, this paper proposes a comprehensive set of embedding techniques that progressively mitigate the problem, thereby improving the model’s performance.

B Details of Entropy Experiment

We provide some details about our entropy comparison experiment in Section 4.3.

Let $A^{(k)} \in \mathbb{R}^{C \times C}$ be the attention matrix for head k in MSA, where C is the number of channels, and $\bar{A} \in \mathbb{R}^{C \times C}$ be the average attention matrix over all heads, where

$$\bar{A} = \frac{1}{h} \sum_{k=1}^h A^{(k)}. \quad (17)$$

Each row \bar{A}_i of \bar{A} is a probability distribution over the channels that channel i attends to.

Entropy of each row $i \in 1, \dots, C$:

$$H_i = - \sum_{j=1}^C \bar{A}_{ij} \log(\bar{A}_{ij}). \quad (18)$$

Average entropy over all rows

$$H_{\text{avg}} = \frac{1}{C} \sum_{i=1}^C H_i \quad (19)$$

$$= - \frac{1}{C} \sum_{i=1}^C \sum_{j=1}^C \bar{A}_{ij} \log(\bar{A}_{ij}). \quad (20)$$

Final formula using the definition of \bar{A}_{ij} as the average across heads

$$H_{\text{avg}} = - \frac{1}{C} \sum_{i=1}^C \sum_{j=1}^C \left(\frac{1}{h} \sum_{k=1}^h A_{ij}^{(k)} \right) \log \left(\frac{1}{h} \sum_{k=1}^h A_{ij}^{(k)} \right). \quad (21)$$

The maximum entropy of a discrete probability distribution is given by:

$$H_{\text{max}} = \log C. \quad (22)$$

This maximum is achieved when the distribution is uniform, i.e., the channel pays even attention to others. In the case of the ETT series datasets, where the number of channels is 7, the maximum entropy is approximately $\log 7 \approx 2.81$. The closer the entropy gets to its maximum value, the more trivial the solution of the self-attention becomes.

C Related Work About Variate Tokenization

The channel-independent mechanism was first introduced by PatchTST (Nie et al. 2023), which segments temporal sequences into patches and applies attention within each channel independently. This approach has inspired a series of follow-up works (Das et al. 2023; Challu et al. 2023; Xu, Zeng, and Xu 2024; Wang et al. 2024a; Dai et al. 2024; Kim et al. 2024; Lin et al. 2024a,b).

In contrast, the channel-dependent mechanism aims to capture inter-channel relationships. This paradigm centers on modeling interactions among channels, employing techniques such as attention (Wang et al. 2024d; Liu et al. 2024c; Zhang and Yan 2023; Xue et al. 2024; Ilbert et al. 2024; Yu et al. 2024b; Liu et al. 2024a; Deng et al. 2024; Liu et al. 2024b; Qiu et al. 2025), graph neural networks (Huang et al. 2023; Yi et al. 2023), convolutional networks (Wang et al. 2023; Wu et al. 2023; Luo and Wang 2024), and MLP-based architectures (Zhang et al. 2022; Chen et al. 2023; Ekambaram et al. 2023; Huang et al. 2024; Han et al. 2024).

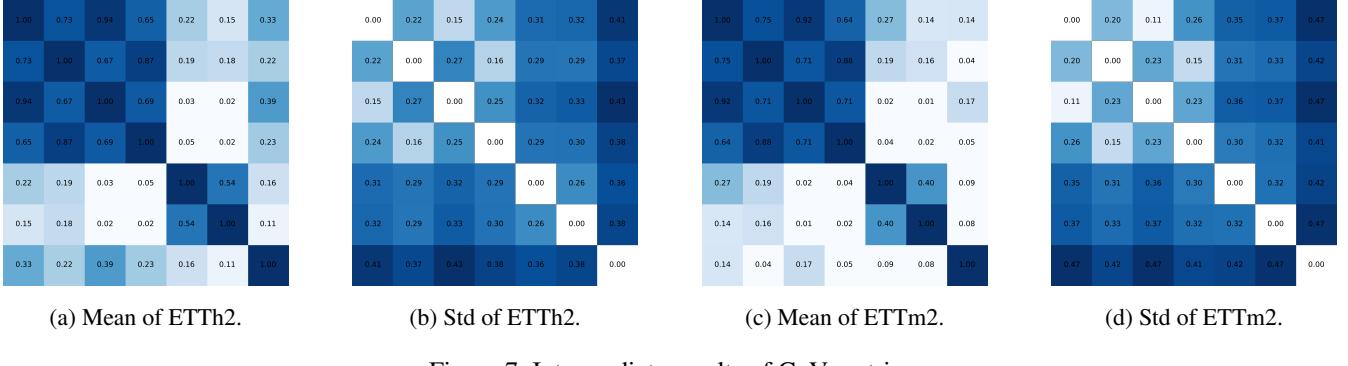


Figure 7: Intermediate results of CoV matrix.

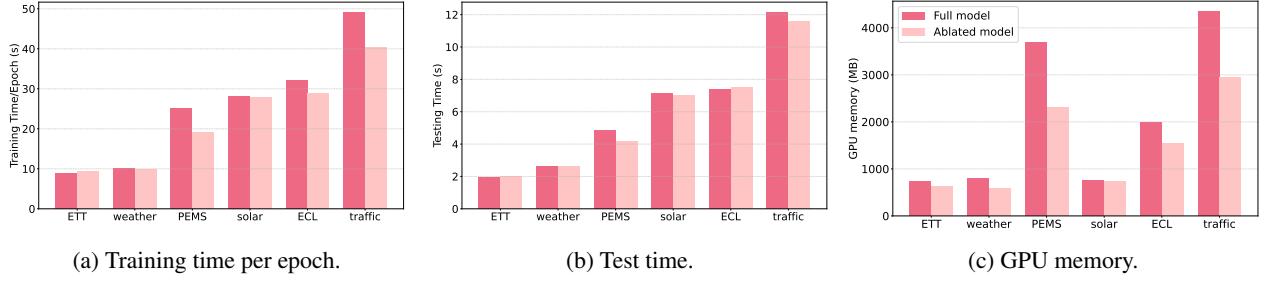


Figure 8: Resource overhead comparison.

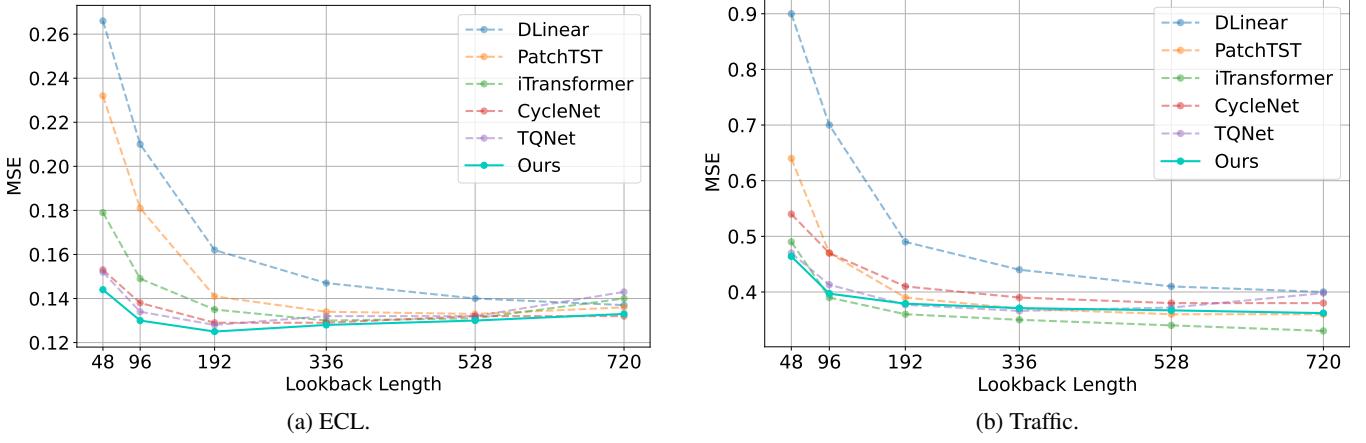


Figure 9: Comparison of MSE for varying lookback lengths on the ECL and Traffic datasets.

D Additional Experiments

Impact of varying history windows. Figure 9 illustrates the prediction errors for different historical window lengths L on the ECL and Traffic datasets. The prediction length is fixed at $H = 96$. Our method consistently achieves superior performance, particularly on the ECL dataset. Our model maintains competitive performance on the Traffic dataset, ranking closely with iTransformer, especially as the look-back length increases. This indicates that our approach effectively utilizes historical information without requiring excessively long input sequences. Intuitively, longer history windows provide more information, which generally enhances prediction performance. However, we observe that the benefits of extending the lookback length diminish beyond 336. Leveraging longer historical dependencies effectively remains an interesting challenge for future research.

Resource overhead. We measure the additional resource overhead of the three auxiliary embeddings. We average the results across four prediction steps for all datasets, with the ETT and PEMS series further averaged across their respective subsets. Figure 8 reveals that the difference in training time per epoch and testing time is less noticeable on datasets with fewer channels. The increase in memory usage is relatively significant, stemming from the additional embedding parameters. Overall, these costs remain acceptable for a consumer-grade GPU. Considering the substantial performance improvement, the resource consumption comparison experiment again validates the effectiveness of our model.

Complete results. We report the complete forecasting results of four steps in Table 5.

E Complete Algorithm Procedure

We provide a pseudocode of EMAformer in Algorithm 1.

F Dataset Statistics

We provide basic statistics for benchmarks, shown in Table 6.

Dataset	# Channels	# Time steps	Frequency	# Period
ETTh1	7	14,400	1 hour	24
ETTh2	7	14,400	1 hour	24
ETTm1	7	57,600	15 mins	96
ETTm2	7	57,600	15 mins	96
ECL	321	26,304	1 hour	168
Solar	137	52,560	10 mins	144
Traffic	862	17,544	1 hour	168
Weather	21	52,696	10 mins	144
PEMS03	358	26,208	5 mins	288
PEMS04	307	16,992	5 mins	288
PEMS07	883	28,224	5 mins	288
PEMS08	170	17,856	5 mins	288

Table 6: Statistics of benchmark datasets.

Algorithm 1: The pseudocode of EMAformer.

Input: Observation $\mathbf{X} \in \mathbb{R}^{L \times C}$;
 Last time step t of the observation window;
 period length P

Output: forecast $\hat{\mathbf{Y}} \in \mathbb{R}^{H \times C}$

```

1: // Variate Tokenization
2:  $\mathbf{E}_x \leftarrow \mathbf{X}^\top \mathbf{W} + \mathbf{b} \in \mathbb{R}^{C \times d}$ 
3: // Channel embedding
4: for  $i = 1$  to  $C$  do
5:    $\mathbf{E}_c^i \leftarrow \text{Lookup}(\Omega_c, i) \in \mathbb{R}^{1 \times d}$ 
6: end for
7:  $\mathbf{E}_c \leftarrow \text{concat}(\mathbf{E}_c^1, \dots, \mathbf{E}_c^C) \in \mathbb{R}^{C \times d}$ 
8: // Phase embedding
9:  $t \leftarrow$  time stamp of the last observed step
10: for  $i = 1$  to  $C$  do
11:    $\mathbf{E}_p^i \leftarrow \text{Lookup}(\Omega_p, t \bmod P) \in \mathbb{R}^{1 \times d}$ 
12: end for
13:  $\mathbf{E}_p \leftarrow \text{concat}(\mathbf{E}_p^1, \dots, \mathbf{E}_p^C) \in \mathbb{R}^{C \times d}$ 
14: // Joint channel-phase embedding
15: for  $i = 1$  to  $C$  do
16:    $\mathbf{E}_{cp}^i \leftarrow \text{Lookup}(\Omega_{cp}, i, t \bmod P) \in \mathbb{R}^{1 \times d}$ 
17: end for
18:  $\mathbf{E}_{cp} \leftarrow \text{concat}(\mathbf{E}_{cp}^1, \dots, \mathbf{E}_{cp}^C) \in \mathbb{R}^{C \times d}$ 
19: // Embedding fusion
20:  $\mathbf{Z}_0 \leftarrow \mathbf{E}_x + \mathbf{E}_c + \mathbf{E}_p + \mathbf{E}_{cp}$ 
21: // Transformer encoder
22: for  $l = 1$  to  $N$  do
23:    $\mathbf{Z}'_l \leftarrow \text{MSA}(\text{LN}(\mathbf{Z}_{l-1})) + \mathbf{Z}_{l-1}$ 
24:    $\mathbf{Z}_l \leftarrow \text{FFN}(\text{LN}(\mathbf{Z}'_l)) + \mathbf{Z}'_l$ 
25: end for
26: // Output projection
27:  $\hat{\mathbf{Y}} \leftarrow (\text{MLP}(\mathbf{Z}_N))^\top \in \mathbb{R}^{H \times C}$ 
28: return  $\hat{\mathbf{Y}}$ 

```

Dataset	Output	Group 1		Dataset	Output	Group 2	
		MSE	MAE			MSE	MAE
ETTh1	96	0.374 ± 0.002	0.390 ± 0.001	Weather	96	0.153 ± 0.001	0.190 ± 0.000
	192	0.428 ± 0.001	0.419 ± 0.001		192	0.204 ± 0.001	0.239 ± 0.001
	336	0.469 ± 0.002	0.439 ± 0.001		336	0.261 ± 0.001	0.282 ± 0.001
	720	0.456 ± 0.003	0.450 ± 0.002		720	0.343 ± 0.000	0.336 ± 0.000
	Avg	0.432 ± 0.002	0.424 ± 0.001		Avg	0.240 ± 0.001	0.262 ± 0.001
ETTh2	96	0.290 ± 0.003	0.335 ± 0.002	PEMS03	12	0.058 ± 0.000	0.157 ± 0.001
	192	0.367 ± 0.001	0.385 ± 0.000		24	0.076 ± 0.001	0.176 ± 0.001
	336	0.414 ± 0.004	0.422 ± 0.002		48	0.095 ± 0.004	0.198 ± 0.003
	720	0.424 ± 0.002	0.439 ± 0.001		96	0.130 ± 0.001	0.229 ± 0.005
	Avg	0.373 ± 0.003	0.395 ± 0.001		Avg	0.089 ± 0.007	0.190 ± 0.003
ETTm1	96	0.304 ± 0.001	0.336 ± 0.000	PEMS04	12	0.064 ± 0.001	0.161 ± 0.001
	192	0.359 ± 0.001	0.365 ± 0.000		24	0.072 ± 0.001	0.170 ± 0.001
	336	0.393 ± 0.001	0.388 ± 0.000		48	0.086 ± 0.001	0.186 ± 0.001
	720	0.467 ± 0.001	0.431 ± 0.000		96	0.103 ± 0.001	0.205 ± 0.001
	Avg	0.381 ± 0.001	0.380 ± 0.000		Avg	0.081 ± 0.001	0.180 ± 0.001
ETTm2	96	0.167 ± 0.001	0.246 ± 0.000	PEMS07	12	0.051 ± 0.001	0.139 ± 0.000
	192	0.233 ± 0.001	0.290 ± 0.000		24	0.060 ± 0.001	0.149 ± 0.000
	336	0.293 ± 0.000	0.329 ± 0.000		48	0.078 ± 0.005	0.165 ± 0.002
	720	0.390 ± 0.000	0.387 ± 0.000		96	0.104 ± 0.003	0.194 ± 0.003
	Avg	0.271 ± 0.000	0.313 ± 0.000		Avg	0.073 ± 0.003	0.162 ± 0.001
ECL	96	0.130 ± 0.000	0.220 ± 0.000	PEMS08	12	0.072 ± 0.000	0.170 ± 0.000
	192	0.150 ± 0.000	0.238 ± 0.000		24	0.099 ± 0.000	0.199 ± 0.000
	336	0.166 ± 0.002	0.255 ± 0.002		48	0.127 ± 0.001	0.212 ± 0.001
	720	0.187 ± 0.001	0.274 ± 0.001		96	0.212 ± 0.007	0.254 ± 0.004
	Avg	0.158 ± 0.001	0.247 ± 0.001		Avg	0.128 ± 0.002	0.208 ± 0.001
Solar	96	0.169 ± 0.004	0.200 ± 0.003	Traffic	96	0.397 ± 0.001	0.238 ± 0.000
	192	0.189 ± 0.003	0.220 ± 0.002		192	0.418 ± 0.001	0.249 ± 0.001
	336	0.224 ± 0.004	0.232 ± 0.001		336	0.435 ± 0.001	0.256 ± 0.001
	720	0.208 ± 0.001	0.243 ± 0.001		720	0.472 ± 0.001	0.276 ± 0.000
	Avg	0.197 ± 0.003	0.224 ± 0.002		Avg	0.430 ± 0.001	0.255 ± 0.001

Table 5: Complete forecasting performance for each dataset. We report the average performance over **6** random seeds, accompanied by the standard deviation to capture variability and assess statistical significance.