TStruct: Learning Shared Temporal Structures for Long-term Time Series Forecasting

Yangze Li[1], **Zhengyang Zhou**[1,2,✉], **Qihe Huang**[1], **Pengkun Wang**[1,2], **Binwu Wang**[1,2], **Yang Wang**[1,2,✉]

[1]University of Science and Technology of China (USTC), Hefei, China
[2]Suzhou Institute for Advanced Research, USTC, Suzhou, China
liyangze@mail.ustc.edu.cn

Abstract

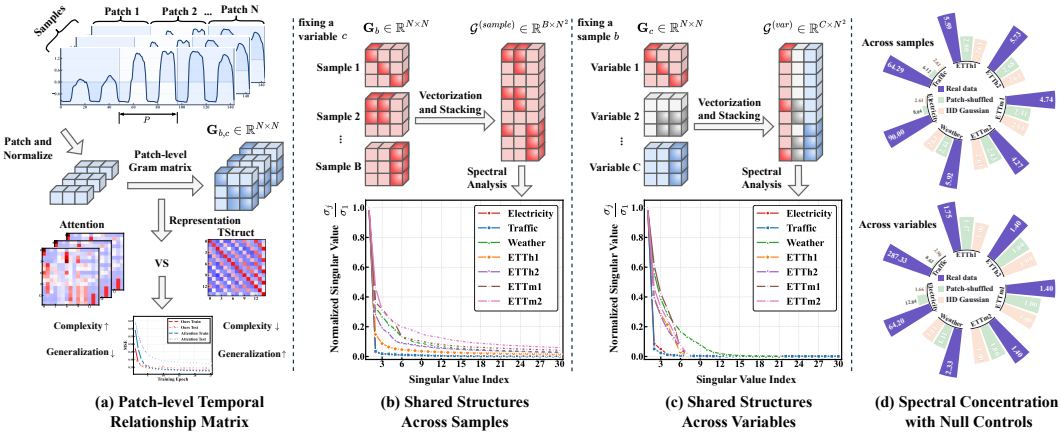
Long-term time series forecasting requires learning temporal dependencies that remain stable and generalizable under temporal distribution shifts. However, real-world time series are often noisy, non-stationary, and heterogeneous, making deep learning models prone to overfitting incidental temporal interactions in the training data. This raises a question: when observations are difficult to align, could temporal relationships be easier to reuse? To investigate this possibility, we analyze patch-level temporal relationship matrices across commonly used time series datasets, revealing dominant patterns shared across different temporal windows and variables. Comparisons with patch-shuffled and IID Gaussian controls further support the presence of reusable temporal relationship structures. Motivated by these findings, we propose **T**emporal **Struct**ure Network (**TStruct**), a simple yet effective network that learns reusable temporal dependency structures and adaptively combines them for each sample and variable. Specifically, TStruct directly parameterizes a set of shared temporal dependency matrices and generates normalized combination weights using dynamic sample information and static variable priors. These weights represent a soft allocation of each sample-variable pair over shared temporal dependency structures, enabling the model to generate adaptive temporal dependency matrices under shared structural constraints, rather than fitting incidental interactions in a fully unconstrained manner. Notably, TStruct consistently outperforms several strong baselines across all evaluated datasets, achieving superior accuracy while maintaining high computational efficiency. Our code is available at https://github.com/yangzeli/TStruct.

1 Introduction

Long-term time series forecasting (LTSF) has widespread applications in real-world scenarios such as energy scheduling, traffic management, weather forecasting, and financial risk control [46, 39, 44, 28]. The goal of LTSF is to accurately model future dynamics over a long prediction horizon. The challenge lies not only in fitting historical observations but also in learning temporal dependency structures that remain effective across future time distributions. However, real-world time series data often have low signal-to-noise ratios (SNR), non-stationarity, and temporal distribution shifts, and local fluctuations or incidental interactions in the training data may not generalize well for future predictions [8, 7, 38, 37]. Therefore, a central question in LTSF is: how can we identify and model temporal dependency structures that remain stable across shifting temporal distributions?

✉ Yang Wang and Zhengyang Zhou are corresponding authors.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).



(a) Patch-level Temporal Relationship Matrix (b) Shared Structures Across Samples (c) Shared Structures Across Variables (d) Spectral Concentration with Null Controls

Figure 1: Patch-level temporal relationship structures. (a) Construction of patch-level relationship matrices and qualitative comparison between attention-based and TStruct-based temporal dependency modeling. (b,c) Concentrated singular value spectra across samples and variables. (d) Higher spectral concentration in real data than in patch-shuffled and IID Gaussian controls supports reusable temporal structures.

In recent years, deep learning methods have driven significant progress in long-term time series forecasting [46, 39, 47, 44, 28, 45, 11, 10, 17, 1, 18, 33, 24, 12, 27, 14, 26]. One category of methods implicitly models temporal dependencies through model architectures. For example, recurrent models pass historical information via hidden states [22, 2], convolutional and TCN models use shared kernels to extract local or multi-scale temporal patterns [23, 35, 6], Transformer-based models estimate pairwise interactions between time tokens explicitly using attention mechanisms [28, 45, 4, 29], while Linear/MLP methods achieve efficient prediction by mixing time or feature dimensions [44, 43, 36, 20, 13]. Despite their strengths in expressive power and efficiency, these methods still face a trade-off between adaptive temporal dependency modeling and forecasting generalization. Highly flexible interaction mechanisms may fit incidental relationships in low-SNR data, while overly fixed transformations struggle to capture heterogeneous temporal dependencies across different samples and variables [18, 17, 37]. A recent line of work directly parameterizes stable temporal patterns, most notably periodicity, through global temporal priors, suggesting that certain temporal structures need not be entirely learned implicitly by generic neural architectures [19, 21, 42, 25, 3]. However, such explicit parameterization has mainly focused on periodic or time-indexed patterns in the input sequence, leaving open whether temporal dependency structures themselves also contain shared and explicitly learnable patterns.

To investigate whether temporal dependencies contain shared structures, we analyze patch-level temporal relationship matrices across commonly used time series datasets. As shown in Figure 1(a), for each sample-variable sequence, we divide the input window into temporal patches and construct a matrix that captures the relationships between relative patch positions. This representation shifts the analysis from raw value fluctuations to relationship patterns among relative patch positions, allowing us to examine whether these patterns recur across samples and variables. We then collect these matrices separately across samples from different temporal positions and across variables for spectral analysis. Figure 1(b,c) shows clear spectral concentration in both settings. This consistent spectral concentration indicates the presence of shared dominant temporal relationship patterns across samples and variables. Figure 1(d) further examines these patterns using patch-shuffled and IID Gaussian controls under the same analysis pipeline. Independent patch shuffling preserves each matrix's relationship values and spectrum while disrupting the alignment of relative patch positions across samples and variables. The resulting decrease in spectral concentration indicates that shared relationship patterns recur at corresponding relative temporal positions. Compared with IID Gaussian inputs, real data also exhibit stronger concentration, indicating shared regularities beyond the random-input reference. Further details are provided in Appendix A.

These observations motivate learning reusable temporal structures and adapting their contributions to each sample and variable. We therefore propose the **T**emporal **Struct**ure Network (**TStruct**). Specifically, TStruct directly parameterizes a set of shared temporal dependency matrices and uses

dynamic sample information and static variable priors to generate normalized combination weights. These weights serve as a soft allocation of each sample-variable pair over the shared dependency structures, combining the shared matrices into a sample- and variable-specific temporal dependency matrix. In this way, TStruct models heterogeneous temporal dependencies under the constraint of shared structures, balancing generalizable structure and adaptive flexibility. In addition, we introduce the Parameter-efficient Channel Adaptation Head (PCAHead), which utilizes a shared low-rank weight generation mechanism to dynamically produce specific prediction heads for each variable.

The contributions of this paper are summarized as follows:

- We reveal that patch-level temporal relationship matrices in real world time series contain dominant structures shared across samples from different temporal positions and across variables.
- We propose TStruct, a concise and effective shared-adaptive temporal dependency modeling method that learns shared temporal dependency matrices and uses dynamic sample information and static variable priors to construct sample- and variable-specific dependencies.
- We conduct extensive experiments to validate the effectiveness of TStruct. The results demonstrate that this simple design consistently outperforms several strong baselines on standard long-term forecasting benchmarks and remains effective under diverse generalization and robustness evaluations, while also exhibiting high computational efficiency.

2 Related Work

2.1 Temporal Dependency Modeling for Time Series Forecasting

Temporal dependency modeling aims to characterize how information interacts across different temporal positions, and is a central problem in long-term time series forecasting. In recent years, Transformer-based and Linear/MLP-based methods have received increasing attention due to their strong performance in long-range dependency modeling and efficient forecasting. Transformer-based methods, such as Informer, Autoformer, PatchTST, Crossformer, and Pathformer [46, 39, 28, 45, 4], usually use attention mechanisms to dynamically estimate pairwise interactions among temporal tokens or patches for each input, thereby modeling long-range temporal dependencies. Their highly flexible input-dependent interactions provide strong adaptivity, but may also capture sample-specific incidental relations in low-SNR time series. Linear/MLP-based methods, such as DLinear, TSMixer, FiTS, SparseTSF, and TimeBase [44, 5, 43, 20, 13], achieve simple and efficient forecasting through shared temporal or feature-wise mixing. These methods demonstrate the strong competitiveness of fixed shared transformations in LTSF, but their adaptability to heterogeneous temporal dependencies remains limited. In contrast, TStruct neither relies on a single fixed mixing rule nor freely estimates interactions for each input. Instead, it generates normalized information aggregation weights through shared temporal dependency structures and sample-variable-conditioned weights, enabling adaptive temporal dependency modeling under shared structural constraints.

2.2 Explicit Temporal Structure Parameterization

Recent studies attempt to explicitly parameterize stable structures in time series, rather than relying entirely on generic neural networks to learn them implicitly from data. CycleNet, TQNet, TimeEmb, MOFO, and GTR [19, 21, 42, 25, 3] mostly model recurring periodic patterns in data through global learnable periodic representations. These studies suggest that directly parameterizing reusable temporal structures can help improve the generalization ability of time series forecasting. However, existing methods mainly focus on periodic patterns in the input sequence or temporal representations. Temporal dependency structures can contain periodicity, but are not equivalent to periodicity itself; they may also characterize more general relationships among relative temporal positions and patch-level interaction patterns [9, 28, 40]. Unlike these methods, TStruct shifts the explicit parameterization target from input-level periodic patterns to temporal dependency structures themselves.

3 Methodology

Problem Definition. The problem of LTSF can be formulated as follows. Let $\mathbf{X} = \{\mathbf{x}_{t-L+1}, \ldots, \mathbf{x}_t\} \in \mathbb{R}^{L \times C}$ denote the sequence of historical observations, where L represents the lookback window length and C denotes the number of variables. At each time step t, the observation vector is defined as $\mathbf{x}_t = \{x_t^{(1)}, \ldots, x_t^{(C)}\}$, where $x_t^{(c)}$ represents the value of the c-th variable. Our objective is to learn a mapping function $\mathcal{F}_\theta$ parameterized by θ to predict the future values for the next H time steps, denoted as $\mathbf{Y} = \{\mathbf{x}_{t+1}, \ldots, \mathbf{x}_{t+H}\} \in \mathbb{R}^{H \times C}$. In this work, we focus on how to learn temporal dependency structures within $\mathcal{F}_\theta$ that remain stable and generalizable across different samples and variables.

3.1 Overview

As shown in Figure 2, TStruct contains three components. Following RevIN [15], TStruct applies instance-wise normalization before encoding and restores the original scale after prediction. Patch Embedding first converts each univariate sequence into patch-level representations, enabling local pattern extraction and reducing the length of temporal modeling [28, 41]. The Temporal Structure Encoder (TSEncoder) then learns shared temporal dependency structures and adaptively combines them using dynamic sample information and static variable priors. The resulting dependency scores are normalized into patch-level aggregation weights, allowing the model to capture sample- and variable-specific temporal dependencies under shared structural constraints. Finally, PCAHead provides a parameter-efficient prediction head that accounts for variable heterogeneity by generating variable-aware forecasting projections.

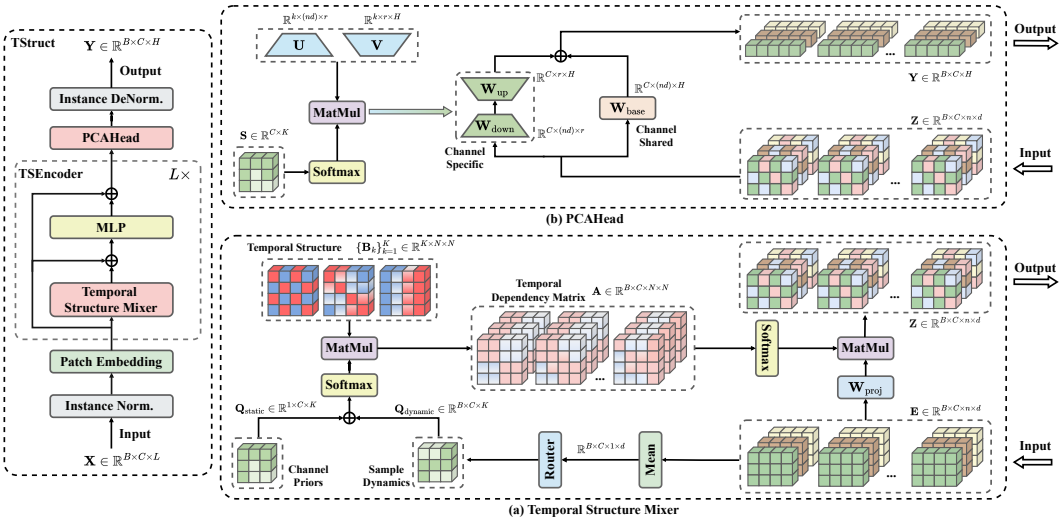


Figure 2: Overall architecture of TStruct. (a) The Temporal Structure Mixer (TSM) generates sample- and variable-specific temporal dependencies by adaptively combining shared temporal dependency structures with dynamic sample information and static variable priors. (b) The PCAHead handles variable heterogeneity by generating variable-aware prediction weights in a parameter-efficient manner.

3.2 Patch Embedding

Patch-based representation has been widely used to capture local temporal patterns and reduce the effective sequence length [28]. Given the input sequence of the c-th variable $\mathbf{x}^{(c)} \in \mathbb{R}^L$, we first segment it into N non-overlapping patches $\mathbf{x}_{\mathrm{p}}^{(c)} \in \mathbb{R}^{N \times P}$, where P is the patch length and $N = L/P$ is the number of patches. For simplicity, we assume that L is divisible by P. Subsequently, a learnable linear projection $\mathbf{W}_{\mathrm{emb}} \in \mathbb{R}^{P \times D}$, shared across all variables, maps these patches into a D-dimensional latent representation:

$$\mathbf{x}_{\mathrm{p}}^{(c)} = \mathrm{Unfold}(\mathbf{x}^{(c)}), \quad \mathbf{E}^{(c)} = \mathbf{x}_{\mathrm{p}}^{(c)} \mathbf{W}_{\mathrm{emb}}, \tag{1}$$

where $\mathbf{E}^{(c)} \in \mathbb{R}^{N \times D}$ denotes the patch-level representation of the c-th variable. This representation provides the basis for TSEncoder to model temporal dependencies among relative patch positions.

3.3 Temporal Structure Encoder

The TSEncoder is designed to model patch-level temporal dependencies in a shared yet adaptive manner. Given the patch-level representation $\mathbf{E} \in \mathbb{R}^{B \times C \times N \times D}$, where B, C, N, and D denote the batch size, number of variables, number of patches, and hidden dimension, respectively, TSEncoder learns temporal dependencies among the N relative patch positions. We first parameterize a set of globally shared temporal dependency structures:

$$\mathcal{B} = \{\mathbf{B}_k\}_{k=1}^{K}, \quad \mathbf{B}_k \in \mathbb{R}^{N \times N}, \tag{2}$$

where K is the number of shared temporal structures. Each $\mathbf{B}_k$ represents a learnable patch-level dependency score pattern.

We initialize the vectorized structure matrices orthogonally, providing distinct initial directions with unit Frobenius norm. This controls the initial scale of the dependency scores before softmax normalization. The importance of initialization is also consistent with findings on learning dynamics in deep linear networks [32]. A detailed initialization study is provided in Appendix C.2.

These shared structures are then combined using sample-variable-specific weights. Specifically, we introduce a learnable static variable prior $\mathbf{Q}_{\text{static}} \in \mathbb{R}^{C \times K}$, which captures variable-specific temporal preferences. A complementary dynamic query is generated by averaging each sample-variable representation along the patch dimension and passing the result through a lightweight router:

$$\mathbf{Q}_{\text{dynamic}} = \text{Router}\left(\text{Mean}_N(\mathbf{E})\right), \tag{3}$$

where $\mathbf{Q}_{\text{dynamic}} \in \mathbb{R}^{B \times C \times K}$. The static and dynamic queries are then fused to obtain normalized combination coefficients:

$$\boldsymbol{\alpha}_{b,c} = \text{Softmax}\left(\mathbf{Q}_{\text{static}}^{(c)} + \mathbf{Q}_{\text{dynamic}}^{(b,c)}\right), \tag{4}$$

where $\boldsymbol{\alpha}_{b,c} \in \mathbb{R}^{K}$ and the softmax is applied over the K shared structures. The coefficients are nonnegative and sum to one, so the resulting dependency score matrix is a convex combination of the shared bases.

For each sample-variable pair, we compose a temporal dependency score matrix from the shared structures:

$$\mathbf{S}_{b,c} = \sum_{k=1}^{K} \alpha_{b,c,k} \mathbf{B}_k, \quad \mathbf{S}_{b,c} \in \mathbb{R}^{N \times N}. \tag{5}$$

The composed scores are converted into normalized temporal aggregation weights:

$$\mathbf{A}_{b,c} = \text{Softmax}_{\text{row}}\left(\mathbf{S}_{b,c}\right). \tag{6}$$

Each row of $\mathbf{A}_{b,c}$ assigns nonnegative weights summing to one to the source patches. Temporal aggregation therefore forms weighted averages of projected historical features, constraining the operator to select and combine information without arbitrary signed amplification.

Finally, the normalized temporal dependency matrix is used to aggregate patch-level features:

$$\widetilde{\mathbf{E}}_{b,c} = \mathbf{A}_{b,c}\left(\mathbf{E}_{b,c}\mathbf{W}_{\text{proj}}\right), \tag{7}$$

where $\mathbf{W}_{\text{proj}} \in \mathbb{R}^{D \times D}$ is a learnable feature projection. The aggregated representation is further processed by a feed-forward network with residual connections:

$$\mathbf{Z}_{b,c} = \widetilde{\mathbf{E}}_{b,c} + \text{MLP}\left(\widetilde{\mathbf{E}}_{b,c}\right). \tag{8}$$

Together, the shared bases and normalized aggregation weights restrict temporal mixing to a common family of dependency patterns learned across samples and variables. This provides an inductive bias toward stable, reusable temporal structures, while allowing their contributions to be adjusted to the input context.

3.4 Parameter-efficient Channel Adaptation Head

PCAHead maps the encoded features to the prediction horizon while balancing variable heterogeneity and parameter efficiency. A fully shared prediction head is efficient but may be insufficient for variable-specific forecasting patterns, whereas assigning an independent head to each variable leads to a parameter cost proportional to C. PCAHead therefore adopts a shared base projection with variable-specific low-rank residual adaptation.

Given the encoder output, we flatten the patch and feature dimensions into $\mathbf{Z} \in \mathbb{R}^{B\times C\times D_{\text{in}}}$, where $D_{\text{in}} = ND$. A shared base projection gives

$$\mathbf{Y}_{\text{base}} = \mathbf{Z}\mathbf{W}_{\text{base}}, \quad \mathbf{W}_{\text{base}} \in \mathbb{R}^{D_{\text{in}}\times H}. \tag{9}$$

For variable-specific adaptation, we define shared low-rank projection banks $\mathbf{U}_k \in \mathbb{R}^{D_{\text{in}}\times r}$ and $\mathbf{V}_k \in \mathbb{R}^{r\times H}$, where $r \ll D_{\text{in}}$. For each variable c, PCAHead learns normalized coefficients $\boldsymbol{\alpha}_c = \text{Softmax}(\mathbf{s}^{(c)})$ and assembles

$$\mathbf{W}^{(c)}_{\text{down}} = \sum_{k=1}^{K} \alpha_{c,k}\mathbf{U}_k, \quad \mathbf{W}^{(c)}_{\text{up}} = \sum_{k=1}^{K} \alpha_{c,k}\mathbf{V}_k. \tag{10}$$

The final prediction is

$$\hat{\mathbf{Y}}^{(c)} = \mathbf{Y}^{(c)}_{\text{base}} + \mathbf{Z}^{(c)}\mathbf{W}^{(c)}_{\text{down}}\mathbf{W}^{(c)}_{\text{up}}. \tag{11}$$

This reduces the head complexity from $O(CD_{\text{in}}H)$ to $O(D_{\text{in}}H + Kr(D_{\text{in}} + H) + CK)$. We zero-initialize the up-projection bank so that the residual branch starts from zero, allowing the model to first rely on the shared base prediction and then gradually learn variable-specific corrections.

4 Experiment

4.1 Experiment Setup

Following prior studies, we use Mean Squared Error (MSE) and Mean Absolute Error (MAE) as evaluation metrics. All methods are evaluated under prediction lengths $H = \{96, 192, 336, 720\}$. The datasets and baseline models used for evaluation are summarized below. Dataset statistics are provided in Appendix A.1, while implementation details and hyperparameter settings are provided in Appendix B.

Datasets. We evaluate TStruct on seven widely used long-term forecasting benchmarks, including Weather, Traffic, Electricity [40], and four ETT subsets [46]. These datasets cover different domains, variable scales, and temporal patterns, providing a comprehensive evaluation of forecasting performance and generalization ability.

Baselines. We compare TStruct with eight strong and recent forecasting baselines, covering different temporal dependency modeling paradigms. Specifically, PatchTST [28] and PhaseFormer [29] represent Transformer-based methods; DLinear [44] represents simple linear forecasting methods; CycleNet [19], TQNet [21], TimeEmb [42], and GTR [3] explicitly model temporal patterns or periodic priors; and DUET [31] is included as a recent strong forecasting baseline. For all baselines, we report the best results reported in their papers or accompanying code repositories, with the corresponding experimental configurations detailed in Appendix B.

4.2 Main results

Table 1 reports the average results over all prediction horizons, with detailed comparisons in Table 10. TStruct achieves the best average MSE on all seven benchmarks and the best average MAE on six, ranking first in 13 of the 14 dataset-metric comparisons. These results demonstrate the effectiveness of shared-adaptive temporal dependency modeling against strong Transformer-based, Linear-based, and explicit temporal-pattern modeling baselines.

On ETT and Weather, TStruct shows clear advantages over existing methods. These datasets often contain non-stationary temporal dynamics and distribution variations, where overly flexible models may fit incidental temporal interactions and overly fixed transformations may be insufficiently

Table 1: Average multivariate time series forecasting results over all prediction horizons. The best results are marked in **bold red**, and the second-best results are marked in blue underline.

Methods	TStruct (ours)		DUET [2025]		PhaseFormer [2026]		GTR [2026]		TimeEmb [2025]		TQNet [2025]		CycleNet [2024]		PatchTST [2023]		DLinear [2023]	
Datasets	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**
ETTh1	**0.390**	**0.415**	0.398	0.419	0.403	**0.415**	0.415	0.419	0.418	0.433	0.420	0.428	0.415	0.426	0.417	0.431	0.425	0.439
ETTh2	**0.331**	**0.381**	0.334	0.383	0.347	0.389	0.346	0.394	0.353	0.396	0.358	0.399	0.345	0.394	0.351	0.395	0.470	0.468
ETTm1	**0.337**	**0.367**	0.338	0.369	0.347	0.374	0.345	0.377	0.345	0.376	0.353	0.388	0.352	0.378	0.349	0.381	0.356	0.378
ETTm2	**0.243**	**0.304**	0.248	0.307	0.251	0.314	0.255	0.315	0.248	0.308	0.258	0.318	0.249	0.311	0.256	0.314	0.259	0.324
Weather	**0.213**	**0.249**	0.218	0.252	0.223	0.261	0.227	0.264	0.218	0.257	0.230	0.266	0.237	0.277	0.224	0.262	0.242	0.293
Electricity	**0.151**	**0.244**	0.157	0.248	0.161	0.250	0.160	0.254	0.158	0.253	0.162	0.256	0.157	0.250	0.162	0.254	0.167	0.264
Traffic	**0.381**	0.252	0.393	0.256	0.387	**0.250**	0.397	0.274	0.399	0.272	0.404	0.282	0.406	0.280	0.396	0.266	0.418	0.287
1st Count	**7**	**6**	0	0	**0**	**2**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

Table 2: Ablation study on the main components of TStruct. Results are averaged over all prediction horizons. Detailed results for each prediction horizon are provided in Table 11.

Dataset	ETTh1		ETTh2		ETTm1		ETTm2		Weather		Electricity		Traffic	
Metric	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**
①	0.425	0.444	0.350	0.395	0.376	0.391	0.259	0.318	0.229	0.265	0.162	0.254	0.403	0.270
②	0.403	0.424	0.341	0.384	0.347	0.372	0.248	0.308	0.218	0.254	0.153	0.246	0.382	0.253
③	0.394	0.417	0.339	0.386	0.341	0.368	0.248	0.307	0.215	0.251	0.152	0.245	0.382	0.253
TStruct	**0.390**	**0.415**	**0.331**	**0.381**	**0.337**	**0.367**	**0.243**	**0.304**	**0.213**	**0.249**	**0.151**	**0.244**	**0.381**	**0.252**

adaptive. The consistent improvements on these datasets indicate that TStruct can better capture stable temporal dependency structures while preserving sample-specific adaptability.

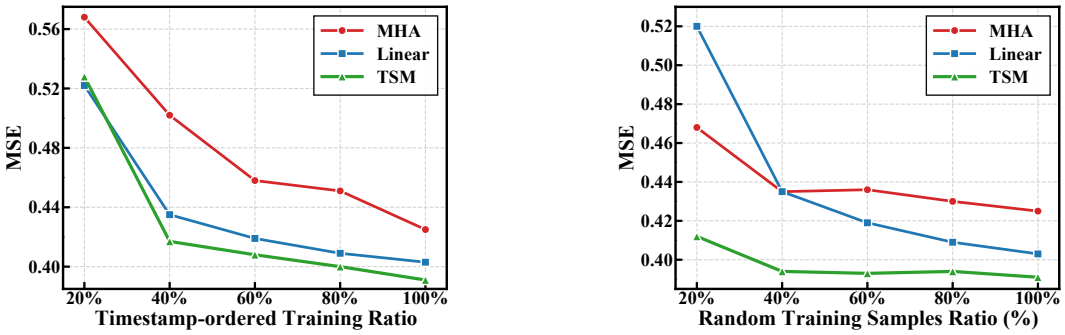
On Electricity and Traffic, Notably, without explicit cross-variable interactions, TStruct achieves the best average MSE on both datasets, with the best average MAE on Electricity and the second-best on Traffic. This result is consistent with our spectral analysis in Appendix A.2: Electricity and Traffic exhibit stronger spectral concentration in patch-level temporal relationship matrices across both samples and variables, suggesting a higher degree of shared temporal dependency structures. Such structures can be effectively captured by TStruct through shared temporal dependency matrices and sample-variable-conditioned adaptation.

4.3 Ablation Study

We evaluate the effectiveness of three key components in TStruct. The full model is compared against three variants: ① **TSM** $\rightarrow$ **MHA** replaces the Temporal Structure Mixer with multi-head attention; ② **TSM** $\rightarrow$ **Linear** replaces it with a shared linear temporal mixer; and ③ **PCAHead** $\rightarrow$ **Linear** replaces PCAHead with a shared linear forecasting head. All TStruct variants in the ablation study are trained with the same objective and training protocol. The summary of these ablation results is reported in Table 2.

On ETT and Weather, TStruct clearly outperforms both ① and ② on these datasets. These datasets exhibit stronger temporal distribution variations, where forecasting requires both shared structures for generalization and adaptive modeling for sample- and variable-specific heterogeneity. MHA provides flexible input-dependent interactions but may fit incidental relations, while a shared linear mixer is stable but insufficiently adaptive. The superior performance of TStruct indicates that shared-adaptive temporal dependency modeling is especially beneficial in such settings.

On Electricity and Traffic, The gap between TStruct and ② becomes smaller on these datasets, while TStruct still clearly outperforms ①. This is consistent with our spectral analysis in Appendix A: Electricity and Traffic exhibit stronger cross-sample and cross-variable spectral concentration in patch-level temporal relationship matrices, suggesting more pronounced shared temporal dependency



(a) Earliest samples.

(b) Random samples.

Figure 3: Generalization analysis on ETTh1. (a) Models are trained with the earliest timestamp-ordered training samples, evaluating temporal distribution shift. (b) Models are trained with randomly sampled training instances, evaluating sample efficiency.

structures. Therefore, simple shared temporal mixing is already highly competitive on these datasets, whereas fully input-dependent attention remains less stable.

Finally, replacing PCAHead with a shared linear head leads to moderate but consistent degradation, showing that variable-aware prediction further improves performance under variable heterogeneity.

4.4 Generalization Analysis

To evaluate generalization, we conduct two reduced-training experiments on ETTh1. We compare the proposed TSM with two variants that replace TSM with Linear or MHA, keeping all other components and experimental settings identical. The earliest sample setting evaluates temporal distribution shift, while random sample setting evaluates sample efficiency.

Temporal Distribution Shift. As shown in Figure 3(a), models are trained using the earliest $20\%, 40\%, 60\%, 80\%, 100\%$ of timestamp-ordered training samples, while validation and test sets remain unchanged. MHA consistently performs worse than Linear and TSM, indicating that fully input-dependent attention is more sensitive to temporally biased training data. TSM is close to Linear when only the earliest 20% samples are used, but clearly outperforms Linear from 40% onward, showing that shared-adaptive temporal dependency modeling improves generalization once sufficient shared structures can be estimated.

Sample Efficiency. As shown in Figure 3(b), TSM achieves the best performance even with only 20% randomly sampled training data and quickly converges around 40%. Notably, TSM trained with 40% data already outperforms MHA and Linear trained with the full training set. This suggests that shared temporal structures improve data efficiency by reducing the need to relearn temporal interactions from individual samples.

4.5 Effect of Look-back Length

To examine how different methods utilize historical context, we vary the look-back length from 48 to 720 and compare TStruct with explicit temporal-pattern parameterization methods, including CycleNet [19], TQNet [21], TimeEmb [42], and GTR [3]. These baselines mainly rely on global periodic representations, which can remain effective even with short input windows because the periodic priors are largely input-independent. However, as the look-back length increases, their improvements become limited and may even degrade when overly long histories introduce noisy or less relevant observations.

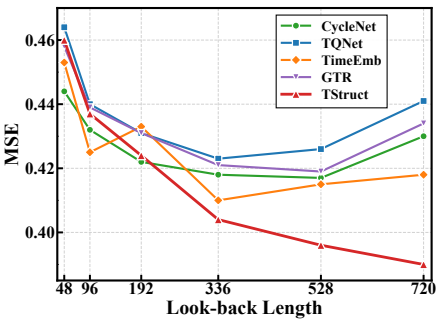


Figure 4: Effect of look-back length on ETTh1.

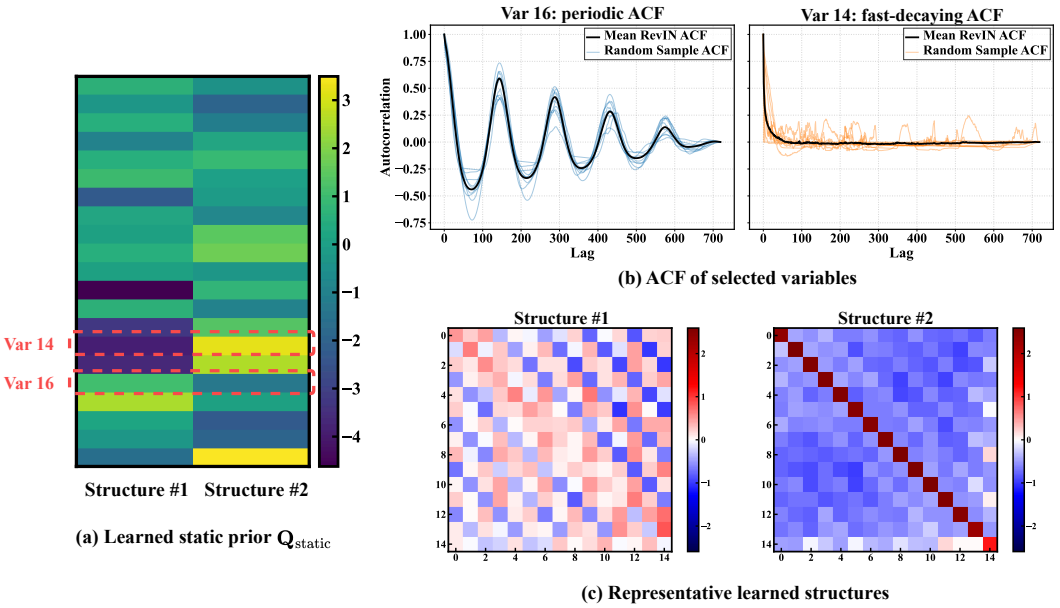


Figure 5: Interpretability analysis on the Weather dataset. Different variables exhibit different ACF patterns and assign different weights to the learned shared temporal structures through Q_{static}. The representative learned structures show distinct patch-level dependency patterns, including broader recurrent relations and local diagonal-dominant continuity.

In contrast, TStruct consistently benefits from longer look-back windows. This suggests that TStruct does not merely rely on fixed global periodic patterns; instead, it explicitly models patch-level temporal dependency structures, which become richer as more historical context is available. This further indicates that patch-level temporal dependency structures can provide a more extensible modeling target than fixed global periodic priors when longer historical context is available.

4.6 Interpretability of Learned Temporal Structures

To examine whether the learned structures are interpretable, we visualize the learned static prior, the autocorrelation functions (ACF) of representative variables, and the corresponding temporal dependency structures on the Weather dataset. As shown in Figure 5, Var 16 exhibits clear periodic autocorrelation, while Var 14 shows rapidly decaying autocorrelation, indicating that these two variables have different temporal characteristics. The learned static prior $\mathbf{Q}_{\text{static}}$ assigns different preferences to these variables over the shared structures. Specifically, Var 16 places higher weight on Structure #1, which captures broader recurrent relations among patches, whereas Var 14 assigns higher weight to Structure #2, which shows strong diagonal dominance and reflects local temporal continuity. This suggests that $\mathbf{Q}_{\text{static}}$ captures variable-specific temporal preferences rather than using the same dependency pattern for all variables. Overall, these observations indicate that TStruct learns reusable temporal dependency structures and adapts them to variables with different temporal characteristics through the static prior. For additional visualizations of learned structure matrices on other datasets, please refer to Appendix D.

5 Conclusion

In this work, we investigated whether temporal relationships can be reused across noisy and heterogeneous time series. Analysis of patch-level relationship matrices reveals dominant patterns shared across temporal windows and variables, with null controls further supporting their reusability. Motivated by these findings, we proposed TStruct, a simple yet effective network that learns shared temporal dependency structures and adaptively combines them for each sample and variable. Experiments on seven forecasting benchmarks show that TStruct consistently outperforms strong baselines while maintaining high computational efficiency. Ablation and generalization analyses further highlight the value of combining shared structures with sample-specific adaptation.

Acknowledgement

This paper is partially supported by the National Natural Science Foundation of China (No.62502488, No.12227901), Natural Science Foundation of Jiangsu Province (BK20240460), the grant from State Key Laboratory of Resources and Environmental Information System. The AI-driven experiments, simulations and model training were performed on the robotic AI-Scientist platform of Chinese Academy of Sciences. The authors declare no competing interests.

References

[1] Abdul Fatir Ansari, Lorenzo Stella, Ali Caner Turkmen, Xiyuan Zhang, Pedro Mercado, Huibin Shen, Oleksandr Shchur, Syama Sundar Rangapuram, Sebastian Pineda Arango, Shubham Kapoor, Jasper Zschiegner, Danielle C. Maddix, Hao Wang, Michael W. Mahoney, Kari Torkkola, Andrew Gordon Wilson, Michael Bohlke-Schneider, and Bernie Wang. Chronos: Learning the language of time series. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=gerNCVqqtR.

[2] Maximilian Beck, Korbinian Pöppel, Markus Spanring, Andreas Auer, Oleksandra Prudnikova, Michael K Kopp, Günter Klambauer, Johannes Brandstetter, and Sepp Hochreiter. xLSTM: Extended long short-term memory. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=ARAxPPIAhq.

[3] Fanpu Cao, Lu Dai, Jindong Han, and Hui Xiong. Enhancing multivariate time series forecasting with global temporal retrieval. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=QUJBPSfyui.

[4] Peng Chen, Yingying ZHANG, Yunyao Cheng, Yang Shu, Yihang Wang, Qingsong Wen, Bin Yang, and Chenjuan Guo. Pathformer: Multi-scale transformers with adaptive pathways for time series forecasting. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=lJkOCMP2aW.

[5] Si-An Chen, Chun-Liang Li, Sercan O Arik, Nathanael Christian Yoder, and Tomas Pfister. TSMixer: An all-MLP architecture for time series forecasting. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856. URL https://openreview.net/forum?id=wbpxTuXgm0.

[6] Luo donghao and wang xue. ModernTCN: A modern pure convolution structure for general time series analysis. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=vpJMJerXHU.

[7] Clive WJ Granger. Investigating causal relations by econometric models and cross-spectral methods. *Econometrica: journal of the Econometric Society*, pages 424–438, 1969.

[8] C.W.J. Granger and P. Newbold. Spurious regressions in econometrics. *Journal of Econometrics*, 2(2):111–120, 1974. ISSN 0304-4076. doi: https://doi.org/10.1016/0304-4076(74)90034-7. URL https://www.sciencedirect.com/science/article/pii/0304407674900347.

[9] James D Hamilton. *Time series analysis*. Princeton University Press, 1994.

[10] Lu Han, Han-Jia Ye, and De-Chuan Zhan. The capacity and robustness trade-off: Revisiting the channel independent strategy for multivariate time series forecasting. *IEEE Trans. Knowl. Data Eng.*, 36(11):7129–7142, 2024.

[11] Qihe Huang, Lei Shen, Ruixin Zhang, Shouhong Ding, Binwu Wang, Zhengyang Zhou, and Yang Wang. CrossGNN: Confronting noisy multivariate time series via cross interaction refinement. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=xOzlW2vUYc.

[12] Qihe Huang, Zhengyang Zhou, Yangze Li, Kuo Yang, Binwu Wang, and Yang Wang. Many minds, one goal: Time series forecasting via sub-task specialization and inter-agent cooperation. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=Uon41HfqR3.

[13] Qihe Huang, Zhengyang Zhou, Kuo Yang, Zhongchao Yi, Xu Wang, and Yang Wang. Timebase: The power of minimalism in efficient long-term time series forecasting. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=GhTdNOMfOD.

[14] Qihe Huang, Zhengyang Zhou, Yangze Li, Jiaming Ma, Kuo Yang, Binwu Wang, Xu Wang, and Yang Wang. STORM: Synergistic cross-scale spatio-temporal modeling for weather forecasting. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=JLF6XDnscF.

[15] Taesung Kim, Jinhee Kim, Yunwon Tae, Cheonbok Park, Jang-Ho Choi, and Jaegul Choo. Reversible instance normalization for accurate time-series forecasting against distribution shift. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=cGDAkQo1C0p.

[16] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR (Poster)*, 2015.

[17] Zhe Li, Shiyi Qi, Yiduo Li, and Zenglin Xu. Revisiting long-term time series forecasting: An investigation on linear mapping, 2026. URL https://arxiv.org/abs/2305.10721.

[18] Zida Liang, Jiayi Zhu, and Weiqiang Sun. Why attention fails: The degeneration of transformers into mlps in time series forecasting, 2025. URL https://arxiv.org/abs/2509.20942.

[19] Shengsheng Lin, Weiwei Lin, Xinyi HU, Wentai Wu, Ruichao Mo, and Haocheng Zhong. Cyclenet: Enhancing time series forecasting through modeling periodic patterns. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=clBiQUgj4w.

[20] Shengsheng Lin, Weiwei Lin, Wentai Wu, Haojun Chen, and Junjie Yang. SparseTSF: Modeling long-term time series forecasting with *1k* parameters. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=54NSHO0lFe.

[21] Shengsheng Lin, Haojun Chen, Haijie Wu, Chunyun Qiu, and Weiwei Lin. Temporal query network for efficient multivariate time series forecasting. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=e24CueVty2.

[22] Shengsheng Lin, Weiwei Lin, Wentai Wu, Feiyu Zhao, Ruichao Mo, and Haotong Zhang. Segrnn: Segment recurrent neural network for long-term time-series forecasting. *IEEE Internet Things J.*, 13(5):9861–9871, 2026.

[23] Minhao LIU, Ailing Zeng, Muxi Chen, Zhijian Xu, Qiuxia LAI, Lingna Ma, and Qiang Xu. SCINet: Time series modeling and forecasting with sample convolution and interaction. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=AyajSjTAzmg.

[24] Peiyuan Liu, Beiliang Wu, Yifan Hu, Naiqi Li, Tao Dai, Jigang Bao, and Shu-Tao Xia. Timebridge: Non-stationarity matters for long-term time series forecasting. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=pyKO0ZZ5lz.

[25] Jiaming Ma, Binwu Wang, Qihe Huang, Guanjun Wang, Pengkun Wang, Zhengyang Zhou, and Yang Wang. Mofo: Empowering long-term time series forecasting with periodic pattern modeling. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=sbvLts2HqR.

[26] Jiaming Ma, Guanjun Wang, Sheng Huang, Kuo Yang, Binwu Wang, Pengkun Wang, and Yang Wang. Spatiotemporal causal decoupling model for air quality forecasting. In *ICASSP 2025 - 2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5, 2025. doi: 10.1109/ICASSP49660.2025.11099015.

[27] Jiaming Ma, Qihe Huang, Haofeng Ma, Guanjun Wang, Sheng Huang, Zhengyang Zhou, Pengkun Wang, Xu Wang, Binwu Wang, and Yang Wang. PHAT: Modeling period heterogeneity for multivariate time series forecasting. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=lr4RlISR6x.

[28] 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.

[29] Yiming Niu, Jinliang Deng, and Yongxin Tong. Phaseformer: From patches to phases for efficient and effective time series forecasting. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=Lk9SqMQzhX.

[30] Xiangfei Qiu, Jilin Hu, Lekui Zhou, Xingjian Wu, Junyang Du, Buang Zhang, Chenjuan Guo, Aoying Zhou, Christian S. Jensen, Zhenli Sheng, and Bin Yang. TFB: towards comprehensive and fair benchmarking of time series forecasting methods. *Proc. VLDB Endow.*, 17(9):2363–2377, 2024.

[31] Xiangfei Qiu, Xingjian Wu, Yan Lin, Chenjuan Guo, Jilin Hu, and Bin Yang. DUET: dual clustering enhanced multivariate time series forecasting. In *KDD (1)*, pages 1185–1196. ACM, 2025.

[32] Mark Rhee, Dhruva Karkada, and Jamie Simon. Deep linear networks are a surprisingly useful toy model of weight-space dynamics. *Learning Mechanics*, 2026. URL https://learningmechanics.pub/deep-linear-nets.

[33] Mingtian Tan, Mike A Merrill, Vinayak Gupta, Tim Althoff, and Thomas Hartvigsen. Are language models actually useful for time series forecasting? In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=DV15UbHCY1.

[34] Hao Wang, Lichen Pan, Yuan Shen, Zhichao Chen, Degui Yang, Yifei Yang, Sen Zhang, Xinggao Liu, Haoxuan Li, and Dacheng Tao. FreDF: Learning to forecast in the frequency domain. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=4A9IdSa1ul.

[35] 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.

[36] 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*, 2024. URL https://openreview.net/forum?id=7oLshfEIC2.

[37] Zheng Wang, Kaixuan Zhang, Wanfang Chen, Xiaonan Lu, Longyuan Li, and Tobias Schlagenhauf. Characteristic root analysis and regularization for linear time series forecasting. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=JTtwGRACte.

[38] Gerhard Widmer and Miroslav Kubat. Learning in the presence of concept drift and hidden contexts. *Mach. Learn.*, 23(1):69–101, 1996.

[39] Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=J4gRj6d5Qm.

[40] Haixu Wu, Tengge Hu, Yong Liu, Hang Zhou, Jianmin Wang, and Mingsheng Long. Timesnet: Temporal 2d-variation modeling for general time series analysis. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=ju_Uqw384Oq.

[41] Xingjian Wu, Xiangfei Qiu, Hanyin Cheng, Zhengyu Li, Jilin Hu, Chenjuan Guo, and Bin Yang. Enhancing time series forecasting through selective representation spaces: A patch perspective. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=BirE0jYKt0.

[42] Mingyuan Xia, Chunxu Zhang, Zijian Zhang, Hao Miao, Qidong Liu, Yuanshao Zhu, and Bo Yang. Timeemb: A lightweight static-dynamic disentanglement framework for time series forecasting. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=sLfMvrkn6T.

[43] Zhijian Xu, Ailing Zeng, and Qiang Xu. FITS: Modeling time series with $10k$ parameters. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=bWcnvZ3qMb.

[44] Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. Are transformers effective for time series forecasting? In *AAAI*, pages 11121–11128. AAAI Press, 2023.

[45] 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.

[46] Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *AAAI*, pages 11106–11115. AAAI Press, 2021.

[47] Tian Zhou, Ziqing Ma, Qingsong Wen, Xue Wang, Liang Sun, and Rong Jin. Fedformer: Frequency enhanced decomposed transformer for long-term series forecasting. In *ICML*, volume 162 of *Proceedings of Machine Learning Research*, pages 27268–27286. PMLR, 2022.

Appendix

Catalogue

1. Introduction **1**

2. Related Work **3**

2.1 Temporal Dependency Modeling for Time Series Forecasting 3

2.2 Explicit Temporal Structure Parameterization 3

3. Methodology **4**

3.1 Overview 4

3.2 Patch Embedding 4

3.3 Temporal Structure Encoder 5

3.4 Parameter-efficient Channel Adaptation Head 6

4. Experiment **6**

4.1 Experiment Setup 6

4.2 Main Results 6

4.3 Ablation Study 7

4.4 Generalization Analysis 8

4.5 Effect of Look-back Length 8

4.6 Interpretability of Learned Temporal Structures 9

5. Conclusion **9**

Appendix **15**

A. Dataset and Temporal Structure Analysis 15

B. Implementation Details 18

C. Additional Experimental Analyses 19

D. Additional Interpretability Visualizations 24

E. Detailed Experimental Results 28

F. Limitations and Future Directions 30

A Dataset and Temporal Structure Analysis

A.1 Dataset Details

We evaluate TStruct on seven widely used long-term forecasting benchmarks, including four ETT subsets, Weather, Electricity, and Traffic. The statistics of these datasets are summarized in Table 3. The ETT datasets [46] contain temperature and power load records collected from electricity transformers in two regions of China from 2016 to 2018. They include two temporal granularities: hourly records, denoted as ETTh1 and ETTh2, and 15-minute records, denoted as ETTm1 and ETTm2. The Weather dataset [40] contains 21 meteorological variables recorded at 10-minute intervals in 2020, including air temperature, humidity, visibility, and other weather-related indicators. The Electricity dataset [40] consists of hourly electricity consumption records from 321 clients, measured in kilowatt-hours (kWh), and is derived from the UCI Machine Learning Repository. The Traffic dataset [40] records hourly road occupancy rates collected by 862 sensors on freeways in the San Francisco Bay Area, reflecting large-scale traffic flow dynamics.

Table 3: Detailed descriptions of datasets.

Dataset	Variables	Timestamps	Split	Prediction Length	Sampling	Information
ETTh1	7	14400	6:2:2	{96,192,336,720}	Hourly	Electricity
ETTh2	7	14400	6:2:2	{96,192,336,720}	Hourly	Electricity
ETTm1	7	57600	6:2:2	{96,192,336,720}	15min	Electricity
ETTm2	7	57600	6:2:2	{96,192,336,720}	15min	Electricity
Weather	21	52696	7:1:2	{96,192,336,720}	10min	Weather
ECL	321	26304	7:1:2	{96,192,336,720}	Hourly	Electricity
Traffic	862	17544	7:1:2	{96,192,336,720}	Hourly	Transport

A.2 Temporal Structure Analysis

Patch-level temporal relationship matrix. Real-world time series often have low-SNR, and temporal interactions observed in individual windows may reflect incidental fluctuations. We hypothesize that these observations nevertheless contain relatively stable temporal structures associated with the underlying evolution, which can be extracted from recurring relationships across windows and variables. To examine this hypothesis, we construct patch-level temporal relationship matrices that describe relationships among relative temporal positions. For each dataset, we first standardize each variable and randomly sample B input windows of length L. Given the b-th input window $\mathbf{X}_b \in \mathbb{R}^{L\times C}$, we apply instance-level normalization along the temporal dimension:

$$\widetilde{\mathbf{X}}_b = \frac{\mathbf{X}_b - \mathrm{Mean}_L(\mathbf{X}_b)}{\mathrm{Std}_L(\mathbf{X}_b) + \epsilon}, \tag{12}$$

where $\mathrm{Mean}_L(\cdot)$ and $\mathrm{Std}_L(\cdot)$ are computed along the temporal dimension for each variable.

Then, for each variable c, the normalized sequence is divided into N non-overlapping patches with patch length P:

$$\mathbf{X}^p_{b,c} = \mathrm{Unfold}(\widetilde{\mathbf{X}}^{(c)}_b) \in \mathbb{R}^{N\times P}, \tag{13}$$

where $N = L/P$ for non-overlapping patches, and we assume L is divisible by P.

Each patch vector is further normalized by its ℓ_2-norm:

$$\overline{\mathbf{X}}^p_{b,c}[i] = \frac{\mathbf{X}^p_{b,c}[i]}{\|\mathbf{X}^p_{b,c}[i]\|_2 + \epsilon}, \quad i = 1, \dots, N. \tag{14}$$

We then construct the patch-level temporal relationship matrix as

$$\mathbf{G}_{b,c} = \overline{\mathbf{X}}^p_{b,c} \left(\overline{\mathbf{X}}^p_{b,c}\right)^\top \in \mathbb{R}^{N\times N}, \tag{15}$$

where $\mathbf{G}_{b,c}[i,j]$ measures the normalized relationship strength between the i-th and j-th relative patch positions.

Shared temporal structures across samples and variables. To examine shared temporal structures, we stack the vectorized relationship matrices across samples and across variables. For the sample-side analysis, each variable c is fixed and the matrices from all B windows are collected; for the variable-side analysis, each sample b is fixed and the matrices from all C variables are collected:

$$\begin{aligned} \mathcal{G}_c^{(\text{sample})} &= [\text{vec}(\mathbf{G}_{1,c}), \ldots, \text{vec}(\mathbf{G}_{B,c})]^\top \in \mathbb{R}^{B \times N^2}, \\ \mathcal{G}_b^{(\text{var})} &= [\text{vec}(\mathbf{G}_{b,1}), \ldots, \text{vec}(\mathbf{G}_{b,C})]^\top \in \mathbb{R}^{C \times N^2}. \end{aligned} \tag{16}$$

Before performing SVD, the rows of both matrices are normalized to compare temporal relationship patterns on a common scale:

$$\overline{\mathcal{G}}[j] = \frac{\mathcal{G}[j]}{\|\mathcal{G}[j]\|_2 + \epsilon}, \tag{17}$$

where $\mathcal{G}$ denotes either stacked matrix and j indexes its rows. This normalization removes differences in vector magnitude while preserving the directions of the temporal relationship representations.

Let $\{\sigma_{c,i}^{(\text{sample})}\}_i$ and $\{\sigma_{b,i}^{(\text{var})}\}_i$ denote the ordered singular values of $\overline{\mathcal{G}}_c^{(\text{sample})}$ and $\overline{\mathcal{G}}_b^{(\text{var})}$, respectively. The averaged spectral energies are defined as

$$\bar{e}_i^{(\text{sample})} = \frac{1}{C} \sum_{c=1}^{C} \left(\sigma_{c,i}^{(\text{sample})}\right)^2, \qquad \bar{e}_i^{(\text{var})} = \frac{1}{B} \sum_{b=1}^{B} \left(\sigma_{b,i}^{(\text{var})}\right)^2. \tag{18}$$

These spectra indicate the extent to which temporal relationship patterns are concentrated in a small number of directions across samples and variables, respectively.

Spectral concentration metrics. We quantify spectral concentration by measuring how many singular components are required to explain 95% of the spectral energy. Given an averaged spectral energy sequence $\{\bar{e}_i\}_{i=1}^R$, the 95% energy rank is defined as

$$r_{95} = \min \left\{ r : \frac{\sum_{i=1}^{r} \bar{e}_i}{\sum_{i=1}^{R} \bar{e}_i} \geq 0.95 \right\}. \tag{19}$$

Here R denotes the reference dimension of the full vectorized representation:

$$R = \begin{cases} \min(C, N^2), & \text{for } \mathcal{G}_b^{(\text{var})}, \\ \min(B, N^2), & \text{for } \mathcal{G}_c^{(\text{sample})}. \end{cases} \tag{20}$$

To make the degree of spectral concentration more intuitive, we report the rank compression factor:

$$\kappa_{95} = \frac{R}{r_{95}}. \tag{21}$$

A larger κ_{95} indicates that 95% of the spectral energy is captured by a smaller fraction of the reference dimension, corresponding to stronger spectral concentration. However, spectral concentration can also arise from the Gram construction itself: symmetry introduces redundant coordinates in the vectorized representation, while patch normalization produces a common unit diagonal across matrices. A reference level is therefore needed to assess the additional concentration associated with temporal structure. To this end, we evaluate two null controls through the same analysis pipeline and compare their κ_{95} values with those of the real data.

Quantitative analysis. Table 4 reports the spectral concentration statistics under $B = 2000$, $L = 720$, and $P = 24$ ($N = 30$), together with two null controls evaluated through the same pipeline. **(1) Patch shuffle** evaluates the role of temporal ordering by independently permuting patch positions for each sample-variable pair, giving $\mathbf{G}_{b,c}^{\text{shuffle}} = \mathbf{\Pi}_{b,c} \mathbf{G}_{b,c} \mathbf{\Pi}_{b,c}^\top$. This preserves the spectrum and similarity values of each Gram matrix while disrupting the correspondence between relative patch positions across matrices. Comparing the resulting concentration with that of real data therefore assesses whether the observed concentration depends on temporal relationships recurring at aligned patch positions. **(2) IID Gaussian** provides a complementary reference by replacing the observed windows with independent Gaussian noise. Applying the same analysis to these inputs gives the spectral concentration that can arise without temporal dependencies, against which the additional concentration in real data can be assessed.

Table 4: Shuffle denotes independent patch permutations and IID denotes independent Gaussian windows. Results are averaged over repeated runs of each null control.

(a) Sample side: across temporal windows

Dataset	Patch size P	Patch number N	Variables C	Real	Shuffle	IID	Real/Shuffle	Real/IID
ETTh1	24	30	7	5.59	2.80	2.61	2.00×	2.14×
ETTh2	24	30	7	5.73	2.68	2.61	2.14×	2.20×
ETTm1	24	30	7	4.74	2.41	2.61	1.96×	1.82×
ETTm2	24	30	7	4.27	2.43	2.61	1.76×	1.64×
Weather	24	30	21	5.92	2.44	2.61	2.43×	2.27×
Electricity	24	30	321	90.00	8.04	2.61	11.20×	34.50×
Traffic	24	30	862	64.29	6.12	2.61	10.50×	24.64×

(b) Variable side: across variables within a window

Dataset	Patch size P	Patch number N	Variables C	Real	Shuffle	IID	Real/Shuffle	Real/IID
ETTh1	24	30	7	1.75	1.17	1.00	1.50×	1.75×
ETTh2	24	30	7	1.40	1.00	1.00	1.40×	1.40×
ETTm1	24	30	7	1.40	1.00	1.00	1.40×	1.40×
ETTm2	24	30	7	1.40	1.00	1.00	1.40×	1.40×
Weather	24	30	21	2.33	1.11	1.11	2.11×	2.11×
Electricity	24	30	321	64.20	12.84	1.66	5.00×	38.60×
Traffic	24	30	862	287.33	8.62	2.90	33.33×	99.00×

Table 4 reveals four findings:

(i) Shared structures at aligned temporal positions. Across all seven datasets, Real/Shuffle is substantially greater than one on both the sample and variable sides. The reduction in concentration after shuffling supports shared temporal relationship patterns that recur at corresponding relative patch positions across windows and variables.

(ii) Concentration beyond the random-input reference. Real/IID is also substantially greater than one across all datasets on both sides. Temporal relationships in real time series are therefore more concentrated than those obtained from independent random inputs under the same analysis pipeline, supporting their representation through a compact set of shared patterns.

(iii) Larger gains in Electricity and Traffic. Electricity and Traffic exhibit substantially larger Real/Shuffle and Real/IID gains than ETT and Weather on both sides. Their temporal relationships show more pronounced concentration relative to their respective controls, with the larger Real/Shuffle gains indicating a greater dependence of this concentration on the original patch-position alignment.

(iv) Shared regularities preserved under shuffling. For Electricity and Traffic, shuffled-data concentration remains well above the corresponding IID reference on both sides, whereas the two controls yield closer values for ETT and Weather. This suggests that shared regularities preserved under patch permutation coexist with reusable temporal relationships tied to the original relative positions.

From temporal relationships to dependency parameterization. The above analysis provides evidence of shared temporal relationship structures across windows and variables. To connect these historical relationships with forecasting, we consider an idealized model in which historical and future patch representations are related by a shared temporal evolution operator.

For a sample-variable context i, let $\mathbf{X}_i \in \mathbb{R}^{N\times d}$ and $\mathbf{Y}_i \in \mathbb{R}^{M\times d}$ denote row-normalized historical and future patch representations in a common state space, where $N = L/P$ and $M = H/P$, assuming both lengths are divisible by P. Suppose

$$\mathbf{Y}_i = \mathbf{T}\mathbf{X}_i + \boldsymbol{\varepsilon}_i, \qquad \mathbb{E}[\boldsymbol{\varepsilon}_i \mid \mathbf{X}_i] = 0, \qquad \mathbf{T} \in \mathbb{R}^{M\times N}, \tag{22}$$

where $\mathbf{T}$ is shared across the contexts under consideration and $\boldsymbol{\varepsilon}_i$ captures residual variation. The historical auto-relation and history–future cross-relation then satisfy

$$\mathbf{G}_i = \mathbf{X}_i\mathbf{X}_i^\top, \qquad \boldsymbol{\Gamma}_i = \mathbf{Y}_i\mathbf{X}_i^\top = \mathbf{T}\mathbf{G}_i + \boldsymbol{\varepsilon}_i\mathbf{X}_i^\top, \tag{23}$$

and hence $\mathbb{E}[\mathbf{\Gamma}_i \mid \mathbf{X}_i] = \mathbf{T}\mathbf{G}_i$.

Motivated by the observed spectral concentration, suppose the historical relation family admits an approximate representation through a mean structure and r shared directions:

$$\mathbf{G}_i = \boldsymbol{\mu}_G + \sum_{\ell=1}^{r} \beta_{i,\ell}\mathbf{Q}_\ell + \mathbf{\Delta}_i, \tag{24}$$

where $\mathbf{\Delta}_i$ denotes the approximation residual. It follows that

$$\mathbf{\Gamma}_i = \mathbf{T}\boldsymbol{\mu}_G + \sum_{\ell=1}^{r} \beta_{i,\ell}(\mathbf{T}\mathbf{Q}_\ell) + \left(\mathbf{T}\mathbf{\Delta}_i + \boldsymbol{\varepsilon}_i\mathbf{X}_i^\top\right). \tag{25}$$

Thus, under shared temporal evolution, the historical directions $\mathbf{Q}_\ell$ give rise to shared directions $\mathbf{T}\mathbf{Q}_\ell$ in the history–future relation family. When the residual terms are small, the same coefficients $\beta_{i,\ell}$ organize both families. This provides a conditional link between reusable historical relationships and a compact description of relationships relevant to forecasting.

Motivated by this connection, TStruct parameterizes temporal aggregation through shared structure matrices:

$$\mathbf{A}_i = \mathrm{Softmax}_{\mathrm{row}}\left(\sum_{k=1}^{K} \alpha_{i,k}\mathbf{B}_k\right), \qquad \widetilde{\mathbf{Z}}_i = \mathbf{A}_i\mathbf{Z}_i, \tag{26}$$

where $\mathbf{Z}_i$ denotes projected historical features. Learned jointly across training observations under forecasting supervision, the shared matrices $\mathbf{B}_k$ define a compact family of temporal aggregation operators $\mathbf{A}_i$ on historical patch coordinates. This parameterization is intended to capture stable temporal dependency patterns relevant to forecasting, while the forecasting head maps the aggregated features to the future horizon. The connection motivates the shared structural parameterization of $\mathbf{A}_i$ without requiring equality with $\mathbf{G}_i$ or $\mathbf{\Gamma}_i$, or inheritance of their spectra. By drawing on patterns shared across observations, TStruct aims to reduce reliance on incidental relationships in noisy time series and improve forecasting generalization. These intended benefits are evaluated in Section 4.4 and Appendix C.4.

B Implementation Details

All experiments performed in this study are conducted on a workstation equipped with NVIDIA RTX PRO 6000 GPUs with 96GB memory. TStruct is implemented in PyTorch 2.8.0. We fix the random seed to 2026 for our runs. Following the standard evaluation protocol [30], we set *drop_last=False* for the data loader during testing to ensure that all test samples are evaluated.

For the main forecasting experiments, TStruct uses a fixed input look-back length of $L = 720$. We train TStruct using the Adam optimizer [16] for at most 25 epochs with an early stopping patience of 10 epochs. The best checkpoint is selected according to the validation loss.

Baseline configurations. To compare the best performance of each method within the evaluated configurations, we retain the training objectives used by the respective methods and allow look-back selection where specified below. The sources and configurations of the results in Table 1 are as follows: **TStruct.** We report our results using the Huber+FFT objective described below and a fixed look-back length of $L = 720$. **DUET.** We use the paper reported results obtained with the L1 objective, selecting the best performing look-back length from $L \in \{96, 192, 336, 512, 720\}$ for each dataset horizon setting. **PhaseFormer.** We use the paper reported results obtained with the Huber objective and $L = 720$. **Other baselines.** We run their released implementations with their native training objectives and select the best performing look-back length from $L \in \{96, 192, 336, 512, 720\}$ for each dataset horizon setting.

Training objective. We optimize TStruct with a hybrid objective that combines Huber loss and FFT loss [34]. Given the prediction $\hat{\mathbf{Y}}$ and ground truth $\mathbf{Y}$, the Huber loss is defined as

$$\mathcal{L}_{\mathrm{Huber}} = \frac{1}{|\Omega|}\sum_{i\in\Omega} \ell_\delta(\hat{y}_i - y_i), \tag{27}$$

where Ω denotes all predicted entries and we set $\delta = 0.1$ in all experiments. The element-wise Huber loss is defined as

$$\ell_\delta(e) = \begin{cases} \frac{1}{2}e^2, & |e| \le \delta, \\ \delta\left(|e| - \frac{1}{2}\delta\right), & |e| > \delta. \end{cases} \tag{28}$$

The FFT loss measures the discrepancy between predictions and ground truth in the frequency domain:

$$\mathcal{L}_{\mathrm{FFT}} = \left\| \mathrm{rFFT}(\hat{\mathbf{Y}}, \mathrm{dim} = 1) - \mathrm{rFFT}(\mathbf{Y}, \mathrm{dim} = 1) \right\|_1, \tag{29}$$

where $\mathrm{rFFT}(\cdot, \mathrm{dim} = 1)$ denotes the real-valued fast Fourier transform along the temporal dimension, and $\|\cdot\|_1$ is implemented as the mean absolute value over all frequency-domain entries. The final training objective is

$$\mathcal{L} = \mathcal{L}_{\mathrm{Huber}} + \lambda_{\mathrm{FFT}}\mathcal{L}_{\mathrm{FFT}}, \tag{30}$$

where λ_{FFT} is selected from $\{0.0, 0.1, 0.2, 0.3\}$.

Evaluation metrics. Following prior LTSF studies [46, 39], we use Mean Squared Error (MSE) and Mean Absolute Error (MAE) as evaluation metrics:

$$\mathrm{MSE} = \frac{1}{|\Omega|}\sum_{i\in\Omega}(\hat{y}_i - y_i)^2, \quad \mathrm{MAE} = \frac{1}{|\Omega|}\sum_{i\in\Omega}|\hat{y}_i - y_i|. \tag{31}$$

Hyperparameter search space. We tune the hyperparameters over the following search space. The patch size is selected from $P \in \{24, 48, 72, 120, 180, 360\}$, the number of encoder layers from $\{1, 3\}$, the hidden dimension from $D \in \{128, 256, 512\}$, and the number of shared temporal structures from $K \in \{1, 2, 4, 5, 8, 16, 32\}$. For regularization, the MLP dropout rate is selected from $\{0.1, 0.2, 0.3, 0.4\}$, the TSM dropout rate from $\{0.0, 0.1, 0.2, 0.3, 0.4\}$, and the PCAhead dropout rate from $\{0.0, 0.1, 0.2\}$. For optimization, the batch size is selected from $\{16, 64, 256, 512\}$, and the learning rate η is selected from $\{0.001, 0.0015, 0.002, 0.005, 0.015\}$.

C Additional Experimental Analyses

C.1 Random Seed Stability

Table 5: Stability analysis under different random seeds.

Seed		2024		2025		2026		2027		2028		Mean		Std	
Dataset	**Metric**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**
ETTh1	96	0.345	0.380	0.344	0.380	0.345	0.381	0.344	0.390	0.345	0.380	0.345	0.382	0.001	0.004
	192	0.382	0.404	0.383	0.406	0.383	0.405	0.384	0.406	0.384	0.406	0.383	0.405	0.001	0.001
	336	0.407	0.421	0.405	0.421	0.405	0.420	0.407	0.422	0.407	0.421	0.406	0.421	0.001	0.001
	720	0.424	0.453	0.426	0.453	0.431	0.456	0.428	0.456	0.424	0.450	0.427	0.454	0.003	0.003
	Avg.	**0.390**	**0.415**	**0.390**	**0.415**	**0.391**	**0.416**	**0.391**	**0.419**	**0.390**	**0.414**	**0.390**	**0.416**	**0.001**	**0.002**
Weather	96	0.137	0.179	0.137	0.179	0.137	0.181	0.138	0.180	0.138	0.180	0.137	0.180	0.001	0.001
	192	0.181	0.225	0.182	0.225	0.181	0.226	0.180	0.225	0.181	0.225	0.181	0.225	0.001	0.000
	336	0.231	0.267	0.231	0.268	0.230	0.267	0.231	0.269	0.230	0.268	0.231	0.268	0.001	0.001
	720	0.306	0.325	0.305	0.325	0.304	0.324	0.305	0.325	0.305	0.326	0.305	0.325	0.001	0.001
	Avg.	**0.214**	**0.249**	**0.214**	**0.249**	**0.213**	**0.250**	**0.214**	**0.250**	**0.214**	**0.250**	**0.214**	**0.249**	**0.000**	**0.000**

To examine the stability of TStruct under different random initializations and stochastic training processes, we repeat experiments with five random seeds, i.e., 2024, 2025, 2026, 2027, and 2028. Table 5 reports the results on ETTh1 and Weather, including the mean and standard deviation across seeds.

The results show that TStruct is highly stable across different random seeds. The standard deviation is consistently very small, mostly ranging from 0.000 to 0.002 for both MSE and MAE. This indicates that the performance of TStruct is not sensitive to random initialization or stochastic optimization, and the observed improvements are reproducible rather than caused by favorable random seeds.

Table 6: Sensitivity analysis under different initialization strategies.

Initialization		Random		Orthogonal		Xavier		Kaiming	
Dataset	Metric	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	96	0.366	0.398	0.345	0.380	0.345	0.380	0.345	0.380
	192	0.421	0.439	0.383	0.405	0.383	0.405	0.383	0.405
	336	0.426	0.442	0.407	0.420	0.407	0.420	0.407	0.420
	720	0.497	0.503	0.433	0.456	0.434	0.459	0.430	0.458
	Avg.	**0.428**	**0.446**	**0.392**	**0.415**	**0.392**	**0.416**	**0.391**	**0.416**
Weather	96	0.142	0.187	0.137	0.181	0.137	0.180	0.137	0.180
	192	0.187	0.230	0.181	0.226	0.182	0.226	0.181	0.225
	336	0.247	0.282	0.230	0.267	0.231	0.267	0.230	0.268
	720	0.327	0.347	0.304	0.324	0.304	0.325	0.305	0.325
	Avg.	**0.226**	**0.262**	**0.213**	**0.250**	**0.214**	**0.250**	**0.213**	**0.250**

C.2 Initialization Study

We study the effect of different initialization strategies for the shared temporal structures $\mathcal{B}$. Specifically, we compare Random, Orthogonal, Xavier, and Kaiming initialization, as reported in Table 6. The results show that the Random baseline yields noticeably higher forecasting errors on both ETTh1 and Weather, whereas Orthogonal, Xavier, and Kaiming achieve consistently strong and comparable performance.

All initialization strategies are applied directly to the $K \times N \times N$ basis tensor. Orthogonal initialization gives the vectorized bases mutually orthogonal directions with unit Frobenius norm, while Xavier and Kaiming scale the sampling variance according to the tensor dimensions. These strategies control the initial magnitude of the dependency scores, which are subsequently converted into aggregation weights through softmax. This provides a plausible explanation for their favorable performance: appropriately scaled initial scores moderate arbitrary preferences over patch positions, allowing temporal dependency patterns to develop through forecasting supervision. The role of initialization scale is also consistent with observations on learning dynamics in deep linear networks [32].

We use Orthogonal initialization as the default. It provides diverse initial structure directions with a common scale, reducing initial redundancy among the shared bases while achieving consistently strong empirical performance.

C.3 Training Objective Comparison

We investigate the effect of the training objective on forecasting performance by evaluating TStruct, DUET, and PhaseFormer under the same two objectives, Huber+FFT and plain MSE. The remaining experimental settings follow Appendix B.

Table 7 shows that TStruct ranks first in MSE on all seven datasets and in MAE on six under both objectives, with PhaseFormer leading in MAE on Traffic. Huber+FFT further improves TStruct's MSE and MAE across all seven datasets compared with plain MSE. Its strong performance under plain MSE supports the effectiveness of shared temporal structure modeling, while the hybrid objective provides additional gains.

C.4 Noise Injection Robustness

To evaluate robustness under local abnormal perturbations, we conduct noise injection experiments on ETTh1. For each normalized input window, whose values are standardized to approximately zero mean and unit variance, we randomly select a continuous segment and inject Gaussian noise into this segment. The noise ratio controls the length of the perturbed segment relative to the input window, while the noise strength controls the magnitude of the injected Gaussian noise. The start position of the noisy segment is randomly sampled within each input window.

We compare TSM and MHA across different noise ratios and strengths, with the results summarized in Table 8. TSM achieves lower MSE and MAE throughout both experiments. As the noise ratio

Table 7: Forecasting performance under two training objectives with $L = 720$ for all methods. The lowest MSE and MAE for each dataset within each objective are highlighted in bold.

Loss	**Huber+FFT**						**Plain MSE**					
Method	**TStruct**		**DUET**		**PhaseFormer**		**TStruct**		**DUET**		**PhaseFormer**	
Datasets	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**
ETTh1	**0.390**	**0.415**	0.430	0.450	0.417	0.425	**0.410**	**0.432**	0.425	0.447	0.431	0.437
ETTh2	**0.331**	**0.381**	0.364	0.410	0.374	0.401	**0.349**	**0.400**	0.358	0.405	0.358	0.402
ETTm1	**0.337**	**0.367**	0.432	0.426	0.351	0.378	**0.345**	**0.373**	0.382	0.409	0.358	0.384
ETTm2	**0.243**	**0.304**	0.282	0.342	0.253	0.314	**0.246**	**0.311**	0.279	0.338	0.266	0.329
Weather	**0.213**	**0.249**	0.218	0.252	0.231	0.271	**0.217**	**0.258**	0.249	0.293	0.231	0.275
Electricity	**0.151**	**0.244**	0.158	0.251	0.159	0.249	**0.157**	**0.251**	0.158	0.256	0.161	0.252
Traffic	**0.381**	0.252	0.387	0.267	0.387	**0.249**	**0.385**	0.264	0.388	0.270	0.393	**0.259**

increases from 0.05 to 0.30, its MSE ranges from 0.392 to 0.400, remaining 0.042–0.047 below MHA. At a noise strength of 8, the MSE gap reaches 0.075, with TSM obtaining 0.404 compared with 0.479 for MHA.

Table 8: Noise injection robustness on ETTh1 under varying noise ratios (left) and strengths (right). Δ denotes the error reduction of TSM relative to MHA.

Setting	**Varying noise ratio**						**Setting**	**Varying noise strength**					
Method	**TSM**		**MHA**		**Δ**		**Method**	**TSM**		**MHA**		**Δ**	
Ratio	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**	**Strength**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**
0.05	0.392	0.417	0.434	0.449	0.042	0.033	2	0.393	0.417	0.437	0.452	0.044	0.035
0.10	0.394	0.418	0.441	0.455	0.047	0.036	3	0.394	0.418	0.441	0.455	0.047	0.036
0.20	0.397	0.425	0.443	0.457	0.046	0.033	5	0.396	0.420	0.434	0.449	0.038	0.029
0.30	0.400	0.427	0.442	0.458	0.042	0.031	8	0.404	0.426	0.479	0.473	0.075	0.047

These results support the robustness of shared temporal structure modeling under local perturbations. By aggregating temporal information through structures learned across observations, TSM provides a shared basis for processing corrupted inputs, which may help explain its consistently lower forecasting errors across the tested noise conditions.

C.5 Hyperparameter Sensitivity

We further analyze the sensitivity of TStruct to two key hyperparameters: the number of shared temporal structures K and the number of patches N. Experiments are conducted on the ETT benchmarks under different prediction horizons.

Figure 6 reports the forecasting performance under different values of K. Overall, TStruct remains stable across a wide range of K. A very small K may limit the diversity of shared temporal structures, making it difficult to represent heterogeneous temporal dependencies. In contrast, an overly large K may introduce redundant structures and increase optimization difficulty. The results show that a moderate number of shared structures is sufficient for strong performance, and we use $K = 4$ as the default setting in our main experiments.

Figure 7 shows the sensitivity to the number of patches N, which controls the resolution of the patch-level temporal dependency matrix. TStruct achieves robust performance within a broad middle range, especially when $N \in [10, 30]$. When N is too small, the temporal relationship matrix becomes overly coarse and may underfit fine-grained temporal dependencies. When N is too large, each patch contains fewer time points, which may make the learned dependency structures more sensitive

to local noise and increase the risk of overfitting. These results indicate that TStruct is robust to reasonable patch-level resolutions while benefiting from a balanced choice of N.

C.6 Efficiency and Complexity Analysis

We provide an additional efficiency analysis on the Electricity dataset to evaluate the accuracy-efficiency trade-off of TStruct. Since TStruct models temporal dependencies at the patch level, we compare it with representative patch-based Transformer baselines, including Crossformer and PatchTST. All models are evaluated under the same setting with input length $L = 720$, patch size and stride $P = S = 48$, and hidden dimension $D = 512$. We report forecasting accuracy, parameter number, multiply-accumulate operations (MACs), GPU memory usage, training time per step, and inference time per batch in Table 9.

Complexity analysis. The efficiency of TStruct mainly comes from its structured temporal dependency modeling. In attention-based patch-level temporal mixing, the dependency score matrix is computed from input-dependent query-key interactions, which introduces a score estimation cost of $O(N^2D)$, where N is the number of patches and D is the hidden dimension. In contrast, TSM constructs temporal dependency scores by combining K shared temporal structures, with a score construction cost of $O(N^2K)$. Since $K \ll D$, TSM provides a more compact way to estimate temporal dependency scores and avoids fully input-dependent pairwise attention computation.

After obtaining the normalized dependency matrix, TSM still performs patch-level feature aggregation, so the overall temporal mixing remains quadratic in N. The key difference is that TSM replaces unconstrained input-dependent attention score estimation with shared-adaptive structure composition. This makes temporal dependency modeling more structured and computationally efficient while preserving sample- and variable-specific adaptability.

Empirical efficiency. As shown in Table 9, TStruct achieves the best MSE while requiring fewer parameters, lower MACs, less GPU memory, and faster training and inference than the patch-based Transformer baselines. Compared with PatchTST, TStruct reduces parameters from 5.496M to 3.324M and MACs from 23.134G to 11.779G, while reducing MSE from 0.171 to 0.151. Compared with Crossformer, TStruct also achieves substantially lower memory usage and faster runtime. These results show that TStruct improves forecasting accuracy with a favorable efficiency profile by replacing fully input-dependent attention with shared-adaptive temporal dependency modeling.

Table 9: Efficiency comparison on the Electricity dataset. Crossformer [45] and PatchTST [28] are included as representative patch-based Transformer baselines. All models are evaluated under the same setting. We report MSE, parameters, MACs, GPU memory, training time per step, and inference time per batch. **Bold** indicates the best result, and underline indicates the second-best result.

Model	MSE	Params (M)	MACs (G)	Memory (GB)	Train Time (s)	Infer Time (ms)
Crossformer	0.171	36.936	47.543	22.932	0.184	75.474
PatchTST	0.171	5.496	23.134	12.848	0.269	326.133
TStruct	**0.151**	**3.324**	**11.779**	**7.143**	**0.059**	**56.059**
Improvement (↓)	**1.13×**	**1.65×**	**1.96×**	**1.80×**	**4.56×**	**1.35×**

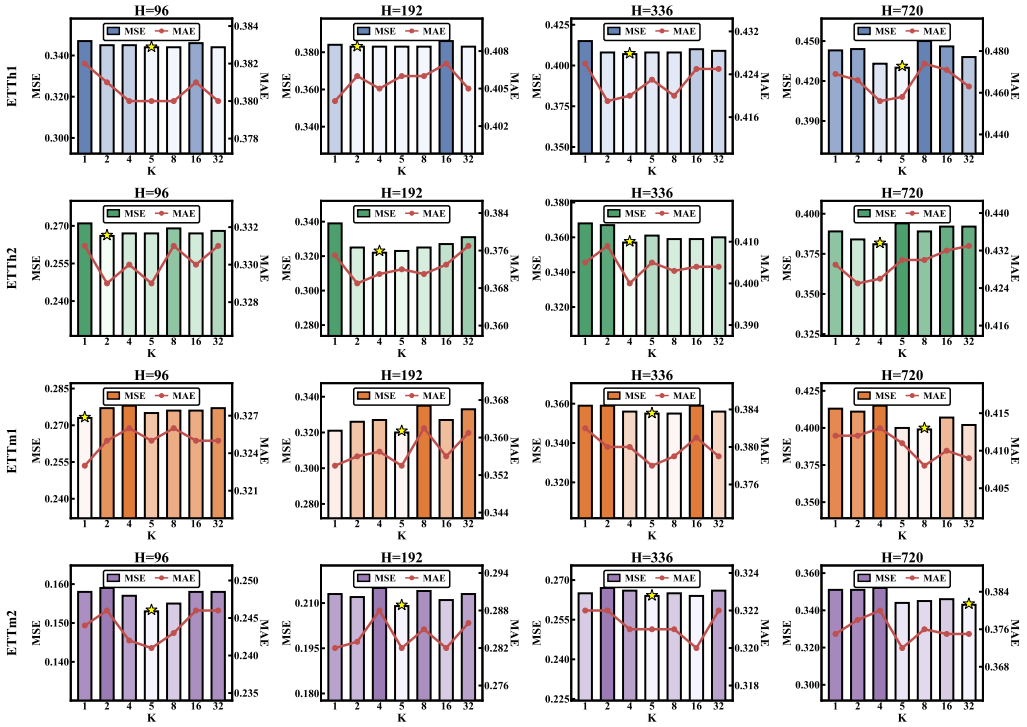


Figure 6: Hyperparameter sensitivity analysis for the number of shared temporal structures K on the ETT benchmarks. Bar plots denote MSE and line plots denote MAE under different prediction horizons.

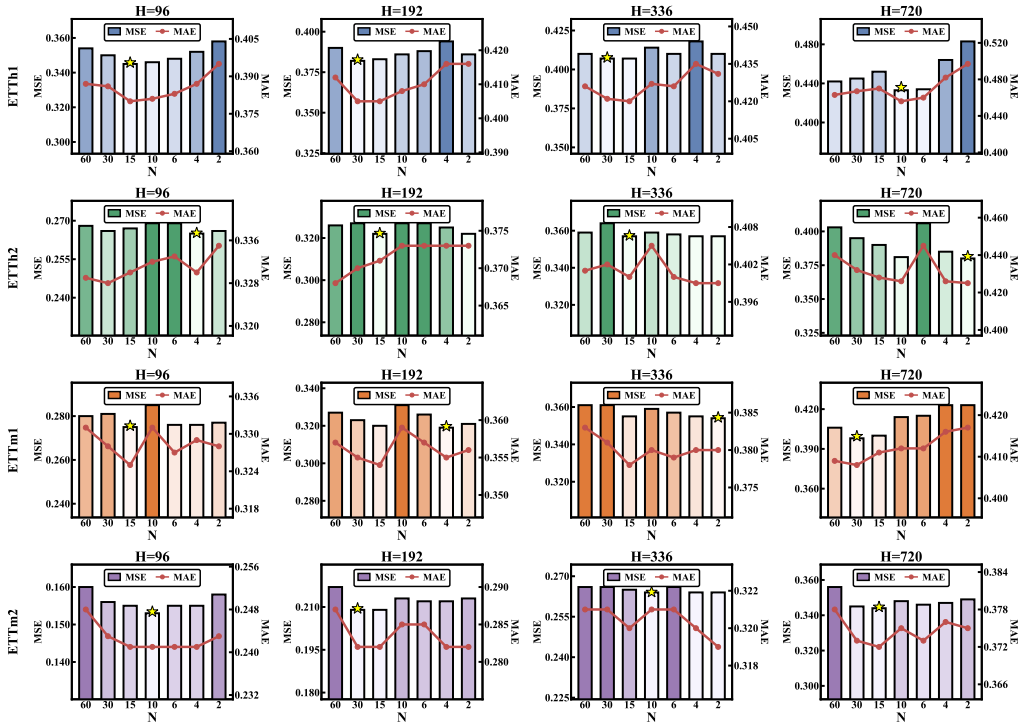


Figure 7: Hyperparameter sensitivity analysis for the number of patches N on the ETT benchmarks. TStruct remains stable within a broad middle range of patch-level resolutions, while overly small or overly large N may lead to underfitting or overfitting.

D Additional Interpretability Visualizations

We provide additional visualizations of the learned temporal structures across all benchmark datasets. Figures 8 to 14 directly visualize the learned shared temporal structure matrices before softmax normalization, as the concentration of weights after softmax can make finer structural differences difficult to distinguish visually. Red and blue indicate higher and lower relative aggregation preferences, respectively, rather than positive and negative dependencies. The visualizations show that the learned structures are not random mixing patterns. Instead, they exhibit structured and reusable temporal dependency patterns across datasets, including diagonal-dominant local continuity, repeated stripe-like interactions, smooth long-range transitions, and more abstract patch-level dependency patterns.

Several observations can be made. First, many structures contain clear diagonal or near-diagonal patterns, indicating that neighboring patches often preserve strong local temporal continuity. Second, repeated stripe-like or checkerboard patterns appear in several datasets, suggesting recurrent interactions between relative patch positions. Third, some structures exhibit smooth low-to-high transitions across rows or columns, reflecting broader temporal dependency tendencies beyond local neighborhoods. These patterns show that TStruct does not rely on a single predefined temporal prior, such as periodicity alone.

The learned structures also vary across datasets. The ETT datasets often contain a mixture of local continuity, alternating recurrent patterns, and smooth transition structures, reflecting heterogeneous temporal dynamics. Weather, Electricity, and Traffic exhibit more regular banded or repeated structures, indicating that TStruct can capture dataset-specific temporal regularities through shared dependency matrices. Overall, these visualizations provide qualitative evidence that the temporal structures learned by TStruct are interpretable, structured, and adaptive to different datasets.

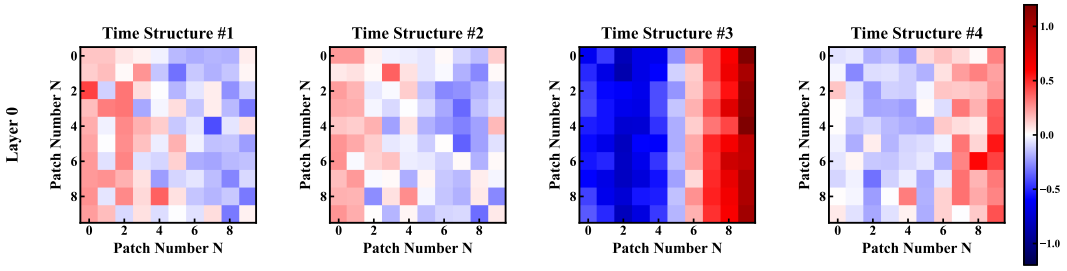


Figure 8: **Learned Temporal Structures on ETTh1.**

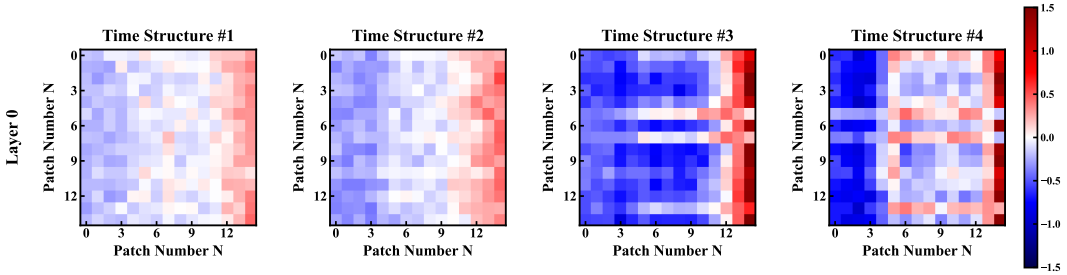


Figure 9: **Learned Temporal Structures on ETTh2.**

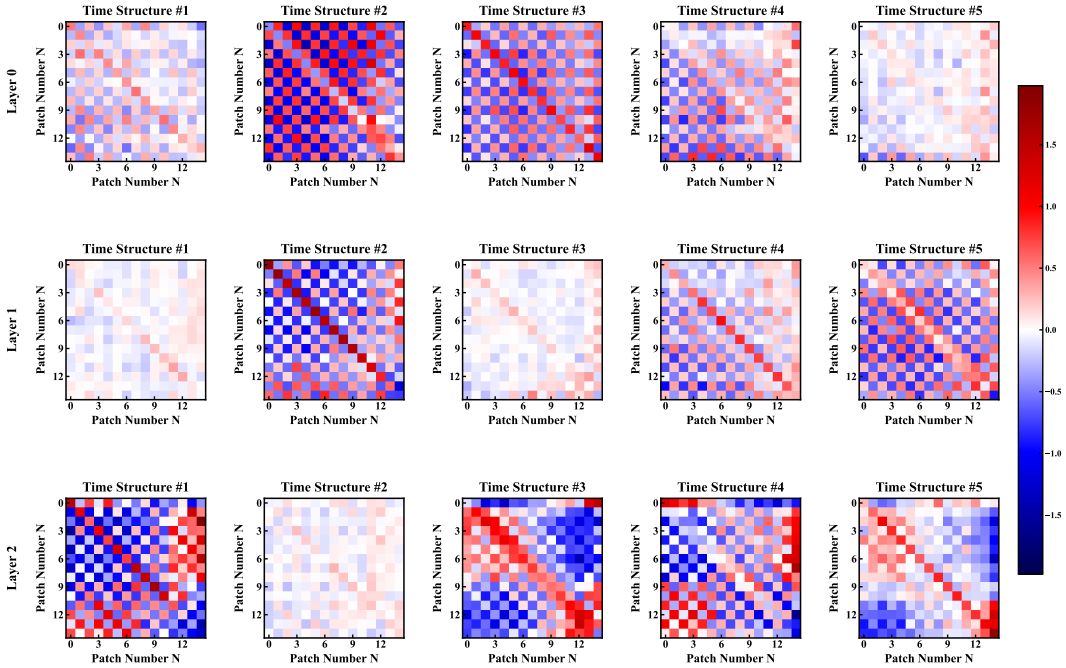


Figure 10: **Learned Temporal Structures on ETTm1.**

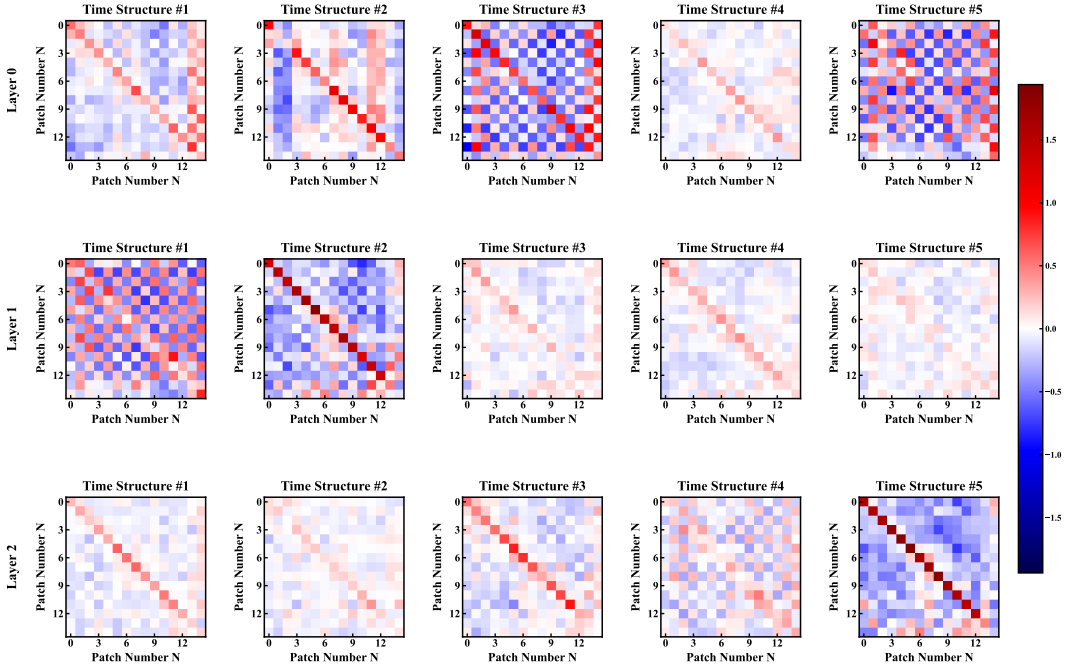


Figure 11: **Learned Temporal Structures on ETTm2.**

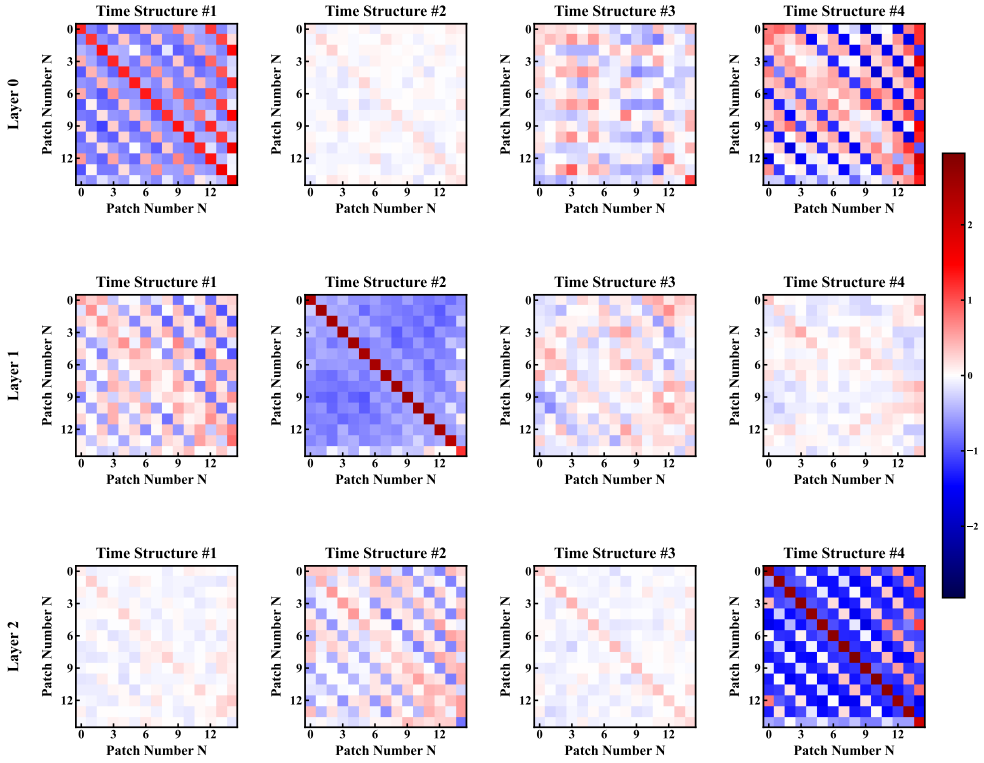


Figure 12: **Learned Temporal Structures on Weather.**

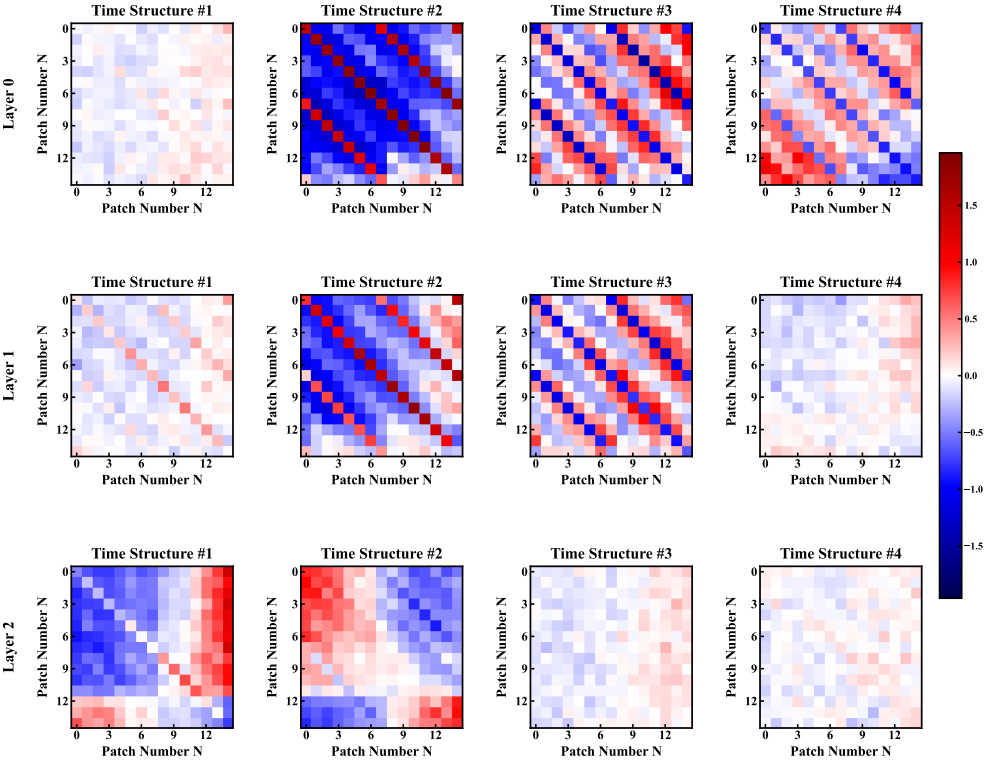


Figure 13: **Learned Temporal Structures on Electricity.**

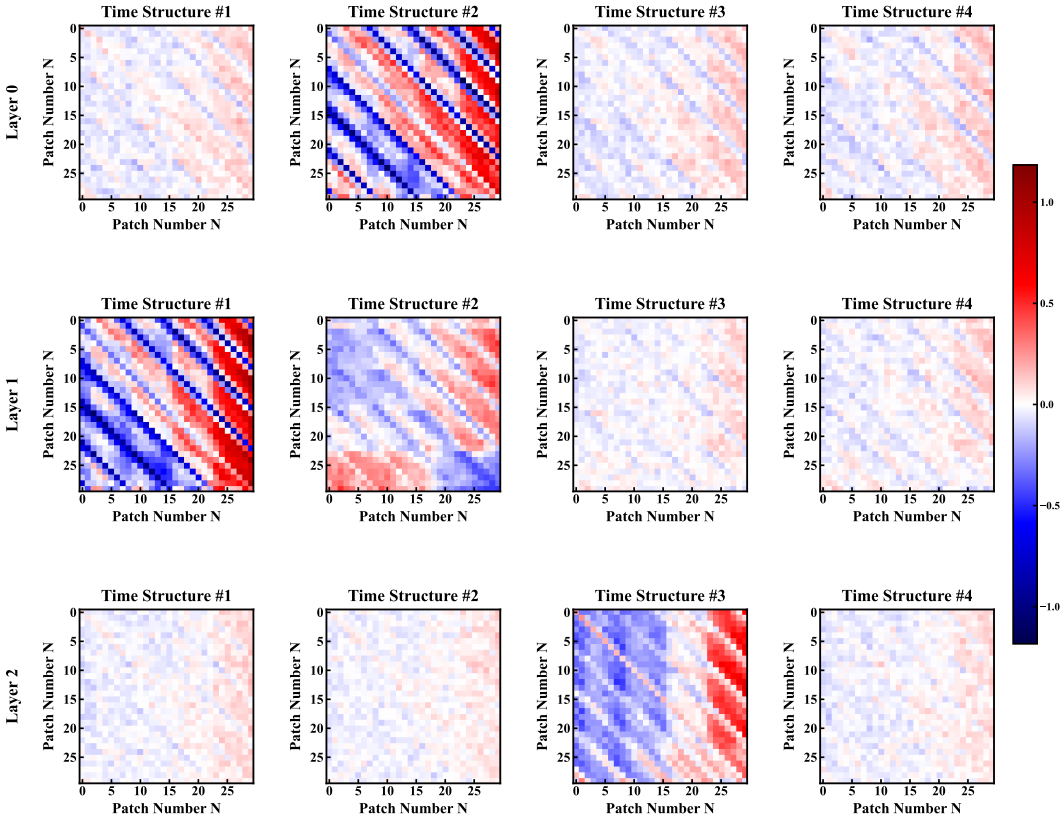


Figure 14: **Learned Temporal Structures on Traffic.**

E Detailed Experimental Results

This section provides the complete experimental results corresponding to the summarized tables in the main paper. While the main text reports average results over all prediction horizons for compactness, here we present the detailed results under each prediction length $H \in \{96, 192, 336, 720\}$. These results provide a more fine-grained comparison across horizons and further support the consistency of TStruct under different forecasting settings.

E.1 Detailed Forecasting Results

Table 10 reports the full forecasting results of TStruct and all baseline methods on seven benchmarks. The results include both MSE and MAE under each prediction horizon, providing the detailed counterpart to the averaged results reported in the main paper.

Table 10: Full multivariate time series forecasting results. The best results are marked in **bold red**, and the second-best results are marked in blue underline.

Methods		TStruct (ours)		DUET [2025]		PhaseFormer [2026]		GTR [2026]		TimeEmb [2025]		TQNet [2025]		CycleNet [2024]		PatchTST [2023]		DLinear [2023]	
Datasets	**Metric**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**
ETTh1	96	**0.345**	**0.380**	0.352	0.384	0.359	0.382	0.367	0.391	0.372	0.400	0.371	0.393	0.374	0.396	0.375	0.399	0.379	0.403
	192	**0.383**	0.405	0.398	0.409	0.397	**0.404**	0.409	0.420	0.413	0.427	0.410	0.418	0.406	0.415	0.414	0.421	0.408	0.419
	336	**0.407**	**0.421**	0.414	0.426	0.425	0.424	0.432	0.434	0.438	0.443	0.432	0.433	0.431	0.430	0.431	0.436	0.440	0.440
	720	**0.427**	0.454	0.429	0.455	0.431	**0.450**	0.451	0.463	0.449	0.462	0.466	0.468	0.450	0.464	0.449	0.466	0.471	0.493
	Avg.	**0.390**	**0.415**	0.398	0.419	0.403	**0.415**	0.415	0.419	0.418	0.433	0.420	0.428	0.415	0.426	0.417	0.431	0.425	0.439
ETTh2	96	**0.267**	**0.330**	0.270	0.336	0.275	0.338	0.274	0.341	0.290	0.348	0.290	0.349	0.271	0.337	0.274	0.337	0.300	0.364
	192	**0.322**	**0.371**	0.332	0.374	0.341	0.376	0.340	0.385	0.347	0.387	0.367	0.397	0.332	0.380	0.348	0.384	0.387	0.423
	336	0.357	0.400	**0.353**	**0.397**	0.369	0.405	0.363	0.409	0.376	0.411	0.363	0.406	0.362	0.408	0.377	0.416	0.490	0.487
	720	**0.381**	0.426	0.382	**0.425**	0.402	0.436	0.405	0.439	0.399	0.439	0.412	0.443	0.415	0.449	0.406	0.441	0.704	0.597
	Avg.	**0.331**	**0.381**	0.334	0.383	0.347	0.389	0.346	0.394	0.353	0.396	0.358	0.399	0.345	0.394	0.351	0.395	0.470	0.468
ETTm1	96	**0.275**	**0.325**	0.279	0.333	0.293	0.344	0.283	0.337	0.293	0.346	0.294	0.345	0.299	0.348	0.289	0.343	0.300	0.345
	192	**0.320**	**0.354**	**0.320**	0.358	0.323	0.361	0.323	0.365	0.326	0.366	0.336	0.372	0.334	0.367	0.329	0.368	0.336	0.366
	336	0.355	0.378	**0.348**	**0.377**	0.358	0.381	0.355	0.384	0.356	0.382	0.368	0.392	0.364	0.387	0.362	0.390	0.367	0.386
	720	**0.400**	0.411	0.405	**0.408**	0.412	0.410	0.420	0.422	0.405	0.410	0.412	0.443	0.410	0.411	0.416	0.423	0.419	0.416
	Avg.	**0.337**	**0.367**	0.338	0.369	0.347	0.374	0.345	0.377	0.345	0.376	0.353	0.388	0.352	0.378	0.349	0.381	0.356	0.378
ETTm2	96	**0.153**	**0.241**	0.161	0.248	0.163	0.256	0.161	0.252	0.163	0.251	0.166	0.255	0.159	0.247	0.165	0.255	0.164	0.255
	192	**0.209**	**0.282**	0.214	0.286	0.219	0.293	0.221	0.292	0.219	0.290	0.222	0.294	0.214	0.286	0.221	0.293	0.224	0.304
	336	**0.264**	0.321	0.267	0.321	0.269	0.326	0.278	0.326	0.265	**0.320**	0.277	0.337	0.268	0.326	0.276	0.327	0.277	0.337
	720	0.344	**0.372**	0.348	0.374	0.351	0.379	0.359	0.388	**0.343**	**0.372**	0.366	0.386	0.353	0.384	0.362	0.381	0.371	0.401
	Avg.	**0.243**	**0.304**	0.248	0.307	0.251	0.314	0.255	0.315	0.248	0.308	0.258	0.318	0.249	0.311	0.256	0.314	0.259	0.324
Weather	96	**0.137**	**0.181**	0.146	0.191	0.148	0.195	0.147	0.199	0.143	0.193	0.155	0.201	0.164	0.220	0.150	0.200	0.170	0.230
	192	**0.181**	**0.226**	0.188	0.231	0.193	0.237	0.192	0.240	0.188	0.237	0.200	0.242	0.209	0.258	0.191	0.239	0.216	0.273
	336	**0.230**	**0.267**	0.234	0.268	0.242	0.278	0.243	0.281	0.236	0.274	0.248	0.284	0.255	0.292	0.242	0.279	0.258	0.307
	720	**0.304**	0.324	0.305	**0.319**	0.309	0.332	0.324	0.337	0.306	0.325	0.318	0.335	0.320	0.338	0.312	0.330	0.323	0.362
	Avg.	**0.213**	**0.249**	0.218	0.252	0.223	0.261	0.227	0.264	0.218	0.257	0.230	0.266	0.237	0.277	0.224	0.262	0.242	0.293
Electricity	96	**0.123**	**0.216**	0.128	0.219	0.129	0.221	0.127	0.222	0.129	0.225	0.129	0.224	0.128	0.223	0.130	0.222	0.140	0.237
	192	**0.141**	**0.232**	0.145	0.235	0.148	0.238	0.147	0.242	0.145	0.241	0.151	0.245	0.143	0.237	0.148	0.240	0.154	0.251
	336	**0.155**	**0.250**	0.163	0.255	0.165	0.257	0.165	0.260	0.161	0.257	0.166	0.261	0.159	0.254	0.167	0.261	0.169	0.268
	720	**0.185**	**0.279**	0.193	0.281	0.201	0.285	0.199	0.292	0.197	0.289	0.201	0.294	0.197	0.287	0.202	0.291	0.204	0.301
	Avg.	**0.151**	**0.244**	0.157	0.248	0.161	0.250	0.160	0.254	0.158	0.253	0.162	0.256	0.157	0.250	0.162	0.254	0.167	0.264
Traffic	96	**0.352**	**0.235**	0.360	0.238	0.361	0.238	0.367	0.257	0.374	0.268	0.362	0.260	0.381	0.266	0.367	0.251	0.395	0.275
	192	**0.368**	0.244	0.383	0.249	0.373	**0.243**	0.382	0.266	0.387	0.268	0.384	0.269	0.394	0.273	0.385	0.259	0.407	0.280
	336	**0.383**	0.253	0.395	0.259	0.385	**0.248**	0.399	0.275	0.401	0.275	0.423	0.297	0.406	0.279	0.398	0.265	0.417	0.286
	720	**0.422**	0.277	0.435	0.278	0.428	**0.270**	0.441	0.298	0.434	0.275	0.447	0.303	0.441	0.300	0.434	0.287	0.454	0.308
	Avg.	**0.381**	0.252	0.393	0.256	0.387	**0.250**	0.397	0.274	0.399	0.272	0.404	0.282	0.406	0.280	0.396	0.266	0.418	0.287
1st Count		**32**	**23**	3	5	0	7	0	0	1	2	0	0	0	0	0	0	0	0

E.2 Detailed Ablation Results

Table 11 reports the full ablation results under different prediction horizons. These results complement the averaged ablation table in the main paper and show that the benefits of the proposed Temporal Structure Mixer and PCAHead remain consistent across forecasting horizons.

Table 11: Complete ablation results of TStruct under different prediction horizons. TSM→MHA replaces the Temporal Structure Mixer with multi-head attention, TSM→Linear replaces it with a shared linear temporal mixer, and PCAHead→Linear replaces the proposed prediction head with a linear head.

Datasets		**ETTh1**		**ETTh2**		**ETTm1**		**ETTm2**		**Weather**		**Electricity**		**Traffic**	
Method	**Metric**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**
TSM → MHA	96	0.367	0.399	0.285	0.350	0.298	0.340	0.191	0.272	0.143	0.189	0.131	0.225	0.377	0.255
	192	0.421	0.440	0.360	0.395	0.344	0.369	0.222	0.293	0.188	0.232	0.148	0.240	0.390	0.262
	336	0.434	0.447	0.360	0.402	0.424	0.426	0.266	0.322	0.260	0.295	0.167	0.258	0.405	0.272
	720	0.478	0.491	0.393	0.433	0.437	0.429	0.356	0.383	0.326	0.344	0.203	0.291	0.439	0.289
	Avg.	0.425	0.444	0.350	0.395	0.376	0.391	0.259	0.318	0.229	0.265	0.162	0.254	0.403	0.270
TSM → Linear	96	0.364	0.393	0.271	0.327	0.278	0.331	0.158	0.243	0.140	0.185	0.125	0.218	0.352	0.235
	192	0.395	0.413	0.338	0.373	0.334	0.362	0.218	0.285	0.187	0.232	0.143	0.234	0.368	0.245
	336	0.418	0.428	0.365	0.404	0.359	0.382	0.266	0.322	0.236	0.273	0.157	0.250	0.383	0.254
	720	0.433	0.461	0.391	0.431	0.418	0.413	0.351	0.381	0.309	0.327	0.187	0.280	0.423	0.277
	Avg.	0.403	0.424	0.341	0.384	0.347	0.372	0.248	0.308	0.218	0.254	0.153	0.246	0.382	0.253
PCAHead → Linear	96	0.348	0.382	0.268	0.331	0.275	0.325	0.160	0.248	0.137	0.179	0.124	0.217	0.353	0.236
	192	0.385	0.405	0.330	0.374	0.323	0.354	0.212	0.283	0.184	0.227	0.142	0.232	0.369	0.245
	336	0.411	0.423	0.364	0.405	0.357	0.379	0.267	0.322	0.230	0.268	0.156	0.250	0.383	0.254
	720	0.431	0.459	0.393	0.432	0.408	0.412	0.354	0.376	0.310	0.328	0.186	0.279	0.423	0.277
	Avg.	0.394	0.417	0.339	0.386	0.341	0.368	0.248	0.307	0.215	0.251	0.152	0.245	0.382	0.253
TStruct	96	**0.345**	**0.380**	**0.267**	**0.330**	**0.275**	**0.325**	**0.155**	**0.241**	**0.137**	**0.181**	**0.123**	**0.216**	**0.352**	**0.235**
	192	**0.383**	**0.405**	**0.322**	**0.371**	**0.320**	**0.354**	**0.209**	**0.282**	**0.181**	**0.226**	**0.141**	**0.232**	**0.368**	**0.244**
	336	**0.407**	**0.421**	**0.357**	**0.400**	**0.355**	**0.378**	**0.265**	**0.320**	**0.230**	**0.267**	**0.155**	**0.249**	**0.383**	**0.253**
	720	**0.427**	**0.454**	**0.381**	**0.426**	**0.400**	**0.411**	**0.344**	**0.372**	**0.304**	**0.324**	**0.185**	**0.279**	**0.422**	**0.277**
	Avg.	**0.390**	**0.416**	**0.331**	**0.381**	**0.337**	**0.367**	**0.243**	**0.303**	**0.213**	**0.249**	**0.151**	**0.244**	**0.381**	**0.252**

F Limitations and Future Directions

TStruct models temporal dependencies through a compact set of shared temporal structure matrices, with controlled initialization and softmax normalized composition and aggregation. This parameterization provides an effective way to learn reusable temporal structures for LTSF. Their adaptation across patch resolutions and transfer across datasets and domains at scale remain directions for further exploration.

Patch resolution dependency. The learned temporal structures are tied to a specific patch resolution. Since each temporal dependency matrix is parameterized over N patch positions, changing the patch size or look-back length can alter the matrix dimensions or the temporal meaning of its coordinates. Although TStruct remains effective under different look-back lengths, directly sharing temporal structures across different patch resolutions remains non-trivial. A promising direction is to develop resolution-adaptive temporal structures that can be transferred, interpolated, or composed across different patch scales.

Temporal structure foundation models. Even after fine-tuning, time series foundation models can struggle to outperform small models trained on individual datasets. One possible explanation is the difficulty of learning input representations that generalize across heterogeneous time series, motivating the exploration of temporal dependency structures as an alternative focus of pretraining. Our analysis provides evidence of reusable temporal relationships across temporal positions and variables, while their transfer and generalization across datasets and domains at scale remain untested. A potential direction is to develop a time series foundation model built around a pretrained dictionary of shared temporal dependency structures, with downstream forecasting tasks selecting and composing these structures to form temporal aggregation operators.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The claims in the abstract and introduction are consistent with the paper's contributions and scope. The paper investigates whether temporal relationship patterns can be reused across different temporal windows and variables, supported by spectral analysis of patch-level temporal relationship matrices and comparisons with patch-shuffled and IID Gaussian controls. Motivated by these findings, we propose TStruct, which explicitly learns shared temporal dependency structures and adaptively combines them using dynamic sample information and static variable priors. The effectiveness of this shared-adaptive design is validated through extensive forecasting experiments, ablation studies, and additional analyses of generalization, robustness, computational efficiency, and interpretability. These results support the paper's main claims within the evaluated long-term forecasting settings.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We present the limitations of this work and potential future directions in Appendix F.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.

- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

   Justification: This paper does not include theoretical assumptions and results.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We provide detailed descriptions of the model architecture, datasets, evaluation metrics, experimental setup, hyperparameter settings, and implementation details needed to reproduce the main results. In addition, we release the source code to further facilitate reproducibility.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.

(c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We have publicly released the complete source code at https://github.com/yangzeli/TStruct/, and all datasets used in the paper are publicly available. The repository and supplemental material provide detailed instructions for dataset preparation, model configuration, and running the experiments needed to reproduce the main results.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: The paper specifies the training and testing details needed to understand the reported results. In particular, Sections A, B, and C provide the data splits, prediction settings, hyperparameter choices, optimizer configuration, implementation details, and additional experimental protocols. These details are provided to ensure that the main results can be clearly interpreted and faithfully reproduced.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: The paper reports statistical stability results for the main experiments. Specifically, Appendix C presents results under multiple random seeds, including the mean and standard deviation, which serve as error bars for assessing the robustness of the reported improvements.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
   - If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We provided basic information about our deployment platform and computing power in Section B.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification: We make sure our research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics.

   Guidelines:

- The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: There is no societal impact of the work.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper poses no such risks.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: The assets used in the paper are properly credited and used according to their licenses and terms. Our TStruct forecasting codebase is adapted from the official open-source implementation of DLinear at https://github.com/cure-lab/LTSF-Linear, which is released under the MIT license. We cite the original DLinear paper in the manuscript and strictly follow the MIT license terms. All datasets used in our experiments are publicly available benchmarks and are used in accordance with their respective licenses and terms of use.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, paperswithcode.com/datasets has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [N/A]

Justification: The paper does not release new assets.

Guidelines:

- The answer [N/A] means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A]

Justification: The paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.

- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.