Time Series Forecasting via Direct Per-Step Probability Distribution Modeling

Linghao Kong, Xiaopeng Hong*

The Faculty of Computing, Harbin Institute of Technology, Harbin, China
leonardokong486@gmail.com, hongxiaopeng@ieee.org

Abstract

Deep neural network-based time series prediction models have recently demonstrated superior capabilities in capturing complex temporal dependencies. However, it is challenging for these models to account for uncertainty associated with their predictions, because they directly output scalar values at each time step. To address such a challenge, we propose a novel model named **inter**leaved dual-branch **P**robability **D**istribution **N**etwork (**interPDN**), which directly constructs discrete probability distributions per step instead of a scalar. The regression output at each time step is derived by computing the expectation of the predictive distribution on a predefined support set. To mitigate prediction anomalies, a dual-branch architecture is introduced with interleaved support sets, augmented by coarse temporal-scale branches for long-term trend forecasting. Outputs from another branch are treated as auxiliary signals to impose self-supervised consistency constraints on the current branch's prediction. Extensive experiments on multiple real-world datasets demonstrate the superior performance of interPDN.

Code — https://github.com/leonardokong486/interPDN

1 Introduction

As a highly challenging task in the field of time series, Time Series Forecasting (TSF) has attracted significant and growing attention in recent years. TSF requires predicting data points over future horizons based on historical observations, while demonstrating strong demand across diverse domains such as finance, energy, healthcare, and transportation. Within the mathematical frameworks employed in TSF tasks, neural network-based prediction models currently represent the most prevalent approach (Wang et al. 2024c).

While most TSF models directly predict a sequence with scalars as demonstrated in Figure 1(a), AdaPTS (Benechehab et al. 2025) outputs the mean and variance of the Gaussian distribution at each time step. LangTime (Niu et al. 2025) also estimates per-step probability distribution to compute Kullback-Leibler divergence from true distribution, which serves as the reward function for reinforcement learning policy. These models output probability distribution parameters at each time step, providing users with interpretable confidence intervals, as shown in Figure 1(b).

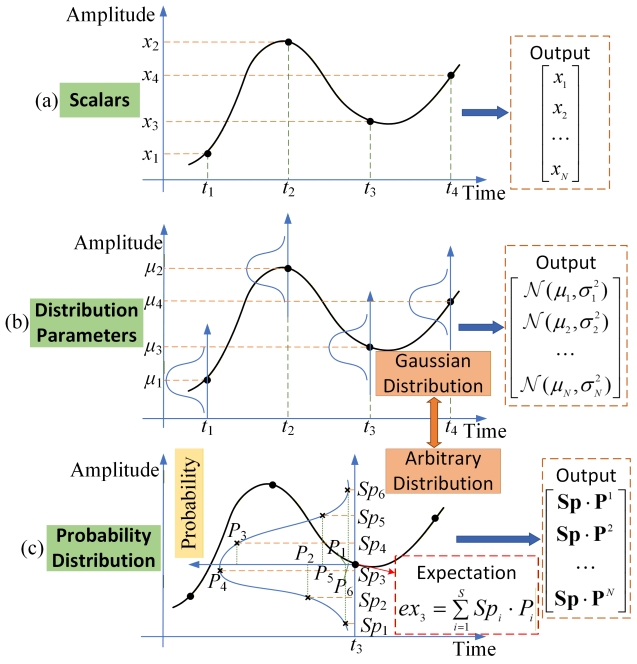

Figure 1: Different ways of per-step time series modeling: (a) Scalar estimation; (b) Probability distribution parameter prediction; (c) Expectation forecasting via discrete probability distribution

Inspired by a crowd counting research (Lin et al. 2025) in Computer Vision (CV), we recast TSF as a direct probabilistic density distribution modeling problem. The backbone does not output a scalar, instead it produces a discrete probability distribution over a predefined support set at each time step. This approach provides advantages for capturing the inherent uncertainty in time series, while avoiding the prior assumption of Gaussian or other distributional forms required by models like AdaPTS. Taking the expectation over the resulting probability distribution yields the prediction for each step, as illustrated in Figure 1(c).

Discrete probability distribution modeling requires quan-

*Corresponding author

tization of output values, which can potentially introduce quantization errors and lead to misclassification at the bin boundary (Sun et al. 2025). Therefore, we propose a dual-branch structure corresponding to interleaved support sets. On one hand, it effectively restricts quantization errors and prevents forecasting anomalies with single branch. On the other hand, the output additional signals enable self-supervised learning. Since the model size doubles while the dataset remains unchanged, the constraints of dual-branch consistency losses are necessary to avoid training issues.

To steer the model towards accurate long-term trend forecasting without overfitting to local details, the interleaved dual-branch architecture is replicated at coarse time scale. The coarse-scale output also serves as an additional self-supervised signal to suppress overfitting caused by doubled model parameters on the original small-scale dataset. Consequently, a consistency loss is applied between the downsampled fine-grained outputs and the coarse-grained outputs.

We summarize our main contributions as follows:

- We transform the TSF task into a direct probability density distribution modeling problem. At each time step, we output a discrete probability distribution, whose expected value serves as the final output.
- To effectively mitigate the quantization error inherent in density distribution modeling, we propose an innovative interleaved dual-branch architecture coupled with corresponding self-supervised constraint.
- Since the model struggles to discern trend evolution in long-term TSF under the normal time scale, we further introduce self-supervised constraints at a coarser scale.
- The proposed interPDN has consistently outperformed state-of-the-art (SOTA) methods in multiple standardized real-world benchmark datasets.

2 Related Work

Statistical models used to lay foundation for traditional TSF models, such as Vector Autoregression (VAR) (Sims 1980) and ARIMA (Box 2013). With the rapid rise of deep neural networks, TSF has drawn on classic models in the field of Natural Language Processing (NLP) and achieved more accurate predictions based on Recurrent Neural Networks (RNNs) (Salinas et al. 2020) and Long-Short-Term Memory networks (LSTMs) (Rangapuram et al. 2018).

Given that conventional TSF models struggle to capture long-term dependencies in sequences and handle non-stationary signals (Wang et al. 2024c), Transformer-based prediction models have gained increasing popularity. Informer (Zhou et al. 2021) introduces the ProbSparse self-attention mechanism and attention distilling, which significantly reduces the computational complexity and memory footprint of Transformer architecture. Autoformer (Wu et al. 2021) specifically targets the periodicity in time series and proposes computing attention between phase-correlated positions across different periods. Unlike most channel-independent Transformer-based models, iTransformer (Liu et al. 2023a) embeds a whole channel into a token to allow modeling variate-wise dependencies. Subsequent research focuses mainly on token organization (Lu, Sun, and Yang 2024), design of attention layers (Lu and Yang 2025), and formulation of key-value pairs (Xue et al. 2023) within Transformer models.

The aforementioned Transformer-based TSF models are typically small-scale, primarily due to the limited volume of time series datasets, making it challenging to train robust large models with strong generalization capabilities (Zhang et al. 2024). Time-LLM (Jin et al. 2023) reprograms time patches into text tokens and employs a frozen LLM as backbone for prediction of time series. Frozen large image inpainting models also exhibit strong zero-shot performance in TSF tasks (Chen et al. 2024a). However, transferring pre-trained models across domains introduces persistent challenges in explainability.

In contrast, compacting models with simple backbones demonstrates superior performance in TSF tasks, such as Multilayer Perceptrons (MLPs) and Convolutional Neural Networks (CNNs). DLinear (Zeng et al. 2023) employs a linear-only architecture to separately model trend and periodicity components in time series. HDMixer (Huang et al. 2024) introduces an adaptive-length patching mechanism that significantly enhances the forecasting capability of MLP architectures. By breaking 1D time series down to 2D images based on periodicity, CNNs effectively expand their receptive fields beyond sequential constraints to capture richer temporal dependencies (Wu et al. 2022). Our model also builds upon a similarly lightweight backbone architecture, while drawing inspiration from existing multi-branch and multi-scale methodologies.

Unlike our interleaved branches, prior TSF models leveraged multi-branch architectures to independently process markedly distinct input features. For instance, after decomposing the input sequence into seasonal and trend components using the moving average technique, separate prediction streams can be employed to forecast each component based on their unique statistical characteristics (Ma et al. 2024; Lin et al. 2024). In terms of learning in frequency-domain, Koopa (Liu et al. 2023b) decomposes time series into time-varying and time-invariant branches, which are modeled separately through local and global Koopman operators. Time-VLM (Zhong et al. 2025) transforms raw time series into images while deriving corresponding text descriptions of input, subsequently feeding temporal, visual, and textual representations into three dedicated channels.

In contrast to our practice of utilizing coarse-grained sequences as self-supervised constraint signals, other TSF models typically apply multi-scale techniques during patching stage (Chen et al. 2024b; Cho and Lee 2025; Tang and Zhang 2025; Han et al. 2025). Unlike parallel multi-scale architectures, TTM (Ekambaram et al. 2024) sequentially incorporates multi-scale patches at distinct hierarchical stages of the model. Pyraformer (Liu et al. 2022) organizes multi-scale time series into a pyramid architecture, performing sparse attention in adjacent steps and layers. Some models also focus on mixing of multi-scale branches, such as AMD (Hu et al. 2025) and TimeMixer (Wang et al. 2024b).

3 Proposed Method

In multivariate TSF, given a historically observed sequence $\mathbf{X}_L = \{\mathbf{x}_{t-L+1}, \cdots\cdots, \mathbf{x}_{t-1}, \mathbf{x}_t\}$ with a length of L, the goal is to predict the target values over the next T time steps $\mathbf{X}_T = \{\mathbf{x}_{t+1}, \mathbf{x}_{t+2}, \cdots\cdots, \mathbf{x}_{t+T}\}$. Herein, $\mathbf{x}_t \in \mathbb{R}^{1\times C}$ denotes the multivariate data at time step t, and C represents the number of channels.

The overall architecture of interPDN is depicted in Figure 2(a). Within each branch, the backbone decomposes the raw input data into seasonal and trend components, which are then processed separately by convolutional and linear modules illustrated in Figure 2(b). To output a per-step probability distribution, probabilistic generation module is introduced. At each distinct time scale, the dual branches produce distributions associated with interleaved support sets, which are shown in Figure 2(c), and their outputs are mixed up by combine modules. The dual branches are replicated at the coarse time scale, so that their outputs guide the model to focus on long-term trends. Since outputs from the additional branch are treated as self-supervised signals, inter-scale and cross-scale consistency losses can constrain the model to prevent prediction deviations.

3.1 Backbone of Each Branch

To address variable lags and cross-channel interference (Zhao and Shen 2024; Yang, Zhu, and Chen 2024) in multivariate TSF, the input multivariate time series $\mathbf{X}_L$ is split into C single-channel sequences $\mathbf{X}_L^i \in \mathbb{R}^{L\times 1}$, following the channel-independent process (Nie et al. 2023). $\mathbf{X}_L^i$ is normalized instance-wise through RevIN method (Kim et al. 2021) to adapt to distribution shift. Using exponential moving average method, trend component $\mathbf{X}_{tr}^i$ is extracted from the single-channel sequence, while the residual becomes seasonal component $\mathbf{X}_{se}^i$.

The seasonal processing stream starts with patching, which is a classic method in TSF for handling longer look-back windows (Nie et al. 2023). Input $\mathbf{X}_{se}^i$ thus turns to $\mathbf{X}_{se,p}^i \in \mathbb{R}^{N\times P}$ with the stride length S_t and patch length P, and the number of patches N can be calculated by $N = \left\lfloor \frac{(L-P)}{S_t} \right\rfloor + 2$. A linear transformation followed by an activation function is applied to the patch count dimension of $\mathbf{X}_{se,p}^i$. Subsequently, 1D convolution is employed along this dimension to aggregate information from adjacent patches. The outputs of the linear block and the convolutional block are then integrated via a ResNet structure (He et al. 2016) and fed into an MLP decoder to obtain $\mathbf{X}_{se,out}^i \in \mathbb{R}^{1\times T}$.

Another stream for trend processing builds upon two linear blocks without any activation layers, thus transforming $\mathbf{X}_{tr}^i$ into $\mathbf{X}_{tr,out}^i \in \mathbb{R}^{1\times T}$. Each linear block requires the application of average pooling and layer normalization subsequent to its fully connected layer. The results of seasonal and trend streams are concatenated along the last dimension to obtain final output $\mathbf{X}_{out}^i \in \mathbb{R}^{1\times 2T}$ of each backbone branch:

$$\mathbf{X}_{out}^i = \mathbf{X}_{tr,out}^i \oplus \mathbf{X}_{se,out}^i. \tag{1}$$

3.2 Probabilistic Generation Module

In conventional TSF models that directly produce scalar predictions, an output projection head is typically used to reduce the dimensionality of $\mathbf{X}_{out}$ and obtain the prediction target $\hat{\mathbf{X}}_T$. However, such output heads fail to account for the uncertainty inherent in TSF, particularly by producing low-confidence predictions at challenging steps in the time sequences. In interPDN, we introduce a probabilistic generation module that outputs per-step discrete probability distribution rather than a scalar. Critically, we make no assumptions about the form of this distribution (e.g., Gaussian, mixture Gaussian, or multinomial or other parametric families). Furthermore, the module explicitly generates the full probability distribution directly, rather than inferring it indirectly via distributional parameters.

A fully connected layer fc_{fine} is employed to expand the dimension of the branch backbone output $\mathbf{X}_{out}^i$ from $2T$ to $S \times T$, where S represents the number of elements in the predefined support set. Subsequently, the output is reshaped:

$$\mathbf{X}_f^i = \mathrm{Reshape}\left(fc_{fine}\left(\mathbf{X}_{out}^i\right)\right), \tag{2}$$

thus forming $\mathbf{X}_f^i \in \mathbb{R}^{T\times S}$.

Given that normalized time-series data tends to follow a normal distribution, applying uniformly spaced partition points to the support set is inadvisable for TSF tasks. It is preferable to partition the sub-intervals of the support set into non-uniform, equal-probability intervals. This strategy maximizes the proximity of data points to support points while minimizing their occurrences in interval centers, thus mitigating categorical errors. The Cumulative Distribution Function (CDF) of the normal distribution and its inverse function can be utilized to solve for breakpoints that divide the restricted interval $[-B, B]$ into $(S-1)$ sub-intervals with equal probability. By arranging these breakpoints in sequence along with the two boundary points $\pm B$, and calculating mean value of each pair of adjacent points, the support set vector $\mathbf{Sp}_1 \in \mathbb{R}^{S\times 1}$ can be obtained. At each time step, the support set remains identical.

In the case of a single branch with the output denoted as $\mathbf{X}_{f,1}^i$, the final prediction $\mathbf{X}_{sp,1}^i$ is obtained by computing the expectation of the output probability distribution over this predefined support set:

$$\mathbf{X}_{sp,1}^i = \langle \mathrm{Softmax}\big(\mathbf{X}_{f,1}^i\big), \mathbf{Sp}_1 \rangle, \tag{3}$$

where $\mathbf{X}_{sp,1}^i \in \mathbb{R}^{T\times 1}$ and $\langle\rangle$ denotes the vector dot product.

3.3 Interleaved Dual Branches

Compared with the probability distribution on a single set, the interleaved design with dual branches is more conducive in improving the robustness of the model. When severe outliers occur in the predictions of one branch, the other branch can be relied upon for correction whose outputs also become additional self-supervised signals. Moreover, when the ground truth at a certain time step falls in the middle of two support points of one branch, quantization error may occur, which is known as boundary effect (Sun et al. 2025). Introducing another branch with interleaved support set thus

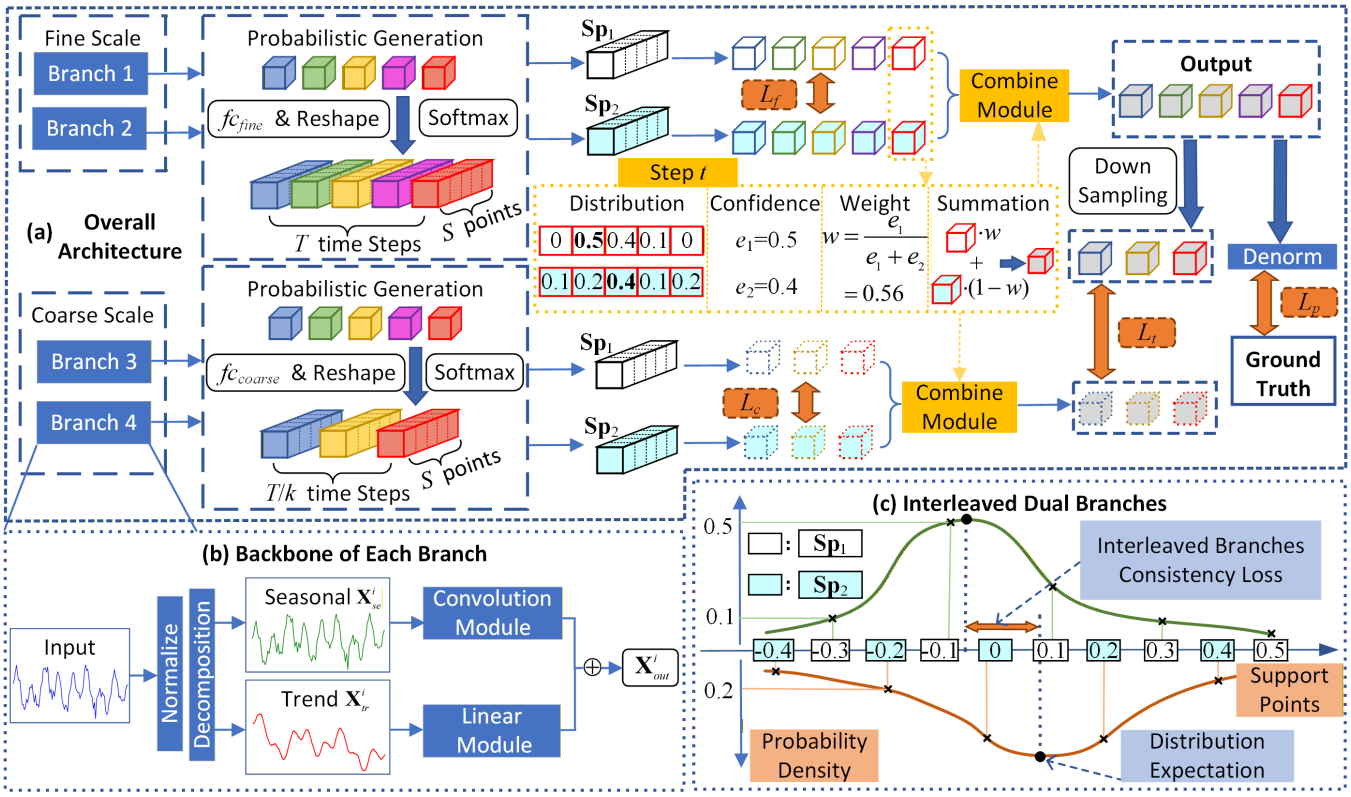

Figure 2: (a) Overall model architecture for direct per-step probability distribution forecasting ; (b) Backbone structure of each branch; (c) Dual-branch design corresponding to interleaved support sets.

mitigates the misclassification near boundaries of adjacent bins. To create another set with interleaved support points, the mean value of adjacent points in $\mathbf{Sp}_1$ is calculated, and a predefined boundary point is simultaneously supplemented to form another support set vector $\mathbf{Sp}_2$.

On the normal time scale, $\mathbf{X}^i_{out,1}$ and $\mathbf{X}^i_{out,2}$ are computed using two branches with independent parameters. Then $\mathbf{X}^i_{f,1}$ and $\mathbf{X}^i_{f,2}$ are derived by mapping through two fully connected layers ($fc_{fine,1}$ and $fc_{fine,2}$) that also have independent parameters. Probability distributions on the second support set are generated through softmax in the last dimension of $\mathbf{X}^i_{f,2}$, and the expectation $\mathbf{X}^i_{sp,2}$ can be calculated as Equation 3.

Unlike existing approaches that aggregate multi-branch outputs through summation (Zhang et al. 2022), we come up with a method based on maximum prediction confidence. First, it is necessary to calculate the maximum probability value $\mathbf{e}_1, \mathbf{e}_2 \in \mathbb{R}^{T\times 1}$ across all support points in the prediction for each time step:

$$\mathbf{e}_1 = \max_{1\le s\le S}\left(\mathrm{Softmax}\left(\mathbf{X}^i_{f,1}\right)\right)_{t,s}, \quad t = 1, 2, \cdots\cdots, T, \tag{4}$$

$$\mathbf{e}_2 = \max_{1\le s\le S}\left(\mathrm{Softmax}\left(\mathbf{X}^i_{f,2}\right)\right)_{t,s}, \quad t = 1, 2, \cdots\cdots, T. \tag{5}$$

Second, the weight coefficient $\mathbf{w}$ is calculated based on the maximum confidence of the two branches:

$$\mathbf{w} = \frac{\mathbf{e}_1}{\mathbf{e}_1 + \mathbf{e}_2}. \tag{6}$$

Third, a weighted sum is performed on the resulting expectations of the two interleaved branches:

$$\mathbf{X}^i_{sp,f} = \mathbf{w} \odot \mathbf{X}^i_{sp,1} + (1 - \mathbf{w}) \odot \mathbf{X}^i_{sp,2}, \tag{7}$$

where $\odot$ denotes the Hadamard product. By applying the RevIN denormalization method to $\mathbf{X}^i_{sp,f} \in \mathbb{R}^{T\times 1}$ and reassembling the separated channels, the final output $\hat{\mathbf{X}}_T \in \mathbb{R}^{T\times C}$ of the model can be obtained.

3.4 Bi-scale Temporal Branches

To further constrain the fusion output on the normal (fine) time scale, we replicate the dual-branch design mentioned in Section 3.2 at the coarse scale to generate auxiliary self-supervised sequences in lower resolution. In this way, the overall model contains a total of four backbone branches with the same architecture but non-shared parameters as shown in the left part of Figure 2(a): two backbone branches used at coarse time scale while another two backbone branches used at fine time scale.

We adjust the projection layer of the probability generation module mentioned in Section 3.2, thus altering the output dimension of this module:

$$\mathbf{X}^i_c = \mathrm{Reshape}\left(fc_{coarse}\left(\mathbf{X}^i_{out}\right)\right). \tag{8}$$

The output dimension of the fully connected layer fc_{coarse} is no longer $T \times S$ but $\frac{T}{k} \times S$ instead, thus generating $\mathbf{X}_c^i \in \mathbb{R}^{\frac{T}{k} \times S}$. k is a positive integer downsampling factor greater than 1. T is divisible by k, otherwise padding must be applied to the dimension of sequence length. All four fully connected layers ($fc_{coarse,1}$, $fc_{coarse,2}$, $fc_{fine,1}$, and $fc_{fine,2}$) employed in the probability generation module do not share any parameters.

The interleaved support sets are the same as $\mathbf{Sp}_1$ and $\mathbf{Sp}_2$, so the subsequent expectation calculation process is identical to that in Equation 3. The expected value calculated at the coarse scale is denoted as $\mathbf{X}_{sp,3}^i, \mathbf{X}_{sp,4}^i \in \mathbb{R}^{\frac{T}{k} \times 1}$. The calculation process of the weight coefficient $\mathbf{w}$ for fusing the forecast results of the two branches is also the same as that in Equations 4 to 6. Final result of the coarse time-scale branch $\mathbf{X}_{sp,c}^i \in \mathbb{R}^{\frac{T}{k} \times 1}$ is generated by the following formula:

$$\mathbf{X}_{sp,c}^i = \mathbf{w} \odot \mathbf{X}_{sp,3}^i + (1 - \mathbf{w}) \odot \mathbf{X}_{sp,4}^i. \tag{9}$$

It should be noted that $\mathbf{X}_{sp,c}^i$ serves solely as a self-supervised signal that constrains the output of the fine-scale branches, and it is not combined with $\hat{\mathbf{X}}_T$.

3.5 Loss Function

Previous TSF models often employ the Mean Square Error (MSE) loss for backpropagation. However, when the prediction length is long, using only the MSE loss as the objective function makes it difficult for the model to capture the continuity and pattern correlation between consecutive time steps. In severe cases, this can lead to direction errors or pattern distortion (Wang et al. 2024a; Kudrat et al. 2025).

InterPDN refers to the loss function adopted by the xPatch prediction model (Stitsyuk and Choi 2025). xPatch assigns different weights to errors at various time steps. In particular, the weights of prediction errors on steps closer to the lookback window are higher, while the weights of errors at more distant time steps decay. The loss between the predictions $\hat{\mathbf{X}}_T$ and the ground truth $\mathbf{X}_T$ is written as L_p, which can be calculated by:

$$L_p\left(\mathbf{X}_T, \hat{\mathbf{X}}_T\right) = \frac{1}{T \cdot C} \sum_{i=1}^{T} \theta(i) \left\|\mathbf{x}_{t+i} - \hat{\mathbf{x}}_{t+i}\right\|_1, \tag{10}$$

where $\theta(i)$ is an error decay function based on the arctangent function.

Since the proposed multi-branch model possesses four times the parameters of the original backbone while the dataset remains unchanged, it is prone to issues such as overfitting and degradation in generalization performance. Hence, similar to self-supervised learning, outputs from another branch can serve as additional supervisory signals to impose consistency constraints on the output of the present branch. Moreover, if results of interleaved set branches are not constrained, it is highly likely that one branch would exhibit high-confidence prediction while the other would perform severe quantization errors in distribution forecasting. Consequently, it can be difficult to achieve the effects of output complementarity or resolve misclassification at bin boundary as mentioned in Section 3.3.

We employ MSE to impose inter-scale consistency constraint loss. We denote by L_f the consistency loss between outputs $\mathbf{X}_{sp,1}$ and $\mathbf{X}_{sp,2}$ of the fine-scale dual branches, while by L_c the consistency loss between outputs $\mathbf{X}_{sp,3}$ and $\mathbf{X}_{sp,4}$ of the coarse-scale dual branches, respectively:

$$L_f\left(\mathbf{X}_{sp,1}, \mathbf{X}_{sp,2}\right) = \frac{1}{T \cdot C} \sum_{i=1}^{C} \left\|\mathbf{X}_{sp,1}^i - \mathbf{X}_{sp,2}^i\right\|_2^2, \tag{11}$$

$$L_c\left(\mathbf{X}_{sp,3}, \mathbf{X}_{sp,4}\right) = \frac{k}{T \cdot C} \sum_{i=1}^{C} \left\|\mathbf{X}_{sp,3}^i - \mathbf{X}_{sp,4}^i\right\|_2^2. \tag{12}$$

Multi-scale prediction branches are at risk of learning contradictory patterns from different time resolutions (Zhang et al. 2025). To enable the coarse-grained branch to guide the fine-scale branch in terms of long-term evolutionary trends, it is necessary to introduce an inter-scale consistency loss function. Due to the discrepancy in the number of time steps between $\mathbf{X}_{sp,f}^i$ and $\mathbf{X}_{sp,c}^i$, it is necessary to downsample $\mathbf{X}_{sp,f}^i$ along the time dimension to form $\mathbf{X}_{d,f}^i$:

$$\mathbf{X}_{d,f}^i = \mathrm{AvgPool}\big(\mathbf{X}_{sp,f}^i, \mathrm{kernel} = k\big), \tag{13}$$

where both kernel size and stride length in the average pooling operation are set to k. We compute the cross-scale consistency loss L_t between $\mathbf{X}_{d,f}^i$ and $\mathbf{X}_{sp,c}^i$ still using MSE:

$$L_t\left(\mathbf{X}_{sp,c}, \mathbf{X}_{d,f}\right) = \frac{k}{T \cdot C} \sum_{i=1}^{C} \left\|\mathbf{X}_{sp,c}^i - \mathbf{X}_{d,f}^i\right\|_2^2. \tag{14}$$

The total loss L_{total} involved in backpropagation consists of the following four components: the loss between the predicted value and the true value, two consistency losses of interleaved set branches, and the cross-scale consistency loss. The calculation formula of L_{total} is as follows:

$$L_{total} = L_p + \alpha \cdot L_f + \beta \cdot L_c + \gamma \cdot L_t, \tag{15}$$

where $\alpha, \beta, \gamma \geq 0$ are the weight coefficients used to balance various kinds of losses.

4 Experiments

4.1 Datasets and Baselines

We conduct extensive experiments on nine real-world datasets spanning electricity transformer monitoring (ETTh1, ETTh2, ETTm1, ETTm2) (Zhou et al. 2021), Electricity, Weather, Traffic, Exchange-rate, and Illness. Detailed information regarding these datasets is provided in Table 3 of Supplementary. On the Illness dataset, experiments are conducted with forecast horizons of 24, 36, 48, and 60. For the other eight datasets, the output sequence lengths are set to 96, 192, 336, and 720. [1] Performance on the test set is evaluated using the metrics MSE and Mean Absolute Error (MAE).

Nine SOTA TSF models serve as baselines, namely xPatch (Stitsyuk and Choi 2025), RAFT (Han et al. 2025), AMD (Hu et al. 2025), MOMENT (Goswami et al. 2024),

[1] Detailed dataset information is provided in *Suppl. A*.

Models	**interPDN (Ours)**		xPatch (2025)		RAFT (2025)		AMD (2025)		MOMENT (2024)		TimeMixer (2024)		iTransformer (2024)		TimesNet (2023)		PatchTST (2023)		DLinear (2023)	
Metric	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	**0.378**	**0.406**	0.395	0.412	0.420	0.436	0.407	0.424	0.418	0.436	0.411	0.423	0.454	0.448	0.458	0.450	0.413	0.431	0.427	0.438
ETTh2	**0.293**	**0.350**	0.301	0.354	0.359	0.409	0.351	0.392	0.352	0.395	0.316	0.384	0.383	0.407	0.414	0.427	0.330	0.379	0.431	0.447
ETTm1	**0.337**	**0.365**	0.344	0.371	0.348	0.378	0.347	0.376	0.344	0.379	0.348	0.376	0.407	0.410	0.400	0.406	0.351	0.381	0.359	0.381
ETTm2	**0.239**	**0.298**	0.243	0.302	0.254	0.320	0.254	0.316	0.259	0.318	0.256	0.316	0.288	0.332	0.291	0.333	0.255	0.315	0.277	0.339
Weather	**0.212**	**0.243**	0.216	0.248	0.241	0.286	0.223	0.263	0.228	0.270	0.222	0.262	0.258	0.278	0.259	0.287	0.226	0.264	0.246	0.300
Traffic	0.391	**0.244**	0.393	0.249	0.401	0.282	0.394	0.272	0.415	0.293	**0.388**	0.263	0.428	0.282	0.620	0.336	0.391	0.264	0.434	0.295
Electricity	**0.149**	**0.241**	0.154	0.245	0.160	0.259	0.159	0.254	0.165	0.260	0.156	0.247	0.178	0.270	0.193	0.295	0.159	0.253	0.166	0.264
Exchange	0.370	0.405	0.373	0.408	0.441	0.441	**0.328**	**0.387**	0.363	0.406	0.471	0.452	0.458	0.469	0.416	0.443	0.405	0.426	0.369	0.418
Illness	**1.393**	**0.708**	1.437	0.721	2.098	0.978	1.597	0.731	2.752	1.108	1.971	0.924	2.947	1.193	2.139	0.931	1.443	0.798	2.169	1.041

Table 1: Performance metrics (MSE and MAE) for multivariate TSF. The best results are highlighted in bold and the second best are underlined. The standard deviation of the metrics across all prediction tasks was controlled within 0.003 with each experiment repeated three times.

Models		**interPDN**		4BSP		BSPDP		IBBPDP		SBPDP		SBSP	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	96	**0.338**	**0.374**	0.349	0.380	0.341	0.376	0.339	0.376	0.344	0.379	0.355	0.382
	192	**0.362**	**0.391**	0.372	0.395	0.365	0.392	0.365	0.392	0.369	0.395	0.381	0.396
	336	**0.378**	**0.406**	0.394	0.410	0.381	0.408	0.382	0.409	0.383	0.410	0.398	0.413
	720	**0.432**	**0.451**	0.449	0.453	0.436	0.453	0.439	0.456	0.445	0.459	0.446	0.456
	avg	**0.378**	**0.406**	0.392	0.411	0.381	0.407	0.381	0.408	0.385	0.411	0.395	0.412
ETTh2	96	**0.223**	**0.297**	0.227	0.301	0.225	0.299	0.224	**0.297**	0.226	0.300	0.229	0.299
	192	**0.267**	**0.328**	0.272	0.333	0.268	0.329	0.268	0.330	0.272	0.333	0.276	0.334
	336	**0.305**	**0.360**	0.310	0.364	0.306	0.361	0.306	**0.360**	0.308	0.363	0.315	0.365
	720	**0.377**	**0.416**	0.380	0.417	0.378	0.417	0.378	0.417	0.383	0.421	0.382	0.418
	avg	**0.293**	**0.350**	0.298	0.355	0.294	0.352	0.294	0.351	0.297	0.354	0.301	0.354
ETTm1	96	**0.277**	**0.331**	0.283	0.337	0.278	0.332	**0.277**	**0.331**	0.288	0.336	0.280	0.333
	192	**0.310**	**0.352**	0.320	0.359	0.313	0.353	0.313	0.353	0.328	0.359	0.318	0.356
	336	**0.348**	**0.372**	0.356	0.381	0.349	0.373	0.350	0.375	0.352	0.378	0.357	0.380
	720	**0.411**	**0.404**	0.424	0.414	**0.411**	**0.404**	0.416	0.407	0.420	0.410	0.422	0.414
	avg	**0.337**	**0.365**	0.347	0.374	0.338	0.366	0.339	0.367	0.347	0.371	0.344	0.371
ETTm2	96	**0.149**	**0.237**	0.151	0.240	0.151	0.239	**0.149**	**0.237**	0.165	0.249	0.151	0.240
	192	**0.208**	**0.277**	0.210	0.280	0.209	0.279	0.209	0.278	0.222	0.288	0.211	0.281
	336	**0.263**	**0.314**	0.264	0.317	**0.263**	0.315	**0.263**	0.315	0.264	0.317	0.267	0.318
	720	**0.336**	**0.362**	0.337	0.366	0.339	0.363	0.339	0.363	0.338	0.365	0.343	0.368
	avg	**0.239**	**0.298**	0.242	0.302	0.241	0.299	0.240	0.298	0.247	0.305	0.243	0.302

Table 2: Ablation results. We respectively ablate the bi-scale branches as well as the interleaved set branches, and report their metrics. We also compare the performance difference between direct scalar estimation head and probability distribution output head under the single-branch configuration. Moreover, we compare interPDN with the network that merely stacks four branches.

TimeMixer (Wang et al. 2024b), iTransformer (Liu et al. 2023a), TimesNet (Wu et al. 2022), PatchTST (Nie et al. 2023), and DLinear (Zeng et al. 2023). Among them, AMD, RAFT and TimeMixer all embody the concept of multi-scale and multi-branch fusion. The backbone of xPatch exhibits a similar architecture to the backbone of each branch in interPDN. To validate whether the lightweight interPDN is capable of outperforming TSF foundation models, we select the SOTA model MOMENT as our baseline.

4.2 Main Results

Our model is implemented within Python version 3.12 PyTorch version 2.3.0. Due to their larger number of channels, the Electricity, Weather and Traffic datasets are evaluated on a system with an Intel Xeon Gold 6348 (14 vCPUs) and an NVIDIA A800 GPU (80GB). Experiments for the remaining six datasets are conducted on a server equipped with an Intel Xeon Gold 6430 (16 vCPUs) and an NVIDIA RTX 4090 GPU (24GB). The operating system is Ubuntu 22.04.

Following the parameter search, Table 1 presents the evaluation metrics for multivariate TSF achieved by interPDN and the other nine baseline models. In terms of average MSE and MAE across four forecast horizons, our model achieves SOTA performance on seven of the nine datasets.[2] With respect to performance improvement margins, interPDN demonstrates a 1.51% lower MAE and a 2.44% lower MSE compared to xPatch. Compared to Transformer-based iTransformer and PatchTST, interPDN achieves a reduction of 35.15% and 5.31% in MSE, along with a reduction of 20.27% and 7.15% in MAE metric, respectively. Com-

[2] *Suppl. C* presents a detailed metric comparison between interPDN and other baseline models across each forecasting horizon.

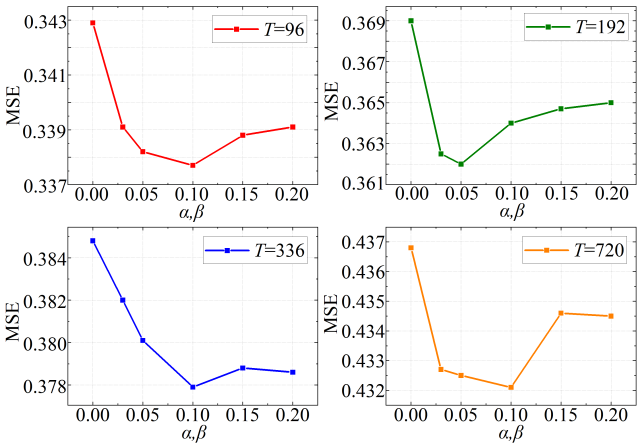

Figure 3: MSE versus α and β at four forecast horizons on ETTh1 dataset

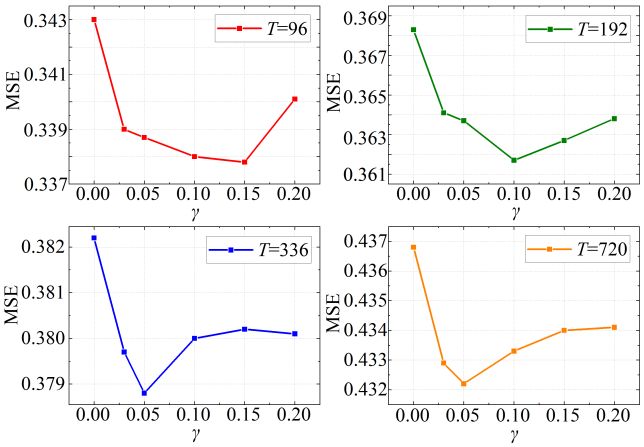

Figure 4: MSE versus γ at four forecast horizons on ETTh1

pared with RAFT, AMD, and MOMENT which are all latest SOTA baseline models, the average reduction rates in MAE are 13.96%, 4.54%, and 15.65%, respectively.

4.3 Ablation Study

We conducted ablation studies across four ETT datasets, respectively evaluating the effectiveness of: Single-Branch Scalar Prediction (SBSP), Single-Branch Probabilistic Distribution Prediction (SBPDP), Interleaved Bi-Branch Probabilistic Distribution Prediction (IBBPDP), Bi-Scale Probabilistic Distribution Prediction(BSPDP), and 4-Branch Scalar Prediction aggregated by averaging (4BSP). The full ablation results are shown in Table 2.

SBSP and SBPDP are both conducted using a single-branch backbone model, adding the scalar estimation head and the per-step discrete distribution output head separately. For 65% of the prediction tasks, employing the probabilistic distribution output head and computing the expectation outperforms SBSP which directly outputs scalar. The standalone probabilistic prediction head offers limited improvements to model performance while, in certain cases, turns to degrade the model predictions.

If the interleaved support sets are employed, IBBP model consistently outperforms both SBSP and SBPDP across all tasks in MSE and MAE. Based on same support set, when augmenting the single-branch probabilistic prediction head with a coarse temporal scale branch as a constraint, this modification also enables BSPDP to yield significant performance improvements. The proposed interPDN combining the interleaved support set design and the bi-scale scheme, universally outperforms the above-mentioned two network architectures on all prediction tasks. Furthermore, to demonstrate that the success of interPDN is not merely due to an increase in parameters, we naively integrate the scalar forecasting results from the four-branch architecture. In fact, on 45% of the forecasting tasks, 4BSP underperforms even SBSP and significantly underperforms interPDN across all tasks. In summary, each branch as well as the probabilistic prediction head within interPDN is essential.

It is also necessary to verify the effectiveness of constraining the outputs of the interleaved branches using loss functions under the same time scale. Adjusting the weighting coefficients preceding L_f and L_c while fixing γ to 0.1, we depict the resulting variation in MSE as illustrated in Figure 3. Since α and β are functionally identical consistency losses, we set $\alpha = \beta$ during our hyperparameter search for simplicity. Across all four TSF tasks on the ETTh1 dataset, MSE is the highest when $\alpha = \beta = 0$. This indicates that without the constraint, the dual branches face difficulties in achieving complementary prediction and effective self-supervision.

Subsequently, while fixing the value of α and β, we adjust the weighting coefficient γ for loss L_t to investigate the impact of the inter-scale consistency loss on model performance. The results in Figure 4 exhibit a similar trend to those in Figure 3, with MSE peaking when $\gamma = 0$. It demonstrates that relying solely on fine-scale branch predictions hinders the model's ability to capture long-term temporal patterns.

5 Conclusion

In summary, the proposed interPDN innovatively employs probabilistic distribution prediction and expectation calculation for TSF tasks, fully considering the uncertainty in backbone outputs. Dual-branch architecture with interleaved support sets is adopted to mitigate quantization errors at adjacent bin boundaries. A coarse time-scale prediction branch is introduced to enable the model to capture long-range dependencies. Building on self-supervised principles, two kinds of consistency losses are applied across branches, effectively alleviating single-branch prediction anomalies. Extensive experiments on multiple real-world benchmarks validate our model's performance, while ablation studies demonstrate the efficacy of probabilistic prediction head, interleaved bi-branch architecture, and inter-scale constraints. This framework advances TSF by offering a scalable and reliable solution without prior distributional assumptions, thus setting a new direction for TSF in real-world applications ranging from energy to healthcare domains.

Acknowledgments

We thank the anonymous reviewers for their insightful and constructive comments. This work was supported by the National Key R&D Program of China (No. 2025YFC3811300), the National Natural Science Foundation of China (62376070, 62076195), and the Fundamental Research Funds for the Central Universities (AUGA5710011522).

References

Benechehab, A.; Feofanov, V.; Paolo, G.; Thomas, A.; Filippone, M.; and Kégl, B. 2025. AdaPTS: Adapting Univariate Foundation Models to Probabilistic Multivariate Time Series Forecasting. *arXiv preprint arXiv:2502.10235*.

Box, G. 2013. Box and Jenkins: time series analysis, forecasting and control. In *A Very British Affair: Six Britons and the Development of Time Series Analysis During the 20th Century*, 161–215. Springer.

Chen, M.; Shen, L.; Li, Z.; Wang, X. J.; Sun, J.; and Liu, C. 2024a. Visionts: Visual masked autoencoders are free-lunch zero-shot time series forecasters. *arXiv preprint arXiv:2408.17253*.

Chen, P.; Zhang, Y.; Cheng, Y.; Shu, Y.; Wang, Y.; Wen, Q.; Yang, B.; and Guo, C. 2024b. Pathformer: Multi-scale transformers with adaptive pathways for time series forecasting. *arXiv preprint arXiv:2402.05956*.

Cho, Y.; and Lee, J.-Y. 2025. CoMRes: Semi-Supervised Time Series Forecasting Utilizing Consensus Promotion of Multi-Resolution. In *The Thirteenth International Conference on Learning Representations*.

Ekambaram, V.; Jati, A.; Dayama, P.; Mukherjee, S.; Nguyen, N.; Gifford, W. M.; Reddy, C.; and Kalagnanam, J. 2024. Tiny time mixers (ttms): Fast pre-trained models for enhanced zero/few-shot forecasting of multivariate time series. *Advances in Neural Information Processing Systems*, 37: 74147–74181.

Goswami, M.; Szafer, K.; Choudhry, A.; Cai, Y.; Li, S.; and Dubrawski, A. 2024. Moment: A family of open time-series foundation models. *arXiv preprint arXiv:2402.03885*.

Han, S.; Lee, S.; Cha, M.; Arik, S. O.; and Yoon, J. 2025. Retrieval augmented time series forecasting. *arXiv preprint arXiv:2505.04163*.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

Hu, Y.; Liu, P.; Zhu, P.; Cheng, D.; and Dai, T. 2025. Adaptive multi-scale decomposition framework for time series forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, 17359–17367.

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

Jin, M.; Wang, S.; Ma, L.; Chu, Z.; Zhang, J. Y.; Shi, X.; Chen, P.-Y.; Liang, Y.; Li, Y.-F.; Pan, S.; et al. 2023. Time-llm: Time series forecasting by reprogramming large language models. *arXiv preprint arXiv:2310.01728*.

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

Kudrat, D.; Xie, Z.; Sun, Y.; Jia, T.; and Hu, Q. 2025. Patch-wise Structural Loss for Time Series Forecasting. *arXiv preprint arXiv:2503.00877*.

Lin, H.; Ma, Z.; Ji, R.; Wang, Y.; Su, Z.; Hong, X.; and Meng, D. 2025. Semi-supervised counting via pixel-by-pixel density distribution modelling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

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

Liu, S.; Yu, H.; Liao, C.; Li, J.; Lin, W.; Liu, A. X.; and Dustdar, S. 2022. Pyraformer: Low-Complexity Pyramidal Attention for Long-Range Time Series Modeling and Forecasting. In *International Conference on Learning Representations*.

Liu, Y.; Hu, T.; Zhang, H.; Wu, H.; Wang, S.; Ma, L.; and Long, M. 2023a. itransformer: Inverted transformers are effective for time series forecasting. *arXiv preprint arXiv:2310.06625*.

Liu, Y.; Li, C.; Wang, J.; and Long, M. 2023b. Koopa: Learning non-stationary time series dynamics with koopman predictors. *Advances in neural information processing systems*, 36: 12271–12290.

Lu, J.; Sun, Y.; and Yang, S. 2024. In-context time series predictor. *arXiv preprint arXiv:2405.14982*.

Lu, J.; and Yang, S. 2025. Linear Transformers as VAR Models: Aligning Autoregressive Attention Mechanisms with Autoregressive Forecasting. *arXiv preprint arXiv:2502.07244*.

Ma, X.; Hong, X.; Lu, S.; and Li, W. 2024. TS3Net: Triple decomposition with spectrum gradient for long-term time series analysis. In *2024 IEEE 40th International Conference on Data Engineering (ICDE)*, 887–900. IEEE.

Nie, Y.; H. Nguyen, N.; Sinthong, P.; and Kalagnanam, J. 2023. A Time Series is Worth 64 Words: Long-term Forecasting with Transformers. In *International Conference on Learning Representations*.

Niu, W.; Xie, Z.; Sun, Y.; He, W.; Xu, M.; and Hao, C. 2025. LangTime: A Language-Guided Unified Model for Time Series Forecasting with Proximal Policy Optimization. *arXiv preprint arXiv:2503.08271*.

Rangapuram, S. S.; Seeger, M. W.; Gasthaus, J.; Stella, L.; Wang, Y.; and Januschowski, T. 2018. Deep state space models for time series forecasting. *Advances in neural information processing systems*, 31.

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

Sims, C. A. 1980. Macroeconomics and reality. *Econometrica: journal of the Econometric Society*, 1–48.

Stitsyuk, A.; and Choi, J. 2025. xPatch: Dual-Stream Time Series Forecasting with Exponential Seasonal-Trend Decomposition. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, 20601–20609.

Sun, Y.; Xie, Z.; Chen, D.; Eldele, E.; and Hu, Q. 2025. Hierarchical classification auxiliary network for time series forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, 20743–20751.

Tang, P.; and Zhang, W. 2025. Unlocking the Power of Patch: Patch-Based MLP for Long-Term Time Series Forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, 12640–12648.

Wang, H.; Pan, L.; Chen, Z.; Yang, D.; Zhang, S.; Yang, Y.; Liu, X.; Li, H.; and Tao, D. 2024a. Fredf: Learning to forecast in the frequency domain. *arXiv preprint arXiv:2402.02399*.

Wang, S.; Wu, H.; Shi, X.; Hu, T.; Luo, H.; Ma, L.; Zhang, J. Y.; and Zhou, J. 2024b. Timemixer: Decomposable multiscale mixing for time series forecasting. *arXiv preprint arXiv:2405.14616*.

Wang, Y.; Wu, H.; Dong, J.; Liu, Y.; Long, M.; and Wang, J. 2024c. Deep time series models: A comprehensive survey and benchmark. *arXiv preprint arXiv:2407.13278*.

Wu, H.; Hu, T.; Liu, Y.; Zhou, H.; Wang, J.; and Long, M. 2022. Timesnet: Temporal 2d-variation modeling for general time series analysis. *arXiv preprint arXiv:2210.02186*.

Wu, H.; Xu, J.; Wang, J.; and Long, M. 2021. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. *Advances in neural information processing systems*, 34: 22419–22430.

Xue, W.; Zhou, T.; Wen, Q.; Gao, J.; Ding, B.; and Jin, R. 2023. Card: Channel aligned robust blend transformer for time series forecasting. *arXiv preprint arXiv:2305.12095*.

Yang, Y.; Zhu, Q.; and Chen, J. 2024. Vcformer: Variable correlation transformer with inherent lagged correlation for multivariate time series forecasting. *arXiv preprint arXiv:2405.11470*.

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

Zhang, X.; Chowdhury, R. R.; Gupta, R. K.; and Shang, J. 2024. Large language models for time series: A survey. *arXiv preprint arXiv:2402.01801*.

Zhang, X.; Jin, X.; Gopalswamy, K.; Gupta, G.; Park, Y.; Shi, X.; Wang, H.; Maddix, D. C.; and Wang, Y. 2022. First de-trend then attend: Rethinking attention for time-series forecasting. *arXiv preprint arXiv:2212.08151*.

Zhang, Z.; Pham, T. D.; An, Y.; Doan, N. P.; Alsharari, M.; Tran, V.-H.; Hoang, A.-T.; Vandierendonck, H.; and Mai, S. T. 2025. WaveletMixer: a multi-resolution wavelets based MLP-mixer for multivariate long-term time series forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, 22741–22749.

Zhao, L.; and Shen, Y. 2024. Rethinking channel dependence for multivariate time series forecasting: Learning from leading indicators. *arXiv preprint arXiv:2401.17548*.

Zhong, S.; Ruan, W.; Jin, M.; Li, H.; Wen, Q.; and Liang, Y. 2025. Time-vlm: Exploring multimodal vision-language models for augmented time series forecasting. *arXiv preprint arXiv:2502.04395*.

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