DeepBooTS: Dual-Stream Residual Boosting for Drift-Resilient Time-Series Forecasting

Daojun Liang [1,2], **Jing Chen** [1,2], **Xiao Wang** [3, 4 *], **Yinglong Wang** [1, 2 *], **Shuo Li** [5, 6]

[1] Key Laboratory of Computing Power Network and Information Security, Ministry of Education, Shandong Computer Science Center (National Supercomputing Center in Jinan), Qilu University of Technology (Shandong Academy of Sciences), Jinan, 250103, China

[2] Shandong Provincial Key Laboratory of Computing Power Internet and Service Computing, Shandong Fundamental Research Center for Computer Science, Jinan, 250103, China

[3] School of Intelligent Manufacturing and Control Engineering, Qilu Institute of Technology, Jinan, 250200, China

[4] Shandong Provincial Key Laboratory of Industrial Big Data and Intelligent Manufacturing, Qilu Institute of Technology, Jinan, 250200, China

[5] Department of Computer and Data Sciences, Case Western Reserve University, Cleveland, OH 44106, USA

[6] Department of Biomedical Engineering, Case Western Reserve University, Cleveland, OH 44106, USA

liangdj@sdas.org, chenj@sdas.org, wangxiao@qlit.edu.cn, wangyinglong@qlu.edu.cn, shuo.li11@case.edu

Abstract

Time-Series (TS) exhibits pronounced non-stationarity. Consequently, most forecasting methods display compromised robustness to concept drift, despite the prevalent application of instance normalization. We tackle this challenge by first analysing concept drift through a bias-variance lens and proving that weighted ensemble reduces variance without increasing bias. These insights motivate DeepBooTS, a novel end-to-end dual-stream residual-decreasing boosting method that progressively reconstructs the intrinsic signal. In our design, each block of a deep model becomes an ensemble of learners with an auxiliary output branch forming a highway to the final prediction. The block-wise outputs correct the residuals of previous blocks, leading to a learning-driven decomposition of both inputs and targets. This method enhances versatility and interpretability while substantially improving robustness to concept drift. Extensive experiments, including those on large-scale datasets, show that the proposed method outperforms existing methods by a large margin, yielding an average performance improvement of 15.8% across various datasets, establishing a new benchmark for TS forecasting.

Code — https://github.com/Anoise/DeepBooTS

Extended version — https://arxiv.org/abs/2511.06893

1 Introduction

Time Series (TS) from natural and engineered systems often evolve under transient conditions (Anderson 1976) and therefore violate stationarity assumptions (Hyndman and Athanasopoulos 2018). Classical statistical models such as ARIMA (Piccolo 1990) and exponential smoothing (Gardner Jr 1985), which depend on fixed statistical properties, struggle to track these shifts (De Gooijer and Hyndman 2006). Deep neural networks have become attractive alternatives because of their powerful nonlinear modeling capabilities (Hornik 1991), yet even the latest attention-based (Zhou et al. 2021; Nie et al. 2022; Liu et al. 2023; Liang et al. 2024b; Liang 2025; Ye et al. 2024; Wang et al. 2024a,b; Liang 2025; Yu et al. 2025a,b) and graph-neural approaches (Wu et al. 2019; Yi et al. 2024b) provide only marginal gains over simple multilayer perceptrons and incur substantial inference overhead (Shao et al. 2022; Zeng et al. 2023; Yi et al. 2024a).

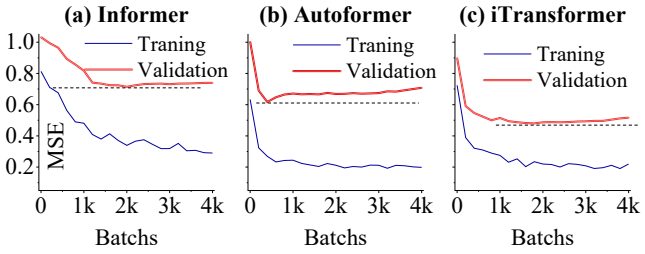
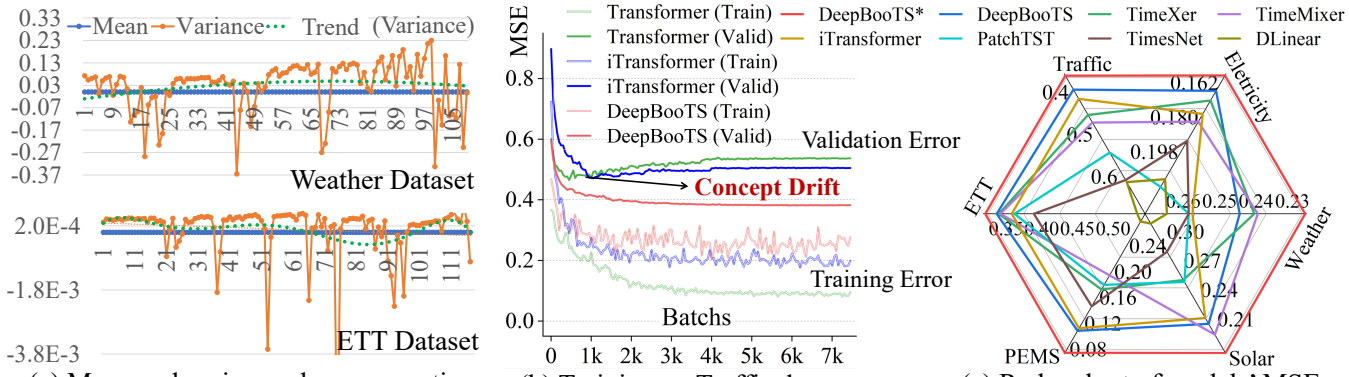


Figure 1: TS forecasting models suffer from concept drift problems, more experiments are provided in Appendix H.

A deeper issue is that leading architectures like Transformers (Vaswani et al. 2017) and their variants are highly susceptible to concept drift—a mismatch between training and testing distributions. Empirical evidence shows that validation error rises early in training even as training error falls, as shown in Fig. 1, a sign that the learned model fails to generalize as the underlying data distribution shifts. Prevailing solutions, including reversible instance normalization (RevIN) (Kim et al. 2021) and temporal re-aggregation, e.g., DLinear (Zeng et al. 2023) and iTransformer (Liu et al. 2023), alleviate mean shifts but leave the variance unstable, which leads to large fluctuations and persistent concept drift. This phenomenon occurs across multiple TS fore-

*Corresponding author.



(a) Mean and variance change over time. (b) Training on Traffic datasets. (c) Radar chart of models' MSE.

Figure 2: (a) Concept drift exists in current TS datasets. (b) Concept drift causes the model training error to decrease while the validation error increases. (c) The performance of DeepBooTS after reducing concept drift.

casting models and datasets, as shown in Fig. 2(a), underscoring the need for an approach that directly targets the high-variance component.

This work addresses the problem by framing concept drift through the bias-variance decomposition. We theoretically show that when the mean bias and noise are fixed, the degree of drift is controlled by the variance of the model predictions. We then prove that ensemble averaging reduces the prediction variance while preserving bias and derive new bounds showing that weighted ensembles achieve a strictly lower error under distribution shift. These results suggest that a properly designed ensemble can substantially mitigate concept drift. Building on this insight, we introduce DeepBooTS, a novel dual-stream residual-decreasing boosting architecture that progressively reinstates the intrinsic values of a non-stationary TS. The key innovations are:

- We rigorously analyze concept drift through the lens of the bias-variance trade-off and prove that even simple deep ensembles can substantially reduce prediction variance without increasing bias.
- An efficient implementation of DeepBooTS is presented. Specifically, the outputs of subsequent blocks subtract the predictions of previous blocks, causing the network to explicitly model and reduce residual errors layer by layer. This residual-learning mechanism is analogous to gradient boosting, but implemented within a deep network, enhancing robustness to distributional shifts.
- A dual-stream decomposition is designed that decomposes both the input and labels, enabling the model to learn complementary representations while enhancing model versatility and interpretability.

We validate DeepBooTS on a wide range of TS benchmarks, including large-scale TS datasets. As shown in Fig. 2(c), our method consistently outperforms state-of-the-art baselines, achieving an average 15.8% improvement. These results, together with our theoretical analysis, demonstrate that DeepBooTS not only advances the accuracy of TS forecasting but also provides a principled solution to the long-standing problem of concept drift.

2 Methodology

2.1 Deep Ensemble helps Alleviate Concept Drift

As analyzed above, RevIN-processed (Z-Score) TS data maintains mean stationarity while exhibiting substantial variance instability. We consider a forecasting task where the data distribution shifts from an initial distribution P_0 to a new distribution P_t at time t. Formally, let $Y = f_0(X) + \varepsilon$ under P_0 (with true regression function f_0), and at time t let $Y = f_t(X) + \varepsilon$ under P_t, where f_t is the new underlying function after drift, $\varepsilon \sim \mathcal{N}(0, \sigma)$. For forecasting results $\hat{Y}$, the estimation error (MSE) of the model is

$$\underbrace{\mathrm{Var}(\hat{Y}) + (\mathrm{Bias}(\hat{Y}))^2 + \sigma^2}_{\text{Test Error}} = \underbrace{\mathbb{E}[(\hat{Y} - Y)^2] + 2\mathbb{E}(\varepsilon(\hat{Y} - \mathscr{Y}))}_{\text{Training Error}}. \tag{1}$$

The proof is given in Appendix B.1. Eq. 1, which is illustrated in Fig. 3(a), shows that when the mean $\mathrm{Bias}(\hat{Y})$ and noise level σ^2 are fixed, the variance of the data governs the extent of concept drift. Thus, models incapable of adapting to large data distribution shifts exhibit high prediction variance. Now, we show that ensemble reduces the variance. Without loss of generality, using MSE as the metric and applying the simple average ensemble method, we have

Theorem 1. *Let $\hat{Y}_1, \cdots, \hat{Y}_N$ be N i.i.d random variables drawn from some unknown distribution, and the ensemble method is $\bar{Y} = \frac{1}{N}\sum_t \hat{Y}^t$. Then,*

$$Bias(\bar{Y}) = Bias(\hat{Y}), \quad Var(\bar{Y}) \leq Var(\hat{Y}). \tag{2}$$

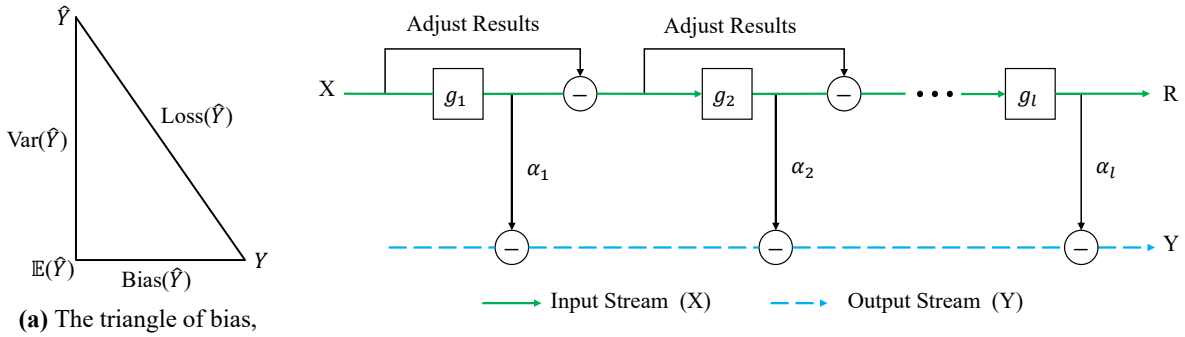
The proof is given in Appendix B.2. Although it adopts the simplest ensemble form, it proves that ensembling reduces variance while conserving bias. If substituting $\hat{f}_0(X)$ (which was trained to approximate f_0 on P_0) for $\hat{Y}$ in Eq. 1, we have

$$\mathrm{MSE}(f_t(X) - \hat{f}_0(X)) = \mathrm{Var}(\hat{f}_0(X)) + (\mathrm{Bias}(\hat{f}_0(X)))^2 + \sigma^2. \tag{3}$$

After the distribution shifts to P_t, the forecasting error is

$$\mathrm{MSE}_{P_t}(f_0(X)) = \mathbb{E}_{P_t}[\mathrm{Bias}_{P_0}(X)^2] + \mathbb{E}_{P_t}[\mathrm{Var}_{P_0}(f_0(X))] + \sigma^2. \tag{4}$$

Let $\mathrm{Var}_{P_t}[X] = \sigma_t^2$ with $c^2 = \sigma_t^2/\sigma^2$ to quantify the change in input variance, thus we have



(a) The triangle of bias, variance and loss (MSE).

(b) Deep ensemble is equivalent to meta-algorithmic Boosting.

Figure 3: (a) Relationship among model's bias, variance, and loss. (b) Deep boosting ensemble learning process.

Theorem 2. *Given a weighted ensemble models* $\hat{f}(X) = \frac{1}{L}\sum_{l=1}^{L}\alpha_l g_l(x)$, *where* $\sum_{l=1}^{L}\alpha_l = 1$, *we have*

$$MSE_{P_t}(\hat{f}_0) - MSE_{P_t}(\hat{f}) \approx \frac{(L-1)(1-\alpha_l)}{L}c^2\sigma_t^2 \geq 0. \quad (5)$$

The proof is given in Appendix B.3. It indicates that the ensemble's MSE under P_t is lower than that of the single one.

2.2 Deep Ensemble Model for TS Forecasting

Based on Theorem 2, we consider a ensemble learner set $\{g_l\}_{l=1}^{L}$ to minimize

$$\mathbb{E}(Y-\hat{f}(X))^2 = \mathbb{E}(Y-\sum_{l=1}^{L}\alpha_l g_l(X))^2, \quad (6)$$

$$s.t.\ \sum_{l=1}^{L}\alpha_l = 1, \text{and}\ \ \alpha_l \geq 0,\ \ l = 1,\cdots,L.$$

Here, the objective in Eq. 6 implies that

$$(Y-\hat{f}(X))^2 = \sum_{l=1}^{L}\alpha_l(Y-g_l(X))^2 - \sum_{l=1}^{L}\alpha_l(\hat{f}(X)-g_l(X))^2. \quad (7)$$

The proof is given in Appendix B.4. Eq. 7 proves that the ensemble generalization error can be reduced by increasing the ambiguity without increasing the bias, since the ambiguity item (the 2nd term on the right), which measures the disagreement among the base predictors, is always nonnegative. Thus, Eq. 7 indicates that we can design an ensemble learning algorithm to reduce concept drift.

Motivated by this, we propose a deep residual-decreasing ensemble method based on prediction variance minimization and TS decomposition principles, named DeepBooTS. As shown in Fig. 3(b), DeepBooTS is a two-stream architecture corresponding to Eq. 7, which is equivalent to a meta-algorithmic Boosting approach, reducing complexity of the model and thus mitigating the risk of concept drift. As shown in Fig. 3, the '*input*' stream is obviously a decomposition of X since

$$X = \sum_{l=0}^{L-1} g_l(X) + R_L, \quad (8)$$

where R_L is the residual term. Further, the aim of the '*output*' stems is to learn L simple learners in a hierarchy where each learner gives more attention (larger weight) to the hard samples from the previous learner. This is equivalent to the Boosting ensemble learning process, where the final prediction is a weighted sum of L simple learners, and the weights are determined by the previous learner. Let $g_l(x)$ denote the l-th learner, and α_l denote the weight of the l-th learner. The overall estimation $\hat{f}(X)$ is a weighted subtraction of the L estimations. For a sample X, we have

$$\hat{f}(X) = i\sum_{l=0}^{\hbar}\alpha_{2l+1}g_{2l+1}(X) - i\sum_{l=0}^{\hbar}\alpha_{2l}g_{2l}(X), \quad (9)$$

$$s.t.\ \ i\sum_{l=0}^{\hbar}\alpha_{2l+1} - i\sum_{l=0}^{\hbar}\alpha_{2l} = 1, \text{and}\ \ \alpha_l \geq 0.$$

where $i = 1$ if $L \bmod 2 = 1$, else $i = -1$, and $\hbar = \lfloor \frac{L}{2} \rfloor$. Similar to Eq 6, Eq. 9 can be rewritten as

$$(Y-\hat{f}(X))^2 = \sum_{l=0}^{\hbar} i(\alpha_{2l+1}-\alpha_{2l})(Y-g_l(X))^2 - \sum_{l=0}^{\hbar} i(\alpha_{2l+1}-\alpha_{2l})(\hat{f}(X)-g_l(X))^2 \quad (10)$$

Obviously, Eq. 10 derived from Eq. 9 is similar to Eq. 7. However, Eq. 9 has more advantages, that is

Theorem 3. *Assume that the estimation error of block* $g_l(X)$ *is* e_l, $e_l \overset{i.i.d}{\sim} \mathcal{N}(0,\nu)$. *Let* $\alpha_l = \alpha$ *be the weight of* g_l, $l \in [0,L]$, *and the covariance of estimations of two different blocks by* μ, *we have*

$$Var(\hat{Y}) < \frac{4}{L}\alpha^2(\nu+\mu). \quad (11)$$

The proof is given in Appendix B.5. Clearly, the variance of DeepBooTS is bound by the estimation error (noise error) of each block, and the covariance between blocks. It is evident that the subtraction adopted in DeepBooTS can reduce the variance, thereby mitigating concept drift. On the contrary, switching the aggregation operation of the output

stream in the DeepBooTS to addition results in an approximate variance of $\frac{4}{L}\alpha^2\nu + 3\alpha^2\mu$, which is much larger than that of subtraction in Eq. 9. This phenomenon is also supported by our experiments in Section 3.5. Furthermore, Theorem 3 also demonstrates that increasing the number of layers L does not escalate the risk of concept drift. It proves that the deep ensemble models can go deeper, and the test error approaches $\text{Bias}(\hat{f}(X))^2 + \sigma^2$ when L is infinite and doesn't consider performance-efficient trade-off.

2.3 Implementation

Here, we employ deep neural networks to implement DeepBooTS. As shown in Fig. 4, the model consists of two primary data streams. One is the input stream decomposed through multiple residual learners, while the other is the output stream that progressively learns the residuals of the supervised signals. Along the way, they pass through multiple learners capable of converting signals. The model is simple and versatile, yet powerful and interpretable. The pseudocode is given in Appendix L, and complexity analysis is given in Appendix G. We now delve into how these properties are incorporated into DeepBooTS.

Backbone: The fundamental building backbone features a fork architecture, which accepts one input X_l and produces two distinct streams, R_l and O_l. Concretely, R_l is the remaining portion of X_l after it has undergone processing within a neural module, which can be expressed as

$$\hat{X}_l = \text{Block}_l(X_l), \qquad R_l = X_l - \hat{X}_l, \tag{12}$$

Equation 12 represents an implicit decomposition of X, which differs from the moving average adopted by (Wu et al. 2021; Zhou et al. 2022; Liang et al. 2023) but is similar to (Oreshkin et al. 2019). Differently, the residual R_l captures what remains unaltered, providing a basis for comparison with the transformed portion.

In the subsequent steps, the intention is to maximize the utilization of the subtracted portion $\hat{X}_l$. First, $\hat{X}_l$ is projected into the same dimension as the label that is anticipated, Y. This process can be expressed as

$$O_l = \text{Predictor}_l(\hat{X}_l), \tag{13}$$

where O_l is the prediction results of the l-th predictor. Then, O_l will be subtracted from the outputs of the next predictor sequentially until the final prediction $\hat{Y}$ is achieved.

Learner: The basic learner can be constructed utilizing widely-used neural structures, such as fully connected, convolutional, and attention layers. A ready-made solution is to utilize Attention to learn subtle relationships between attributes. As analyzed in Section 2.2, we implement a corrective measure by subtracting the output from the input, i.e.,

$$\hat{X}_{l,1} = \text{Attention}_{\theta_{l,1}}(X_{l,1}), \quad R_{l,1} = X_{l,1} - \delta\hat{X}_{l,1}, \tag{14}$$

where δ is 1 when the module exists, otherwise it is 0. δ serves to eliminate the Attention layer when it exerts adverse effects, thereby enabling the unimpeded flow of input towards the FeedForward (FF) layers:

$$\hat{X}_{l,2} = \text{FF}_{\theta_{l,2}}(R_{l,1}), \qquad R_{l,2} = X_{l,2} - \hat{X}_{l,2}. \tag{15}$$

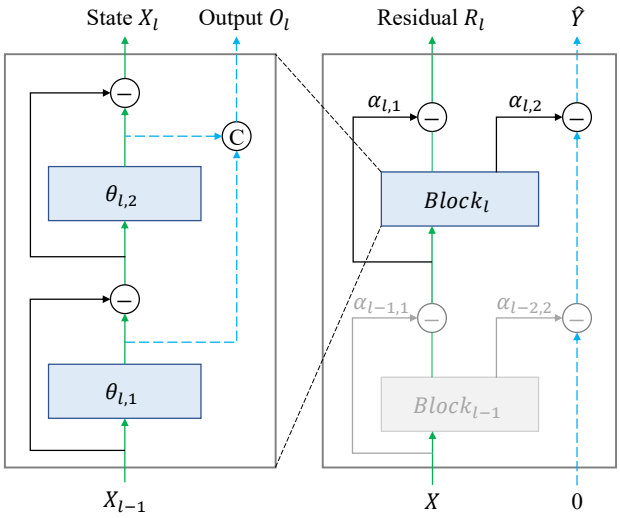


Figure 4: The implementation of DeepBooTS.

Also, the Attention can be implemented in the frequency domain by Fast Fourier Transform (FFT) to achieve lightweight and efficient processing, as

$$Q_l, K_l, V_l = \text{FFT}_{\theta_Q,\theta_K,\theta_V}(X_l), \tag{16}$$

$$\hat{X}_{l,1} = \text{FFT}^{-1}(\phi(Q_l K_l^T)V). \tag{17}$$

where ϕ is the activation function.

Coefficient: Upon being scaled by a coefficient (α_l), the dual streams are directed toward the next block or projected into the output space. Here, we adopt a learnable gating mechanism as the coefficient. The aim is to let each learner regulate the stream-transmission pace autonomously. Thus, for input and output streams, we have

$$X_{l+1} = \varphi(\theta_1(R_{l,2})) \cdot \theta_2(R_{l,2}), \tag{18}$$

$$O_{l+1} = \varphi(\theta_3([\hat{X}_{l,1}, \hat{X}_{l,2}])) \cdot \theta_4([\hat{X}_{l,1}, \hat{X}_{l,2}]), \tag{19}$$

where θ_1 and θ_2 are learnable parameters, φ is the sigmoid, and the brackets '[ ]' are a concatenation operation.

3 Experiments

We conduct a comprehensive comparison with the 18 latest SOTA methods, including TimeXer (Wang et al. 2024b), TimeMixer (Wang et al. 2024a), iTransformer (Liu et al. 2023), PatchTST (Nie et al. 2022), Crossformer (Zhang and Yan 2022), SCINet (Liu et al. 2022a), DLinear (Zeng et al. 2023), FEDformer (Zhou et al. 2022) for multivariate forecasting, and FreTS (Yi et al. 2024c), PSLD (Liang et al. 2024d), and FourierGNN (Yi et al. 2024b) for large-scale TS datasets, as well as Periodformer (Liang et al. 2024c), Autoformer (Wu et al. 2021), Informer (Zhou et al. 2021), LogTrans (Li et al. 2019), Reformer (Kitaev, Kaiser, and Levskaya 2020), N-BEATS (Oreshkin et al. 2019) and N-Hits (Challu et al. 2023) for univariate forecasting. More experimental settings can be found in Appendix D.

Model	DeepBooTS*		DeepBooTS		TimeXer		TimeMixer		iTransformer		PatchTST		Crossformer		DLinear		FEDformer	
Metric	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETT	**0.346**	**0.378**	0.362	0.382	0.365	0.388	0.367	0.388	0.383	0.407	0.381	0.397	0.942	0.684	0.559	0.515	0.437	0.449
Traffic	**0.373**	**0.261**	0.406	0.270	0.466	0.287	0.484	0.297	0.428	0.282	0.481	0.304	0.550	0.304	0.625	0.383	0.610	0.376
ELC	**0.158**	**0.252**	0.166	0.259	0.171	0.270	0.182	0.272	0.178	0.270	0.205	0.290	0.244	0.334	0.212	0.300	0.214	0.327
Weather	**0.227**	**0.266**	0.245	0.271	0.241	0.271	0.240	0.271	0.258	0.279	0.259	0.281	0.259	0.315	0.265	0.317	0.309	0.360
Solar	**0.197**	**0.240**	0.227	0.250	0.272	0.300	0.216	0.280	0.233	0.262	0.270	0.307	0.641	0.639	0.330	0.401	0.291	0.381
PEMS	**0.075**	**0.178**	0.109	0.218	0.141	0.225	0.138	0.242	0.113	0.221	0.180	0.291	0.169	0.281	0.278	0.375	0.213	0.327
1^{st} or 2^{st}	**30**	**30**	15	22	3	4	8	2	3	2	0	0	3	0	1	0	0	0

Table 1: Multivariate TS forecasting results (average). The results for all forecasting lengths are provided in Appendix M.

Model	DeepBooTS		Periodformer		FEDformer		Autoformer		Informer		LogTrans		Reformer	
Metric	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	**0.072**	**0.206**	0.093	0.237	0.111	0.257	0.105	0.252	0.199	0.377	0.345	0.513	0.624	0.600
ETTh2	**0.185**	**0.337**	0.192	0.343	0.206	0.350	0.218	0.364	0.243	0.400	0.252	0.408	3.472	1.283
ETTm1	**0.052**	**0.172**	0.059	0.201	0.069	0.202	0.081	0.221	0.281	0.441	0.231	0.382	0.523	0.536
ETTm2	0.118	0.254	**0.115**	**0.253**	0.135	0.278	0.130	0.271	0.147	0.293	0.130	0.277	0.136	0.288
Traffic	**0.132**	**0.212**	0.150	0.233	0.177	0.270	0.261	0.365	0.309	0.388	0.341	0.417	0.375	0.434
Electricity	0.314	0.401	**0.298**	**0.389**	0.347	0.434	0.414	0.479	0.372	0.444	0.410	0.473	0.352	0.435
Weather	**0.0015**	**0.0293**	0.0017	0.0317	0.008	0.067	0.0083	0.0700	0.0033	0.0438	0.0059	0.0563	0.0115	0.0785
Exchange	0.429	0.453	**0.353**	**0.434**	0.499	0.512	0.578	0.537	1.511	1.029	1.350	0.810	1.028	0.812
1^{st} Count	**24**	**27**	14	12	0	1	0	0	0	0	2	0	0	0

Table 2: Univariate TS forecasting results (average). The results for all forecasting lengths are provided in Appendix N.

3.1 Main Experimental Results

All datasets are adopted for both multivariate (multivariate predict multivariate) and univariate (univariate predicts univariate) tasks. The detailed information pertaining to the datasets can be located in Appendix C. The models used in the experiments are evaluated over a wide range of prediction lengths to compare performance on different future horizons: 96, 192, 336 and 720. Please refer to Appendix K for more experiments on the full ETT dataset.

Multivariate Results: The results for multivariate TS forecasting are outlined in Table 1, with the optimal results highlighted in **red** and the second-best results emphasized with underlined. Due to variations in input lengths among different methods, for instance, PatchTST and DLinear employing an input length of 336, while Crossformer and Periodformer search for the input length without surpassing the maximum setting (720 in Crossformer, 144 in Periodformer), we have configured two versions of DeepBooTS: 336 for DeepBooTS* and 96 for DeepBooTS.

As shown in Table 1, DeepBooTS achieves the consistent SOTA performance across all datasets and prediction length configurations. Compared with the latest models acknowledged for their exceptional average performance, the proposed DeepBooTS demonstrates an average performance increase of **15.8%**, achieving a substantial performance improvement. Obviously, it achieves advanced performance, averaging **30** items across six datasets, with an improvement in average performance on each dataset. These experimental results confirm that the proposed DeepBooTS demonstrates superior prediction performance across different datasets with varying horizons. For a more detailed analysis, please refer to Appendix O.

Univariate Results: The average results for univariate TS forecasting are shown in Table 2. It is evident that the proposed DeepBooTS continues to maintain a SOTA performance across various prediction length settings compared to the benchmarks. In summary, compared with the hyperparameter-searched Periodformer, DeepBooTS yields an average **4.8%** reduction across five datasets, and it achieves an average of **26** best terms. For example, under the input-96-predict-96 setting, DeepBooTS yields a reduction of **11.2%** (0.143→0.127) in MSE for Traffic. Obviously, the experimental results again verify the superiority of DeepBooTS on univariate TS forecasting tasks.

3.2 Evaluation on Monash TS Datasets

Further, we evaluate the proposed method on 7 Monash TS datasets (Godahewa et al. 2021) (e.g., NN5, M4 and Sunspot, etc.) and 7 diverse metrics (e.g., MAPE, sMAPE, MASE and Quantile, etc.) to systematically evaluate our model. All experiments are compared under the same input length (e.g., I=96) and output lengths (e.g., O={96, 192, 336 and 720}). As shown in Table 3, the proposed DeepBooTS emerged as the frontrunner, achieving a score of **41 out of 54**. Please refer to Appendices Q and R for details about the definition and experimental settings, respectively.

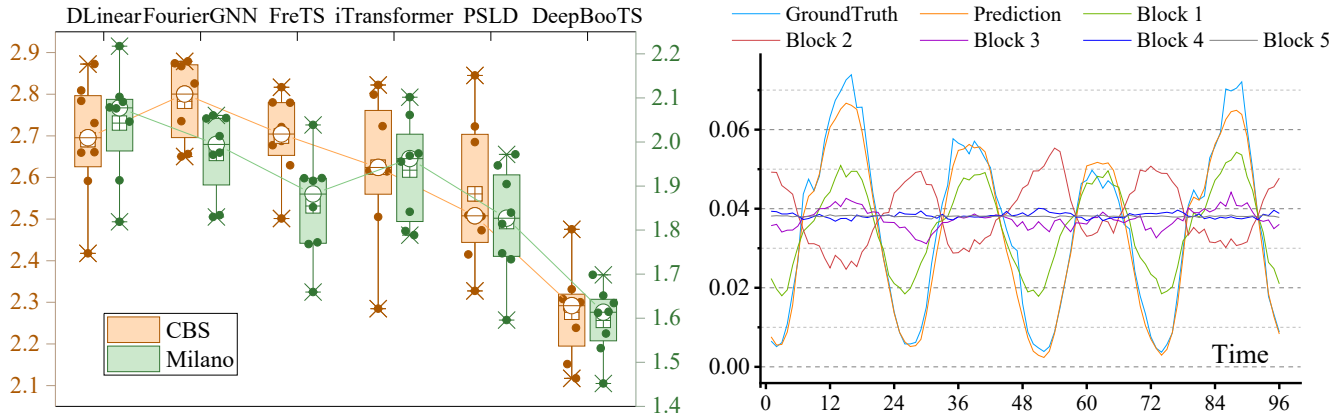
3.3 Performance on Large-Scale Datasets

The performance comparisons for large-scale TS datasets are summarized in Fig. 5(a), including the CBS dataset with 4,454 nodes (17GB) and the Milano dataset with 10,000 nodes (19GB). On these datasets, it is not feasible to place all

Mulvariate	ILI						Oik_Weather						NN5					
Metric	MSE	MAE	RMSP	sMAPE	MASE	Q75	MSE	MAE	RMSP	sMAPE	MASE	Q75	MSE	MAE	RMSP	sMAPE	MASE	Q75
DeepBooTS	**2.016**	**0.86**	**0.367**	**0.650**	**0.574**	**0.955**	0.689	**0.618**	**0.773**	**1.082**	0.888	**0.628**	**0.719**	**0.578**	**0.541**	**0.861**	**0.530**	**0.559**
iTransformer	2.262	0.958	0.415	0.703	0.648	1.107	0.720	0.632	0.781	1.090	0.935	0.636	0.723	0.583	0.554	0.877	0.542	0.565
DLinear	2.915	1.188	0.574	0.865	0.791	1.524	0.701	0.637	0.805	1.185	0.943	0.641	1.448	0.929	0.969	1.304	0.835	0.978
Autoformer	3.187	1.224	0.596	0.909	0.740	1.349	0.853	0.712	0.883	1.201	0.992	0.704	0.851	0.659	0.636	0.975	0.603	0.637
Informer	5.208	1.576	0.792	1.183	0.907	2.304	**0.67**	0.625	0.795	1.126	**0.818**	0.640	0.967	0.727	0.739	1.081	0.662	0.722
Univariate	M4 Hourly						Us_births						Saugeenday					
DeepBooTS	**0.223**	**0.287**	**0.230**	**0.509**	**0.276**	**0.311**	**0.364**	**0.431**	**0.248**	0.445	**0.363**	**0.361**	1.194	0.544	**0.808**	**1.053**	1.232	0.674
iTransformer	0.310	0.390	0.376	0.656	0.366	0.410	0.378	0.439	0.251	**0.438**	0.365	0.388	1.206	0.552	0.832	1.066	1.258	0.687
N-Beats	0.337	0.423	0.392	0.700	0.390	0.493	0.670	0.638	0.409	0.625	0.546	0.834	1.028	0.543	0.962	1.387	0.943	0.572
N-Hits	0.346	0.418	0.402	0.687	0.396	0.442	0.538	0.578	0.364	0.540	0.464	0.802	**0.982**	**0.532**	0.865	1.189	**0.858**	**0.562**
Autoformer	0.635	0.611	0.614	0.881	0.518	0.741	0.901	0.757	0.447	0.789	0.737	0.788	1.252	0.640	1.058	1.370	1.057	0.674

* All results are averaged across all prediction lengths. The results for all prediction lengths and the experimental settings are provided in Appendix R. The definitions of all metrics are provided in Appendix Q.

Table 3: Mulvariate and univariate forecasting results with diverse metrics on Monash TS datasets.



(a) Performance on Large TS Datasets (b) Visualization (Block=5, embed_dim=512)

Figure 5: (a) Comparisons on large-scale TS datasets. (b) Visualization depicting the output of each block in DeepBooTS.

nodes on a single GPU for long-term TS forecasting. Therefore, we adopted the random partitioning strategy (Liang et al. 2024a) for our experiments. Please refer to Appendices C.3. and D for details. Compared to the latest advanced PSLD, the proposed DeepBooTS yields an overall MSE reduction of **8.9%** and **6.2%** on the CBS and Milano datasets, respectively. In addition, we are pleased to report that our method has been successfully applied to real-world scenarios, e.g., energy and power, showing substantial practical value. For more experiments, please refer to Appendix F.

3.4 Generality

To investigate DeepBooTS's generality as a universal architecture, we substituted its original Attention with other novel Attention mechanisms to observe the resulting changes in model performance. As shown in Fig. 6, after harnessing the newly invented Attention within DeepBooTS, its performance exhibited considerable variation. E.g., the average MSE of Prob-Attention (Zhou et al. 2021) on the Electricity and Weather datasets witnessed a reduction of **48.2%** (0.311→0.161) and **61.7%** (0.634→0.243), respectively, surpassing Full-Attention and achieving new SOTA performance. Furthermore, Period-Attention (Liang et al. 2023), Auto-Correlation (Wu et al. 2021), and Flow-Attention (Wu et al. 2022b) all exhibit commendable performance on the aforementioned datasets, with their average performance surpassing that of Full-Attention. The conducted experiments suggest that DeepBooTS can serve as a versatile architecture, amenable to the integration of novel modules, thereby facilitating the enhancement of performance in TS forecasting.

3.5 Effectiveness

To validate the effectiveness of DeepBooTS components, we conduct comprehensive ablation studies encompassing both component replacement and component removal experiments, as shown in Fig. 7. We utilize signs '+' and '-' to denote the utilization of addition or subtraction operations during the aggregation process of the input or output streams. In cases involving only input streams, it becomes evident that the model's average performance is superior when employing subtraction (-X) compared to when employing addition

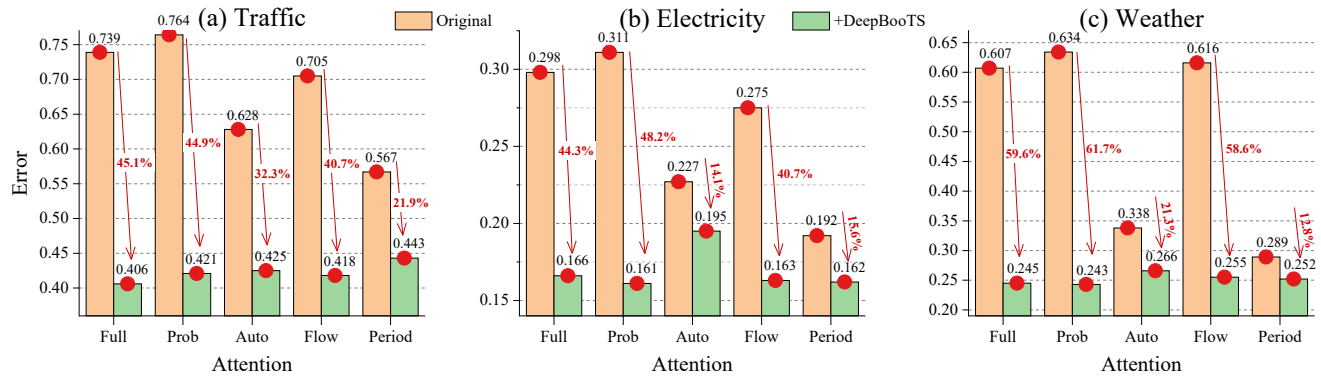


Figure 6: Ablation studies of DeepBooTS using various Attention. All results are averaged across all prediction lengths. The tick labels of the X-axis are the abbreviation of Attention types. The detailed setup and results are provided in Appendix Q.

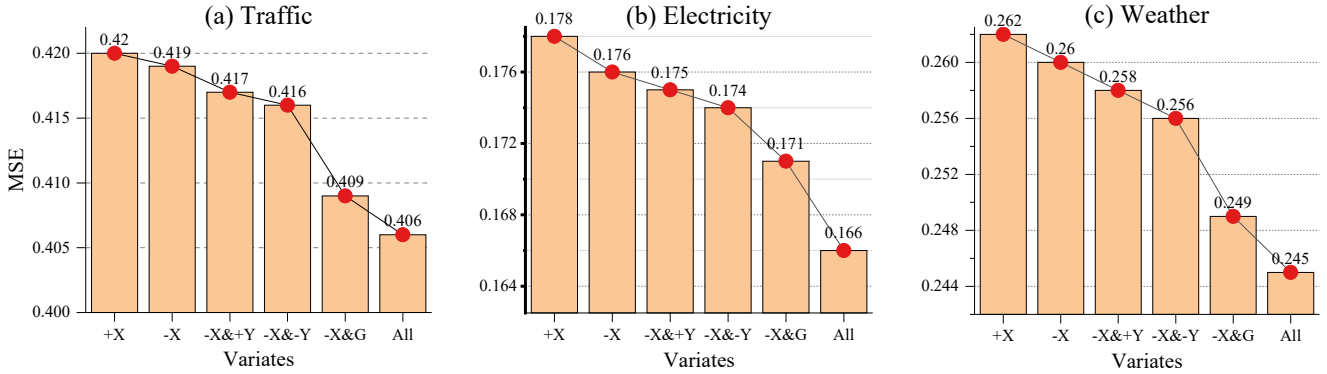


Figure 7: Ablation studies on various components of DeepBooTS. All results are averaged across all prediction lengths. The variables *X* and *Y* represent the input and output streams, while the signs '+' and '-' denote the addition or subtraction operations used when the streams' aggregation. The letter 'G' denotes adding a learnable gating (Coefficient) to the output of each block.

(+X). E.g., on the Weather dataset, forecast error is reduced by **2.3%** (0.262→0.256). Moreover, with the introduction of a high-speed output stream to the model, shifting the aggregation method of the output stream from addition (+Y) to subtraction (-Y) is poised to further enhance the model's performance. Afterward, incorporating gating mechanisms (G) into the model holds the potential to improve predictive performance again, e.g., forecast error is reduced by **4.9%** (0.262→0.249) on the Weather dataset. In summary, integrating the advantages of the aforementioned components has the potential to significantly boost the model's performance across the board.

3.6 Interpretability

The intrinsic characteristic of DeepBooTS lies in the alignment of the output from each block with the final output. This alignment, in turn, expedites the interpretability of the model's learning process. As depicted in Fig. 5(b), the output of each block in DeepBooTS is visualized. It becomes evident that each block discerns and assimilates meaningful patterns. Specifically, when the embedding dimension is low, each block must learn salient patterns. Further, when the depth of the model is increased, it becomes evident that the amplitude of the shallow block decreases, and numerous components are transferred to the deep block. For a more detailed analysis, please refer to Appendix P.

3.7 Reduce Variance and Go Deeper

By adjusting model hyperparameters and injecting noise, we generated a diverse set of predictions from latest advanced models. Then, these predictions are statistically analyzed to compute the prediction mean and variance. As shown in Fig. 8(a), DeepBooTS achieves superior performance and the smallest prediction variance, while other models exhibit weaker alignment with ground truth and higher variance. This validates the effectiveness of our theory and model design. Meanwhile, these results also confirm that DeepBooTS is less sensitive to hyperparameters, with details presented in Appendix I. Furthermore, given DeepBooTS's robustness against high-variance, it can be designed with considerable depth. Fig. 8(b) shows the scenarios when models go deeper. Serious overfitting happens when the number of iTransformer blocks is increased from 4 to 8. However, even with the DeepBooTS blocks deepened to 16, it continues to exhibit excellent performance.

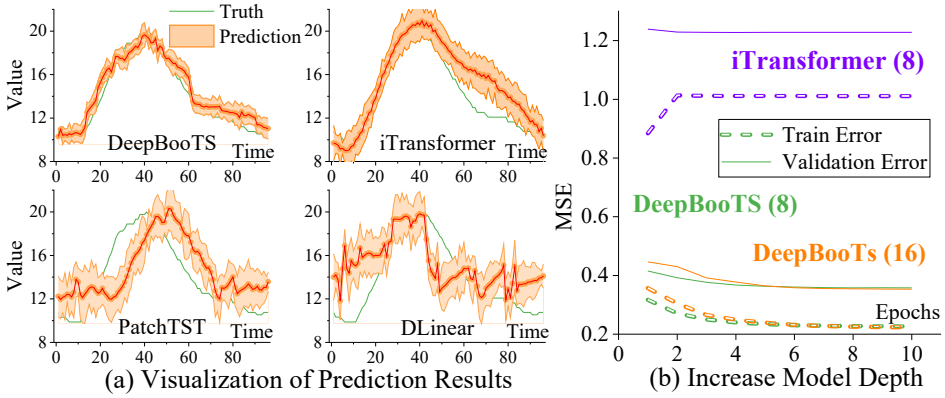


Figure 8: Comparisons of model's variance and depth.

4 Conclusion

In this paper, we theoretically analyze the impact of concept drift in TS forecasting from a bias-variance perspective. We demonstrate that the deep ensemble is capable of reducing this issue. Based on this, we propose a novel dual-stream residual-decreasing Boosting ensemble approach, termed DeepBooTS, and demonstrate its efficacy in variance reduction. DeepBooTS facilitates the learning-driven and progressive decomposition of both the input and output streams, thereby empowering the model resilience against concept drift. Extensive experiments show that DeepBooTS achieves SOTA performance, and it is less sensitive to hyperparameters and can be designed very deep with enhanced interpretability. Furthermore, DeepBooTS can serve as a versatile framework to enhance other models' performance.

Acknowledgments

This work was supported by Project of Key R&D Program of Shandong Province, China (2025CXGC010107, 2025CXPT095), Taishan Scholars Program: NO.tspd20240814, the Pilot Project for the Integration of Science, Education and Industry of Qilu University of Technology (Shandong Academy of Sciences)(2025ZDZX01), National Key R&D Program of China (2024YFB3312302), and the Qilu Institute of Technology for Talent Project (QIT24TP027), and the Young Talent of Lifting engineering for Science and Technology in Shandong (SDAST2025QTB008).

References

Ahamed, M. A.; and Cheng, Q. 2024. Timemachine: A time series is worth 4 mambas for long-term forecasting.

Anderson, O. D. 1976. Time-series. *Journal of the Royal Statistical Society. Series D (The Statistician)*, 25(4): 308–310.

Baier, L.; Hofmann, M.; Kühl, N.; Mohr, M.; and Satzger, G. 2020. Handling Concept Drifts in Regression Problems–the Error Intersection Approach.

Barlacchi, G.; De Nadai, M.; Larcher, R.; Casella, A.; Chitic, C.; Torrisi, G.; Antonelli, F.; Vespignani, A.; Pentland, A.; and Lepri, B. 2015. A multi-source dataset of urban life in the city of Milan and the Province of Trentino. *Scientific data*, 2(1): 1–15.

Castro-Neto, M.; Jeong, Y.-S.; Jeong, M.-K.; and Han, L. D. 2009. Online-SVR for short-term traffic flow prediction under typical and atypical traffic conditions. *Expert Systems with Applications*, 36(3): 6164–6173.

Challu, C.; Olivares, K. G.; Oreshkin, B. N.; Ramirez, F. G.; Canseco, M. M.; and Dubrawski, A. 2023. Nhits: Neural hierarchical interpolation for time series forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, 6989–6997.

Cleveland, R. B.; Cleveland, W. S.; McRae, J. E.; and Terpenning, I. 1990. STL: A seasonal-trend decomposition. *J. Off. Stat*, 6(1): 3–73.

Connor, J. T.; Martin, R. D.; and Atlas, L. E. 1994. Recurrent neural networks and robust time series prediction. *IEEE Transactions on Neural Networks*, 5(2): 240–254.

Darlow, L. N.; Deng, Q.; Hassan, A.; Asenov, M.; Singh, R.; Joosen, A.; Barker, A.; and Storkey, A. 2024. DAM: Towards a Foundation Model for Forecasting. In *The Twelfth International Conference on Learning Representations*.

De Gooijer, J. G.; and Hyndman, R. J. 2006. 25 years of time series forecasting. *International journal of forecasting*, 22(3): 443–473.

Gardner Jr, E. S. 1985. Exponential smoothing: The state of the art. *Journal of forecasting*, 4(1): 1–28.

Garza, A.; Challu, C.; and Mergenthaler-Canseco, M. 2023. TimeGPT-1.

Godahewa, R. W.; Bergmeir, C.; Webb, G. I.; Hyndman, R.; and Montero-Manso, P. 2021. Monash Time Series Forecasting Archive. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.

Goodfellow, I.; Bengio, Y.; and Courville, A. 2016. *Deep learning*. MIT press.

Hornik, K. 1991. Approximation capabilities of multilayer feedforward networks. *Neural networks*, 4(2): 251–257.

Hyndman, R. J.; and Athanasopoulos, G. 2018. *Forecasting: principles and practice*. OTexts.

Jia, Y.; Lin, Y.; Yu, J.; Wang, S.; Liu, T.; and Wan, H. 2024. PGN: The RNN's New Successor is Effective for Long-Range Time Series Forecasting. *Advances in Neural Information Processing Systems*, 37: 84139–84168.

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. 2024. Time-LLM: Time Series Forecasting by Reprogramming Large Language Models. In *The Twelfth International Conference on Learning Representations*.

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

Kingma, D. P.; and Ba, J. 2015. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*. Santiago de Cuba.

Kitaev, N.; Kaiser, L.; and Levskaya, A. 2020. Reformer: The Efficient Transformer. In *8th International Conference on Learning Representations (ICLR)*. Ababa, Ethiopia.

Korycki, Ł.; and Krawczyk, B. 2023. Adversarial concept drift detection under poisoning attacks for robust data stream mining. *Machine Learning*, 112(10): 4013–4048.

Lai, G.; Chang, W.-C.; Yang, Y.; and Liu, H. 2018. Modeling long- and short-term temporal patterns with deep neural networks. In *The 41st international ACM SIGIR conference on research & development in information retrieval (SIGIR)*, 95–104. Ann Arbor, MI, USA.

Lea, C.; Vidal, R.; Reiter, A.; and Hager, G. D. 2016. Temporal convolutional networks: A unified approach to action segmentation. In *European conference on computer vision*, 47–54. Springer.

LeCun, Y.; Bottou, L.; Bengio, Y.; and Haffner, P. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11): 2278–2324.

Li, F.; Feng, J.; Yan, H.; Jin, G.; Yang, F.; Sun, F.; Jin, D.; and Li, Y. 2023. Dynamic graph convolutional recurrent network for traffic prediction: Benchmark and solution. *ACM Transactions on Knowledge Discovery from Data*, 17(1): 1–21.

Li, S.; Jin, X.; Xuan, Y.; Zhou, X.; Chen, W.; Wang, Y.-X.; and Yan, X. 2019. Enhancing the Locality and Breaking the Memory Bottleneck of Transformer on Time Series Forecasting. In *Advances in 33rd Neural Information Processing Systems (NeurIPS)*, volume 32, 5243–5253. Vancouver, Canada.

Liang, D. 2025. DistPred: A Distribution-Free Probabilistic Inference Method for Regression and Forecasting. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 1*, 753–764.

Liang, D.; Zhang, H.; Wang, J.; Yuan, D.; and Zhang, M. 2024a. Act Now: A Novel Online Forecasting Framework for Large-Scale Streaming Data.

Liang, D.; Zhang, H.; Yuan, D.; Ma, X.; Li, D.; and Zhang, M. 2023. Does Long-Term Series Forecasting Need Complex Attention and Extra Long Inputs?

Liang, D.; Zhang, H.; Yuan, D.; Zhang, B.; and Zhang, M. 2024b. Minusformer: Improving Time Series Forecasting by Progressively Learning Residuals.

Liang, D.; Zhang, H.; Yuan, D.; and Zhang, M. 2024c. Periodformer: An efficient long-term time series forecasting method based on periodic attention. *Knowledge-Based Systems*, 304: 112556.

Liang, D.; Zhang, H.; Yuan, D.; and Zhang, M. 2024d. Progressive Supervision via Label Decomposition: An long-term and large-scale wireless traffic forecasting method. *Knowledge-Based Systems*, 305: 112622.

Liaw, A.; Wiener, M.; et al. 2002. Classification and regression by randomForest. *R News*, 2(3): 18–22.

Liu, M.; Zeng, A.; Chen, M.; Xu, Z.; Lai, Q.; Ma, L.; and Xu, Q. 2022a. Scinet: Time series modeling and forecasting with sample convolution and interaction. In *Advances in Neural Information Processing Systems*, 5816–5828.

Liu, X.; Hu, J.; Li, Y.; Diao, S.; Liang, Y.; Hooi, B.; and Zimmermann, R. 2024. Unitime: A language-empowered unified model for cross-domain time series forecasting. In *Proceedings of the ACM Web Conference 2024*, 4095–4106.

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

Liu, Y.; Wu, H.; Wang, J.; and Long, M. 2022b. Non-stationary transformers: Exploring the stationarity in time series forecasting. *Advances in Neural Information Processing Systems*, 35: 9881–9893.

Ma, H.; Chen, Y.; Zhao, W.; Yang, J.; Ji, Y.; Xu, X.; Liu, X.; Jing, H.; Liu, S.; and Yang, G. 2024. A Mamba Foundation Model for Time Series Forecasting.

Makridakis, S.; Spiliotis, E.; and Assimakopoulos, V. 2020. The M4 Competition: 100,000 time series and 61 forecasting methods. *International Journal of Forecasting*, 36(1): 54–74.

Ni, Z.; Yu, H.; Liu, S.; Li, J.; and Lin, W. 2023. Basisformer: Attention-based time series forecasting with learnable and interpretable basis. *Advances in Neural Information Processing Systems*, 36: 71222–71241.

Nie, Y.; Nguyen, N. H.; Sinthong, P.; and Kalagnanam, J. 2022. A Time Series is Worth 64 Words: Long-term Forecasting with Transformers. In *The Eleventh International Conference on Learning Representations*.

Oreshkin, B. N.; Carpov, D.; Chapados, N.; and Bengio, Y. 2019. N-BEATS: Neural basis expansion analysis for interpretable time series forecasting. In *International Conference on Learning Representations*.

Pham, Q.; Liu, C.; Sahoo, D.; and Hoi, S. C. 2023. Learning Fast and Slow for Online Time Series Forecasting. In *8th International Conference on Learning Representations (ICLR)*.

Piccolo, D. 1990. A distance measure for classifying ARIMA models. *Journal of time series analysis*, 11(2): 153–164.

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.

Shabani, M. A.; Abdi, A. H.; Meng, L.; and Sylvain, T. 2022. Scaleformer: Iterative Multi-scale Refining Transformers for Time Series Forecasting. In *The Eleventh International Conference on Learning Representations*.

Shao, Z.; Zhang, Z.; Wang, F.; Wei, W.; and Xu, Y. 2022. Spatial-temporal identity: A simple yet effective baseline for multivariate time series forecasting. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, 4454–4458.

Taylor, S. J.; and Letham, B. 2018. Forecasting at scale. *The American Statistician*, 72(1): 37–45.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L. u.; and Polosukhin, I. 2017. Attention is all you need. In *Advances in 31st Neural Information Processing Systems (NeurIPS)*, volume 30, 6000–6010. Long Beach, USA.

Wang, S.; Wu, H.; Shi, X.; Hu, T.; Luo, H.; Ma, L.; Zhang, J. Y.; and ZHOU, J. 2024a. TimeMixer: Decomposable Multiscale Mixing for Time Series Forecasting. In *The Twelfth International Conference on Learning Representations*.

Wang, Y.; Wu, H.; Dong, J.; Qin, G.; Zhang, H.; Liu, Y.; Qiu, Y.; Wang, J.; and Long, M. 2024b. Timexer: Empowering transformers for time series forecasting with exogenous variables. *Advances in Neural Information Processing Systems*, 37: 469–498.

Wen, Q.; Chen, W.; Sun, L.; Zhang, Z.; Wang, L.; Jin, R.; Tan, T.; et al. 2023. Onenet: Enhancing time series forecasting models under concept drift by online ensembling. *Advances in Neural Information Processing Systems*, 36: 69949–69980.

Wu, H.; Hu, T.; Liu, Y.; Zhou, H.; Wang, J.; and Long, M. 2022a. TimesNet: Temporal 2D-Variation Modeling for General Time Series Analysis. In *The Eleventh International Conference on Learning Representations*.

Wu, H.; Wu, J.; Xu, J.; Wang, J.; and Long, M. 2022b. Flowformer: Linearizing Transformers with Conservation Flows. In *International Conference on Machine Learning*.

Wu, H.; Xu, J.; Wang, J.; and Long, M. 2021. Autoformer: Decomposition Transformers with Auto-Correlation for Long-Term Series Forecasting. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 34, 22419–22430. Virtual Conference.

Wu, Z.; Pan, S.; Long, G.; Jiang, J.; and Zhang, C. 2019. Graph WaveNet for Deep Spatial-Temporal Graph Modeling. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*. International Joint Conferences on Artificial Intelligence Organization.

Xiaming Chen, Y. J.; Qiang, S.; Hu, W.; and Jiang, K. 2015. Analyzing and Modeling Spatio-Temporal Dependence of Cellular Traffic at City Scale. In *Communications (ICC), 2015 IEEE International Conference on*.

Ye, W.; Deng, S.; Zou, Q.; and Gui, N. 2024. Frequency adaptive normalization for non-stationary time series forecasting. *Advances in Neural Information Processing Systems*, 37: 31350–31379.

Yi, K.; Fei, J.; Zhang, Q.; He, H.; Hao, S.; Lian, D.; and Fan, W. 2024a. Filternet: Harnessing frequency filters for time series forecasting. *Advances in Neural Information Processing Systems*, 37: 55115–55140.

Yi, K.; Zhang, Q.; Fan, W.; He, H.; Hu, L.; Wang, P.; An, N.; Cao, L.; and Niu, Z. 2024b. FourierGNN: Rethinking multivariate time series forecasting from a pure graph perspective. *Advances in Neural Information Processing Systems*, 36.

Yi, K.; Zhang, Q.; Fan, W.; Wang, S.; Wang, P.; He, H.; An, N.; Lian, D.; Cao, L.; and Niu, Z. 2024c. Frequency-domain MLPs are more effective learners in time series forecasting. *Advances in Neural Information Processing Systems*, 36.

Yu, E.; Lu, J.; Wang, K.; Yang, X.; and Zhang, G. 2025a. Drift-aware Collaborative Assistance Mixture of Experts for Heterogeneous Multistream Learning.

Yu, E.; Lu, J.; Yang, X.; Zhang, G.; and Fang, Z. 2025b. Learning Robust Spectral Dynamics for Temporal Domain Generalization. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

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, C.; Bengio, S.; Hardt, M.; Recht, B.; and Vinyals, O. 2021. Understanding deep learning (still) requires rethinking generalization. *Communications of the ACM*, 64(3): 107–115.

Zhang, Q.; Yu, C.; Wang, H.; Yan, Y.; Cao, Y.; Yiu, S.-M.; Wu, T.; and Yin, H. 2025. FLDmamba: Integrating Fourier and Laplace Transform Decomposition with Mamba for Enhanced Time Series Prediction.

Zhang, Y.; and Yan, J. 2022. Crossformer: Transformer utilizing cross-dimension dependency for multivariate time series forecasting. In *The Eleventh International Conference on Learning Representations*.

Zhou, H.; Zhang, S.; Peng, J.; Zhang, S.; Li, J.; Xiong, H.; and Zhang, W. 2021. Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting. In *Proceedings of the 35th AAAI Conference on Artificial Intelligence (AAAI)*, volume 35, 11106–11115. Virtual Conference.

Zhou, T.; Ma, Z.; Wen, Q.; Wang, X.; Sun, L.; and Jin, R. 2022. FEDformer: Frequency enhanced decomposed transformer for long-term series forecasting. In *Proceedings of the 39th International Conference on Machine Learning (ICML)*, volume 162, 27268–27286. Baltimore, Maryland.

Zhou, T.; Niu, P.; Sun, L.; Jin, R.; et al. 2023. One fits all: Power general time series analysis by pretrained lm. *Advances in neural information processing systems*, 36: 43322–43355.