ForeNet: Unlocking Long-Term Series Forecasting in High-Dimensional Scenario via Forest Structure

Xinyu Li, Hao Xu, Zhiheng Yang, Hongxiang Zhou, Hong Lu, Xin Wang†, Jin Zhao
School of Computer Science, Shanghai Key Laboratory of Intelligent Information Processing, Fudan University

***Abstract*—Facing the key challenge in multivariate long-term series prediction, namely effectively modeling the long-term dependencies among high-dimensional variables, we propose an innovative forest network (ForeNet). Firstly, we construct a polytree, which takes variable as leaf node, convolution as edge, and progressive fusion as the root node. Polytree models the dependencies among variables from the bottom up, adopting a local-to-global progressive learning strategy. Moreover, polytree embeds the entire sequence as input channels for convolution, allowing interactions across arbitrary time steps between variables. Thereby, polytree could model long-term dependencies. Then, we construct multiple polytrees with varying branching factors, utilizing the self-attention mechanism to assign weights and combine polytrees into a forest. Forest adopts an ensemble learning strategy to capture complex patterns hidden under high-dimensional variables. Combining progressive and ensemble strategies, ForeNet could effectively address the key challenge mentioned at the beginning. Extensive experiments show that ForeNet could reduce the average MSE by up to 11.53% compared to the baseline, which verifies the effectiveness of ForeNet. Source code is available at: https://github.com/lxy-PhD2022/ForeNet**

***Index Terms*—Time series forecasting, Multivariate long-term forecasting, Forest**

I. Introduction

Time series is widely employed in multimedia, such as audio and speech. In recent years, due to the importance of multivariate long-term series prediction, it has become the focus of research. In the real world, many scenarios involve high-dimensional variables with complex and long-term impacts. Correlations, causal relationships, and lag effects among these variables complicate accurate predictions. Therefore, the key challenge is to effectively model the long-term dependencies among high-dimensional variables.

One category of the existing methods is the channel-independent methods [1]–[5]. They have no explicit variables interaction and thereby have difficulty in learning the relationships among variables. Another category of methods [6]–[8] learns the sample point relationships across variables at each time step, but due to the use of scattered time point data, they often lack sufficient variable information. This results in imprecise variable relationships learning. There is also a category of methods that utilize the Transformer model [9] to learn the relationships of sub-sequences or sequences across variables [10] [11]. However, they have difficulty in deeply mining the long-term dependency relationships among variables. Furthermore, due to the computational complexity of Transformer being quadratic with respect to the number of its inputs, it is difficult to be applicable in scenarios with high-dimensional variables.

In order to improve prediction accuracy, we need to effectively model the long-term dependencies among high-dimensional variables. Therefore, we introduce an innovative approach known as the Forest Network (ForeNet), which centers around three key strategies: progressive modeling, ensemble, and interaction. We introduce as follows in turn.

We begin with the progressive strategy, which also encompasses elements of interaction. In ForeNet, the input variable sequences are fed to each tree in the forest as leaf nodes, where each leaf node represents a single variable sequence. Multiple adjacent leaf nodes merge upward into a parent node, facilitating interaction between variables. Convolution, naturally suited for feature extraction and multi-item merging, is employed to amalgamate child nodes, thus serving as the edges of the tree, with the in-degree of each parent node equal to the size of the convolution kernel.

Since the convolution operates across the variable dimension, facilitating interactions across arbitrary time steps between variables becomes challenging, which is crucial for effectively modeling long-term dependencies between variables. To address this, we embed the entire variable sequence as input channels for the convolution, allowing arbitrary time step interactions between different variables. This configuration enables us to effectively model the long-term dependencies between variables. Through successive merging, all variable features and long-term dependencies are extracted up to the root node, forming a polytree.

Next, we introduce the ensemble strategy, which also involves elements of the interaction strategy. We configure trees with different merging scales (in-degrees of parent nodes) and integrate them into a forest. The varying merging scales represent different scales of variable interaction and degrees of progression, thereby endowing the forest with stronger capabilities for pattern extraction and generalization compared to individual trees.

A key challenge in effective integration is the identification of differences among the trees, which allows us to assign different levels of confidence to all trees, thereby enhancing the aforementioned capabilities. To achieve it, we employ a Transformer to weight and amplify the differences between the trees. We use it to compute the relevance of the root nodes and

†represents corresponding author, whose email is xinw@fudan.edu.cn. This work is partially supported by the National Natural Science Foundation of China (Project No. 62472101) and Shanghai Municipal Science and Technology Commission (Project No. 22dz1204900).

merge them into the forest. The computational complexity of the used Transformer is $O(N)$ because the parallel number of trees (tokens) increases linearly as variables increase in deep learning framework. By adopting these strategies, ForeNet reduces the average mean squared error (MSE) by up to 11.53% on authoritative benchmarks compared to the baseline. Our main contributions are as follows.

- We employ a polytree structure to learn variable relationships progressively. It allows free interactions across arbitrary time steps between variables. This effectively addresses the challenge of capturing long-term dependencies across variables.
- We construct a forest consisting of different polytrees, which show preferences for processing different data features according to their structural differences. By ensembling polytrees with different preferences for data features, ForeNet could gain insights into the complex patterns hidden deep within high-dimensional variables.
- We apply the Transformer to ensemble polytrees into a forest. Here, Transformer's computational complexity is $O(N)$ because its number of tokens is not affected by the number of variables and the length of the sequence. This effectively solves the curse of dimensionality problem that Transformer faces in high-dimensional variable scenarios.

II. RELATED WORK

The existing methods could be roughly divided into the following categories. In the first category of methods: PatchTST [1] explores a more efficient application way of Transformer and proposes patch-level self-attention; TimesNet [2] captures temporal 2D-variations by transforming the 1D time series into 2D space; DLinear [3] demonstrates that the linear layer can effectively capture temporal relationships; RLinear [4] further proves the effectiveness of linear layers, reversible normalization, and channel independence; and TiDE [5] leverages MLP to build an efficient encoder-decoder architecture. However, because they have no explicit variables interaction, they have difficulty in learning the relationships among variables.

In the second category of methods: Informer [6] proposes a more efficient ProbSparse self-attention; Autoformer [7] performs autocorrelation fusion on the most similar parts of series; FEDformer [8] learns by randomly selecting a fixed number of high and low frequency components. Due to the use of scattered time point data, they often lack sufficient variable information. This results in imprecise variable relationships learning.

In the third category of methods: Crossformer [10] proposes two-stage attention to model the sub-sequence relationships across variables; MSGNet [11] calculates the positive and negative correlations of different scale periodic sub-sequences. Since they learn only partial information about each variable, they have difficulty in deeply mining the long-term dependency. Moreover, due to the computational complexity of Transformer being quadratic with respect to the number of variables, they are difficult to be applicable in scenarios with high-dimensional variables.

III. FORENET

A. Problem Definition

In multivariate time series forecasting, the inputs are multiple time series $\boldsymbol{x} \in R^{C \times L}$, where C is variate number and L is look-back window. For each single series of $c\text{-}th$ variate (channel) $\boldsymbol{x}^{(c)} \in R^{1 \times L}$, the goal is to forecast T future values $\boldsymbol{y}^{(c)} = (x_{L+1}^{(c)}, \ldots, x_{L+T}^{(c)}) \in R^{1 \times T}$. Therefore, the multivariate prediction result $\boldsymbol{y} \in R^{C \times T}$.

B. Workflow Overview

To effectively mitigate the impact of non-stationary sequences, we first apply instance normalization [12] to the input, normalizing each sample point to have zero mean and unit standard deviation.

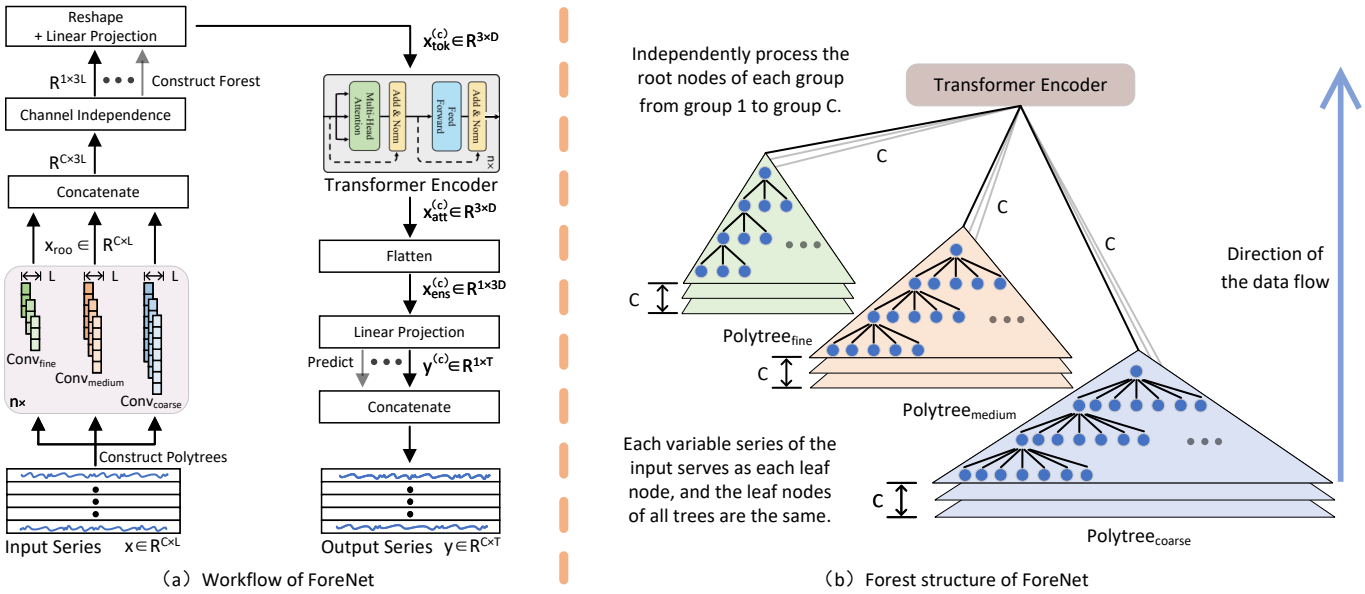
As shown in Fig. 1(a), ForeNet first employs convolutions of multiple scales to construct multiple polytrees. During this process, it conducts n times convolutions on variable dimension C. Then, it concatenates the output of convolutions and constructs a forest in a channel-independent manner. It reshapes the root nodes of each variable (located in different colored trees in (b)) into a multi-channel form and linearly maps them into tokens. After that, it feeds these tokens into a Transformer encoder to weight different trees. Subsequently, Transformer's outputs for each variable are concatenated and passed through a linear layer to form the predicted sequence. Finally, the predicted sequences for all variables are concatenated to produce the output sequence.

C. Polytree Construction

As shown in Fig. 1(a), the input is first fed into a convolution. Taking any convolution in the figure as an example, it operates along the variable dimension (vertical direction), merging information from adjacent variables. In this process, input variables act as leaf nodes, and the convolution serves as an edge that merges them upward into parent nodes. The convolution kernel slides along the variable dimension, selecting a set of leaf nodes to merge into a parent node, thus building one layer of the tree.

Following this method, applying convolutions to the new layer of parent nodes allows us to construct a multi-layered tree. Since the dimension of input variables must strictly correspond to the predicted results, it is necessary to maintain the number of variables constant during convolution. To this end, we apply zero-padding along the variable dimension during the convolution process. Consequently, the number of both root nodes and leaf nodes is C. For ease of understanding in Fig. 1(b), we depict the generation process of these C root nodes as overlapping trees under a uniform color. Thus, the actual computational complexity is not as high as it appears in Fig. 1(b).

However, if the convolution has only one input channel - that is, handling a single time step (horizontal direction) at once - the semantic information of each variable can only



(a) Workflow of ForeNet (b) Forest structure of ForeNet

Fig. 1. (a) We construct polytrees through convolution and then ensemble a forest using Transformer encoder. (b) Polytrees is a progressive learning structure from local to global. Forest is an ensemble learning structure with different feature processing preferences. The polytrees of the same color are formed in one time convolution using zero-padding, not through separate C convolutions, thus the computational burden is not as high.

be extracted from a single point. Formally, the convolution kernel is denoted as $kernel \in R^{1\times K}$, where 1 represents the number of input channels, and K is the kernel size. Due to the lack of sufficient variable information, it encounters the same issues as existing methods [6]–[8], [13], namely the inaccurate learning of variable relationships. On the other hand, this setup prevents free interactions across arbitrary time steps between variables. This is a key challenge in multivariate long-term series forecasting because long-term dependencies across variables refer to dependencies between arbitrary time steps. If they can not interact freely, it becomes impossible to model long-term dependencies across variables, thereby hindering accurate predictions.

To address this key challenge, we embed the entire variable sequence into the convolution's input channels, enabling the kernel to extract comprehensive semantic information from the variables. Thus, we could effectively model the long-term dependencies between variables by interacting arbitrary time step across variables. Formally, the convolution kernel is denoted as $kernel \in R^{L\times K}$, where L represents the number of input channels, which is equal to the length of input variable series.

Through the aforementioned convolution process, we could construct polytrees shown in Fig. 1(b), with input variables flowing upwards from the bottom of the tree layer by layer. The parent nodes of the tree are the merged features of their child nodes, representing the relationships between variables. The parent nodes of lower layers merging into upper layers could further integrate their features. This progressive strategy allows for the learning of variable relationships from local to global, and its strategy of free interaction captures long-term dependencies across variables. Thus, the polytrees we construct could model complex variable relationships more meticulously and accurately than existing methods. This construction process could be formally expressed as follows.

$$\boldsymbol{x}_{\text{par}} = Conv_1(\boldsymbol{x}_{\text{inp}}), \boldsymbol{x}_{\text{roo}} = Conv_{n-1}(\boldsymbol{x}_{\text{par}}), \tag{1}$$

where $\boldsymbol{x}_{\text{inp}}$, $Conv_1(\cdot)$, and $\boldsymbol{x}_{\text{par}}$ represent the input multivariate sequence, the 1st convolution, and the parent nodes, respectively. In addition, $\boldsymbol{x}_{\text{roo}}$ and $Conv_{n-1}(\cdot)$ represent the root nodes and the 2nd to n-th convolution, respectively. n is the number of polytree's layers. All convolutions are performed along the channel dimension with zero-padding, while their convolution kernel $kernel \in R^{L\times K}$, where L represents the number of input channels, which is equal to the length of input multivariate sequence. Thus, $\boldsymbol{x}_{\text{inp}}, \boldsymbol{x}_{\text{par}}, \boldsymbol{x}_{\text{roo}} \in R^{C\times L}$.

D. Forest Construction

As shown in Fig. 1(a), we set multiple convolutions ($Conv_{coarse}$, $Conv_{medium}$, $Conv_{fine}$) with kernel sizes ranging from small to large. Since the size of the convolution kernel represents the scope of learning, this arrangement allows for including learning granularity from coarse to fine. Consequently, as depicted in Fig. 1(b), we obtain multiple polytrees ($Polytree_{coarse}$, $Polytree_{medium}$, $Polytree_{fine}$), each with different in-degrees at their parent nodes. Due to this structural difference, these trees show preferences for processing different data features. By leveraging this and ensembling them into a forest, we accommodate diverse feature preferences. Thus, the forest is capable of gaining insights into the complex patterns hidden deep within high-dimensional variables.

The key challenge in forest construction is identifying the differences between the trees. Specifically, if the forest can not identify the differences among the trees, it will assign the

TABLE I
STATISTICS OF BENCHMARK DATASETS.

Datasets	Weather [7]	Traffic [7]	Electricity [7]	Solar [14]	PEMS03 [14]	PEMS04 [14]	PEMS07 [14]	PEMS08 [14]
Variates	21	862	321	137	358	307	883	170
Timesteps	52696	17544	26304	52560	26209	16992	28224	17856
Granularity	10min	1hour	1hour	10min	5min	5min	5min	5min

same level of confidence to all trees, thereby losing the ability to generalize and extract complex patterns. To address the key challenge, we employ a Transformer to weight and amplify the differences between the trees. We use it to compute the relevance of the root nodes and merge them into the forest.

As shown in Fig. 1(a), we first concatenate convolution's outputs (root nodes) along the time dimension, with a shape of $R^{C\times 3L}$. Then, we adopt channel-independent, that is, for the concatenated one, Transformer independently operates on the each variable and shares parameters. The shape of operation objective is $R^{1\times 3L}$. On each variable dimension, we split the concatenated root nodes and concatenate them along the channel dimension, with a shape of $R^{3\times L}$. Then we independently embed them as tokens $\boldsymbol{x}_{\text{tok}}^{(c)} \in R^{3\times D}$ by linear projection. Here, c represents $c\text{-}th$ variable and D is the embedding dimention.

We feed tokens into Transformer encoder, where the self-attention mechanism learns the correlations between root nodes at the same variable dimension, and the feed forward network learns the temporal dependencies within the root nodes. This process could be formally expressed as follows.

$$\boldsymbol{Q}_{\text{h}}^{(\text{c})} = \left(\boldsymbol{x}_{\text{tok}}^{(c)}\right)^T \boldsymbol{W}_{\text{h}}^{\text{Q}}, \boldsymbol{K}_{\text{h}}^{(\text{c})} = \left(\boldsymbol{x}_{\text{tok}}^{(c)}\right)^T \boldsymbol{W}_{\text{h}}^{\text{K}}, \boldsymbol{V}_{\text{h}}^{(\text{c})} = \left(\boldsymbol{x}_{\text{tok}}^{(c)}\right)^T \boldsymbol{W}_{\text{h}}^{\text{V}}, \tag{2}$$

where $\boldsymbol{Q}_{\text{h}}^{(\text{c})}$, $\boldsymbol{K}_{\text{h}}^{(\text{c})}$, and $\boldsymbol{V}_{\text{h}}^{(\text{c})}$ represent query, key, and value of $c\text{-}th$ variable, respectively; $\boldsymbol{W}_{\text{h}}^{\text{Q}}$, $\boldsymbol{W}_{\text{h}}^{\text{K}}$, and $\boldsymbol{W}_{\text{h}}^{\text{V}}$ is projection matric in multi-head attention space, respectively; h represents the $h\text{-}th$ head; $\boldsymbol{W}_{\text{h}}^{\text{Q}}, \boldsymbol{W}_{\text{h}}^{\text{K}}, \boldsymbol{W}_{\text{h}}^{\text{V}} \in R^{D\times d_k}$, where d_k is projection dimension.

$$\boldsymbol{O}_{\text{h}}^{(\text{c})} = Softmax\left(\frac{\boldsymbol{Q}_{\text{h}}^{(\text{c})}\left(\boldsymbol{K}_{\text{h}}^{(\text{c})}\right)^T}{\sqrt{d_k}}\right)\boldsymbol{V}_{\text{h}}^{(\text{c})}, \tag{3}$$

where $\boldsymbol{O}_{\text{h}}^{(\text{c})}$ and $Softmax$ represent the output of self-attention and the softmax operation, respectively.

$$\boldsymbol{x}_{\text{att}}^{(\text{c})} = LN(FFN(LN(\boldsymbol{Res}^{(\text{c})})) + LN(\boldsymbol{Res}^{(\text{c})})), \tag{4}$$

where $\boldsymbol{Res}^{(\text{c})} = \boldsymbol{O}_{\text{h}}^{(\text{c})} + \boldsymbol{x}_{\text{tok}}^{(\text{c})}$ and it is a residue connection; $\boldsymbol{x}_{\text{att}}^{(\text{c})}$, LN, and FFN represent the output of Transformer encoder, layer normalization, and feed-forward network, respectively.

Thus, we could construct a forest as shown in Fig. 2(b). For each variable dimension, the Transformer encoder assigns different weights to different trees, identifying the differences between them. Consequently, the forest could capture complex patterns hidden within high-dimensional variables by ensembling different feature processing preferences. The number of tokens equals the number of trees with different structures, rather than the number or length of the input variable sequences. The former is constant, while the latter is variable. In the deep learning framework, all variables are processed in parallel. Thus, the computational complexity of the employed Transformer encoder is $O(N)$.

Next, we flatten the learning results along the time dimension and obtain the predicted sequence through a linear layer. Finally, we concatenate the predicted sequences of each variable dimension along the variable dimension to produce the output result. The process could be formally expressed as follows.

$$\boldsymbol{x}_{\text{ens}}^{(\text{c})} = Flatten(\boldsymbol{x}_{\text{att}}^{(\text{c})}), \tag{5}$$

where $\boldsymbol{x}_{\text{ens}}^{(\text{c})}$ and $Flatten$ represent the ensembled root node information and flatten operation, respectively.

$$\boldsymbol{y}^{(c)} = Linear(\boldsymbol{x}_{\text{ens}}^{(\text{c})}), \boldsymbol{y} = Concat(\boldsymbol{y}^{(c)}), \tag{6}$$

where $\boldsymbol{y}^{(c)}$, $\boldsymbol{y}$, $Linear$, and $Concat$ represent the predicted $c\text{-}th$ variable series, predicted multivariate time series, linear layer, and concatenation, respectively. In addition, $\boldsymbol{y}^{(c)} \in R^{1\times T}$ and $\boldsymbol{y} \in R^{C\times T}$,where T is prediction length.

E. Loss Function

We adopt the MSE loss to constrain the predictions to be close to the ground truth. MSE loss is widely used in long-term series prediction task. Its formalized representation is as follows.

$$\mathcal{L} = \mathbb{E}_{\boldsymbol{x}}\frac{1}{C}\sum_{c=1}^{C}\left\|\boldsymbol{y}_{L+1:L+T}^{(c)} - \boldsymbol{x}_{L+1:L+T}^{(c)}\right\|_2^2, \tag{7}$$

where $\boldsymbol{x}_{L+1:L+T}^{(c)}$, $\boldsymbol{y}_{L+1:L+T}^{(c)}$, c, and C represent the single series including future T time steps $(x_{L+1}, \ldots, x_{L+T})$ in training set, single series including predicted T time steps, $c\text{-}th$ variate, and the number of variates, respectively; $c \in [1, C]$.

IV. EXPERIMENTS

A. Experimental Setup

1) Datasets: We extensively conduct experiments on 8 authoritative public datasets from high-dimensional variable scenarios of the real-world. They could comprehensively verify the effectiveness of the methods. TABLE I shows the statistics of these datasets. Due to the fact that ILI [7], Exchange-Rate [15], and ETT [6] have too few variables and do not belong to the high-dimensional variable scenario targeted in this paper, we do not select them.

2) Baselines: Our baselines are numerous high-impact methods of multivariate long-term forecasting, including Transformer-based methods [1], [7], [8], [10], [11], Linear-based methods [3]–[5], and CNN-based methods [2].

TABLE II
MULTIVARIATE LONG-TERM FORECASTING RESULTS. THE BEST RESULTS ARE IN BOLD AND THE SECOND BEST ARE UNDERLINED.

Models	ForeNet		MSGNet [11]		RLinear [4]		TiDE [5]		PatchTST [1]		TimesNet [2]		DLinear [3]		Crossformer [10]		FEDformer [8]		Autoformer [7]	
Venue	-		AAAI 24		ARXIV 24		ARXIV 24		ICLR 23		ICLR 23		AAAI 23		ICLR 23		ICML 22		NeurIPS 21	
Metric	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Electricity	**0.181**	**0.272**	0.194	0.300	0.219	0.298	0.252	0.344	0.216	0.318	0.193	0.295	0.212	0.300	0.244	0.334	0.214	0.327	0.227	0.338
Traffic	**0.468**	0.315	0.661	0.384	0.627	0.378	0.761	0.473	0.529	0.341	0.620	0.336	0.625	0.383	0.550	**0.304**	0.610	0.376	0.628	0.379
Weather	**0.244**	**0.274**	0.249	0.278	0.272	0.291	0.271	0.320	0.265	0.286	0.259	0.287	0.265	0.317	0.259	0.315	0.309	0.360	0.338	0.382
Solar	**0.252**	**0.280**	0.269	0.295	0.369	0.355	0.347	0.418	0.270	0.307	0.301	0.319	0.330	0.401	0.641	0.639	0.292	0.381	0.885	0.711
PEMS03	**0.141**	**0.246**	0.160	0.258	0.495	0.472	0.326	0.420	0.180	0.291	0.147	0.248	0.278	0.375	0.169	0.282	0.213	0.327	0.667	0.601
PEMS04	0.130	**0.236**	0.133	0.247	0.526	0.491	0.353	0.437	0.195	0.307	**0.129**	0.241	0.295	0.388	0.209	0.314	0.232	0.338	0.610	0.590
PEMS07	**0.123**	0.231	0.125	0.229	0.505	0.478	0.380	0.440	0.211	0.303	0.125	**0.226**	0.329	0.396	0.235	0.315	0.165	0.283	0.367	0.451
PEMS08	**0.169**	**0.258**	0.211	0.295	0.529	0.487	0.441	0.464	0.280	0.321	0.193	0.271	0.379	0.416	0.268	0.307	0.286	0.358	0.814	0.659

3) Implementation Details: ForeNet is implemented using PyTorch on two 2080Ti GPUs. We utilize the Adam optimizer with a learning rate set at 0.0005 for PEMS [14] or 0.0001 for others. The batch size is set at 32. Due to space limitations, please review the complete hyper-parameters and implementation details in our codes.

B. Quantitative Experiment

Following baselines [11] [2], the look-back window size L is set at 96, while the prediction window T is set within $\{12, 24, 48, 96\}$ for PEMS [14] and $\{96, 192, 336, 720\}$ for other datasets. We calculate the MSE and mean absolute error (MAE) and show the average results of each dataset in TABLE II.

Firstly, it is evident that ForeNet has the lowest overall error across 8 datasets, which broadly validates the effectiveness of the proposed progressive, ensemble, and interaction strategies. Secondly, compared to other methods that also facilitate variable interactions, ForeNet achieves lower MSE and MAE scores of 0.169 and 0.258 respectively on the PEM08 dataset, while MSGNet [11] records 0.211 and 0.295, and Crossformer [10] scores 0.268 and 0.307. ForeNet reduces MSE by about 19.91% compared to MSGNet and 36.94% compared to Crossformer. Similarly, ForeNet reduces MAE by about 12.54% compared to MSGNet and 15.96% compared to Crossformer. On the Traffic dataset, although ForeNet's MAE ranked second, its MSE decreased by 29.20% and 14.91% compared to MSGNet and Crossformer, respectively. These demonstrate that ForeNet models more accurate long-term dependencies across variables, thanks to the more effective structure of its polytrees and forest.

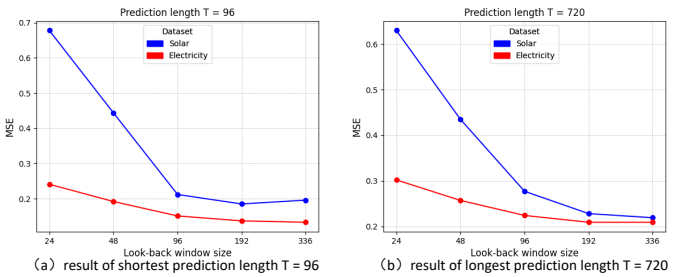


Fig. 2. MSE variation of the increasing look-back window sizes L.

Compared to channel-independent methods such as RLinear [4], TiDE [5], PatchTST [1], TimesNet [2], and DLinear [3], ForeNet on the Traffic dataset shows a decrease in MSE of 25.36%, 38.50%, 11.53%, 24.52%, and 25.12% respectively. Moreover, its MAE is also lower than these methods. This demonstrates that ForeNet's modeling of variable relationships is better equipped to handle high-dimensional variable scenarios, further validating its effectiveness.

In addition , we conduct an experiment of "MSE variation with increasing look-back window sizes L" to verify the long-term prediction ability of ForeNet. As shown in Fig. 2 (a) and (b), the MSE consistently decreases as L increases. The trend is most pronounced in the Solar dataset, with the Electricity dataset also showing stable decrease. Notably, in the Solar dataset of (a), a difference occurs at $L = 336$ where the MSE reverses trend, while no corresponding anomaly appears in (b). This is because for its data characteristics, excessively long inputs do not provide benefits for short predictions and can indeed be detrimental. The optimal limit for sequence length that ForeNet could utilize does not exceed 336. Conversely, a prediction of 720 steps could effectively use an input length of 336, where a moderate input length provides benefits for longer predictions.

Similarly, the characteristics of the Electricity data lead to the plateau observed at $L = 336$ in (b). In this scenario, the optimal sequence length that ForeNet could utilize does not exceed 336. These indicate that ForeNet effectively leverages longer input sequences to enhance its long-term forecasting accuracy. These also suggest that its enhancement has a boundary effect, and the optimal input length for ForeNet should be either 192 or 336.

C. Ablation Study

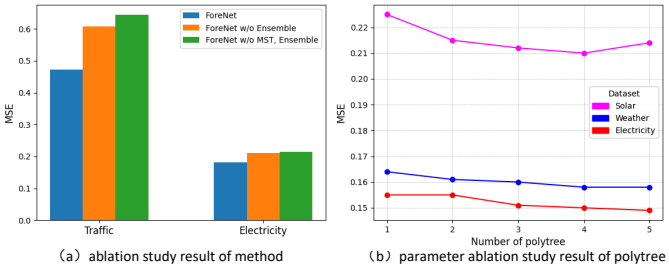


Fig. 3. Ablation study results. In (a), MST and Ensemble represent the multiple structure polytrees and ensembling them into a forest, respectively.

Fig. 3(a) visualizes the results of the ablation study, with the MSE averaged across all prediction lengths. Polytree is

ForeNet's basic working unit, which can not be reduced. The label "ForeNet" represents the method with its full structure intact, and it achieves the lowest MSE. "w/o Ensemble" indicates the absence of integration of trees with diverse structures into a forest, leading to a significant increase in MSE, especially noticeable on the Traffic dataset with more variables.

This demonstrates that the forest structure is superior and that our ensemble strategy is effective. Furthermore, "w/o MST, Ensemble" denotes a further reduction of the tree structure to a single type, resulting in another increase in MSE. This suggests that trees with multiple structures optimize predictions through different feature processing preferences.

To explore the optimal number of tree structures for the forest, we conducted a parameter ablation experiment as shown in Fig. 3(b). The overall MSE decreases steadily as the number of tree structures increases, reaching a bottleneck when it equals 5. This suggests that the optimal forest configuration consists of 4 or 5 different tree structures, and adding more may lead to overfitting and an increase in error.

TABLE III
OVERHEAD COMPARISON OF THE TRAINING TIME (S/EPOCH), PARAMETER NUMBER (M), AND GPU MEMORY UTILIZATION (GB). THE BEST RESULTS ARE IN BOLD AND THE SECOND BEST ARE UNDERLINED.

Models		ForeNet			MSGNet			TimesNet		
Venue		-			AAAI 24			ICLR 23		
Metric		Time	Param	GPU	Time	Param	GPU	Time	Param	GPU
Traffic	96	**51.4**	**1.8**	**7.8**	359.3	27.6	21.6	1278.6	301.7	16.9
	192	**54.8**	**1.9**	**7.8**	361.3	27.7	21.6	1629.6	301.7	18.2
	336	**60.1**	**1.9**	**8.0**	600.4	39.7	18.8	2014.4	301.7	14.1
	720	**72.0**	**2.1**	**8.3**	598.5	39.8	18.6	3064.0	301.8	15.4
	AVG	**59.6**	**1.9**	**8.0**	479.9	33.7	20.1	1996.7	301.7	16.1
Electricity	96	**39.0**	**1.1**	**4.3**	385.9	18.9	12.7	911.9	150.3	15.2
	192	**63.2**	**1.9**	**3.4**	386.0	18.9	12.7	1175.1	150.3	10.7
	336	**44.4**	**1.2**	**4.4**	567.9	27.7	16.7	1478.4	150.3	14.6
	720	**51.1**	**1.4**	**4.6**	566.6	27.8	16.7	2060.9	150.4	11.6
	AVG	**49.4**	**1.4**	**4.2**	476.6	23.3	14.7	1406.6	150.3	13.0

D. Resource Overhead Analysis

TABLE III shows a detailed comparison of the time-space overheads. It could be seen that ForeNet leads the comparison methods by an order of magnitude in all indicators, strongly verifying the high efficiency of ForeNet.

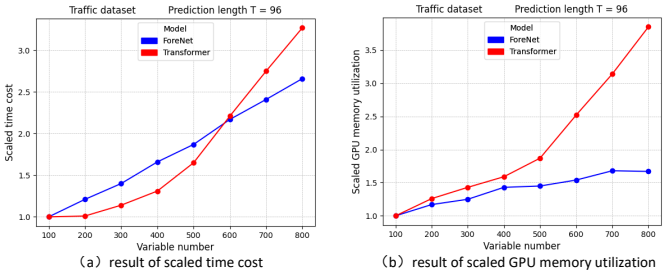


Fig. 4. Observation experiment on computational complexity. Transformer is exponential, while ForeNet is linear.

As shown in Fig. 4(a) and (b), we linearly increase the input variables and observe the corresponding growth rate of the model's space-time overheads. The original Transformer encoder, treating variables as tokens, exhibits a consistent exponential increase. In contrast, ForeNet shows a consistent linear trend, especially with a slower rise in spatial overheads. These results validate that ForeNet could reduce the computational complexity of the Transformer encoder to $O(N)$. Also, these show that ForeNet is more suitable for scenarios with high-dimensional variables.

V. CONCLUSION

To effectively modeling the long-term dependencies among high-dimensional variables, we propose an innovative forest network (ForeNet). By employing progressive, ensemble, and interaction strategies, ForeNet could capture long-term dependencies across variables and gain insights into the complex patterns hidden deep within high dimensional variables. In addition, the computational complexity of the Transformer encoder in ForeNet is $O(N)$. Extensive experiments show that ForeNet could reduce the average MSE by up to 11.53% compared to the baseline, which verifies the effectiveness of ForeNet.

REFERENCES

[1] Yuqi Nie, Nam H Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam, "A time series is worth 64 words: Long-term forecasting with transformers," *arXiv preprint arXiv:2211.14730*, 2022.

[2] Haixu Wu, Tengge Hu, Yong Liu, Hang Zhou, Jianmin Wang, and Mingsheng Long, "Timesnet: Temporal 2d-variation modeling for general time series analysis," *arXiv preprint arXiv:2210.02186*, 2022.

[3] Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu, "Are transformers effective for time series forecasting?," in *Proceedings of the AAAI conference on artificial intelligence*, 2023, vol. 37, pp. 11121–11128.

[4] Zhe Li, Shiyi Qi, Yiduo Li, and Zenglin Xu, "Revisiting long-term time series forecasting: An investigation on linear mapping," *arXiv preprint arXiv:2305.10721*, 2023.

[5] Abhimanyu Das, Weihao Kong, Andrew Leach, Shaan Mathur, Rajat Sen, and Rose Yu, "Long-term forecasting with tide: Time-series dense encoder," *arXiv preprint arXiv:2304.08424*, 2023.

[6] Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang, "Informer: Beyond efficient transformer for long sequence time-series forecasting," in *Proceedings of the AAAI conference on artificial intelligence*, 2021, vol. 35, pp. 11106–11115.

[7] Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long, "Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting," *Advances in neural information processing systems*, vol. 34, pp. 22419–22430, 2021.

[8] Tian Zhou, Ziqing Ma, Qingsong Wen, Xue Wang, Liang Sun, and Rong Jin, "Fedformer: Frequency enhanced decomposed transformer for long-term series forecasting," in *International conference on machine learning*. PMLR, 2022, pp. 27268–27286.

[9] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[10] Yunhao Zhang and Junchi Yan, "Crossformer: Transformer utilizing cross-dimension dependency for multivariate time series forecasting," in *The eleventh international conference on learning representations*, 2023.

[11] Wanlin Cai, Yuxuan Liang, Xianggen Liu, Jianshuai Feng, and Yuankai Wu, "Msgnet: Learning multi-scale inter-series correlations for multivariate time series forecasting," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2024, vol. 38, pp. 11141–11149.

[12] Taesung Kim, Jinhee Kim, Yunwon Tae, Cheonbok Park, Jang-Ho Choi, and Jaegul Choo, "Reversible instance normalization for accurate time-series forecasting against distribution shift," in *International Conference on Learning Representations*, 2021.

[13] Shizhan Liu, Hang Yu, Cong Liao, Jianguo Li, Weiyao Lin, Alex X Liu, and Schahram Dustdar, "Pyraformer: Low-complexity pyramidal attention for long-range time series modeling and forecasting," in *International conference on learning representations*, 2021.

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

[15] Guokun Lai, Wei-Cheng Chang, Yiming Yang, and Hanxiao Liu, "Modeling long-and short-term temporal patterns with deep neural networks," in *The 41st international ACM SIGIR conference on research & development in information retrieval*, 2018, pp. 95–104.