Partial-Multivariate Model for Forecasting

Jaehoon Lee[1] **Hankook Lee**[1,2] **Sungik Choi**[1] **Sungjun Cho**[1] **Moontae Lee**[1,3]
[1]LG AI Research [2]Sungkyunkwan University [3]University of Illinois Chicago

Abstract

When solving forecasting problems including multiple time-series features, existing approaches often fall into two extreme categories, depending on whether to utilize inter-feature information: univariate and complete-multivariate models. Unlike univariate cases which ignore the information, complete-multivariate models compute relationships among a complete set of features. However, despite the potential advantage of leveraging the additional information, complete-multivariate models sometimes underperform univariate ones. Therefore, our research aims to explore a middle ground between these two by introducing what we term *Partial-Multivariate* models where a neural network captures only partial relationships, that is, dependencies within subsets of all features. To this end, we propose PMformer, a Transformer-based partial-multivariate model, with its training algorithm. We demonstrate that PMformer outperforms various univariate and complete-multivariate models, providing a theoretical rationale and empirical analysis for its superiority. Additionally, by proposing an inference technique for PMformer, the forecasting accuracy is further enhanced. Finally, we highlight other advantages of PMformer: efficiency and robustness under missing features.

1 Introduction

Time-series forecasting is a fundamental machine learning task that aims to predict future events based on past observations, requiring to capture temporal dynamics. A forecasting problem often includes interrelated multiple variables (*e.g.*, multiple market values in stock price forecasting). For decades, the forecasting task with multiple time-series features has been of great importance in various applications such as health care (Nguyen et al., 2021; Jones et al., 2009), meteorology (Sanhudo et al., 2021; Angryk et al., 2020), and finance (Qiu et al., 2020; Mehtab, Sen, 2021).

For this problem, there have been developed a number of deep-learning models, including linear models (Chen et al., 2023a; Zeng et al., 2022), state-space models (Liang et al., 2024; Gu et al., 2022), recurrent neural networks (RNNs) (Lin et al., 2023b; Du et al., 2021), convolution neural networks (CNNs) (Wang et al., 2023; Liu et al., 2022a), and Transformers (Zhou et al., 2021; Liu et al., 2022b). These models are typically categorized by the existence of modules that capture inter-feature information, falling into two extremes: *(i)* univariate and *(ii)* complete-multivariate models[1]. While univariate models only capture temporal dependencies, complete-multivariate ones incorporate additional modules that account for complete dependencies among all given features.

However, although complete-multivariate models have potential advantages over univariate ones by additionally utilizing inter-feature information, some studies have demonstrated that complete-multivariate models are sometimes inferior to univariate ones in time-series forecasting. (Zeng et al., 2022; Zhang et al., 2023a; Xu et al., 2024) For example, in Table 1, PatchTST (a univariate

[1]To differentiate from existing multivariate models that capture dependencies among a complete set of features, we refer to them as complete-multivariate models, while our new approaches, which capture partial dependencies by sampling subsets of features, are termed partial-multivariate models.

Preprint. Under review.

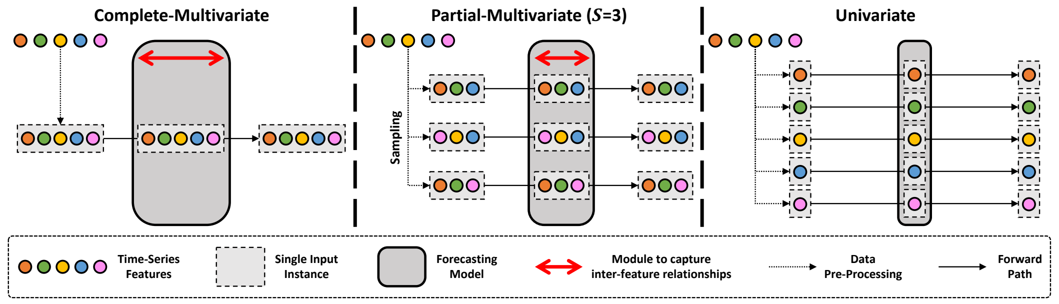


Figure 1: Visualization of three types of models. While the complete-multivariate model processes a complete set of features simultaneously taking into account their relationships, the univariate model, which treats each feature as separate inputs for a shared neural network, disregards relationships. However, in the partial-multivariate model, several subsets of size S are sampled from a complete feature set and relationships are captured only within each subset — note that a single neural network is shared by all sampled subsets.

Transformer-based model) (Nie et al., 2023) outperforms Crossformer (a complete-multivariate Transformer-based model) (Zhang, Yan, 2023) by a large margin.

Contribution. Our research goal is to explore a middle ground between these two extreme types, which we call a *Partial-Multivariate* model. Complete-multivariate models process all features simultaneously with an inter-feature module computing their complete relationships. On the other hand, typically in a univariate model, each feature is pre-processed into separate inputs, and a single neural network is shared by all features. (Nie et al., 2023; Wang et al., 2024; Lee et al., 2024) In contrast to the two extreme cases, our partial-multivariate model includes a single neural network that captures dependencies within subsets of features (*i.e.*, partial dependencies) and is shared by all sampled subsets. The differences among the three models are illustrated in Figure 1. To implement the concept of partial-multivariate models, we propose *Partial-Multivariate Transformer*, PMformer. Inspired by Nie et al. (2023), PMformer can capture any partial relationship with a shared Transformer by tokenizing features individually and computing attention maps for selected features. Additionally, we propose training algorithms for PMformer based on random sampling or partitioning, under a usual assumption that the prior knowledge on how to select subsets of features is unavailable.

In experiments, we demonstrate that PMformer outperforms existing complete-multivariate or univariate models, attaining the best forecasting accuracy against 20 baselines. To explain the superiority of our partial-multivariate model against the other two types, we provide a theoretical analysis based on McAllester's bound on generalization errors (McAllester, 1999), suggesting the following two reasons: *(i)* higher entropy in posterior distributions of hypotheses of partial-multivariate models than complete-multivariate ones and *(ii)* the increased training set size for partial-multivariate models against complete-multivariate and univariate ones. Our analysis is supported by our empirical results. On top of that, to improve forecasting performance further, we introduce a simple inference technique for PMformer based on the fact that the probability that a specific event occurs at least once out of several trials increases as the number of trials gets large. Finally, we show other useful properties of PMformer: efficient inter-feature attention costs against other Transformers including inter-feature attention modules, and robustness under missing features compared to complete-multivariate models. To sum up, our contributions are summarized as follows:

- To the best of our knowledge, for the first time, we introduce the novel concept of *Partial-Multivariate* models in the realm of time-series forecasting, devising Transformer-based PMformer. To train PMformer, we propose a train algorithm based on random sampling or partitioning.
- The abundant experimental results demonstrate that our PMformer achieves the best performance against 20 baselines. Furthermore, we provide a theoretical analysis for the excellence of our model and provide empirical results supporting this analysis.
- We propose an inference technique for PMformer to enhance forecasting accuracy further, inspired by the relationships between the number of trials and probabilities of sampling a specific subset at least once. At last, we discover some useful properties of PMformer against complete-multivaraite models: efficient inter-feature attention costs and robustness under missing features.

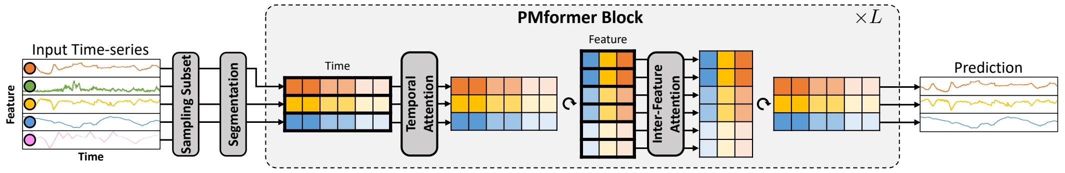


Figure 2: Architecture of Partial-Multivariate Transformer (PMformer). To emphasize row-wise attention operations, we enclose each row within bold frames before feeding them into the attention modules. In this figure, the subset size S is 3.

2 Related Works

To solve the forecasting problem with multiple features, it is important to discover not only temporal but also inter-feature relationships. As for inter-feature relationships, existing studies often aim to capture full dependencies among a complete set of features, which we call complete-multivariate models. For example, some approaches encode all features into a single hidden vector, which is then decoded back into feature spaces after some processes. This technique has been applied to various architectures, including RNNs (Che et al., 2016), CNNs (Bai et al., 2018), state-space models (Gu et al., 2022), and Transformers (Wu et al., 2022). Conversely, other complete-multivariate studies have developed modules to explicitly capture these relationships. For instance, Zhang, Yan (2023) computes $D \times D$ attention matrices among D features by encoding each feature into a separate token, while Wu et al. (2020) utilizes graph neural networks with graphs of inter-feature relationships. Additionally, Chen et al. (2023a) parameterizes a weight matrix $W \in \mathbb{R}^{D \times D}$, where each element in the i-th row and j-th column represents the relationship between the i-th and j-th features.

Unlike complete-multivariate models which fully employ inter-feature information, new models have recently been developed: univariate models. (Zeng et al., 2022; Xu et al., 2024; Nie et al., 2023; Wang et al., 2024; Lee et al., 2024) These models capture temporal dynamics but ignore the inter-feature information by processing each of D features separately as independent inputs. It is worth noting that univariate models usually include a single neural network shared by all features. Existing models typically either utilize inter-feature information among all features or ignore it entirely. In contrast, we propose a completely different approach to using inter-feature information, where a shared model captures relationships only within subsets of features, named partial-multivariate models.

3 Method

In this section, we first formulate partial-multivariate forecasting models. Subsequently, we introduce **Partial-Multivariate Transformer (PMformer)** with its training algorithm and inference technique. Finally, we provide a theoretical analysis to explain the superiority of PMformer.

3.1 Partial-Multivariate Forecasting Model

In this section, we provide the formulation of the partial-multivariate forecasting model. To simplify a notation, we denote the set of integers from N to M (inclusive of N and exclusive of M) as $[N : M]$ (*i.e.*, $[N : M] \coloneqq \{N, N+1, \ldots, M-1\}$). Also, when the collection of numbers is given as indices for vectors or matrices, it indicates selecting all indices within the collection. (*e.g.*, $x_{t=[0,T],d=[0,D]} \coloneqq \{\{x_{t,d}\}_{t\in[0:T]}\}_{d\in[0:D]}$). Let $\mathbf{x}_{t,d} \in \mathbb{R}$ the t-th observation of the d-th feature, and $\mathbf{x}_{[0:T],d}$ and $\mathbf{x}_{[T:T+\tau],d}$ the d-th feature's historical inputs and ground truth of future outputs with T and τ indicating the length of past and future time steps, respectively. Assuming D denotes the number of features, then a partial-multivariate forecasting model f is formulated as follows:

$$\hat{\mathbf{x}}_{[T:T+\tau],\mathbf{F}} = f(\mathbf{x}_{[0:T],\mathbf{F}}, \mathbf{F}), \qquad \mathbf{F} \in \mathcal{F}^{all} = \{\mathbf{F} | \mathbf{F} \subset [0:D], |\mathbf{F}| = S\}. \tag{1}$$

After sampling a subset $\mathbf{F}$ of size S from a complete set of features $[0 : D]$, a model f takes feature indices in $\mathbf{F}$ and their historical observations $\mathbf{x}_{[0:T],\mathbf{F}}$ as input, forecasting the future of the selected features $\hat{\mathbf{x}}_{[T:T+\tau],\mathbf{F}}$. In this formulation, a single model f is shared by any $\mathbf{F} \in \mathcal{F}^{all}$. In Transformer-based models, $\mathbf{F}$ is typically encoded using positional encoding schemes. It is worth noting that from this formulation, univariate and complete-multivariate can be represented by extreme cases of S where $S = 1$ and $S = D$, respectively. Our research goal is to explore the middle ground with partial-multivariate models where $1 < S < D$.

3.2 PMformer

For complete-multivariate cases, the architectures are required to capture perpetually unchanging (*i.e.*, static) relationships among a complete set of features. In other words, $\mathbf{F}$ in equation (1) is always the same as $[0:D]$. However, for partial-multivariate cases, $\mathbf{F}$ can vary when re-sampled, requiring to ability to deal with dynamic relationships. Therefore, inspired by recent Transformer-based models using segmentation (Nie et al., 2023; Zhang, Yan, 2023), we devise PMformer which addresses this problem by encoding each feature into individual tokens and calculating attention maps only with the feature tokens in $\mathbf{F}$. The overall architecture is illustrated in Figure 2.

After sampling $\mathbf{F}$ in equation (1), the historical observations of selected features $\mathbf{x}_{[0:T],\mathbf{F}} \in \mathbb{R}^{T\times S}$ are encoded into latent tokens $\mathbf{h}^{(0)} \in \mathbb{R}^{N_S\times S\times d_h}$ via a segmentation process where N_S is the number of segments and d_h is hidden size. The segmentation process is formulated as follows:

$$\mathbf{h}^{(0)}_{b,i} = \texttt{Linear}(\mathbf{x}_{[\frac{bT}{N_S}:\frac{(b+1)T}{N_S}],\mathbf{F}_i}) + \mathbf{e}^{Time}_{b} + \mathbf{e}^{Feat}_{\mathbf{F}_i}, \quad b\in[0,N_S], \quad i\in[0,S], \tag{2}$$

where $\mathbf{F}_i$ denotes the i-th element in $\mathbf{F}$. A single linear layer maps observations into latent space with learnable time-wise and feature-wise positional embeddings, $\mathbf{e}^{Time} \in \mathbb{R}^{N_S\times d_h}$ and $\mathbf{e}^{Feat} \in \mathbb{R}^{D\times d_h}$. In most scenarios, we can reasonably assume the input time span T to be divisible by N_S by adjusting T during data pre-processing or padding with zeros as in Zhang, Yan (2023) and Nie et al. (2023).

Subsequently, $\mathbf{h}^{(0)}$ is processed through L PMformer blocks. Each block is formulated as follows:

$$\bar{\mathbf{h}}^{(\ell-1)} = \mathbf{h}^{(\ell-1)} + \texttt{Feature-Attention}(\mathbf{h}^{(\ell-1)}, \texttt{Temporal-Attention}(\mathbf{h}^{(\ell-1)})), \tag{3}$$

$$\mathbf{h}^{(\ell)} = \bar{\mathbf{h}}^{(\ell-1)} + \texttt{MLP}(\bar{\mathbf{h}}^{(\ell-1)}), \quad \ell = 1,\dots,L. \tag{4}$$

MLP in equation (4) operates both feature-wise and time-wise, resembling the feed-forward networks found in the original Transformer (Vaswani et al., 2017). As shown in equation (3), there are two types of attention modules:

$$\forall i \in [0:S], \quad \texttt{Temporal-Attention}(\mathbf{h})_{[0:N_S],i} = \texttt{MHSA}(\mathbf{h}_{[0:N_S],i}, \mathbf{h}_{[0:N_S],i}, \mathbf{h}_{[0:N_S],i}), \tag{5}$$

$$\forall b \in [0:N_S], \quad \texttt{Feature-Attention}(\mathbf{h},\mathbf{v})_{b,[0:S]} = \texttt{MHSA}(\mathbf{h}_{b,[0:S]}, \mathbf{h}_{b,[0:S]}, \mathbf{v}_{b,[0:S]}). \tag{6}$$

$\texttt{MHSA}(\mathbf{Q},\mathbf{K},\mathbf{V})$ denotes the multi-head self-attention layer like in Vaswani et al. (2017) where $\mathbf{Q}, \mathbf{K}$, and $\mathbf{V}$ are queries, keys and values. While temporal attention is responsible for capturing temporal dependencies, feature attention mixes representations among features in $\mathbf{F}$.

Starting with initial representations $\mathbf{h}^{(0)}$, PMformer encoder with L blocks generates final representations $\mathbf{h}^{(L)}$. These representations are then passed through a decoder to forecast future observations. Similar to Nie et al. (2023), the concatenated representations $\mathbf{h}^{(L)}_{[0:N_S],i}$ are mapped to future observations $\mathbf{x}_{[T,T+\tau],\mathbf{F}_i}$ via a single linear layer.

3.3 Training Algorithm for PMformer

To train PMformer, the process to sample $\mathbf{F}$ from a complete set of features is necessary. Ideally, the sampling process would select mutually correlated features. However, prior knowledge about the relationships between features is usually unavailable. Therefore, we use random sampling where $\mathbf{F}$ is sampled fully at random, leading to training Algorithm 1 with $use\_random\_partition = False$ where N_U is the number of sampled subsets per iteration — note that for-loop in while-loop can be dealt with parallelly with attention masking techniques. However, this training algorithm may result in redundancy or omission in each iteration, as some features might be selected multiple times while others might never be chosen across the N_U trials. To address this issue, we propose a training algorithm based on random partitioning (see Algorithm 1 with $use\_random\_partition = True$). In this algorithm, D features are partitioned into

Algorithm 1: Training Algorithm 1

```
Input: # of features $D$, # of subsets $N_U$,
       Past obs. $\mathbf{x}_{[0:D]}$, Future obs. $\mathbf{y}_{[0:D]}$
while is_converge do
    Sample all $\mathbf{F}(g)$ with random partition;
    for $g \leftarrow 0$ to $N_U - 1$ do
        if use_random_partition then
            $\mathbf{F} = \mathbf{F}(g)$
        else
            Sample $\mathbf{F}$ from [0:D]
        $\hat{\mathbf{y}}_{\mathbf{F}} = \texttt{PMformer}(\mathbf{x}_{\mathbf{F}}, \mathbf{F})$;
        $\text{Loss}_g = \texttt{MSE}(\hat{\mathbf{y}}_{\mathbf{F}}, \mathbf{y}_{\mathbf{F}})$;
    $\text{Loss} = \sum_{g\in[0:N_U]} \text{Loss}_g / N_U$;
    Train PMformer with Loss;
return Trained PMformer
```

$N_U = D/S$ disjoint subsets $\{\mathbf{F}(g)\}_{g\in[0:N_U]}$ [2] where $\mathbf{F}(g) \subset [0:D], |\mathbf{F}(g)| = S, \bigcap_{g\in[0:N_U]} \mathbf{F}(g) = \phi, \bigcup_{g\in[0:N_U]} \mathbf{F}(g) = [0:D]$. This scheme can minimize the redundancy and omission of features in each iteration. We adopt the training algorithm based on random partitioning as our main training algorithm. Appendix E provides a comparison of these two algorithms in empirical experiments.

3.4 Inference Technique for PMformer

After training PMformer, we can measure inference score using Algorithm 1 without line 11 where $use\_random\_partition = True$. During inference time, it is important to group mutually significant features together. Attention scores among all features can provide information on feature relationships, but evaluating these scores results in a high computational cost, $\mathcal{O}(D^2)$. To avoid this, we propose a simple inference technique that doesn't require such expensive computations. We evaluate future predictions by repeating the inference process based on random partitioning N_I times and averaging N_I outputs to obtain the final outcomes — note that the inference process based on random partitioning achieves efficient costs because of computing attention within subsets of size S.

This technique relies on the principle that the probability of a specific event occurring at least once out of N_I trials increases as N_I grows. Let $P(\mathbf{F}_*) = p$ be the probability that we sample a specific subset $\mathbf{F}_*$ from all possible cases. The probability of sampling $\mathbf{F}_*$ at least once out of N_I trials is $1-(1-p)^{N_I}$. Given that $0 \le p \le 1$, $1-(1-p)^{N_I}$ increases as N_I increases. By treating a specific subset $\mathbf{F}_*$ as one that includes mutually significant features, our inference technique with a large N_I increases the likelihood of selecting a subset including highly correlated features at least once, thereby improving forecasting performance. This is supported by our empirical results in Section 4.3.

3.5 Theoretical Analysis on PMformer

In this section, we provide theoretical reasons for superiority of our partial-multivariate models over complete-multivariate and univariate ones, based on PAC-Bayes framework, similar to other works (Woo et al., 2023; Amit, Meir, 2019; Valle-Pérez, Louis, 2020). Let a neural network f be a partial-multivariate model which samples subsets $\mathbf{F}$ of S size from a complete set of D features as defined in equation (1). Also, $\mathcal{T}$ is a training dataset which consists of m instances sampled from the true data distribution. $\mathcal{H}$ denotes the hypothesis class of f with $\mathbf{P}(h)$ and $\mathbf{Q}(h)$ representing the prior and posterior distributions over the hypotheses h, respectively. Then, based on McAllester (1999), the generalization bound of a partial-multivariate model f is given by:

Theorem 1. *Under some assumptions, with probability at least $1-\delta$ over the selection of the sample $\mathcal{T}$, we have the following for generalized loss $l(\mathbf{Q})$ under posterior distributions $\mathbf{Q}$.*

$$l(\mathbf{Q}) \le \sqrt{\frac{-H(\mathbf{Q}) + \log\frac{1}{\delta} + \frac{5}{2}\log m + 8 + C}{2m-1}}, \tag{7}$$

where $H(\mathbf{Q})$ is the entropy of $\mathbf{Q}$, (i.e., $H(\mathbf{Q}) = E_{h\sim\mathbf{Q}}[-\log \mathbf{Q}(h)]$) and C is a constant.

In equation (7), the upper bound depends on m and $-H(\mathbf{Q})$, both of which are related to S. Selecting subsets of S size from D features leads to $|\mathcal{F}^{all}| = \binom{D}{S}$ possible cases, affecting m (*i.e.*, $m \propto |\mathcal{F}^{all}|$), where $\mathcal{F}^{all}$ is a pool including all possible $\mathbf{F}$. This is because each subset is regarded as a separate instance as in Figure 1. Also, the following theorem reveals relationships between S and $-H(\mathbf{Q})$:

Theorem 2. *Let $H(\mathbf{Q}_S)$ be the entropy of a posterior distribution $\mathbf{Q}_S$ with subset size S. For S_+ and S_- satisfying $S_+ > S_-$. $H(\mathbf{Q}_{S_+}) \le H(\mathbf{Q}_{S_-})$.*

Theorem 2 is intuitively connected to the fact that capturing dependencies within large subsets of size S_+ is usually harder tasks than the case of small S_-, because more relationships are captured in the case of S_+. As such, the region of hypotheses that satisfies conditions for such hard tasks would be smaller than the one that meets the conditions for a simple task. In other words, probabilities of a posterior distribution $\mathbf{Q}_{S_+}$ might be centered on a smaller region of hypotheses than $\mathbf{Q}_{S_-}$, leading to decreasing the entropy of $\mathbf{Q}_{S_+}$. We refer the readers to Appendix A for full proofs and assumptions for the theorems.

[2]We assume that D is divisible by S. If not, we can handle such cases by repeating some features, as explained in Appendix B.

Given the unveiled impacts of S on m and $-H(\mathbf{Q})$, we can estimate S_* which is S leading to the lowest upper-bound. When considering only the influence of m, S_* is $D/2$, resulting in the largest $|\mathcal{F}^{all}| = \binom{D}{S}$. On the other hand, considering only that of $-H(\mathbf{Q})$, S_* is 1, because $-H(\mathbf{Q})$ decrease as S decreases. Therefore, considering both effects simultaneously, we can think $1 < S_* < D/2$, which means partial-multivariate models ($1 < S < D$) are better than univariate models ($S = 1$) and complete-multivariate ($S = D$) and the best S_* is between 1 and $D/2$. This analysis is supported by our empirical experimental results in Section 4.3. As of now, since we do not evaluate $H(\mathbf{Q})$ exactly, we cannot compare the magnitudes of effects by m and $-H(\mathbf{Q})$, leaving it for future work. Nevertheless, our analysis from the sign of correlations between S and two factors in the upper-bound still is of importance in that it aligns with our empirical observations.

4 Experiments

4.1 Experimental Setup

We mainly follow the experiment protocol of previous forecasting tasks with multiple features, (Zhou et al., 2021). A detailed description of datasets, baselines, and hyperparameters is in Appendix C.

Datasets. We evaluate PMformer and other methods on the seven real-world datasets with $D > 1$: (*i*-*iv*) ETTh1, ETTh2, ETTm1, and ETTm2 ($D = 7$), (*v*) Weather ($D = 21$), (*vi*) Electricity ($D = 321$), and (*vii*) Traffic ($D = 862$). For each dataset, four settings are constructed with different forecasting lengths τ, which is in $\{96, 192, 336, 720\}$.

Baselines. Various complete-multivariate and univariate models are included in our baselines. For complete-multivariate baselines, we use Crossformer (Zhang, Yan, 2023), FEDformer (Zhou et al., 2022), Pyraformer (Liu et al., 2022b), Informer (Zhou et al., 2021), TSMixer (Chen et al., 2023a), MICN (Wang et al., 2023), TimesNet (Wu et al., 2023), and DeepTime (Woo et al., 2023). On the other hand, univariate ones include NLinear, DLinear (Zeng et al., 2022), and PatchTST (Nie et al., 2023). Furthermore, we compare PMformer against concurrent complete-multivariate (ModernTCN (donghao, xue wang, 2024), JTFT (Chen et al., 2023b), GCformer (Zhao et al., 2023), CARD (Xue et al., 2023), Client (Gao et al., 2023), and PETformer (Lin et al., 2023a)) and univariate models (FITS (Xu et al., 2024), PITS (Lee et al., 2024), and TimeMixer (Wang et al., 2024)). According to code accessibility or fair experimental setting with ours, we select these concurrent models.

Other settings. PMformer is trained with mean squared error (MSE) between ground truth and forecasting outputs. Also, we use MSE and mean absolute error (MAE) as evaluation metrics, and mainly report MSE. The MAE scores of experimental results are available in Appendix G.1. After training each method with five different random seeds, we measure the scores of evaluation metrics in each case and report the average scores. For the subset size S, we use $S = 3$ for ETT datasets, $S = 7$ for Weather, $S = 30$ for Electricity, and $S = 20$ for Traffic, satisfying $1 < S < D/2$. Also, for the inference technique of PMformer, we set N_I to 3.

4.2 Forecasting Result

Table 1 shows the test MSE of representative baselines along with the PMformer. PMformer outperforms univariate and complete-multivariate baselines in 27 out of 28 tasks and achieves the second place in the remaining one. We also provide visualizations of forecasting results of PMformer and some baselines in Appendix G.2, which shows the superiority of PMformer. On top of that, PMformer is compared to the nine concurrent baselines in Table 2. PMformer shows top-1 performance in 10 cases and top-2 in 12 cases out of 12 cases, attaining a 1.167 average rank. The scores in Table 1 and 2 are measured with $N_I = 3$, and in Appendix F, we provide another forecasting result which shows that our PMformer still outperforms other baselines even with $N_I = 1$.

4.3 Analysis

In this section, we provide some analysis on our PMformer. We refer the readers to Appendix G for additional experimental results.

Table 1: MSE scores of main forecasting results. The best score in each experimental setting is in boldface and the second best is underlined.

Data		Partial-Multivariate	Univariate			Complete-Multivariate							
		PMformer	PatchTST	Dlinear	Nlinear	Crossformer	FEDformer	Informer	Pyraformer	TSMixer	DeepTime	MICN	TimesNet
ETTh1	τ=96	**0.361**	<u>0.370</u>	0.375	0.374	0.427	0.376	0.941	0.664	**0.361**	0.372	0.828	0.465
	192	**0.396**	0.413	0.405	0.408	0.537	0.423	1.007	0.790	<u>0.404</u>	0.405	0.765	0.493
	336	**0.400**	0.422	0.439	0.429	0.651	0.444	1.038	0.891	<u>0.420</u>	0.437	0.904	0.456
	720	**0.412**	0.447	0.472	<u>0.440</u>	0.664	0.469	1.144	0.963	0.463	0.477	1.192	0.533
ETTh2	96	**0.269**	<u>0.274</u>	0.289	0.277	0.720	0.332	1.549	0.645	<u>0.274</u>	0.291	0.452	0.381
	192	**0.323**	0.341	0.383	0.344	1.121	0.407	3.792	0.788	<u>0.339</u>	0.403	0.554	0.416
	336	**0.317**	<u>0.329</u>	0.448	0.357	1.524	0.400	4.215	0.907	0.361	0.466	0.582	0.363
	720	**0.370**	0.379	0.605	0.394	3.106	0.412	3.656	0.963	0.445	0.576	0.869	<u>0.371</u>
ETTm1	96	**0.282**	0.293	0.299	0.306	0.336	0.326	0.626	0.543	<u>0.285</u>	0.311	0.406	0.343
	192	**0.325**	0.333	0.335	0.349	0.387	0.365	0.725	0.557	<u>0.327</u>	0.339	0.500	0.381
	336	**0.352**	0.369	0.369	0.375	0.431	0.392	1.005	0.754	<u>0.356</u>	0.366	0.580	0.436
	720	<u>0.401</u>	0.416	0.425	0.433	0.555	0.446	1.133	0.908	0.419	**0.400**	0.607	0.527
ETTm2	96	**0.160**	0.166	0.167	0.167	0.338	0.180	0.355	0.435	<u>0.163</u>	0.165	0.238	0.218
	192	**0.213**	0.223	0.224	0.221	0.567	0.252	0.595	0.730	<u>0.216</u>	0.222	0.302	0.282
	336	**0.262**	0.274	0.281	0.274	1.050	0.324	1.270	1.201	<u>0.268</u>	0.278	0.447	0.378
	720	**0.336**	<u>0.361</u>	0.397	0.368	2.049	0.410	3.001	3.625	0.420	0.369	0.549	0.444
Weather	96	**0.142**	0.149	0.176	0.182	0.150	0.238	0.354	0.896	<u>0.145</u>	0.169	0.188	0.179
	192	**0.185**	0.194	0.220	0.225	0.200	0.275	0.419	0.622	<u>0.191</u>	0.211	0.231	0.230
	336	**0.235**	0.245	0.265	0.271	0.263	0.339	0.583	0.739	<u>0.242</u>	0.255	0.280	0.276
	720	**0.305**	0.314	0.323	0.338	<u>0.310</u>	0.389	0.916	1.004	0.320	0.318	0.358	0.347
Electricity	96	**0.125**	<u>0.129</u>	0.140	0.141	0.135	0.186	0.304	0.386	0.131	0.139	0.177	0.186
	192	**0.142**	<u>0.147</u>	0.153	0.154	0.158	0.197	0.327	0.386	0.151	0.154	0.195	0.208
	336	**0.154**	0.163	0.169	0.171	0.177	0.213	0.333	0.378	<u>0.161</u>	0.169	0.213	0.210
	720	**0.176**	<u>0.197</u>	0.203	0.210	0.222	0.233	0.351	0.376	<u>0.197</u>	0.201	0.204	0.231
Traffic	96	**0.345**	<u>0.360</u>	0.410	0.410	0.481	0.576	0.733	2.085	0.376	0.401	0.489	0.599
	192	**0.370**	<u>0.379</u>	0.423	0.423	0.509	0.610	0.777	0.867	0.397	0.413	0.493	0.612
	336	**0.385**	<u>0.392</u>	0.436	0.435	0.534	0.608	0.776	0.869	0.413	0.425	0.496	0.618
	720	**0.426**	<u>0.432</u>	0.466	0.464	0.585	0.621	0.827	0.881	0.444	0.462	0.520	0.654
Avg. Rank		**1.036**	2.893	5.357	5.071	8.036	7.821	11.429	11.286	<u>2.821</u>	4.607	9.000	8.179

Table 2: Test MSE of PMformer compared to concurrent models.

Method		ETTm2				Weather				Electricity				Avg. Rank
		τ=96	192	336	720	96	192	336	720	96	192	336	720	
Partial-Multivariate	PMformer	<u>0.160</u>	**0.213**	**0.262**	**0.336**	**0.142**	**0.185**	**0.235**	**0.305**	**0.125**	<u>0.142</u>	**0.154**	**0.176**	**1.167**
Univariate	PITS	0.163	0.215	<u>0.266</u>	<u>0.342</u>	0.154	0.191	0.245	0.309	0.132	0.147	0.162	0.199	6.000
	FITS	0.164	0.217	0.269	0.347	0.145	0.188	<u>0.236</u>	0.308	0.135	<u>0.142</u>	0.163	0.200	5.083
	TimeMixer	0.164	0.223	0.279	0.359	0.147	0.189	0.241	0.310	0.129	**0.140**	0.161	0.194	6.083
Complete-Multivariate	JTFT	0.164	0.219	0.272	0.353	<u>0.144</u>	<u>0.186</u>	0.237	<u>0.307</u>	0.131	0.144	<u>0.159</u>	0.186	4.333
	GCformer	0.163	0.217	0.268	0.351	0.145	0.187	0.244	0.311	0.132	0.152	0.168	0.214	6.083
	CARD	**0.159**	<u>0.214</u>	<u>0.266</u>	0.379	0.145	0.187	0.238	0.308	0.129	0.154	0.161	<u>0.185</u>	<u>3.917</u>
	Client	0.167	0.220	0.268	0.356	0.153	0.195	0.246	0.314	0.131	0.153	0.170	0.200	8.250
	PETformer	<u>0.160</u>	0.217	0.274	0.345	0.146	0.190	0.241	0.314	<u>0.128</u>	0.144	<u>0.159</u>	0.195	5.000
	ModernTCN	0.166	0.222	0.272	0.351	0.149	0.196	0.238	0.314	0.129	0.143	0.161	0.191	6.250

Empirical result supporting the theoretical analysis. In Section 3.5, we think that S_* leading to the best forecasting performance is between 1 and $D/2$. To validate this analysis, we provide Table 3, which shows that partial-multivariate settings ($1 < S < D$) outperform others with $S = 1$ or D, in most cases. On top of that, our analysis is further supported by the U-shaped plots in Figure 3 where the best MSE is achieved when $1 < S < D/2$ and the worst one is in $S \in \{1, D\}$.

On top of that, we conduct another experiment in Figure 4 where we adjust the size of subsets' pool ($|\mathcal{F}^{all}|$) while fixing S. For training of original PMformer, $\mathcal{F}^{all}$ consists of all possible subsets, leading to $|\mathcal{F}^{all}| = \binom{D}{S}$. However, for this experiment, we limit $|\mathcal{F}^{all}|$ into $\alpha \times N_U$ by randomly removing some subsets from all possible cases where N_U is the number of sampling a subset in each iteration and $\alpha \in \{1, 400, 1600, 6400, \text{Max}\}$. 'Max' denotes α leading to $|\mathcal{F}^{all}| = \binom{D}{S}$. Figure 4 shows that as $|\mathcal{F}^{all}|$ increases, forecasting performance improves. These experimental results align with our theoretical analyses that $|\mathcal{F}^{all}|$ is proportional to a training set size m and large m leads to low upper-bounds on generalization errors.

Analysis on the inference technique. In Section 3.4, we think that large N_I (the repeating number of the inference process based on random partitioning) would improve forecasting results by increasing the probabilities that sampled subsets include mutually significant features at least once out of N_I trials. Figure 5(a) is aligned with our thought, showing monotonically decreasing test MSE as N_I

Table 3: Comparison among three types of models by adjusting S in PMformer.

PMformer Variants		ETTh2 ($D = 7$)				Weather ($D = 21$)				Electricity ($D = 321$)				Traffic ($D = 862$)			
		τ=96	192	336	720	96	192	336	720	96	192	336	720	96	192	336	720
Univariate	$S = 1$	0.272	0.325	0.318	0.374	**0.141**	0.186	0.237	0.308	0.128	0.146	0.163	0.204	0.368	0.388	0.404	0.441
Partial-Multivariate	$1 < S < D$	**0.269**	**0.323**	**0.317**	**0.370**	0.142	**0.185**	**0.235**	**0.305**	**0.125**	**0.142**	**0.154**	**0.176**	**0.345**	**0.370**	**0.385**	**0.426**
Complete-Multivariate	$S = D$	**0.269**	0.325	0.318	0.371	0.146	0.192	0.244	0.307	0.129	0.147	0.163	0.204	0.363	0.383	0.394	0.441

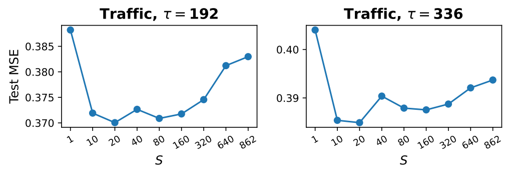


Figure 3: Test MSE by changing S.

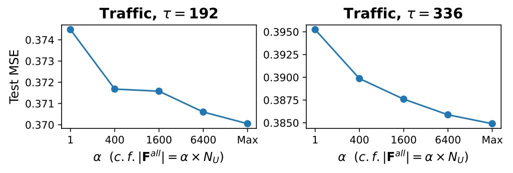
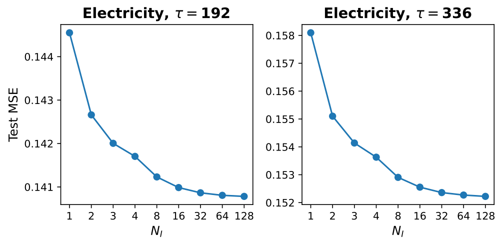
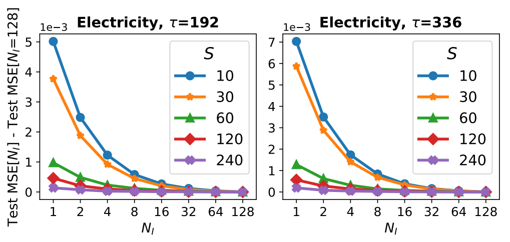


Figure 4: Test MSE by changing $|\mathbf{F}^{all}|$, fixing S.



(a) Sensitivity to N_I



(b) Changes in the effect of N_I when S increases

Figure 5: The effect of N_I on test MSE when (a) S is fixed to the selected hyperparameter and (b) S changes. For (b), the y axis shows the difference of test MSE between when $N_E \in \{1, 2, 4, 8, 16, 32, 64, 128\}$ and $N_E = 128$.

Table 4: Test MSE of PMformer with various inference techniques — note that all variants of PMformer are trained with the same algorithms as ours. To identify relevance (significance) of features to others, we utilize attention scores.

Inference Technique	Electricity ($D = 321$)				Traffic ($D = 862$)			
	τ=96	192	336	720	96	192	336	720
Proposed Technique with $N_I = 3$ (Ours)	**0.125**	**0.142**	**0.154**	**0.176**	**0.345**	**0.370**	**0.385**	**0.426**
Sampling A Subset of Mutually Significant Features	0.132	0.148	0.178	0.205	0.352	0.372	0.386	0.428
Sampling A Subset of Mutually Insignificant Features	0.135	0.167	0.174	0.235	0.377	0.410	0.410	0.444

gets large. In Figure 5(b), we investigate relationships between the feature subset size S and N_I by measuring performance gain by increasing N_I in various S. This figure shows that the effect of increasing N_I tends to be smaller, as S increases. We think this is because a single subset $\mathbf{F}$ with large S can contain a number of features, so mutually significant features can be included in such large subsets at least once only with few repetitions.

Besides the inference technique based on random selection, we explore another technique which samples subsets of mutually important features by selecting some keys with the highest attention scores per query. We compare this technique to the counterpart which selects keys based on the lowest attention score. In Table 4, we provide the forecasting MSE of each inference technique. — note that only the inference method is different while the training algorithm remains the same as the original one in Algorithm 1. In that an inference technique utilizing the highest attention scores outperforms one with the lowest ones, attention scores are helpful in identifying relationships between features to some extent. However, our proposed method based on random partitioning achieves the best forecasting performance. Furthermore, identifying relationships between features requires high-cost attention computation which calculates full attention between D features, leading to $\mathcal{O}(D^2)$. On the other hand, our proposed inference technique doesn't incorporate such high-cost computations but just repeats low-cost ones N_I times, each of which has $O(SD)$ costs for computing inter-feature relationships — note that $S < D/2$. In Appendix D, we elaborate on the details of why PMformer achieves $\mathcal{O}(SD)$. Therefore, against the inference technique with information of attention scores, our proposed one with random partitioning is superior in terms of efficiency and forecasting accuracy.

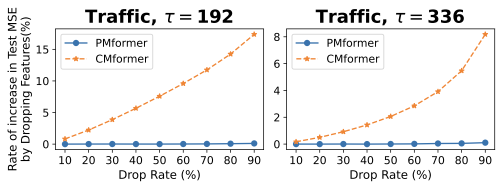


Figure 6: Increasing rate of test MSE by dropping $n\%$ features in PMformer or Complete-Multivariate Transformer (CMformer).

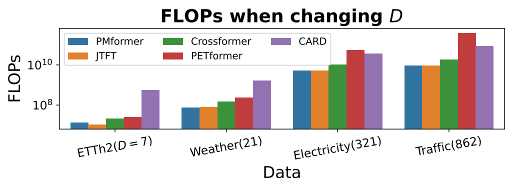


Figure 7: FLOPs of self-attention for inter-feature dependencies in various Transformers when changing D.

Other Advantages of PMformer. In the real world, some features in time series are often missing. Inspired by the works that address irregular time series where observations at some time steps (Che et al., 2016; Kidger et al., 2020) are missing, we randomly drop some features of input time series in the inference stage and measure the increasing rate of test MSE in undropped features. For comparison, we use the original PMformer and a complete-multivariate version of PMformer (CMformer) by setting S to D. PMformer can address the missingness by simply excluding missing features in the random sampling process, while CMformer has no choice but to pad dropped features with zeros. In Figure 6, unlike the other case, PMformer maintains its forecasting performance, regardless of the drop rate of the features. This robust characteristic gives PMformer more applicability in real-world situations where some features are not available.

For Transformers with inter-feature attention modules (PMformer, Crossformer, JTFT, PETformer, and CARD), we compare the costs of their inter-feature modules using floating point operations (FLOPs) in Figure 7. When naïvely computing inter-feature attention like PETformer, the attention cost is $\mathcal{O}(D^2)$ where D is the number of features. On the other hand, due to capturing only partial relationships, the attention cost of PMformer is reduced to $\mathcal{O}(SD)$ where S is the size of each subset. In Appendix D, we elaborate on the details of the reason why the inter-feature module in PMformer achieves $\mathcal{O}(SD)$. Given that small S is enough to generate good forecasting performance (*e.g.*, S = 20∼30 for 300∼800 features), the attention cost is empirically efficient. As a result, PMformer achieves the lowest or the second lowest FLOPs compared to others, as shown in Figure 7. Although Crossformer, JTFT, and CARD achieve $O(RD)$ complexities of inter-feature attention with low-rank approximations where R is the rank, our PMformer shows quite efficient costs, compared to them.

5 Conclusion

Various models have been developed to address the forecasting problem with multiple variables. However, most studies focus on two extremes: univariate or complete-multivariate models. To explore the middle ground between them, our research introduces a novel concept, partial-multivariate models, devising PMformer. PMformer captures dependencies only within subsets of a complete feature set using a single inter-feature attention module shared by all subsets. To train PMformer under usual situations without prior knowledge on subset selection, we propose a training algorithm based on random sampling or partitioning. Extensive experiments show that PMformer outperforms 20 baseline models. To explain PMformer's superior performance, we theoretically analyze the upper-bound on generalization errors of PMformer compared to complete-multivariate and univariate ones and provide empirical results supporting the results of the theoretical analysis. Additionally, we enhance forecasting accuracy by introducing a simple inference technique for PMformer. Finally, we highlight PMformer's useful characteristics in terms of the efficiency of inter-feature attention and robustness under missing features against complete-multivariate models.

Limitation. Further theoretical analysis is needed to more accurately explain partial-multivariate models, such as precisely calculating the entropy of posterior distributions and relaxing certain assumptions. Despite these limitations, our research still remains significant as it introduces the concept of partial-multivariate models for the first time and provides theoretical analyses that align with empirical results.

Broader Impacts. Our work might have positive effects by benefiting those who devise foundation models for times series because different time series vary in the number of features and our feature sampling scheme where the sampled subset size is always S can overcome this heterogeneity. As for the negative ones, we think the negative effects of forecasting well are still under-explored.

References

Amit Ron, Meir Ron. Meta-Learning by Adjusting Priors Based on Extended PAC-Bayes Theory. 2019.

Angryk Rafal A, Martens Petrus C, Aydin Berkay, Kempton Dustin, Mahajan Sushant S, Basodi Sunitha, Ahmadzadeh Azim, Cai Xumin, Filali Boubrahimi Soukaina, Hamdi Shah Muhammad, others . Multivariate time series dataset for space weather data analytics // Scientific data. 2020. 7, 1. 227.

Bai Shaojie, Kolter J. Zico, Koltun Vladlen. An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling. 2018.

Bhanja Samit, Das Abhishek. Impact of Data Normalization on Deep Neural Network for Time Series Forecasting. 2019.

Che Zhengping, Purushotham Sanjay, Cho Kyunghyun, Sontag David, Liu Yan. Recurrent Neural Networks for Multivariate Time Series with Missing Values. 2016.

Chen Si-An, Li Chun-Liang, Yoder Nate, Arik Sercan O., Pfister Tomas. TSMixer: An all-MLP Architecture for Time Series Forecasting. 2023a.

Chen Yushu, Liu Shengzhuo, Yang Jinzhe, Jing Hao, Zhao Wenlai, Yang Guangwen. A Joint Time-frequency Domain Transformer for Multivariate Time Series Forecasting. 2023b.

Cybenko G. Approximation by superpositions of a sigmoidal function // Mathematics of Control, Signals and Systems. Dec 1989. 2, 4. 303–314.

Domingos Pedro. A few useful things to know about machine learning // Commun. ACM. oct 2012. 55, 10. 78–87.

Du Yuntao, Wang Jindong, Feng Wenjie, Pan Sinno, Qin Tao, Xu Renjun, Wang Chongjun. AdaRNN: Adaptive Learning and Forecasting of Time Series. 2021.

Gao Jiaxin, Hu Wenbo, Chen Yuntian. Client: Cross-variable Linear Integrated Enhanced Transformer for Multivariate Long-Term Time Series Forecasting. 2023.

Gu Albert, Goel Karan, Ré Christopher. Efficiently Modeling Long Sequences with Structured State Spaces. 2022.

Jones Spencer S, Evans R Scott, Allen Todd L, Thomas Alun, Haug Peter J, Welch Shari J, Snow Gregory L. A multivariate time series approach to modeling and forecasting demand in the emergency department // Journal of biomedical informatics. 2009. 42, 1. 123–139.

Kidger Patrick, Morrill James, Foster James, Lyons Terry. Neural Controlled Differential Equations for Irregular Time Series // Advances in Neural Information Processing Systems. 2020.

Lee Seunghan, Park Taeyoung, Lee Kibok. Learning to Embed Time Series Patches Independently // The Twelfth International Conference on Learning Representations. 2024.

Li Shiyang, Jin Xiaoyong, Xuan Yao, Zhou Xiyou, Chen Wenhu, Wang Yu-Xiang, Yan Xifeng. Enhancing the Locality and Breaking the Memory Bottleneck of Transformer on Time Series Forecasting. 2020.

Liang Aobo, Jiang Xingguo, Sun Yan, Lu Chang. Bi-Mamba4TS: Bidirectional Mamba for Time Series Forecasting. 2024.

Lim Bryan, Arik Sercan O., Loeff Nicolas, Pfister Tomas. Temporal Fusion Transformers for Interpretable Multi-horizon Time Series Forecasting. 2020.

Lin Shengsheng, Lin Weiwei, Wu Wentai, Wang Songbo, Wang Yongxiang. PETformer: Long-term Time Series Forecasting via Placeholder-enhanced Transformer. 2023a.

Lin Shengsheng, Lin Weiwei, Wu Wentai, Zhao Feiyu, Mo Ruichao, Zhang Haotong. SegRNN: Segment Recurrent Neural Network for Long-Term Time Series Forecasting. 2023b.

Liu Minhao, Zeng Ailing, Chen Muxi, Xu Zhijian, Lai Qiuxia, Ma Lingna, Xu Qiang. SCINet: Time Series Modeling and Forecasting with Sample Convolution and Interaction. 2022a.

Liu Shizhan, Yu Hang, Liao Cong, Li Jianguo, Lin Weiyao, Liu Alex X., Dustdar Schahram. Pyraformer: Low-Complexity Pyramidal Attention for Long-Range Time Series Modeling and Forecasting // International Conference on Learning Representations. 2022b.

McAllester David A. PAC-Bayesian model averaging // Proceedings of the Twelfth Annual Conference on Computational Learning Theory. New York, NY, USA: Association for Computing Machinery, 1999. 164–170. (COLT '99).

Mehtab Sidra, Sen Jaydip. Stock Price Prediction Using Convolutional Neural Networks on a Multivariate Time Series. VIII 2021.

Nguyen Hieu M, Turk Philip J, McWilliams Andrew D. Forecasting COVID-19 hospital census: A multivariate time-series model based on local infection incidence // JMIR Public Health and Surveillance. 2021. 7, 8. e28195.

Nie Yuqi, Nguyen Nam H., Sinthong Phanwadee, Kalagnanam Jayant. A Time Series is Worth 64 Words: Long-term Forecasting with Transformers. 2023.

Qiu Jiayu, Wang Bin, Zhou Changjun. Forecasting stock prices with long-short term memory neural network based on attention mechanism // PLOS ONE. 01 2020. 15. e0227222.

Rasul Kashif, Ashok Arjun, Williams Andrew Robert, Ghonia Hena, Bhagwatkar Rishika, Khorasani Arian, Bayazi Mohammad Javad Darvishi, Adamopoulos George, Riachi Roland, Hassen Nadhir, Biloš Marin, Garg Sahil, Schneider Anderson, Chapados Nicolas, Drouin Alexandre, Zantedeschi Valentina, Nevmyvaka Yuriy, Rish Irina. Lag-Llama: Towards Foundation Models for Probabilistic Time Series Forecasting. 2024.

Sanhudo Luís, Rodrigues Joao, Vasconcelos Filho Enio. Multivariate time series clustering and forecasting for building energy analysis: Application to weather data quality control // Journal of Building Engineering. 2021. 35. 101996.

Shao Zezhi, Wang Fei, Zhang Zhao, Fang Yuchen, Jin Guangyin, Xu Yongjun. HUTFormer: Hierarchical U-Net Transformer for Long-Term Traffic Forecasting. 2023.

Tolstikhin Ilya, Houlsby Neil, Kolesnikov Alexander, Beyer Lucas, Zhai Xiaohua, Unterthiner Thomas, Yung Jessica, Steiner Andreas, Keysers Daniel, Uszkoreit Jakob, Lucic Mario, Dosovitskiy Alexey. MLP-Mixer: An all-MLP Architecture for Vision. 2021.

Valle-Pérez Guillermo, Louis Ard A. Generalization bounds for deep learning. 2020.

Vaswani Ashish, Shazeer Noam, Parmar Niki, Uszkoreit Jakob, Jones Llion, Gomez Aidan N, Kaiser Ł ukasz, Polosukhin Illia. Attention is All you Need // Advances in Neural Information Processing Systems. 30. 2017.

Wang Huiqiang, Peng Jian, Huang Feihu, Wang Jince, Chen Junhui, Xiao Yifei. MICN: Multi-scale Local and Global Context Modeling for Long-term Series Forecasting // The Eleventh International Conference on Learning Representations. 2023.

Wang Shiyu, Wu Haixu, Shi Xiaoming, Hu Tengge, Luo Huakun, Ma Lintao, Zhang James Y., ZHOU JUN. TimeMixer: Decomposable Multiscale Mixing for Time Series Forecasting // The Twelfth International Conference on Learning Representations. 2024.

Woo Gerald, Liu Chenghao, Sahoo Doyen, Kumar Akshat, Hoi Steven. Learning Deep Time-index Models for Time Series Forecasting // Proceedings of the 40th International Conference on Machine Learning. 2023.

Wu Haixu, Hu Tengge, Liu Yong, Zhou Hang, Wang Jianmin, Long Mingsheng. TimesNet: Temporal 2D-Variation Modeling for General Time Series Analysis. 2023.

Wu Haixu, Xu Jiehui, Wang Jianmin, Long Mingsheng. Autoformer: Decomposition Transformers with Auto-Correlation for Long-Term Series Forecasting. 2022.

Wu Zonghan, Pan Shirui, Long Guodong, Jiang Jing, Chang Xiaojun, Zhang Chengqi. Connecting the Dots: Multivariate Time Series Forecasting with Graph Neural Networks. 2020.

Xu Lei, Skoularidou Maria, Cuesta-Infante Alfredo, Veeramachaneni Kalyan. Modeling Tabular data using Conditional GAN. 2019.

Xu Zhijian, Zeng Ailing, Xu Qiang. FITS: Modeling Time Series with $10k$ Parameters // The Twelfth International Conference on Learning Representations. 2024.

Xue Wang, Zhou Tian, Wen Qingsong, Gao Jinyang, Ding Bolin, Jin Rong. Make Transformer Great Again for Time Series Forecasting: Channel Aligned Robust Dual Transformer. 2023.

Yu Chengqing, Wang Fei, Shao Zezhi, Sun Tao, Wu Lin, Xu Yongjun. DSformer: A Double Sampling Transformer for Multivariate Time Series Long-term Prediction. 2023.

Yun Chulhee, Bhojanapalli Srinadh, Rawat Ankit Singh, Reddi Sashank J., Kumar Sanjiv. Are Transformers universal approximators of sequence-to-sequence functions? 2020.

Zeng Ailing, Chen Muxi, Zhang Lei, Xu Qiang. Are Transformers Effective for Time Series Forecasting? 2022.

Zhang YiFan, Wen Qingsong, Wang Xue, Chen Weiqi, Sun Liang, Zhang Zhang, Wang Liang, Jin Rong, Tan Tieniu. OneNet: Enhancing Time Series Forecasting Models under Concept Drift by Online Ensembling // Thirty-seventh Conference on Neural Information Processing Systems. 2023a.

Zhang Yunhao, Yan Junchi. Crossformer: Transformer Utilizing Cross-Dimension Dependency for Multivariate Time Series Forecasting // The Eleventh International Conference on Learning Representations. 2023.

Zhang Zhenwei, Wang Xin, Gu Yuantao. SageFormer: Series-Aware Graph-Enhanced Transformers for Multivariate Time Series Forecasting. 2023b.

Zhao YanJun, Ma Ziqing, Zhou Tian, Sun Liang, Ye Mengni, Qian Yi. GCformer: An Efficient Framework for Accurate and Scalable Long-Term Multivariate Time Series Forecasting. 2023.

Zhou Haoyi, Zhang Shanghang, Peng Jieqi, Zhang Shuai, Li Jianxin, Xiong Hui, Zhang Wancai. Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting. 2021.

Zhou Tian, Ma Ziqing, Wen Qingsong, Wang Xue, Sun Liang, Jin Rong. FEDformer: Frequency Enhanced Decomposed Transformer for Long-term Series Forecasting. 2022.

donghao Luo, xue wang. ModernTCN: A Modern Pure Convolution Structure for General Time Series Analysis // The Twelfth International Conference on Learning Representations. 2024.

A Proof

A.1 Proof for Theorem 1

Starting from McAllester's bound on generalization errors (McAllester, 1999), we derive generalization bound in Theorem 1. Before getting into the main part, we define some notations. Let a neural network f be a partial-multivariate model which samples subsets $\mathbf{F}$ consisting of S features from a complete set of D features as defined in equation (1). $\mathcal{H}$ denotes hypothesis class of f, and $\mathbf{P}(h)$ and $\mathbf{Q}(h)$ are a prior and posterior distribution over the hypotheses h, respectively. Also, $(\mathbf{x}, \mathbf{y})$ is a input-output pair in an entire dataset and $(\mathbf{x}^{\mathcal{T}}, \mathbf{y}^{\mathcal{T}})$ is a pair in a training dataset $\mathcal{T}$ with m instances sampled from the entire dataset. At last, $\hat{\mathbf{y}} = f(\mathbf{x})$ is the output value of a neural network f, and $l(\mathbf{Q})$ and $\hat{l}(\mathbf{Q}, \mathcal{T})$ are generalized and empirical training loss under posterior distributions $\mathbf{Q}$ and training datasets $\mathcal{T}$.

Subsequently, we list assumptions for proof:

Assumption 1. *The maximum and minimum values of* $\mathbf{y}$ *are known and min-max normalization is applied to* $\mathbf{y}$ *(i.e.,* $0 \leq \mathbf{y} \leq 1$*).*

Assumption 2. *The output values of a neural network are assumed to be between 0 and 1, (i.e.,* $0 \leq \hat{\mathbf{y}} \leq 1$*).*

Assumption 3. *For posterior distributions* $\mathbf{Q}$*,* $\mathbf{Q}$ *is pruned. In other words, we set* $\mathbf{Q}(h) = 0$ *for hypotheses* h *where* $\mathbf{Q}(h) < \mathbf{P}(h)$ *and renormalize it.*

Assumption 4. *For any hypothesis* h*,* $\mathbf{P}(h) > \omega$ *where* ω *is the minimum probabilities in* $\mathbf{P}(h)$ *and* $\omega > 0$*.*

Assumption 5. *For posterior distributions* $\mathbf{Q}$ *and training datasets* $\mathcal{T}$*,* $\hat{l}(\mathbf{Q}, \mathcal{T}) \approx 0$*.*

Given that min-max normalization has been often used in time-series domains with empirical minimum and maximum values (Bhanja, Das, 2019), Assumption 1 can be regarded as a reasonable one. Also, by equipping the last layer with some activation functions such as Sigmoid or Tanh (hyperbolic tangent) like Xu et al. (2019) and adequate post-processing, Assumption 2 can be satisfied.[3] As for Assumption 3, according to (McAllester, 1999), it might have very little effects on $\mathbf{Q}$. Finally, because Transformers can universally approximate any continuous sequence-to-sequence function (Yun et al., 2020), (possibly, extended to general deep neural networks with the universal approximation theorem (Cybenko, 1989)), any hypothesis h can be approximated with proper parameters in f. Thus, we can assume $\mathbf{P}(h) > w > 0$ for any h when sampling the initial parameters of f from the whole real-number space (Assmuption 4). Also with proper training process and this universal approximation theorem, $\hat{l}(\mathbf{Q}, \mathcal{T})$ might approximate to zero (Assumption 5). With these assumptions, the proof for Theorem 1 is as follows:

Proof. Let MSE be a loss function l. Then, according to Assumption 1 and 2, $0 \leq l(h, (\mathbf{x}, \mathbf{y})) \leq 1$ for any data instance $(\mathbf{x}, \mathbf{y})$ and hypothesis h. Then, with probability at least $1 - \delta$ over the selection of the sample $\mathcal{T}$ of size m, we have the following for $\mathbf{Q}$ (McAllester, 1999):

$$l(\mathbf{Q}) \leq \hat{l}(\mathbf{Q}, \mathcal{T}) + \sqrt{\frac{D(\mathbf{Q}\|\mathbf{P}) + \log\frac{1}{\delta} + \frac{5}{2}\log m + 8}{2m - 1}}, \tag{8}$$

where $D(\mathbf{Q}\|\mathbf{P})$ denotes Kullback-Leibler divergence from distribution $\mathbf{Q}$ to $\mathbf{P}$. Due to Assumption 5, $\hat{l}(\mathbf{Q}, \mathcal{T}) \approx 0$. Also, because $E[\log\frac{1}{\mathbf{P}(h)}] < \log E[\frac{1}{\mathbf{P}(h)}] < \log\frac{1}{\omega} = C$ with Jensen's inequality and Assumption 4, $D(\mathbf{Q}\|\mathbf{P}) = E_{h\sim\mathbf{Q}}[\log\frac{\mathbf{Q}(h)}{\mathbf{P}(h)}] = E[\log\mathbf{Q}(h)] + E[\log\frac{1}{\mathbf{P}(h)}] < E[\log\mathbf{Q}(h)] + C$. Therefore, we can derive Theorem 1 by substituting $\hat{l}(\mathbf{Q}, \mathcal{T})$ and $D(\mathbf{Q}\|\mathbf{P})$ with 0 and $E[\log\mathbf{Q}(h)] + C$, respectively:

$$l(\mathbf{Q}) \leq \sqrt{\frac{E_{h\sim\mathbf{Q}}[\log\mathbf{Q}(h)] + \log\frac{1}{\delta} + \frac{5}{2}\log m + 8 + C}{2m - 1}}. \tag{9}$$

□

[3] Assumption 1 and 2 can be considered somewhat strong but should be satisfied to utilize McAllester's bound widely used for estimating generalization errors (Valle-Pérez, Louis, 2020; Amit, Meir, 2019). When the conditions of McAllester's bound are relaxed, we can also relax our assumptions.

Based on this theorem, we provide a theoretical analysis which is the impact of S on m and $-H(Q)$. However, an additional assumption is required to make the rationale valid as follows:

Assumption 6. *For the region of hypothesis h' where $\mathbf{Q}(h') > 0$, the prior distribution satisfies $\log \frac{1}{\mathbf{P}(h')} \leq C_{max}$ where C_{max} is small enough to be ignored in upper-bound.*

It is possible that the upper-bound is dominated by $C \to \infty$ when $w \to 0$. As such, $P(h)$ needs to be distributed properly over the region of hypothesis h' where $\mathbf{Q}(h') > 0$ not to result in $C \to \infty$, leading to Assumption 6. This assumption can be satisfied when the prior distribution is non-informative which is natural in Bayesian statistics under the assumption that prior knowledge is unknown (i.e. $P(h) \propto 1$). For any countable set of all possible inputs $\{\mathbf{x}_i\}_{i=1}^{N}$, probabilities of each h can be represented as $p(h) = \prod_{i=1}^{N} p(\hat{\mathbf{y}}_i^h|\mathbf{x}_i)$ where $\hat{\mathbf{y}}_i^h = f_h(\mathbf{x}_i)$ is the output of a function f_h under hypothesis h (Domingos, 2012). Because $0 \leq \hat{\mathbf{y}}_i^h \leq 1$ (Assumption 2) and $p(\hat{\mathbf{y}}_i^h|\mathbf{x}_i)$ is a uniform distribution under the non-informative assumption, $p(\hat{\mathbf{y}}_i^h|\mathbf{x}_i) = 1$. As such, the prior distribution under the non-informative assumption is $\mathbf{P}(h) = 1$, leading to $C_{max} = 0$ which is small enough not to dominate upper-bound. On top of that, we can indirectly solve this problem by injecting appropriate inductive biases in the form of architectures or regularizers, which can help to allocate more probability to each hypothesis (*i.e.*, increase ω) by reducing the size of the whole hypothesis space $\mathcal{H}$.

A.2 Proof for Theorem 2

To provide a proof for Theorem 2, we first prove Lemma 1. For Lemma 1, we need the following assumption:

Assumption 7. *A neural network f models models $p(\mathbf{y}|\mathbf{x})$ where $(\mathbf{x}, \mathbf{y})$ is an input-output pair.*

By regarding the output of a neural network $\hat{\mathbf{y}}$ as mean of normal or Student's t-distribution like in Rasul et al. (2024), Assumption 7 can be satisfied. Then, Lemma 1 and a proof are as follows:

Lemma 1. *Let $\hat{l}(\mathbf{Q}_S, \mathcal{T}_S)$ be a training loss with posterior distributions $\mathbf{Q}_S$ and a training dataset $\mathcal{T}_S$ when a subset size is S. Accordingly, $\hat{l}(\mathbf{Q}_S, \mathcal{T}_S) < \epsilon$ with small ϵ is a training objective. Then, for S_+ and S_- where $S_+ > S_-$, $\mathbf{Q}_{S_+}$ satisfies both $\hat{l}(\mathbf{Q}_{S_+}, \mathcal{T}_{S_+}) < \epsilon$ and $\hat{l}(\mathbf{Q}_{S_+}, \mathcal{T}_{S_-}) < \epsilon$. (On the other hands, $\mathbf{Q}_{S_-}$ is required to satisfy only $\hat{l}(\mathbf{Q}_{S_-}, \mathcal{T}_{S_-}) < \epsilon$.)*

Proof. Let S_+ and S_- be subset size where $S_+ > S_-$. $\mathbf{F}_{S_-}$ be any subset of S_- size sampled from a complete set of features, and $\mathbf{F}_{S_+}$ is any subset of S_+ size among ones that satisfy $\mathbf{F}_{S_-} \subset \mathbf{F}_{S_+}$. $\mathbf{F}_R$ is the set of elements that are in $\mathbf{F}_{S_+}$ but not in $\mathbf{F}_{S_-}$ (*i.e.*, $\mathbf{F}_R = \mathbf{F}_{S_+} - \mathbf{F}_{S_-}$). $\hat{l}(\mathbf{Q}_S, \mathcal{T}_S)$ is a training loss value with posterior distributions $\mathbf{Q}_S$ and a training dataset $\mathcal{T}_S$ when a subset size is S. Then, after training process satisfying $\hat{l}(\mathbf{Q}_{S_+}, \mathcal{T}_{S_+}) < \epsilon$ where ϵ is a small value, we can say that f under $\mathbf{Q}_{S_+}$ outputs the true value of $p(\mathbf{y}_{\mathbf{F}_{S_+}}|\mathbf{x}_{\mathbf{F}_{S_+}})$, according to Assumption 7. In the following process, we demonstrate that $p(\mathbf{y}_{\mathbf{F}_{S_-}}|\mathbf{x}_{\mathbf{F}_{S_-}})$ can be derived from $p(\mathbf{y}_{\mathbf{F}_{S_+}}|\mathbf{x}_{\mathbf{F}_{S_+}}) = p(\mathbf{y}_{\mathbf{F}_{S_-}}, \mathbf{y}_{\mathbf{F}_R}|\mathbf{x}_{\mathbf{F}_{S_-}}, \mathbf{x}_{\mathbf{F}_R})$:

$$\int_{\mathbf{y}_{\mathbf{F}_R}} E_{\mathbf{x}_{\mathbf{F}_R}|\mathbf{x}_{\mathbf{F}_{S_-}}} [p(\mathbf{y}_{\mathbf{F}_{S_-}}, \mathbf{y}_{\mathbf{F}_R}|\mathbf{x}_{\mathbf{F}_{S_-}}, \mathbf{x}_{\mathbf{F}_R})] d\mathbf{y}_{\mathbf{F}_R}, \tag{10}$$

$$= \int_{\mathbf{y}_{\mathbf{F}_R}} \int_{\mathbf{x}_{\mathbf{F}_R}} p(\mathbf{y}_{\mathbf{F}_{S_-}}, \mathbf{y}_{\mathbf{F}_R}|\mathbf{x}_{\mathbf{F}_{S_-}}, \mathbf{x}_{\mathbf{F}_R}) p(\mathbf{x}_{\mathbf{F}_R}|\mathbf{x}_{\mathbf{F}_{S_-}}) d\mathbf{x}_{\mathbf{F}_R} d\mathbf{y}_{\mathbf{F}_R}, \tag{11}$$

$$= p(\mathbf{y}_{\mathbf{F}_{S_-}}|\mathbf{x}_{\mathbf{F}_{S_-}}), \tag{12}$$

In that expectation can be approximated by an empirical mean with sufficient data and integral can be addressed with discretization, we can think that $p(\mathbf{y}_{\mathbf{F}_{S_-}}|\mathbf{x}_{\mathbf{F}_{S_-}})$ can be derived from $p(\mathbf{y}_{\mathbf{F}_{S_+}}|\mathbf{x}_{\mathbf{F}_{S_+}})$. According to this fact, f under $\mathbf{Q}_+$ should be able to output not only true $p(\mathbf{y}_{\mathbf{F}_{S_+}}|\mathbf{x}_{\mathbf{F}_{S_+}})$ but also true $p(\mathbf{y}_{\mathbf{F}_{S_-}}|\mathbf{x}_{\mathbf{F}_{S_-}})$. Therefore, we conclude that $\mathbf{Q}_+$ have to satisfy both $\hat{l}(\mathbf{Q}_{S_+}, \mathcal{T}_{S_+}) < \epsilon$ and $\hat{l}(\mathbf{Q}_{S_+}, \mathcal{T}_{S_-}) < \epsilon$. □

With Lemma 1, we provide a proof for Theorem 2:

Proof. Let h be a hypothesis on a space defined when a subset size is S. Then, we can denote a posterior distribution which is trained to decrease $\hat{l}(\mathbf{Q}_S, \mathcal{T}_S)$ as follows:

$$\mathbf{Q}(h_S) = p(h_S|c_S = 1), \quad \text{where} \;\; c_S = \begin{cases} 1, & \hat{l}(h, \mathcal{T}_S) < \epsilon, \\ 0, & \text{otherwise}, \end{cases} \tag{13}$$

According to Lemma 1, for S_+ and S_- where $S_+ > S_-$, the posterior distributions of two cases can be represent as $\mathbf{Q}(h_{S_+}) = p(h_{S_+}|c_{S_+} = 1, c_{S_-} = 1)$ and $\mathbf{Q}(h_{S_-}) = p(h_{S_-}|c_{S_-} = 1)$, respectively. With the following two assumptions, we can prove Theorem 2:

Assumption 8. *hypotheses h_{S_+} and h_{S_-} have similar distributions after training with $\mathcal{T}_{S_-}$ (i.e., $p(h_{S_+}|c_{S_-} = 1) \approx p(h_{S_-}|c_{S_-} = 1)$).*

Assumption 9. *Prior distributions are nearly non-informative (i.e., $P(h) \propto 1$).*

Assumption 8 can be considered reasonable because we can make the training process of a model of subset size S_+ very similar to that of subset size S_- with a minimal change in architecture such as input and output masking. Also, as for Assumption 9, non-informative prior is usually used under usual situations without prior knowledge in Bayesian statistics.

$\mathbf{Q}(h_S)$ can be expanded as $p(h_S|c_S) \propto p(c_S|h_S)p(h_S) \propto p(c_S|h_S)$, according to Assumption 9. Because we exactly know whether to satisfy $\hat{l}(h, \mathcal{T}_S) < \epsilon$ given h, $p(c_S|h_S)$ is 1 when a given h_S satisfies c_S or 0, otherwise. Thus, $\mathbf{Q}(h_S)$ are defined as follows:

$$\mathbf{Q}(h_S) = p(h_S|c_S = 1) = \begin{cases} \eta_S, & c_S = 1 \text{ given } h_S, \\ 0, & \text{otherwise}, \end{cases} \tag{14}$$

Similarly, $\mathbf{Q}(h_{S_+})$ and $\mathbf{Q}(h_{S_-})$ can be expanded as $p(h_{S_+}|c_{S_+}, c_{S_-}) \propto p(c_{S_+}, c_{S_-}|h_{S_+})p(h_{S_+}) \propto p(c_{S_+}, c_{S_-}|h_{S_+})$ and $p(h_{S_-}|c_{S_-}) = p(h_{S_+}|c_{S_-}) \propto p(c_{S_-}|h_{S_+})p(h_{S_+}) \propto p(c_{S_-}|h_{S_+})$, according to Assumption 8 and 9. A region of hypothesis satisfying both $c_{S_+} = 1$ and $c_{S_-} = 1$ is smaller than that satisfying either of them. Because the probability of h in a region satisfying conditions has the same value and $\int_h p(h)dh = 1$ is maintained, h in the small region is allocated higher probabilities than h in the large one. Therefore, $\eta_{S_+} > \eta_{S_-}$ and the entropy $H(\mathbf{Q}_{S_-})$ is larger than $H(\mathbf{Q}_{S_+})$:

□

So far, we have finished a proof for Theorem 2. We additionally provide Theorem 3 which is a variant of Theorem 2 where Assumption 9 can be relaxed while proposing the relationships between $H(\mathbf{Q}_{S_-})$ and $H(\mathbf{Q}_{S_+})$ in the expectation level:

Theorem 3. *for S_+ and S_- satisfying $S_+ > S_-$, $H(\mathbf{Q}_{S_+}) \leq H(\mathbf{Q}_{S_-})$ in expectation over c_{S_+}.*

Proof. Let $\tilde{h}_{S_+}$ be the $h_{S_+}|c_{S_-} = 1$ (*i.e.*, $\mathbf{Q}(h_{S_+}) = p(h_{S_+}|c_{S_+} = 1, c_{S_-} = 1) = p(\tilde{h}_{S_+}|c_{S_+} = 1)$). Then, $H(\tilde{h}_{S_+}|c_{S_+})$ can be expanded as follows:

$$H(p(\tilde{h}_{S_+}|c_{S_+})), \tag{15}$$

$$= H(p(c_{S_+}|\tilde{h}_{S_+})) + H(p(\tilde{h}_{S_+})) - H(p(c_{S_+})), \quad (\because \text{Bayes' rule for conditional entropy states}), \tag{16}$$

$$= H(p(\tilde{h}_{S_+})) - H(p(c_{S_+})) \quad (\because \text{when } h \text{ is given, we know whether to satisfy } \hat{l}(h, \mathcal{T}_{S_+}) < \epsilon. \; (i.e., H(p(c_{S_+}|\tilde{h}_{S_+})) = 0), \tag{17}$$

From this expansion, we can derive $H(p(\tilde{h}_{S_+}|c_{S_+})) \leq H(p(\tilde{h}_{S_+}))$ because entropy of $p(c_{S_+})$ must be larger than 0 (*i.e.*, $H(p(c_{S_+})) \geq 0$). By substituting $H(\mathbf{Q}_{S_-})$ for $H(p(\tilde{h}_{S_+}))$ according to Assumption 8 and $E_{c_{S_+}}[H(\mathbf{Q}_{S_+})]$ for $H(p(\tilde{h}_{S_+}|c_{S_+}))$, we can derive Theorem 3.

Also, based on Chebyshev's inequality, we can calculate the least probabilities at which $H(\mathbf{Q}_{S_+}) < H(\mathbf{Q}_{S_-})$ are satisfied, given the variance $\sigma^2 = Var_{c_{S_+}}[H(p(\tilde{h}_{S_+}|c_{S_+}))]$:

$$p\left[H(\mathbf{Q}_{S_+}) < H(\mathbf{Q}_{S_-})\right] \quad \leq \quad 1 - \frac{\sigma^2}{(H(p(c_{S_+})))^2} \tag{18}$$

□

B How to Handle Non-Divisible Cases of PMformer with Random Partitioning

In this section, we further elaborate on how to deal with the cases where the number of features D is not divisible by the size of subsets S. We simply repeat some randomly chosen features and augment them to the original input time series, in order to make the total number of features divisible by S. After finishing the forecasting procedure with the augmented inputs, we drop augmented features from outputs. The details are delineated in Algorithm 2.

Algorithm 2: How to handle non-divisible cases of PMformer with random partitioning

```
Input: # of features $D$, Subset size $S$, Past obs. $\mathbf{x}_{[0:D]}$
1  $\mathbf{V} = \{0, 1, ..., D-1\}$;   $N_U = \lceil \frac{D}{S} \rceil$;   $R = D \% S$;
2  if $R \neq 0$ then
3      Randomly split $\mathbf{V}$ into $\mathbf{V}^+, \mathbf{V}^-$, where $|\mathbf{V}^+| = D - R, |\mathbf{V}^-| = R, \mathbf{V}^+ \cap \mathbf{V}^- = \phi$;
4      Get $\{\mathbf{F}(g)\}_{g \in [0, N_U - 1]}$ by randomly partitioning $\mathbf{V}^+$;
5      $\mathbf{V}^{++} = \{v_i | v_i$is a random sample from $\mathbf{V}^+$ without replacement, $i = [0, S - R]\}$;
6      $\mathbf{F}(N_U - 1) = \mathbf{V}^- \cup \mathbf{V}^{++}$
7  else
8      Get $\{\mathbf{F}(g)\}_{g \in [0, N_U]}$ by randomly partitioning $\mathbf{V}$;
9  for $g \leftarrow 0$ to $N_U - 1$ do
10     $\hat{\mathbf{y}}_{\mathbf{F}(g)} = \mathtt{PMformer}(\mathbf{x}_{\mathbf{F}(g)}, \mathbf{F}(g))$;
11 if $R \neq 0$ then
12     Remove features of $\mathbf{V}^{++}$ from $\hat{\mathbf{y}}_{\mathbf{F}(N_U - 1)}$;
13 Sort $\{\hat{\mathbf{y}}_{\mathbf{F}(g)}\}_{g \in [0, N_U]}$ by feature index and get $\hat{\mathbf{y}}_{[0:D]}$;
14 return Predicted future observations $\hat{\mathbf{y}}_{[0:D]}$;
```

C Details of Experimental Environments

C.1 Datasets

We evaluate PMformer on 7 benchmark datasets for time series forecasting with multiple variables. The normalization and train/val/test splits are also the same with PatchTST (Nie et al., 2023) which is our main baseline. The information of each dataset is as follows:

- **(1-2) ETTh1,2**[4] (Electricity Transformer Temperature-hourly): They have 7 indicators in the electric power long-term deployment, such as oil temperature and 6 power load features. This data is collected for 2 years and the granularity is 1 hour. Different numbers denote different counties in China. Train/val/test is 12/4/4 months and the number of time steps is 17,420.
- **(3-4) ETTm1,2** (Electricity Transformer Temperature-minutely): This dataset is exactly the same with **ETTh1,2**, except for granularity. The granularity of these cases is 15 minutes. The number of time steps is 69,680.
- **(5) Weather**[5]: It has 21 indicators of weather including temperature, humidity, precipitation, and air pressure. It was recorded for 2020, and the granularity is 10 minutes. The ratio of train/val/test is 0.7/0.1/0.2 and the number of time steps is 52,696.
- **(6) Electricity**[6]: In this dataset, information about hourly energy consumption from 2012 to 2014 is collected. Each feature means the electricity consumption of one client, and there are 321 clients in total. The ratio of train/val/test is 0.7/0.1/0.2 and the number of time steps is 26,304.
- **(7) Traffic**[7]: Traffic dataset pertains to road occupancy rates. It encompasses hourly data collected by 862 sensors deployed on San Francisco freeways during the period spanning from 2015 to 2016. The ratio of train/val/test is 0.7/0.1/0.2 and the number of time steps is 17,544.

[4]https://github.com/zhouhaoyi/ETDataset

[5]https://www.bgc-jena.mpg.de/wetter/

[6]https://archive.ics.uci.edu/ml/datasets/ElectricityLoadDiagrams20112014

[7]http://pems.dot.ca.gov

C.2 Software & Hardware Environments and Computation Time

We conduct experiments on this software and hardware environments for M-LTSF: PYTHON 3.7.12, PYTORCH 2.0.1, and NVIDIA GEFORCE RTX 3090. For each training of PMformer, it takes about $1 \sim 4$ hours with one or two GPUs according to the number of features. In total, it takes about one months to complete our projects with 16 GPUs.

C.3 Baselines

We select 3 univariate baselines: PatchTST, NLinear, and DLinear. As for complete-multivariate ones, there are many candidates including Lim et al. (2020); Wu et al. (2022); Li et al. (2020). Among them, our choices are Crossformer, FEDformer, Informer, Pyraformer, TSMixer, DeepTime, MICN, and TimesNet, considering their performance and meanings in the forecasting tasks.

- **(1) PatchTST** (Nie et al., 2023): It uses segmentation with separate tokenization where different features are allocated to different tokens and doesn't consider any relationship between different features.
- **(2) DLinear** (Zeng et al., 2022): A single linear layer mapping past observations into future observations with a decomposition trick.
- **(3) NLinear** (Zeng et al., 2022): A single linear layer mapping past observations into future observations with a normalization trick that subtracts the last value of input observations from input and adds the value to the output.
- **(4) Crossformer** (Zhang, Yan, 2023): It use similar segmentation to **PatchTST** and and two types of attention, one of which is self-attention for temporal dependencies and the other is for inter-feature relationships. It reduces the complexity of self-attention for inter-feature relationships using routers with low-rank approximation concepts.
- **(5) FEDformer** (Zhou et al., 2022): Using the sparsity in frequency domains, it tries to reduce the quadratic complexity of self-attention layers to a linear one.
- **(6) Informer** (Zhou et al., 2021): By estimating KL divergence between query-key distribution and uniform distribution, it discerns useful and useless information. By using only useful information, it achieves log-linear complexity. Also, a new type of decoder was proposed, which generates forecasting outputs at once.
- **(7) Pyraformer** (Liu et al., 2022b): It has hierarchical structures with different resolutions, leading to linear complexity of self-attention.
- **(8) TSMixer** (Chen et al., 2023a): Using the concept of MLP-Mixer in vision domains (Tolstikhin et al., 2021), it was devised to explore the abilities of linear layers in the forecasting tasks.
- **(9) DeepTime** (Woo et al., 2023): It solves the problem where INRs are hard to be generalized in time-series forecasting tasks, with a meta-optimization framework.
- **(10) MICN** (Wang et al., 2023): To capture both local and global patterns from time series efficiently, it extracts patterns with down-sampled convolution and isometric convolution. Also, multi-scale structures are used to capture more diverse patterns.
- **(11) TimesNet** (Wu et al., 2023): Building upon the multi-periodicity of time series, it regards time series as not 1d but 2d structures and aims to figure out intra-period and inter-period relationships.

Furthermore, we find concurrent works for time-series forecasting and include them as our baselines. Among Chen et al. (2023b); Zhao et al. (2023); Xue et al. (2023); Gao et al. (2023); Zhang et al. (2023b); Shao et al. (2023); Yu et al. (2023); Lin et al. (2023a); Lee et al. (2024); donghao, xue wang (2024); Xu et al. (2024); Wang et al. (2024), we select PITS, FITS, TimeMixer, JTFT, GCformer, CARD, Client, PETformer, and ModernTCN as our baselines because they have the same experimental settings with ours[8] or their executable codes are available to run models in our settings.

- **(1) PITS** (Lee et al., 2024): This paper proposed new a self-supervised representation learning strategy for time series with neural networks not directly considering patch-dependence. Through the contrastive learning, adjacent time series information can be captured efficiently.

[8]We decide that it has the same experimental setting with ours when the scores of some baselines are the same.

- **(2) FITS** (Xu et al., 2024): This paper introduced FITS, which is effective but efficient for time series tasks, based on the fact that time series can be dealt with in the complex frequency domain.
- **(3) TimeMixer** (Wang et al., 2024): This paper is similar to **TSMixer**. However, it has distinct differences in that it utilizes a decomposition scheme and considers multi-scale time series.
- **(4) JTFT** (Chen et al., 2023b): Similar to **Crossformer**, segmentation and two types of Transformers are employed. Before a Transformer takes input, it pre-processes input time series. It only encodes a fixed length of recent observations into tokens and sparse frequency information extracted from the whole input into tokens, rather than encodes the whole input directly. This leads to efficient self-attention for temporal dependencies. Also, with a low-rank approximation scheme, it reduces the complexity of self-attention for inter-feature dependencies.
- **(5) GCformer** (Zhao et al., 2023): To overcome the limitations of Transformers that they cannot deal with long time series well, it combines a convolutional branch for global information and Transformer-based branch for local, recent information.
- **(6) CARD** (Xue et al., 2023): With a dual Transformer, it can capture various dependencies across temporal, feature, and hidden dimensions. On top of that, the author devised a robust loss function to relieve overfitting issues in M-LTSF.
- **(7) Client** (Gao et al., 2023): This method has two parts, one of which is a linear model to capture temporal trends and the other is self-attention for inter-feature dependencies.
- **(8) PETformer** (Lin et al., 2023a): Based on **Crossformer** architecture, it introduced placeholder enhancement technique (PET). Thanks to PET, PETformer can forecast with only encoders (*i.e.*, without decoder).
- **(9) ModernTCN** (donghao, xue wang, 2024): Because many existing CNN-based methods don't show the good performance in time series forecasting tasks, this paper tried to modify the traditional CNNs into ModernTCN including maintaining the variable dimension, DWConv, and ConvFFN.

As for evaluation metrics of baseline methods, we repeat the scores when the scores of the same experimental settings as ours are available. Otherwise, we measure evaluation scores with their official codes and best hyperparameters in our experimental environments. The scores of PatchTST, FEDformer, Pyraformer, and Informer are from Nie et al. (2023), and those of TSMixer and NLinear (DLinear) are from Chen et al. (2023a) and Zeng et al. (2022), respectively. Also, for PITS, FITS, TimeMixer, JTFT, GCformer, CARD, PETformer, and ModernTCN, we repeat the score reported in each paper. For Crossformer[9], MICN, TimesNet, Client[10], and DeepTime[11], we measure new scores in the same experimental environments with ours. When training Crossformer, we convert a Transformer-based encoder into a linear-based encoder for fair comparison to PMformer, because the latter usually has better performance than the former.

C.4 HyperParameters

The details of hyperparameters used in the PMformer are delineated in this section. The first hyperparameter is the length of input time steps T. We regard it as hyperparameters which is common in recent literature for time-series forecasting (Liu et al., 2022b; Zhang, Yan, 2023). The range of T is $\{512, 1024\}$. Also, the number of segments N_S is in $\{8,16,32,64\}$ and the dropout ratio r_{dropout} is in $\{0.1, 0.2, 0.3, 0.4, 0.7\}$. The hidden dimension d_h is in $\{32,64,128,256,512\}$. The number of heads in self-attention n_h is in $\{2,4,8,16\}$ and the number of layers L is in $\{1,2,3\}$. d_{ff} is the hidden size of feed-forward networks in each PMformer layer and in $\{32,64,128,256,512\}$. Also, batch size is 128, 128, 16, and 12 for ETT, Weather, Electricity, and Traffic datasets, respectively. Finally, we set the learning rate and training epochs to 10^{-3} and 100, respectively. Finally, we use Adam optimizer to train our model. The selected best hyperparameters of PMformer are in Table 5.

[9]`https://github.com/Thinklab-SJTU/Crossformer`

[10]For MICN, TimesNet, and Client, we use the same code from `https://github.com/daxin007/Client/tree/main`.

[11]`https://github.com/salesforce/DeepTime`

Table 5: Selected hyperparameters of PMformer.

Data	τ	T	N_S	r_{dropout}	d_h	n_h	L	d_{ff}
ETTh1	96	512	64	0.7	128	4	1	256
	192	512	64	0.7	32	4	1	256
	336	512	64	0.7	64	8	1	64
	336	512	64	0.7	64	8	1	64
ETTh2	96	512	64	0.7	512	4	1	256
	192	1024	64	0.7	512	2	1	256
	336	1024	64	0.7	64	16	1	256
	720	512	64	0.7	64	16	1	128
ETTm1	96	512	64	0.2	256	2	2	256
	192	512	64	0.1	64	8	1	128
	336	512	64	0.2	64	2	2	64
	720	1024	64	0.7	64	4	1	128
ETTm2	96	1024	64	0.7	512	2	1	64
	192	1024	64	0.7	128	4	1	32
	336	1024	64	0.4	128	2	1	32
	720	1024	64	0.7	256	4	1	32
Weather	96	512	64	0.2	128	8	3	256
	192	512	64	0.2	128	16	3	256
	336	512	64	0.4	128	16	3	512
	720	512	64	0.4	128	2	1	256
Electricity	96	512	64	0.3	256	8	1	256
	192	512	64	0.2	256	4	2	256
	336	512	64	0.2	128	4	3	256
	720	512	64	0.2	256	4	3	256
Traffic	96	512	8	0.2	512	2	3	512
	192	512	8	0.1	256	4	3	512
	336	512	8	0.2	256	2	3	256
	720	512	8	0.2	512	4	3	512

Table 6: Test MSE and MAE of training PMformer using a training algorithm with random sampling or partitioning

Score	Training Algorithm	ETTh1 ($D = 7$)				ETTh2 (7)				ETTm1 (7)				ETTm2 (7)			
		96	192	336	720	96	192	336	720	96	192	336	720	96	192	336	720
MSE	Random Partitioning	**0.361**	**0.396**	**0.400**	**0.412**	**0.269**	**0.323**	**0.317**	**0.370**	**0.282**	**0.325**	**0.352**	**0.401**	**0.160**	**0.213**	**0.262**	**0.336**
	Random Sampling	0.362	0.397	**0.400**	**0.412**	0.273	**0.323**	**0.317**	0.371	0.283	**0.325**	**0.352**	0.403	0.162	0.214	0.263	0.337
MAE	Random Partitioning	**0.390**	**0.414**	**0.421**	**0.442**	**0.332**	**0.369**	**0.378**	**0.416**	0.340	**0.365**	**0.385**	**0.408**	**0.253**	**0.290**	**0.325**	**0.372**
	Random Sampling	0.391	0.415	**0.421**	**0.442**	0.334	**0.369**	0.380	**0.416**	**0.339**	**0.365**	**0.385**	0.409	0.254	0.291	0.326	0.373

Score	Training Algorithm	Weather (21)				Electricity (321)				Traffic (862)			
		96	192	336	720	96	192	336	720	96	192	336	720
MSE	Random Partitioning	**0.142**	0.185	**0.235**	**0.305**	**0.125**	0.142	**0.154**	**0.176**	**0.345**	**0.370**	**0.385**	**0.426**
	Random Sampling	**0.142**	**0.184**	0.237	**0.305**	0.126	**0.141**	**0.154**	0.180	0.347	**0.370**	0.386	0.427
MAE	Random Partitioning	**0.193**	0.237	**0.277**	**0.328**	**0.222**	0.240	0.256	**0.278**	**0.245**	**0.255**	**0.265**	**0.287**
	Random Sampling	0.195	**0.236**	0.278	0.329	**0.222**	**0.239**	**0.255**	0.281	0.246	0.256	**0.265**	**0.287**

D Theoretical Complexity of Inter-Feature Attention in PMformer

In this section, we elaborate on the reason why the theoretical complexity of inter-feature attention in PMformer is $\mathcal{O}(SD)$ where D is the number of features and S is the subset size. Attention cost in each subset is $\mathcal{O}(S^2)$. Because random partitioning generates $N_U \approx \frac{D}{S}$ subsets, the final complexity is $N_U\mathcal{O}(S^2) = \frac{D}{S}\mathcal{O}(S^2) = \mathcal{O}(SD)$.

E The Effect of Training PMformer with Random Sampling or Partitioning

In this section, we provide the experimental results where we train PMformer using a training algorithm with random sampling or partitioning (*i.e.*, $use\_random\_partition = False$ or $True$ in Algorithm 1). As shown in Table 6, these two ways are comparable in terms of forecasting performance — note that we adopt the training algorithm based on random partitioning for our main experiments.

F The Performance of PMformer with $N_I = 1$

In Table 7, we conduct the main experiments including PMformer with $N_I = 1$ which is the number of repeating an inference process based on random partitioning. In this experiment, we include some

Table 7: MSE scores of main forecasting results including PMformer wiht $N_I = 1$.

Data		MSE					MAE				
		PMformer	PatchTST	Nlinear	TSMIxer	DeepTime	PMformer	PatchTST	Nlinear	TSMIxer	DeepTime
ETTh1	96	**0.361**	$\underline{0.370}$	0.374	**0.361**	0.372	**0.391**	0.400	0.394	$\underline{0.392}$	0.398
	192	**0.393**	0.413	0.408	$\underline{0.404}$	0.405	**0.409**	0.429	$\underline{0.415}$	0.418	0.419
	336	**0.404**	0.422	0.429	$\underline{0.420}$	0.437	**0.417**	0.440	$\underline{0.427}$	0.431	0.442
	720	**0.412**	0.447	$\underline{0.440}$	0.463	0.477	**0.442**	0.468	$\underline{0.453}$	0.472	0.493
ETTh2	96	**0.270**	$\underline{0.274}$	0.277	$\underline{0.274}$	0.291	**0.332**	$\underline{0.337}$	0.338	0.341	0.350
	192	**0.321**	0.341	0.344	$\underline{0.339}$	0.403	**0.369**	0.382	$\underline{0.381}$	0.385	0.427
	336	**0.317**	$\underline{0.329}$	0.357	0.361	0.466	**0.380**	$\underline{0.384}$	0.400	0.406	0.475
	720	**0.371**	$\underline{0.379}$	0.394	0.445	0.576	**0.416**	$\underline{0.422}$	0.436	0.470	0.545
ETTm1	96	$\underline{0.286}$	0.293	0.306	**0.285**	0.311	$\underline{0.343}$	0.346	0.348	**0.339**	0.353
	192	$\underline{0.328}$	0.333	0.349	**0.327**	0.339	$\underline{0.368}$	0.370	0.375	**0.365**	0.369
	336	**0.354**	0.369	0.375	$\underline{0.356}$	0.366	$\underline{0.387}$	0.392	0.388	**0.382**	0.391
	720	$\underline{0.403}$	0.416	0.433	0.419	**0.400**	**0.409**	0.420	0.422	$\underline{0.414}$	$\underline{0.414}$
ETTm2	96	**0.160**	0.166	0.167	$\underline{0.163}$	0.165	**0.250**	0.256	0.255	$\underline{0.252}$	0.259
	192	**0.213**	0.223	0.221	$\underline{0.216}$	0.222	**0.288**	0.296	0.293	$\underline{0.290}$	0.299
	336	**0.262**	0.274	0.274	$\underline{0.268}$	0.278	$\underline{0.325}$	0.329	0.327	**0.324**	0.338
	720	**0.336**	$\underline{0.361}$	0.368	0.420	0.369	**0.372**	0.394	$\underline{0.384}$	0.422	0.400
Weather	96	**0.142**	0.149	0.182	$\underline{0.145}$	0.169	**0.194**	$\underline{0.198}$	0.232	$\underline{0.198}$	0.227
	192	**0.186**	0.194	0.225	$\underline{0.191}$	0.211	**0.238**	$\underline{0.241}$	0.269	0.242	0.266
	336	**0.236**	0.245	0.271	$\underline{0.242}$	0.255	**0.278**	0.282	0.301	$\underline{0.280}$	0.304
	720	**0.305**	$\underline{0.314}$	0.338	0.320	0.318	**0.328**	$\underline{0.334}$	0.348	0.336	0.357
Electricity	96	**0.127**	$\underline{0.129}$	0.141	0.131	0.139	$\underline{0.224}$	**0.222**	0.237	0.229	0.239
	192	**0.145**	$\underline{0.147}$	0.154	0.151	0.154	$\underline{0.244}$	**0.240**	0.248	0.246	0.253
	336	**0.158**	0.163	0.171	$\underline{0.161}$	0.169	$\underline{0.260}$	**0.259**	0.265	0.261	0.270
	720	**0.181**	$\underline{0.197}$	0.210	$\underline{0.197}$	0.201	**0.283**	$\underline{0.290}$	0.297	0.293	0.300
Traffic	96	**0.347**	$\underline{0.360}$	0.410	0.376	0.401	**0.247**	$\underline{0.249}$	0.279	0.264	0.280
	192	**0.372**	$\underline{0.379}$	0.423	0.397	0.413	$\underline{0.258}$	**0.256**	0.284	0.277	0.285
	336	**0.387**	$\underline{0.392}$	0.435	0.413	0.425	$\underline{0.267}$	**0.264**	0.290	0.290	0.292
	720	**0.430**	$\underline{0.432}$	0.464	0.444	0.462	$\underline{0.289}$	**0.286**	0.307	0.306	0.312
Avg. Rank		**1.107**	2.786	4.321	$\underline{2.571}$	4.036	**1.357**	$\underline{2.714}$	3.464	2.750	4.607

baselines showing decent forecasting performance. As Table 7 shows, despite $N_I = 1$, PMformer still gives better results than baselines.

G Additional Experiments

G.1 Additional Experimental Results in Tabular Forms

In this section, we provide additional results for existing experiments, such as experiments with other datasets and MAE evaluation metrics. Table 8, Table 9, and Table 10 are additional results for Table 1, Table 3, and Table 4, respectively. Furthermore, Table 11 provides the standard deviation information of PMformer in forecasting accuracy of Table 8.

G.2 Additional Visualization

Like Appendix G.1, this section provides additional visualizations with other datasets or models for existing ones. Figure 8 is for Figure 3, Figure 9 for Figure 4, Figure 10 for Figure 5(a), Figure 11 for Figure 5(b), and Figure 12 for Figure 6. Furthermore, Figure 13 shows the forecasting results of PMformer, PatchTST, and Crossformer. We select these baselines because they have similar architecture to PMformer, such as segmentation or inter-feature attention modules. Our method captures temporal dynamics better than baselines.

Table 8: MSE and MAE scores of main forecasting results. ‘former’ included in some model names is abbreviated to ‘f.’. Also, ‘P.M.’, , ‘U.’, and ‘C.M.’ denote partial-multivariate, univariate, and complete-multivariate, respectively. (Additional results for Table 1)

Score	Data		P.M.	U.			C.M.							
			PMf.	PatchTST	Dlinear	Nlinear	Crossf.	FEDf.	Inf.	Pyraf.	TSMixer	DeepTime	MICN	TimesNet
MSE	ETTh1	96	**0.361**	0.370	0.375	0.374	0.427	0.376	0.941	0.664	**0.361**	0.372	0.828	0.465
		192	**0.396**	0.413	0.405	0.408	0.537	0.423	1.007	0.790	0.404	0.405	0.765	0.493
		336	**0.400**	0.422	0.439	0.429	0.651	0.444	1.038	0.891	0.420	0.437	0.904	0.456
		720	**0.412**	0.447	0.472	0.440	0.664	0.469	1.144	0.963	0.463	0.477	1.192	0.533
	ETTh2	96	**0.269**	0.274	0.289	0.277	0.720	0.332	1.549	0.645	0.274	0.291	0.452	0.381
		192	**0.323**	0.341	0.383	0.344	1.121	0.407	3.792	0.788	0.339	0.403	0.554	0.416
		336	**0.317**	0.329	0.448	0.357	1.524	0.400	4.215	0.907	0.361	0.466	0.582	0.363
		720	**0.370**	0.379	0.605	0.394	3.106	0.412	3.656	0.963	0.445	0.576	0.869	0.371
	ETTm1	96	**0.282**	0.293	0.299	0.306	0.336	0.326	0.626	0.543	0.285	0.311	0.406	0.343
		192	**0.325**	0.333	0.335	0.349	0.387	0.365	0.725	0.557	0.327	0.339	0.500	0.381
		336	**0.352**	0.369	0.369	0.375	0.431	0.392	1.005	0.754	0.356	0.366	0.580	0.436
		720	0.401	0.416	0.425	0.433	0.555	0.446	1.133	0.908	0.419	**0.400**	0.607	0.527
	ETTm2	96	**0.160**	0.166	0.167	0.167	0.338	0.180	0.355	0.435	0.163	0.165	0.238	0.218
		192	**0.213**	0.223	0.224	0.221	0.567	0.252	0.595	0.730	0.216	0.222	0.302	0.282
		336	**0.262**	0.274	0.281	0.274	1.050	0.324	1.270	1.201	0.268	0.278	0.447	0.378
		720	**0.336**	0.361	0.397	0.368	2.049	0.410	3.001	3.625	0.420	0.369	0.549	0.444
	Weather	96	**0.142**	0.149	0.176	0.182	0.150	0.238	0.354	0.896	0.145	0.169	0.188	0.179
		192	**0.185**	0.194	0.220	0.225	0.200	0.275	0.419	0.622	0.191	0.211	0.231	0.230
		336	**0.235**	0.245	0.265	0.271	0.263	0.339	0.583	0.739	0.242	0.255	0.280	0.276
		720	**0.305**	0.314	0.323	0.338	0.310	0.389	0.916	1.004	0.320	0.318	0.358	0.347
	Electricity	96	**0.125**	0.129	0.140	0.141	0.135	0.186	0.304	0.386	0.131	0.139	0.177	0.186
		192	**0.142**	0.147	0.153	0.154	0.158	0.197	0.327	0.386	0.151	0.154	0.195	0.208
		336	**0.154**	0.163	0.169	0.171	0.177	0.213	0.333	0.378	0.161	0.169	0.213	0.210
		720	**0.176**	0.197	0.203	0.210	0.222	0.233	0.351	0.376	0.197	0.201	0.204	0.231
	Traffic	96	**0.345**	0.360	0.410	0.410	0.481	0.576	0.733	2.085	0.376	0.401	0.489	0.599
		192	**0.370**	0.379	0.423	0.423	0.509	0.610	0.777	0.867	0.397	0.413	0.493	0.612
		336	**0.385**	0.392	0.436	0.435	0.534	0.608	0.776	0.869	0.413	0.425	0.496	0.618
		720	**0.426**	0.432	0.466	0.464	0.585	0.621	0.827	0.881	0.444	0.462	0.520	0.654
	Avg. Rank		**1.036**	2.893	5.357	5.071	8.036	7.821	11.429	11.286	2.821	4.607	9.000	8.179
MAE	ETTh1	96	**0.390**	0.400	0.399	0.394	0.448	0.415	0.769	0.612	0.392	0.398	0.607	0.466
		192	**0.414**	0.429	0.416	0.415	0.520	0.446	0.786	0.681	0.418	0.419	0.575	0.479
		336	**0.421**	0.440	0.443	0.427	0.588	0.462	0.784	0.738	0.431	0.442	0.621	0.473
		720	**0.442**	0.468	0.490	0.453	0.612	0.492	0.857	0.782	0.472	0.493	0.736	0.525
	ETTh2	96	**0.332**	0.337	0.353	0.338	0.615	0.374	0.952	0.597	0.341	0.350	0.462	0.423
		192	**0.369**	0.382	0.418	0.381	0.785	0.446	1.542	0.683	0.385	0.427	0.528	0.445
		336	**0.378**	0.384	0.465	0.400	0.980	0.447	1.642	0.747	0.406	0.475	0.556	0.422
		720	**0.416**	0.422	0.551	0.436	1.487	0.469	1.619	0.783	0.470	0.545	0.667	0.424
	ETTm1	96	0.340	0.346	0.343	0.348	0.387	0.390	0.560	0.510	**0.339**	0.353	0.434	0.381
		192	**0.365**	0.370	**0.365**	0.375	0.419	0.415	0.619	0.537	**0.365**	0.369	0.500	0.403
		336	0.385	0.392	0.386	0.388	0.449	0.425	0.741	0.655	**0.382**	0.391	0.549	0.438
		720	**0.408**	0.420	0.421	0.422	0.532	0.458	0.845	0.724	0.414	0.414	0.560	0.488
	ETTm2	96	0.253	0.256	0.260	0.255	0.393	0.271	0.462	0.507	**0.252**	0.259	0.331	0.307
		192	**0.290**	0.296	0.303	0.293	0.519	0.318	0.586	0.673	**0.290**	0.299	0.374	0.352
		336	0.325	0.329	0.342	0.327	0.732	0.364	0.871	0.845	**0.324**	0.338	0.478	0.407
		720	**0.372**	0.394	0.421	0.384	1.170	0.420	1.267	1.451	0.422	0.400	0.554	0.450
	Weather	96	**0.193**	0.198	0.237	0.232	0.224	0.314	0.405	0.556	0.198	0.227	0.258	0.237
		192	**0.237**	0.241	0.282	0.269	0.267	0.329	0.434	0.624	0.242	0.266	0.295	0.279
		336	**0.277**	0.282	0.319	0.301	0.328	0.377	0.543	0.753	0.280	0.304	0.337	0.310
		720	**0.328**	0.334	0.362	0.348	0.363	0.409	0.705	0.934	0.336	0.357	0.399	0.353
	Electricity	96	**0.222**	**0.222**	0.237	0.237	0.234	0.302	0.393	0.449	0.229	0.239	0.294	0.290
		192	**0.240**	**0.240**	0.249	0.248	0.262	0.311	0.417	0.443	0.246	0.253	0.306	0.301
		336	**0.256**	0.259	0.267	0.265	0.283	0.328	0.422	0.443	0.261	0.270	0.324	0.314
		720	**0.278**	0.290	0.301	0.297	0.328	0.344	0.427	0.445	0.293	0.300	0.317	0.329
	Traffic	96	**0.245**	0.249	0.282	0.279	0.265	0.359	0.410	0.468	0.264	0.280	0.317	0.325
		192	**0.255**	0.256	0.287	0.284	0.277	0.380	0.435	0.467	0.277	0.285	0.319	0.332
		336	0.265	**0.264**	0.296	0.290	0.291	0.375	0.434	0.469	0.290	0.292	0.317	0.332
		720	0.287	**0.286**	0.315	0.307	0.325	0.375	0.466	0.473	0.306	0.312	0.326	0.348
	Avg. Rank		**1.214**	2.964	5.643	3.821	8.000	8.214	11.464	11.393	2.857	5.357	9.071	7.571

Table 9: Test MSE and MAE of three types of models by adjusting S in PMformer. (Additional results for Table 3)

Score	PMformer Variants		ETTh1 ($D=7$)				ETTh2 (7)				ETTm1 (7)				ETTm2 (7)			
			96	192	336	720	96	192	336	720	96	192	336	720	96	192	336	720
MSE	Univariate	$S=1$	**0.361**	**0.393**	0.404	0.420	0.272	0.325	0.318	0.371	0.288	0.335	0.358	0.403	0.161	**0.213**	0.265	0.338
	Partial-Multivariate	$1<S<D$	**0.361**	0.396	**0.400**	**0.412**	**0.269**	**0.323**	**0.317**	**0.370**	**0.282**	**0.325**	**0.352**	**0.401**	**0.160**	**0.213**	**0.262**	**0.336**
	Complete-Multivariate	$S=D$	**0.361**	0.395	0.401	0.413	**0.269**	0.325	0.318	0.371	0.299	0.350	0.377	0.402	0.161	**0.213**	0.265	0.338
MAE	Univariate	$S=1$	**0.390**	**0.410**	**0.419**	0.446	0.334	0.373	0.380	0.418	0.344	0.371	0.386	0.409	**0.253**	**0.290**	0.328	0.376
	Partial-Multivariate	$1<S<D$	**0.390**	0.414	0.421	**0.442**	**0.332**	**0.369**	**0.378**	**0.416**	**0.340**	**0.365**	**0.385**	**0.408**	**0.253**	**0.290**	**0.325**	**0.372**
	Complete-Multivariate	$S=D$	**0.390**	0.413	0.420	**0.442**	**0.332**	0.371	0.380	**0.416**	0.353	0.382	0.396	**0.408**	**0.253**	**0.290**	0.327	0.374

Score	PMformer Variants		Weather (21)				Electricity (321)				Traffic (862)			
			96	192	336	720	96	192	336	720	96	192	336	720
MSE	Univariate	$S=1$	**0.141**	0.186	0.237	0.308	0.128	0.146	0.163	0.204	0.368	0.388	0.404	0.441
	Partial-Multivariate	$1<S<D$	0.142	**0.185**	**0.235**	**0.305**	**0.125**	**0.142**	**0.154**	**0.176**	**0.345**	**0.370**	**0.385**	**0.426**
	Complete-Multivariate	$S=D$	0.146	0.192	0.244	0.307	0.129	0.147	0.163	0.204	0.363	0.383	0.394	0.441
MAE	Univariate	$S=1$	0.195	0.239	0.279	0.333	0.223	0.242	0.260	0.297	0.257	0.265	0.277	0.299
	Partial-Multivariate	$1<S<D$	**0.193**	**0.237**	**0.277**	**0.328**	**0.222**	**0.240**	**0.256**	**0.278**	**0.245**	**0.255**	**0.265**	**0.287**
	Complete-Multivariate	$S=D$	0.199	0.243	0.286	0.331	0.228	0.246	0.259	0.297	0.257	0.269	0.276	0.303

Table 10: Test MSE and MAE of PMformer with various inference techniques. (Additional results for Table 4)

Score	Inference Technique	Electricity ($D=321$)				Traffic (862)			
		τ=96	192	336	720	96	192	336	720
MSE	Proposed Technique with $N_I=3$ (Ours)	**0.125**	**0.142**	**0.154**	**0.176**	**0.345**	**0.370**	**0.385**	**0.426**
	Sampling A Subset of Mutually Significant Features	0.132	0.148	0.178	0.205	0.352	0.372	0.386	0.428
	Sampling A Subset of Mutually Insignificant Features	0.135	0.167	0.174	0.235	0.377	0.410	0.410	0.444
MAE	Proposed Technique with $N_I=3$ (Ours)	**0.222**	**0.240**	**0.256**	**0.278**	**0.245**	**0.255**	**0.265**	**0.287**
	Sampling A Subset of Mutually Significant Features	0.231	0.247	0.285	0.302	0.251	0.259	0.267	0.289
	Sampling A Subset of Mutually Insignificant Features	0.237	0.268	0.276	0.329	0.267	0.285	0.289	0.308

Table 11: Main forecasting results of PMformer with standard deviation

	Score	ETTh1 ($D=7$)				ETTh2 (7)				ETTm1 (7)				ETTm2 (7)			
		96	192	336	720	96	192	336	720	96	192	336	720	96	192	336	720
PMformer	MSE	0.361 $\pm$0.001	0.396 $\pm$0.002	0.400 $\pm$0.001	0.412 $\pm$0.001	0.269 $\pm$0.001	0.323 $\pm$0.002	0.317 $\pm$0.001	0.370 $\pm$0.001	0.282 $\pm$0.002	0.325 $\pm$0.001	0.352 $\pm$0.001	0.401 $\pm$0.000	0.160 $\pm$0.001	0.213 $\pm$0.001	0.262 $\pm$0.002	0.336 $\pm$0.001
	MAE	0.390 $\pm$0.001	0.414 $\pm$0.002	0.421 $\pm$0.001	0.442 $\pm$0.002	0.332 $\pm$0.001	0.369 $\pm$0.002	0.378 $\pm$0.001	0.416 $\pm$0.001	0.340 $\pm$0.001	0.365 $\pm$0.001	0.385 $\pm$0.000	0.408 $\pm$0.000	0.253 $\pm$0.001	0.290 $\pm$0.001	0.325 $\pm$0.001	0.372 $\pm$0.001

	Score	Weather (21)				Electricity (321)				Traffic (862)			
		96	192	336	720	96	192	336	720	96	192	336	720
PMformer	MSE	0.142 $\pm$0.000	0.185 $\pm$0.000	0.235 $\pm$0.001	0.305 $\pm$0.001	0.125 $\pm$0.000	0.142 $\pm$0.001	0.154 $\pm$0.001	0.176 $\pm$0.003	0.345 $\pm$0.001	0.370 $\pm$0.001	0.385 $\pm$0.001	0.426 $\pm$0.001
	MSE	0.193 $\pm$0.001	0.237 $\pm$0.000	0.277 $\pm$0.001	0.328 $\pm$0.001	0.222 $\pm$0.001	0.240 $\pm$0.001	0.256 $\pm$0.000	0.278 $\pm$0.003	0.245 $\pm$0.001	0.255 $\pm$0.000	0.265 $\pm$0.000	0.287 $\pm$0.001

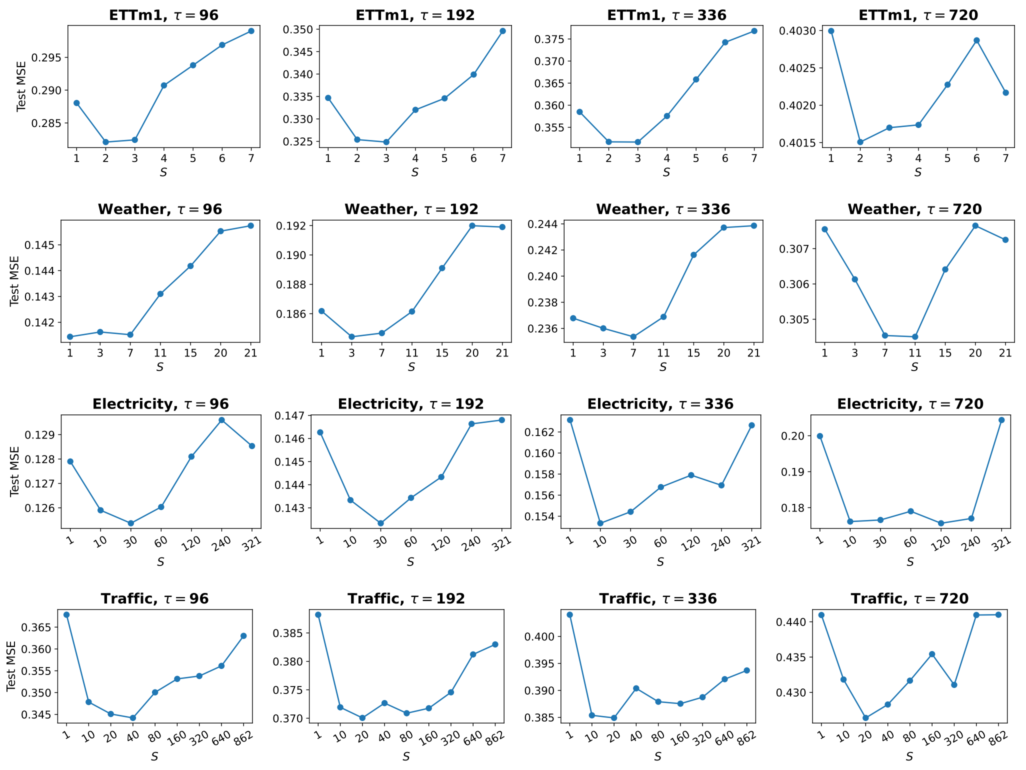


Figure 8: Sensitivity to S. (Additional results for Figure 3)

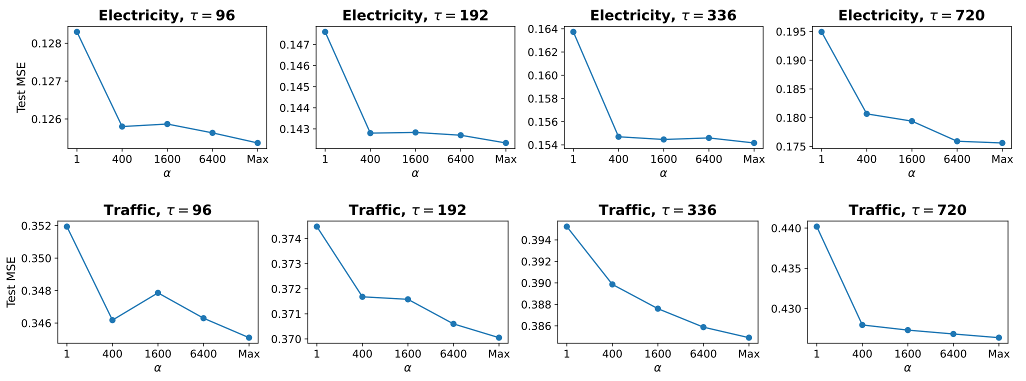


Figure 9: Sensitivity to $|\mathbf{F}^{all}| = \alpha \times N_U$. (Additional results for Figure 4)

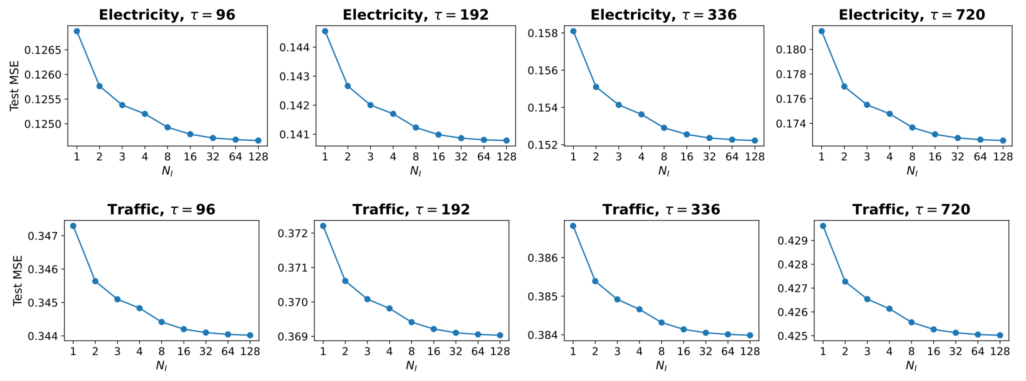


Figure 10: Sensitivity to N_I. (Additional results for Figure 5(a))

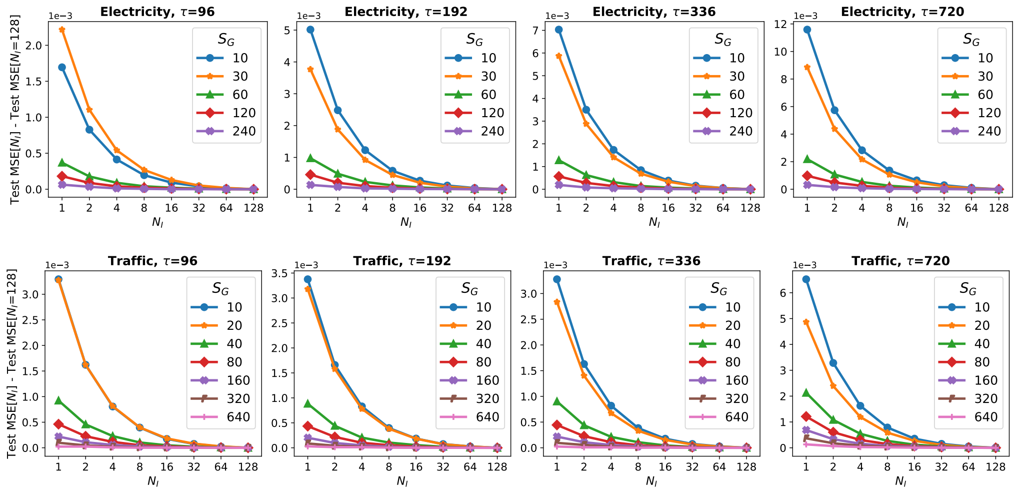


Figure 11: Changes in the effect of N_I on forecasting performance when S increases. (Additional results for Figure 5(b))

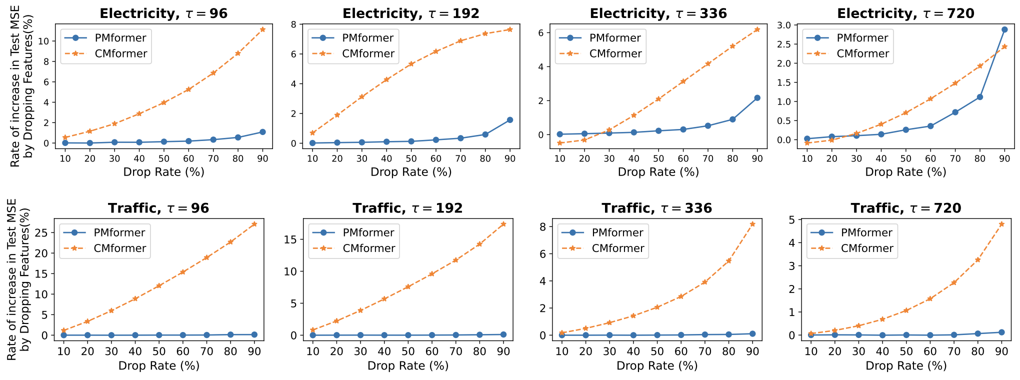


Figure 12: Increasing rate of test MSE by dropping $n\%$ features in PMformer or Complete-Multivariate Transformer (CMformer). (Additional results for Figure 6)

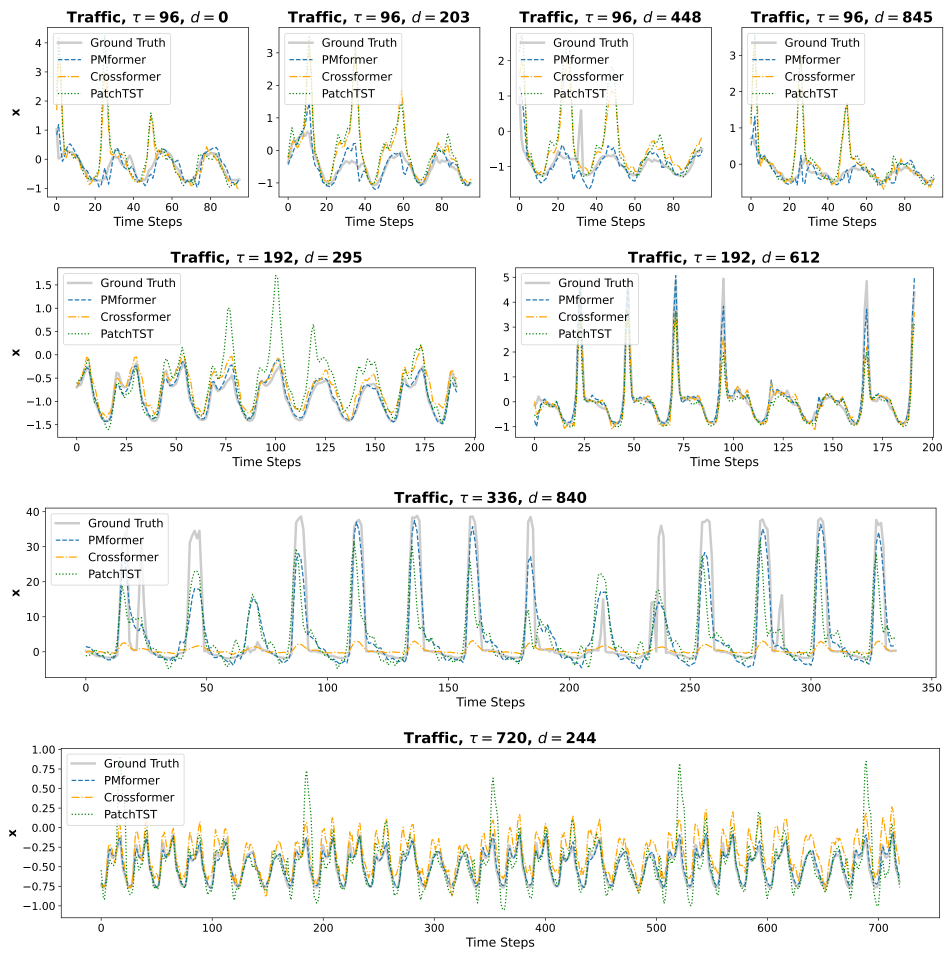


Figure 13: Forecasting results of various segment-based transformers (Crossformer, PatchTST, and PMformer). Dotted lines and dotted-dashed lines denote baselines, dashed lines denote PMformer, and solid lines denote ground truth. τ denotes the length of time steps in future outputs and d denotes a feature index.