arXiv:2408.10483v1 [cs.LG] 20 Aug 2024

PRformer: Pyramidal Recurrent Transformer for Multivariate Time Series Forecasting

Yongbo Yu[1], **Weizhong Yu**[1], **Feiping Nie**[1], **Xuelong Li**[2]

[1]Northwestern Polytechnical University
[2]Institute of Artificial Intelligence (TeleAI),China Telecom Corp Ltd
yongboyu@mail.nwpu.edu.cn, yuwz05@mail.xjtu.edu.c,feipingnie@gmail.com,li@nwpu.edu.cn

Abstract

The self-attention mechanism in Transformer architecture, invariant to sequence order, necessitates positional embeddings to encode temporal order in time series prediction. We argue that this reliance on positional embeddings restricts the Transformer's ability to effectively represent temporal sequences, particularly when employing longer lookback windows. To address this, we introduce an innovative approach that combines Pyramid RNN embeddings(PRE) for univariate time series with the Transformer's capability to model multivariate dependencies. PRE, utilizing pyramidal one-dimensional convolutional layers, constructs multiscale convolutional features that preserve temporal order. Additionally, RNNs, layered atop these features, learn multiscale time series representations sensitive to sequence order. This integration into Transformer models with attention mechanisms results in significant performance enhancements. We present the PRformer, a model integrating PRE with a standard Transformer encoder, demonstrating state-of-the-art performance on various real-world datasets. This performance highlights the effectiveness of our approach in leveraging longer lookback windows and underscores the critical role of robust temporal representations in maximizing Transformer's potential for prediction tasks. Code is available at this repository: https://github.com/usualheart/PRformer.

Introduction

Time series forecasting finds extensive applications in various domains such as meteorology, transportation, energy, finance, etc. In earlier years, models based on Recurrent Neural Networks (Cho et al. 2014; Lai et al. 2018; Salinas et al. 2020) were popular for time series forecasting. In recent years, following the success of Transformer models in natural language processing(Vaswani et al. 2017) and image processing (Dosovitskiy et al. 2021), researchers have explored the application of Transformers to time series prediction, achieving notable success with models such as Autoformer (Wu et al. 2021)and FedFormer (Zhou et al. 2022). However, recent studies have revealed that Transformers are outperformed by linear models in time series prediction, exemplified by DLinear (Zeng et al. 2023). This paper posits that the primary reason for this lies in the existing Transformers' dependence on time encoding to determine temporal positions, which may not effectively capture sequential patterns, resulting in deficiencies in time series prediction. This limitation also hinders the utilization of longer time windows, as increasing the window length often leads to a drastic performance decline. Additionally, the computational complexity of Transformers concerning sequence length is $O(n^2d)$, preventing linear expansion with length and thus restricting the use of longer lookback windows.

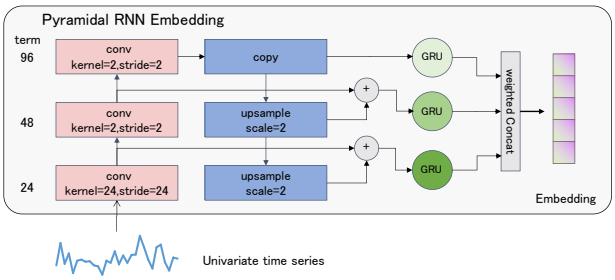


Figure 1: Pyramidal RNN Embedding (PRE) Architecture.The actual pyramid structure is generated based on the settings.

We contend that Recurrent Neural Networks (RNNs) may offer solutions to these issues with Transformers. On one hand, the inherent structure of RNNs is well-suited for sequential data, capable of handling variable-length time steps, with hidden units serving as representations of time series. On the other hand, the complexity of RNNs is $O(nd^2)$, linearly expanding with sequence length, making them advantageous for processing longer time series. These characteristics of RNNs address the shortcomings of Transformer models. However, RNNs have their own limitations, such as fixed hidden layer sizes, and issues like gradient vanishing and exploding(Pascanu, Mikolov, and Bengio 2013). Therefore, the integration of RNNs and Transformers holds significant importance in achieving complementary strengths for effective time series forecasting.

In addition to sequentiality, time series data often exhibit multi-periodic characteristics. Convolutional Neural Networks, through multiple convolutional layers, can progressively construct larger-scale features, thereby capturing dependencies in time series data with long periods(Wu et al. 2022a). To capture multi-period dependencies in time series data, this paper proposes constructing a pyramidal multiscale convolutional feature(Lin et al. 2017) for time series.

The pyramidal structure consists of two parts: bottom-up and top-down. The bottom-up process constructs various cycles from small to large through one-dimensional convolution, representing the elevation of scale. In convolutional neural networks, higher layers of convolution often represent abstract semantic features, which in time series data imply changes in large-scale temporal cycles. Macroscopic changes are generally more deterministic than microscopic changes, making them easier to learn the underlying logic of changes(Hoel, Albantakis, and Tononi 2013). The top-down process gradually upsamples from the top layer downward. Since small-scale changes usually do not deviate too much from large-scale changes, this approach can transfer features from large cycles to small cycles as constraints. Finally, at each scale, the bottom-up and top-down features are added to obtain convolutional feature representations that are both finely detailed and constrained by macroscopic information.

Based on these considerations, this paper proposes an improved approach to Transformer prediction: first, construct pyramidal multi-scale convolutional features for univariate time series, then input the convolutional features of each scale into an RNN(Cho et al. 2014) to learn multi-scale time series representations. By convoluting before using RNN, it can transform the original long sequence data into short sequence data, favorable for fully leveraging the advantages of RNN in modeling time series data. This method of learning univariate time series representations is named Pyramidal Rnn Embedding(PRE). PRE is combined with the standard Transformer encoder, learning dependencies between multiple variables through a multi-head self-attention mechanism, and finally outputting predictions through linear projection, resulting in a model named PRformer.

Our contributions are as follows:

- **Proposing a method for extracting multi-scale temporal representations.** We introduce a novel module called Pyramidal RNN Embedding (PRE), which learns multi-scale temporal features by combining a pyramidal structure with RNN. PRE captures temporal dependencies at different periods and scales, enabling it to obtain time series representations that fuse multi-scale dependency information. The computational complexity of PRE grows linearly with sequence length, ensuring high computational efficiency.
- **Proposing the use of PRE to enhance the performance of Transformer predictors.** We employ PRE to learn representations of univariate time series, which serve as input to the Transformer model. This approach effectively addresses the deficiencies of Transformers in capturing positional information relationships within sequences. Through extensive experiments combining PRE with three Transformer variants, we validate the significant improvement that PRE brings to Transformer-based predictors.
- **Proposing PRformer, a time series prediction model that combines PRE with the standard Transformer encoder.** By integrating PRE with the Transformer encoder, we introduce PRformer, a powerful time series prediction model. PRformer achieves state-of-the-art performance on several real-world datasets, surpassing existing Transformer-based architecture predictors and even outperforming linear prediction models. Moreover, we observe that PRformer's performance improves as the sequence length increases, highlighting its scalability and effectiveness in handling long-term dependencies.

Related work

Transformer-Based Time Series Forecasting The Transformer has achieved widespread success in various tasks such as natural language processing (Vaswani et al. 2017) and computer vision (Dosovitskiy et al. 2021). In recent years, the Transformer has also been applied to time series forecasting with continuous improvements,such as Informer (Zhou et al. 2021),Autoformer (Wu et al. 2021) ,Pyraformer (Liu et al. 2021),FedFormer (Zhou et al. 2022),PatchTST (Nie et al. 2023) and so on. The improvements on the Transformer mainly involve designing new attention modules (Wu et al. 2021; Zhou et al. 2022; Liu et al. 2021; Zhou et al. 2021) and modifying the network architecture.

RNN for Time Series Forecasting RNN has a strong affinity with sequence data in terms of its structure. With their gating mechanisms, RNN can adaptively regulate the influence of historical information by selectively depending on or ignoring past states. For a significant period, RNN-based methods have held an important position in the field of time series forecasting (Lai et al. 2018; Salinas et al. 2020; Hewage et al. 2020). The use of RNN in time series forecasting helps capture the dependencies across different periods. GRU(Cho et al. 2014) controls the hidden state h over time through reset and update gates, enabling it to learn "soft" periods. However, due to the fixed size of hidden units, RNN suffers from the gradient vanishing problem when modeling extremely long sequences. Nonetheless, RNN still possesses unique advantages, as its computational complexity is $O(nd^2)$,which is linear with respect to the length of the time series. Recently, some work try to mitigate the gradient vanishing problem(Li et al. 2018) and applied RNN to long sequence modeling(Ma et al. 2023; Peng et al. 2023).

Patch-Based Time Series Forecasting In computer vision, ViT(Dosovitskiy et al. 2021) is a milestone work that segments images into 16x16 patches and feeds them into the Transformer model, resulting in significant success. Patch-based approaches have also been influential in natural language processing, speech, and other fields(Bao et al. 2021; Hsu et al. 2021). In the domain of time series, PatchTST(Nie et al. 2023) first divides time series into patches and feeds them into the Transformer model, achieving improved prediction accuracy compared to previous Transformer-based approaches. Patching is similar to one-dimensional convolution, allowing the extraction of local semantic information from the data. Moreover, patching combined with convolution can divide long sequences into shorter segments, which facilitates their modeling using RNN(Lai et al. 2018; Hewage et al. 2020).

Multi-Scale Feature Extraction (Hoel, Albantakis, and Tononi 2013) proposes that there are better causal relationships at a macroscopic level in terms of the causal changes in complex systems. This insight can be applied to time series

forecasting, where incorporating features from the macroscopic level may lead to better predictions. In the field of image processing, feature pyramid networks(Lin et al. 2017) construct multi-scale features by combining convolution and upsampling, resulting in improved object detection accuracy. This concept also has important applications in weather forecasting, where constructing multi-scale features from storm data has been shown to enhance storm prediction(Yang and Yuan 2023). In this paper, we propose constructing pyramidal convolutional features for univariate time series data to represent different time scales. RNN is applied to extract temporal representations from each scale, and the resulting representations are fused to obtain a multi-scale temporal representation, thereby improving time series forecasting performance.

PRformer

We primarily address the task of predicting multivariate time series, formalized as follows: Given a historical multivariate time series $X \in \mathbb{R}^{L \times C}$ as input, the model aims to output future time series predictions $Y \in \mathbb{R}^{H \times C}$. Here, C represents the number of variables in the time series, L signifies the time step length of the historical sequence, and H indicates the time step length for prediction.

Model Architecture

The proposed PRformer is illustrated in Figure 2.The model employs a vanilla Transformer encoder as its core architecture, comprising an embedding layer, multi-head attention, and a final projection layer. We introduce a Pyramid Recurrent Neural Network to extract embeddings for each univariate time series, emphasizing learning representations. The Transformer encoder focuses on learning relationships between multiple variables, obtaining representations for multivariate time series. Following the Transformer encoding, the final prediction results are obtained through a simple linear projection layer.

In the PRformer, the overall formalized process for predicting the future sequence $Y \in \mathbb{R}^{H \times C}$ based on the historical sequence $X \in \mathbb{R}^{L \times C}$ is expressed as follows:

$$\begin{aligned} \mathbf{h}_i^0 &= \mathrm{PREmbedding}(\mathbf{X}_{:,i}), \quad i = 0, ..., C-1 \\ \mathbf{H}^{l+1} &= \mathrm{Encoder}(\mathbf{H}^l), \quad l = 0, \cdots, L-1, \\ \hat{\mathbf{Y}}_{:,i} &= \mathrm{Projection}(\mathbf{h}_i^L), \quad i = 0, ..., C-1 \end{aligned} \tag{1}$$

Here, $H = \{h_1, ..., h_C\} \in \mathbb{R}^{C \times D}$ represents embeddings for C variables, with each variable embedding having dimension D. The embeddings, generated through Pyramid RNN embedding(PRE), undergo processing in the Transformer encoder. In each encoder layer, the multi-head attention mechanism is employed to learn relationships between multiple variables and generate new representations. As PRE embedding already captures the sequential relationships of the sequence, additional positional embeddings are unnecessary for representing the entire sequence of a univariate variable.

Pyramid RNN Embedding (PRE)

The overall architecture of Pyramid RNN Embedding (PRE) is shown in Figure 1. The input of PRE is a univariate time series, and the output is the embedded representation of the time series. The PRE consists of a Pyramid Temporal Convolution module and a Multi-Scale RNN module. The Pyramid Temporal Convolution module is used to learn multi-scale convolutional features of the temporal data, and compress the length of the time series through convolution, making it easier for the RNN to handle. The Multi-Scale RNN module is used to learn the sequence dependencies at different scales on the temporal dimension, and obtain the final multi-scale temporal embedding.

Pyramid Convolution Block Periodicity is an important characteristic of time series data, and time series often have multiple different sized periods. In many cases, large periods contain small periods, such as weeks, months, seasons, and years. Previous time series predictors did not pay much attention to the relationship between large and small periods. In fact, there are dependencies between different-sized periods, and large periods may determine the fluctuation range of small periods, such as the influence of seasons on the climate. In addition, large-scale changes have less randomness compared to small-scale changes, making them less susceptible to noise interference and potentially yielding more robust prediction results. Based on this consideration, we propose the Pyramid Temporal Convolution module, and subsequent experiments have shown that it helps improve the performance of time series prediction.

Bottom-up pathway Through multiple layers of 1D convolution, a temporal feature pyramid is constructed, aligning each layer's data point period with the predetermined periodic lengths. Given the multiplicative growth pattern of the configured periodic lengths, the kernel size for each convolutional layer is equivalent to the multiple of the current layer's period and the subsequent layer's period length. The stride size for each layer's convolution is set equal to the kernel size. To facilitate downsampling, the output channels for all convolutional layers are maintained uniformly. The following equation illustrates the process of a univariate sequence x_{l-1} from the (l-1)-th layer through the one-dimensional convolutional kernel of the l-th layer.

$$\begin{aligned} \mathbf{K}_l &= \left\lfloor \frac{\text{Window}^l}{\text{Window}^{l-1}} \right\rfloor \\ \mathbf{x}_l &= \mathrm{Conv1d}(\mathbf{x}_{l-1}, \text{kernel} = \mathbf{K}_l, \text{stride} = \mathbf{K}_l) \end{aligned} \tag{2}$$

Top-down pathway and lateral connections The top-down pathway samples from higher levels of the pyramid layers to obtain features with larger time scales and stronger semantics. Then, the same sized temporal features from the bottom-up pathway and the top-down pathway are fused through lateral connections. The features from the bottom-up pathway have smaller scale and higher resolution. Through fusion, the large-scale temporal changes can guide the small-scale changes, while maintaining a smaller time series scale to improve prediction accuracy. The following equation shows the process of generating the feature of the $(l-1)$th layer.

$$\begin{aligned} \mathbf{x}_{l-1}^{'} &= \mathrm{upsample}(\mathbf{x}_l^{'}) \\ \mathbf{x}_{l-1}^{\text{out}} &= \mathbf{x}_{l-1}^{\prime} + \mathbf{x}_{l-1} \end{aligned} \tag{3}$$

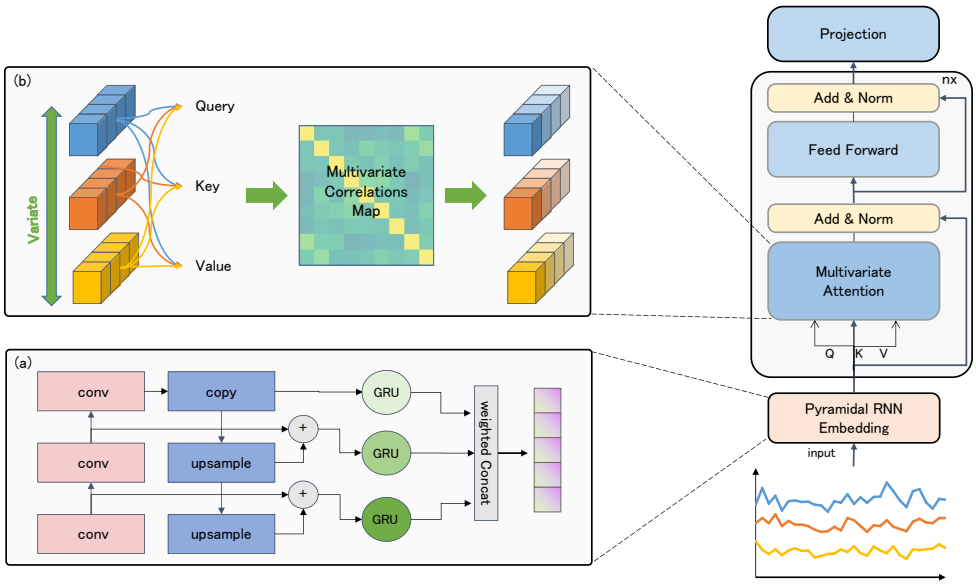


Figure 2: The overall architecture of PRformer utilizes a Transformer encoder as its backbone, culminating in the generation of prediction results through a simple linear projection. (a) Pyramidal RNN Embedding (PRE) block. The initial sequences of diverse variables are independently fed into PRE block to acquire distinct representations in the form of embeddings. (b) multi-head self-attention is employed on the embeddings of multiple variables to capture intricate interdependencies among them.

Multi-Scale RNN Block The convolutional features from the Pyramid Convolution module still have a temporal structure for each scale, but the sequence length has been greatly reduced compared to the original sequence, which makes it easier to be processed by the RNN. The Multi-Scale RNN module is used to learn the sequence dependencies at different scales on the temporal dimension, and obtain the final multi-scale temporal embedding, where D represents the embedding dimension of each variable. As shown in Figure 1, the convolutional features of each scale are processed by a separate GRU module. The hidden dimension of the GRU is D divided by the number of scales, and the last hidden unit is used as the temporal representation of that scale. The temporal representation $h^{(i)}$ of each scale is first multiplied by the weight coefficient β_i, and then concatenated to form the final multi-scale temporal embedding h. The weight coefficient β_i is obtained by applying a softmax with temperature parameter control to α_i, where α_i is initialized to $1/l$ and can be adaptively adjusted through training gradients. The temperature parameter T of the softmax is used to amplify the differences in α_i, making the coefficient β_i sharper and increasing the differences in weight coefficients between different scale temporal representations.

$$
\begin{aligned}
h^{(i)} &= \mathrm{GRU}(x_i^{\mathrm{out}}, D|\text{layer\_num}), \quad i = 1, \ldots, l \\
\beta_i &= \frac{\exp(\alpha_i/T)}{\sum_{j=1}^{l} \exp(\alpha_j/T)}, \quad i = 1, \ldots, l \\
h &= \mathrm{concat}(\beta_1 \cdot h^{(1)}, \beta_2 \cdot h^{(2)}, \ldots, \beta_l \cdot h^{(l)})
\end{aligned} \tag{4}
$$

Pyramid Convolutional Layer Configuration Method The configuration of pyramid convolutional layers is determined based on a set of periodic lengths, where each layer represents a distinct period. These layers extend from the bottom to the top, with periodic lengths increasing multiplicatively. The selection of periodic lengths is contingent upon the frequency of the time series dataset and the specific application scenario. For instance, in the case of the electrical power dataset ETTh1, with a temporal sampling frequency of 1 hour, one may opt for a set of periodic lengths such as 24, 48, and 96, corresponding to cycles of 1 day, 2 days, and 4 days, respectively. This configuration of the pyramid convolutional layers aligns more closely with real-world conditions, facilitating a more effective capture of cyclic dependencies.

Transformer Encoder Multivariate Attention

After the multivariate time series is processed by PRE, we obtain the embedded token $H = \{h_1, ..., h_C\} \in \mathbb{R}^{C \times D}$, where C represents the number of variables and D represents the embedding dimension of each variable. The obtained multi-variate embedded token H is then input into a regular Transformer encoder for processing. Each head in the multi-head attention mechanism linearly projects H to obtain queries matrices Q_k, keys matrices K_k, and values matrices V_k, where d_k represents the projected dimension of each head. Then, a scaled dot product is applied to obtain the attention output $\mathbf{O}_k \in \mathbb{R}^{C \times d_k}$:

$$
(\mathbf{O}_k)^T = \mathrm{Softmax}\left(\frac{Q_k K_k^T}{\sqrt{d_k}}\right) V_k \tag{5}
$$

The multi-head attention block also includes LayerNorm layers and a feed forward network with residual connections. It generates the representation $z^{(i)} \in \mathbb{R}^{C \times D}$ for each variable. In each Encoder layer, the multi-head attention mechanism is able to learn the relationships between different variables and generate new representations for each variable. Finally, a channel-wise linear projection is applied to obtain the prediction results $(\hat{y}_{L+1}^{(i)}, ..., \hat{y}_{L+T}^{(i)}) \in \mathbb{R}^{1 \times H}$ for each variable. The prediction results are concatenated to obtain the final prediction result $\hat{Y} \in \mathbb{R}^{H \times C}$.

Loss Function and Normalization

We choose to use the MAE loss function to train the model. The MAE loss function is defined as follows:

$$\mathcal{L}(Y, \bar{Y}) = \frac{1}{HC} \sum_{t=1}^{H} \sum_{i=1}^{C} |\bar{y}_t^{(i)} - y_t^{(i)}| \tag{6}$$

Instance Normalization. In this paper, we use reversible instance normalization(Kim et al. 2022)(RevIN)to process the time series data to alleviate distribution shift issues. Similar to standardization, RevIN scales the sequence data of each variable to $N(\beta_i, \gamma_i)$ through learnable parameters. RevIN is applied to normalize the time series data before entering the model, and then applied again to reverse normalize the predicted results to obtain the final prediction results.

Complexity Analysis of PRformer

Table 1: Complexity analysis of different Transformer prediction models. S is the stride length of PatchTST.

Methods	Time	Memory
Transformer	$O(L^2)$	$O(L^2)$
Informer	$O(L \log L)$	$O(L \log L)$
Reformer	$O(L \log L)$	$O(L \log L)$
PatchTST	$O(\frac{L^2}{S^2})$	$O(\frac{L^2}{S^2})$
PRformer	$O(\frac{L}{W} + D^2)$	$O(\frac{L}{W} + D^2)$
PRE+Informer	$O(\frac{L}{W} + D \log D)$	$O(\frac{L}{W} + D \log D)$
PRE+Reformer	$O(\frac{L}{W} + D \log D)$	$O(\frac{L}{W} + D \log D)$

The PRformer primarily consists of two components: the PRE and the Transformer encoder. The time complexity of the PRE is related to the length of the historical sequence and is $O(\frac{L}{W})$, where W is the period length of the bottom layer in the pyramid structure. For some real-world data recorded in hours, W is typically set to 24, representing a daily period, significantly reducing the time complexity. The space complexity of PRE is associated with the dimensionality of the embedding, D, and is $O(D)$. Both the time and space complexities of the Transformer encoder are $O(D^2)$. Typically, D is chosen such that $D \leq L$, hence the overall time complexity of the PRformer is $O(\frac{L}{W} + D^2)$ and the space complexity is $O(\frac{L}{W} + D^2)$. The table 1 compares the time complexity and memory usage of various Transformer models. Compared to the state-of-the-art Transformer predictor model, PatchTST, and the original Transformer, the PRformer has lower complexity. It is noteworthy that the complexity of the PRformer grows linearly with sequence length, enabling it to utilize longer lookback windows under equivalent hardware conditions, thereby enhancing predictive performance. Furthermore, the key module of PRformer, PRE, can be combined with Informer or Reformer to further reduce complexity and memory usage. Experiments in section show that such combinations can also significantly improve the predictive performance of the original models.

EXPERIMENTS

In various time series forecasting applications, we comprehensively evaluate PRformer to validate the generalization of the proposed model. We also assess the effectiveness of the pyramidal RNN embedding layer in enhancing various Transformer predictors, confirming the ability of the pyramidal RNN embedding to learn effective representations of univariate time series.

Datasets We assess the performance of PRformer on 8 popular datasets, including Weather, Traffic, Electricity, Solar-Energy and 4 ETT datasets (ETTh1, ETTh2, ETTm1, ETTm2). These datasets represent diverse domains and intervals, offering a comprehensive assessment environment.The characteristics of these datasets are summarized in Table 2. Of particular note are the Electricity and Traffic datasets, which consist of over 300 variables. For such datasets, modeling the dependencies between multiple variables is crucial, and our proposed PRformer excels in this aspect.

Table 2: Summary of datasets.

Datasets	Weather	Traffic	Electricity	ETTh1	ETTh2	ETTm1	ETTm2	Solar-Energy
Channels	21	862	321	7	7	7	7	137
Frequency	10 mins	1 hour	1 hour	1 hour	1 hour	15 mins	15 mins	10 mins
Timesteps	52,696	17,544	26,304	17,420	17,420	69,680	69,680	36601

Baselines and metrics We select SOTA Transformer-based models and some representative non-Transformer-based model in the time series forecasting field as baselines, including iTransformer (Liu et al. 2024), PatchTST (Nie et al. 2023), Crossformer (Zhang and Yan 2022), FEDformer (Zhou et al. 2022), Autoformer (Wu et al. 2021), Informer (Zhou et al. 2021),Dlinear (Zeng et al. 2023) and TimesNet (Wu et al. 2022a).Two commonly used evaluation metrics, Mean Squared Error (MSE) and Mean Absolute Error (MAE), serve as the assessment criteria.

Performance promotion with PRE

To validate the effectiveness and generalization capability of PRE embedding, we apply PRE to Transformer(Vaswani et al. 2017) and its variants (primarily addressing the quadratic complexity issue of the self-attention mechanism): Reformer (Kitaev, Kaiser, and Levskaya 2020), Informer(Zhou et al. 2021),and Flowformer (Wu et al. 2022b). The experimental results reveal surprising and promising findings.Table 3 present the performance improvements achieved by applying PRE. Notably, PRE consistently enhances the performance of various Transformer variants. Overall, a performance improvement of 45.88% is achieved on the Transformer, 50.79%

Table 3: Performance promotion obtained by PRE.

Models		Transformer (2017)		Reformer (2020)		Informer (2021)		Flowformer (2022)	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Electricity	Original	0.277	0.372	0.338	0.422	0.311	0.397	0.267	0.359
	+PRE	0.156	0.247	0.163	0.258	0.162	0.257	0.161	0.256
	Promotion	43.86%	33.74%	51.80%	38.98%	47.92%	35.38%	39.63%	28.81%
Traffic	Original	0.665	0.363	0.741	0.422	0.764	0.416	0.750	0.421
	+PRE	0.383	0.239	0.394	0.273	0.394	0.273	0.392	0.272
	Promotion	42.44%	34.30%	46.81%	35.23%	48.46%	34.44%	47.79%	35.28%
Weather	Original	0.657	0.572	0.803	0.656	0.634	0.548	0.286	0.308
	+PRE	0.225	0.257	0.224	0.264	0.231	0.269	0.227	0.268
	Promotion	65.79%	55.16%	72.14%	59.79%	63.61%	50.84%	20.63%	12.92%
Promotion Average		45.88%		50.79%		46.78%		30.84%	

on the Reformer, 46.78% on the Informer, and 30.84% on the Flowformer. This indicates that PRE can effectively represent univariate time series learning and enhance the performance of Transformer-based predictors. In the future, the concept of PRE can be widely practiced in Transformer-based predictors to take advantage of the growing efficient attention mechanisms.

Long-term Time Series Forecasting

The forecasting results for multivariate long-term time series are presented in Table 4, with the best outcomes highlighted in bold and the second-best underlined. The proposed PRformer model has achieved state-of-the-art performance across most scenarios, securing the top performance in 25 out of 32 MSE metrics and in 28 out of 32 MAE metrics. Even when compared to the previously state-of-the-art Transformer-based forecaster, PatchTST, the PRformer demonstrated comprehensive performance improvements. This enhancement in performance was particularly notable in high-dimensional time series datasets, such as the Traffic and Solar-Energy datasets. Against the Crossformer model, which explicitly captures multivariate correlations, the PRformer's performance improvement exceeded 70% in some cases, showcasing its effectiveness in modeling dependencies among multiple variables. We attribute this efficiency to the use of PRE for time series representation learning, which alleviates the limitations of Transformer positional encodings, thereby leading to improved performance. This is also why the PRformer exhibited comprehensive advantages over advanced linear models and CNN-based models, underscoring the success of the PRformer design.

Model Analysis

Ablation Experiment To validate the effectiveness of the components of PRformer, we conducted ablation experiments. Three variants of PRformer were tested: (1) PRformer V1: we replaced the multi-head self-attention module in the Transformer with a linear layer, (2) PRformer V2: we replaced the generation of time series representations for each variable with a linear projection instead of PRE, (3) PRformer V3: we only used the bottom layer of the temporal convolution pyramid to extract univariate time series representations. We used PatchTST as the state-of-the-art benchmark for the Transformer-based model. If the ablated version outperformed PatchTST, it is indicated in bold numbers. From Table 5, it can be observed that 12/32 instances of PRformer V2 showed improvement, 18/32 instances of PRformer V1 showed improvement, and 23/32 instances of PRformer V3 showed improvement. The best performance was achieved by the PRformer model proposed in this paper, with 26/32 instances showing improvement. By comparing PRformer V2 and PRformer, we validated the effectiveness of the PRE embedding compared to a simple linear layer; by comparing PRformer V1 and PRformer, we confirmed the effectiveness of the Transformer's self-attention module for learning relationships between multiple variables; by comparing PRformer V3 and PRformer, we verified the effectiveness of building multiscale temporal representations through the pyramid structure.

Lookback Window Analysis

The ability to use a longer review window reflects the model's ability to capture long-term dependency relationships. Many previous Transformer-based models have difficulty in utilizing longer review windows, as evidenced by unchanged or decreased predictive performance when the review window is increased, leading to doubts about the quality of Transformer models. PRformer uses multiscale pyramidal convolution and RNN to learn univariate time series representation. Since RNN is structurally aware of the sequence order of sequence data and has a high compatibility with time series data, and pyramidal convolution can turn long sequences into shorter sequences to alleviate the gradient vanishing problem of RNN, we speculate that PRformer has the ability to use longer review windows.The experimental results in the Figure 3 confirm this speculation: as the review window length increases, the prediction error of the ordinary Transformer model remains unchanged or increases, while the prediction error of PRformer shows a significant downward trend. This indicates that PRformer can utilize longer time series to improve overall predictive performance.

Model Efficiency

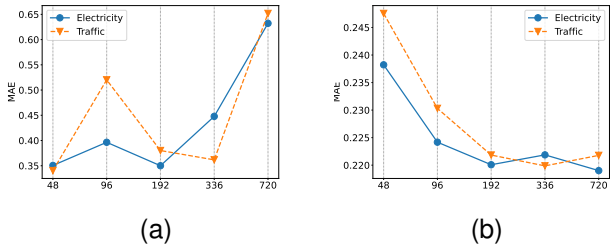
To verify the actual efficiency of PRformer, we compared

Table 4: Results for multivariate long-term series forecasting on seven datasets with input length $L = 720$ and prediction length $H \in (96, 192, 336, 720)$. A lower value indicates better performance, the best results are highlighted in bold and the second are underlined.

Models		PRformer (2024)		iTransformer (2024)		PatchTST (2023)		Crossformer (2023)		FEDformer (2022)		Autoformer (2021)		Informer (2021)		Dlinear (2023)		TimesNet (2023)	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE		
ETTm1	96	**0.278**	**0.333**	0.334	0.368	0.293	0.346	0.404	0.426	0.326	0.390	0.505	0.475	0.626	0.560	0.299	0.343	0.338	0.375
	192	**0.324**	**0.361**	0.377	0.391	0.333	0.370	0.450	0.451	0.365	0.415	0.553	0.496	0.725	0.619	0.335	0.365	0.374	0.387
	336	**0.362**	**0.384**	0.426	0.420	0.369	0.392	0.532	0.515	0.392	0.425	0.621	0.537	1.005	0.741	0.369	0.386	0.410	0.411
	720	0.426	0.425	0.491	0.459	**0.416**	**0.420**	0.666	0.589	0.446	0.458	0.671	0.561	1.133	0.845	0.425	0.421	0.478	0.450
ETTm2	96	**0.162**	**0.245**	0.180	0.264	0.166	0.256	0.287	0.366	0.180	0.271	0.255	0.339	0.355	0.462	0.167	0.260	0.187	0.267
	192	**0.219**	**0.286**	0.250	0.309	0.223	0.296	0.414	0.492	0.252	0.318	0.281	0.340	0.595	0.586	0.224	0.303	0.249	0.309
	336	**0.272**	**0.326**	0.311	0.348	0.274	0.329	0.597	0.542	0.324	0.364	0.339	0.372	1.270	0.871	0.281	0.342	0.321	0.351
	720	**0.359**	**0.383**	0.412	0.407	0.362	0.385	1.730	1.042	0.410	0.420	0.422	0.419	3.001	1.267	0.397	0.421	0.408	0.403
ETTh1	96	**0.354**	**0.383**	0.386	0.405	0.370	0.400	0.423	0.448	0.376	0.415	0.449	0.459	0.941	0.769	0.375	0.399	0.384	0.402
	192	**0.397**	**0.410**	0.441	0.436	0.413	0.429	0.471	0.474	0.423	0.446	0.500	0.482	1.007	0.786	0.405	0.416	0.436	0.429
	336	0.427	**0.428**	0.487	0.458	**0.422**	0.440	0.570	0.546	0.444	0.462	0.521	0.496	1.038	0.784	0.439	0.443	0.491	0.469
	720	0.489	0.492	0.503	0.491	**0.447**	**0.468**	0.653	0.621	0.469	0.492	0.514	0.512	1.144	0.857	0.472	0.490	0.521	0.500
ETTh2	96	**0.268**	**0.327**	0.297	0.349	0.274	0.337	0.745	0.584	0.332	0.374	0.358	0.397	1.549	0.952	0.289	0.353	0.340	0.374
	192	**0.332**	**0.370**	0.380	0.400	0.341	0.382	0.877	0.656	0.407	0.446	0.456	0.452	3.792	1.542	0.383	0.418	0.402	0.414
	336	0.361	0.395	0.428	0.432	**0.329**	**0.384**	1.043	0.731	0.400	0.447	0.482	0.486	4.215	1.642	0.448	0.465	0.452	0.452
	720	0.396	0.429	0.427	0.445	**0.379**	**0.422**	1.104	0.763	0.412	0.469	0.515	0.511	3.656	1.619	0.605	0.551	0.462	0.468
Electricity	96	**0.127**	**0.217**	0.148	0.240	0.129	0.222	0.151	0.251	0.186	0.302	0.201	0.317	0.304	0.393	0.140	0.237	0.168	0.272
	192	0.148	**0.237**	0.162	0.253	**0.147**	0.240	0.163	0.262	0.197	0.311	0.222	0.334	0.327	0.417	0.153	0.249	0.184	0.289
	336	**0.161**	**0.252**	0.178	0.269	0.163	0.259	0.195	0.288	0.213	0.328	0.231	0.338	0.333	0.422	0.169	0.267	0.198	0.300
	720	**0.185**	**0.275**	0.225	0.317	0.197	0.290	0.224	0.316	0.233	0.344	0.254	0.361	0.351	0.427	0.203	0.301	0.220	0.320
Traffic	96	**0.353**	**0.222**	0.395	0.268	0.360	0.249	0.522	0.290	0.576	0.359	0.613	0.388	0.733	0.410	0.410	0.282	0.593	0.321
	192	**0.372**	**0.233**	0.417	0.276	0.379	0.256	0.530	0.293	0.610	0.380	0.616	0.382	0.777	0.435	0.423	0.287	0.617	0.336
	336	**0.385**	**0.241**	0.433	0.283	0.392	0.264	0.558	0.305	0.608	0.375	0.622	0.337	0.776	0.434	0.436	0.296	0.629	0.336
	720	**0.421**	**0.258**	0.467	0.302	0.432	0.286	0.589	0.328	0.621	0.375	0.660	0.408	0.827	0.466	0.466	0.315	0.640	0.350
Weather	96	**0.144**	**0.187**	0.174	0.214	0.149	0.198	0.158	0.230	0.238	0.314	0.266	0.336	0.354	0.405	0.176	0.237	0.172	0.220
	192	**0.188**	**0.233**	0.221	0.254	0.194	0.241	0.206	0.277	0.275	0.329	0.307	0.367	0.419	0.434	0.220	0.282	0.219	0.261
	336	**0.241**	**0.274**	0.278	0.296	0.245	0.282	0.272	0.335	0.339	0.377	0.359	0.395	0.583	0.543	0.265	0.319	0.280	0.306
	720	0.326	**0.332**	0.358	0.349	**0.314**	0.334	0.398	0.418	0.389	0.409	0.419	0.428	0.916	0.705	0.323	0.362	0.365	0.359
Solar-Energy	96	**0.171**	**0.201**	0.203	0.237	0.234	0.286	0.310	0.331	0.242	0.342	0.884	0.711	0.206	0.229	0.290	0.378	0.250	0.292
	192	**0.193**	**0.215**	0.233	0.261	0.267	0.310	0.734	0.725	0.285	0.380	0.834	0.692	0.235	0.258	0.320	0.398	0.296	0.318
	336	**0.217**	**0.228**	0.248	0.273	0.290	0.315	0.750	0.735	0.282	0.376	0.941	0.723	0.261	0.271	0.353	0.415	0.319	0.330
	720	**0.229**	**0.237**	0.249	0.275	0.289	0.317	0.769	0.765	0.357	0.427	0.882	0.717	0.264	0.279	0.356	0.413	0.338	0.337
1^{st} Count		25	28	0	0	7	4	0	0	0	0	0	0	0	0	0	0	0	0

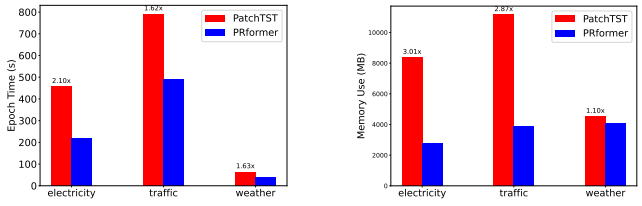
Table 5: Ablation studies: multivariate long-term series forecasting result with input length $L = 720$ and prediction length $H \in (96, 192, 336, 720)$.Evaluation and comparison of three PRformer variants against baseline across datasets: electricity, traffic, weather, and ETTh2. Instances of enhanced performance over baseline are emphasized in bold.

Models		PRformer								PatchTST(2023)	
		PRformer		PRformer V1		PRformer V2		PRformer V3			
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Electricity	96	**0.127**	**0.217**	0.132	**0.219**	0.132	0.222	0.130	**0.218**	0.129	0.222
	192	0.148	**0.237**	0.147	**0.235**	0.149	0.240	0.148	**0.237**	0.147	0.240
	336	**0.161**	**0.252**	0.165	**0.254**	**0.161**	**0.252**	**0.161**	**0.253**	0.163	0.259
	720	**0.185**	**0.275**	**0.196**	**0.283**	**0.176**	**0.269**	**0.190**	**0.279**	0.197	0.290
Traffic	96	**0.353**	**0.222**	**0.358**	**0.230**	**0.344**	**0.227**	**0.344**	**0.222**	0.360	0.249
	192	**0.372**	**0.233**	**0.377**	**0.238**	**0.370**	**0.239**	**0.372**	**0.233**	0.379	0.256
	336	**0.385**	**0.241**	0.397	**0.248**	**0.385**	**0.247**	**0.382**	**0.239**	0.392	0.264
	720	**0.421**	**0.258**	0.442	**0.271**	**0.425**	**0.265**	**0.412**	**0.257**	0.432	0.286
Weather	96	**0.144**	**0.187**	**0.145**	**0.183**	0.162	0.202	0.152	**0.191**	0.149	0.198
	192	**0.188**	**0.233**	**0.193**	**0.230**	0.205	0.248	0.196	**0.236**	0.194	0.241
	336	**0.241**	**0.274**	0.250	**0.275**	0.251	0.284	**0.243**	**0.275**	0.245	0.282
	720	0.326	**0.332**	0.355	0.343	0.357	0.349	0.318	**0.329**	0.314	0.334
ETTh2	96	**0.268**	**0.327**	0.278	**0.328**	0.295	0.353	**0.267**	**0.327**	0.274	0.337
	192	**0.332**	**0.370**	0.341	**0.371**	0.360	0.393	**0.338**	**0.373**	0.341	0.382
	336	0.361	0.395	0.366	0.396	0.396	0.419	0.368	0.399	0.329	0.384
	720	0.396	0.429	0.408	0.434	0.446	0.467	0.397	0.428	0.379	0.422
count		26		18		12		23		-	



(a) (b)

Figure 3: The MAE results (Y-axis) of models with different lookback window sizes (X-axis) of long-term forecasting (T=96) on the Traffic and Electricity datasets. (a) Transformer results; (b) PRformer results.



(a) (b)

Figure 4: In (a) we present the training times for one epoch on three datasets. In (b) we illustrate the corresponding memory usage. The experiments were conducted under equivalent hardware conditions and parameter configurations.

the memory usage and runtime of PatchTST and PRformer on the same GPU, maintaining identical lookback window length, prediction length, and batch size. The results from Figure 4 reveal that the runtime of PRformer is nearly half that of PatchTST, while its memory usage ranges between 1/3 and 0.9 of PatchTST's. This indicates that under equivalent hardware conditions, PRformer can accommodate a larger batch size, thereby further accelerating the training and inference speeds. Additionally, PRformer is capable of utilizing longer lookback windows, enabling it to learn dependencies over a larger range and thus further enhance its performance. This flexibility provides more opportunities for real-world applications and extensions.

Conclusion

In this paper, we explore a new direction for applying Transformers to time series prediction. We propose using a Pyramidal RNN Embedding (PRE) module to replace the positional encoding in Transformers, addressing the deficiencies of Transformers in encoding sequential positional information relationships. Through experiments combining PRE with three Transformer variants, we validate the significant improvement that PRE brings to Transformer architecture predictors. Furthermore, we propose PRformer, a time series predictor that combines PRE and Transformers. Comprehensive empirical experiments on eight long-term prediction benchmarks demonstrate the superiority of our approach. Additionally, PRE enables the computational complexity of Transformers to grow linearly with sequence length, resulting in better computational efficiency. We hope that this work will promote future research on the integration of RNNs and Transformers in time series modeling.

References

Bao, H.; Dong, L.; Piao, S.; and Wei, F. 2021. Beit: Bert pre-training of image transformers. *arXiv preprint arXiv:2106.08254*.

Bianchi, F. M.; Scardapane, S.; Løkse, S.; and Jenssen, R. 2021. Reservoir Computing Approaches for Representation and Classification of Multivariate Time Series. *IEEE Transactions on Neural Networks and Learning Systems*, 32(5): 2169–2179.

Cho, K.; Merrienboer, B. v.; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H.; and Bengio, Y. 2014. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; Uszkoreit, J.; and Houlsby, N. 2021. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *International Conference on Learning Representations (ICLR)*.

Graves, A.; Wayne, G.; and Danihelka, I. 2014. Neural turing machines. *arXiv preprint arXiv:1410.5401*.

Hewage, P.; Behera, A.; Trovati, M.; Pereira, E.; Ghahremani, M.; Palmieri, F.; and Liu, Y. 2020. Temporal convolutional neural (TCN) network for an effective weather forecasting using time-series data from the local weather station. *Soft Computing*, 24: 16453–16482.

Hoel, E. P.; Albantakis, L.; and Tononi, G. 2013. Quantifying causal emergence shows that macro can beat micro. *Proceedings of the National Academy of Sciences*, 110(49): 19790–19795.

Hsu, W.-N.; Bolte, B.; Tsai, Y.-H. H.; Lakhotia, K.; Salakhutdinov, R.; and Mohamed, A. 2021. Hubert: Self-supervised speech representation learning by masked prediction of hidden units. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 29: 3451–3460.

Khrulkov, V.; Novikov, A.; and Oseledets, I. V. 2018. Expressive power of recurrent neural networks. In *International Conference on Learning Representations (ICLR)*.

Kim, T.; Kim, J.; Tae, Y.; Park, C.; Choi, J.-H.; and Choo, J. 2022. Reversible Instance Normalization for Accurate Time-Series Forecasting against Distribution Shift. In *International Conference on Learning Representations (ICLR)*.

Kingma, D. P.; and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kitaev, N.; Kaiser, Ł.; and Levskaya, A. 2020. Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451*.

Lai, G.; Chang, W.-C.; Yang, Y.; and Liu, H. 2018. Modeling Long- and Short-Term Temporal Patterns with Deep Neural Networks. In *Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, 95–104.

Li, S.; Li, W.; Cook, C.; Zhu, C.; and Gao, Y. 2018. Independently Recurrent Neural Network (IndRNN): Building a Longer and Deeper RNN. In *Computer Vision and Pattern Recognition (CVPR)*, 5457–5466.

Li, Y.; Gault, R.; and McGinnity, T. M. 2022. Probabilistic, Recurrent, Fuzzy Neural Network for Processing Noisy Time-Series Data. *IEEE Transactions on Neural Networks and Learning Systems*, 33(9): 4851–4860.

Lin, T.-Y.; Dollár, P.; Girshick, R. B.; He, K.; Hariharan, B.; and Belongie, S. J. 2017. Feature Pyramid Networks for Object Detection. In *Computer Vision and Pattern Recognition (CVPR)*, 936–944.

Liu, S.; Yu, H.; Liao, C.; Li, J.; Lin, W.; Liu, A. X.; and Dustdar, S. 2021. Pyraformer: Low-complexity pyramidal attention for long-range time series modeling and forecasting. In *International conference on learning representations*.

Liu, Y.; Hu, T.; Zhang, H.; Wu, H.; Wang, S.; Ma, L.; and Long, M. 2024. iTransformer: Inverted Transformers Are Effective for Time Series Forecasting. In *The Twelfth International Conference on Learning Representations*.

Ma, X.; Zhou, C.; Kong, X.; He, J.; Gui, L.; Neubig, G.; May, J.; and Zettlemoyer, L. 2023. Mega: Moving Average Equipped Gated Attention. In *International Conference on Learning Representations (ICLR)*, volume abs/2209.10655.

Myers, C.; Rabiner, L.; and Rosenberg, A. 1980. Performance tradeoffs in dynamic time warping algorithms for isolated word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 28(6): 623–635.

Nie, Y.; Nguyen, N. H.; Sinthong, P.; and Kalagnanam, J. 2023. A Time Series is Worth 64 Words: Long-term Forecasting with Transformers. In *International Conference on Learning Representations (ICLR)*, volume abs/2211.14730.

Pascanu, R.; Mikolov, T.; and Bengio, Y. 2013. On the difficulty of training recurrent neural networks. In *International Conference on Machine Learning (ICML)*, 1310–1318.

Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32.

Peng, B.; Alcaide, E.; Anthony, Q.; Albalak, A.; Arcadinho, S.; Biderman, S.; Cao, H.; Cheng, X.; Chung, M.; Derczynski, L.; Du, X.; Grella, M.; Gv, K.; He, X.; Hou, H.; Kazienko, P.; Kocon, J.; Kong, J.; Koptyra, B.; Lau, H.; Lin, J.; Mantri, K. S. I.; Mom, F.; Saito, A.; Song, G.; Tang, X.; Wind, J.; Woźniak, S.; Zhang, Z.; Zhou, Q.; Zhu, J.; and Zhu, R.-J. 2023. RWKV: Reinventing RNNs for the Transformer Era. In *Findings of the Association for Computational Linguistics: EMNLP 2023*.

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

Van der Maaten, L.; and Hinton, G. 2008. Visualizing data using t-SNE. *Journal of machine learning research*, 9(11).

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L.; and Polosukhin, I. 2017. Attention is All you Need. In *Conference on Neural Information Processing Systems (NeurIPS)*, 5998–6008.

Wu, H.; Hu, T.; Liu, Y.; Zhou, H.; Wang, J.; and Long, M. 2022a. Timesnet: Temporal 2d-variation modeling for general time series analysis. In *International Conference on Learning Representations*, volume abs/2210.02186.

Wu, H.; Wu, J.; Xu, J.; Wang, J.; and Long, M. 2022b. Flowformer: Linearizing transformers with conservation flows. *arXiv preprint arXiv:2202.06258*.

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

Yang, S.; and Yuan, H. 2023. A Customized Multi-Scale Deep Learning Framework for Storm Nowcasting. *Geophysical Research Letters*, 50(13): e2023GL103979.

Zeng, A.; Chen, M.; Zhang, L.; and Xu, Q. 2023. Are Transformers Effective for Time Series Forecasting? In *AAAI Conference on Artificial Intelligence (AAAI)*, 11121–11128.

Zhang, Y.; and Yan, J. 2022. Crossformer: Transformer utilizing cross-dimension dependency for multivariate time series forecasting. In *The eleventh international conference on learning representations*.

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

Zhou, T.; Ma, Z.; Wen, Q.; Wang, X.; Sun, L.; and Jin, R. 2022. FEDformer: Frequency Enhanced Decomposed Transformer for Long-term Series Forecasting. In *International Conference on Machine Learning (ICML)*, 27268–27286.

Limitations of Positional Encoding

Transformer relies on positional encoding to determine the position of each timestamp(including relative and absolute distances), which may limit its capabilities and raise questions about its effectiveness compared to linear models (Zeng et al. 2023). The original positional encoding method in Transformers is as follows:

$$PE_t^{(i)} = \begin{cases} \sin\left(\frac{1}{10000^{2k/d_{model}}}t\right), & \text{if } i = 2k \\ \cos\left(\frac{1}{10000^{2k/d_{model}}}t\right), & \text{if } i = 2k+1 \end{cases}, \quad (7)$$
$$i = 0, 1, 2, 3, \ldots, d_{model} - 1$$

This design has the following property: the dot product of two positional encodings depends only on the offset $\triangle t$ and is independent of the absolute positions. Proof:

$$
\begin{aligned}
& PE_t^T \cdot PE_{t+\triangle t} \\
&= \sum_{i=0}^{\frac{d_{model}}{2}-1} [PE_t^{(2i)} \cdot PE_{t+\triangle t}^{(2i)} + PE_t^{(2i+1)} \cdot PE_{t+\triangle t}^{(2i+1)}] \\
&= \sum_{i=0}^{\frac{d_{model}}{2}-1} [\sin(w_i t) \cdot \sin(w_i(t+\triangle t)) + \cos(w_i t) \cdot \cos(w_i(t+\triangle t))] \\
&= \sum_{i=0}^{\frac{d_{model}}{2}-1} \cos(w_i \triangle t)
\end{aligned}
\tag{8}
$$

This property makes the distance measurement generated by positional encoding translation-invariant. When modeling time series data, traditional Transformers attempt to learn the variation patterns of sequences by combining multivariate data at different moments through positional encoding. However, real-world data is often non-stationary, and the patterns of data variation change significantly over time (Li, Gault, and McGinnity 2022). Adopting the assumption of translation invariance makes it difficult for the self-attention mechanism of Transformers to learn the correct temporal dependencies. The DTW algorithm (Myers, Rabiner, and Rosenberg 1980) is a good example in this regard. By finding the optimal alignment path, DTW allows sequences to "stretch" to obtain more accurate sequence similarity. Due to the translation invariance of positional encoding, Transformers cannot dynamically adjust like the DTW algorithm to adapt to the similarity relationships of non-stationary sequences, thus limiting performance improvement.

Compared to Transformers, the recurrent structure of RNN does not have fixed positional encoding and can adaptively adjust to sequences of different lengths, which helps in handling non-stationary time series. In addition, RNN have strong function fitting capabilities (Graves, Wayne, and Danihelka 2014; Khrulkov, Novikov, and Oseledets 2018) and have the potential to transform non-stationary sequences into representations (Bianchi et al. 2021) that are more conducive to Transformer learning. Based on these considerations, we combine RNN with PRE to transform non-stationary univariate time series into representations in vector space, and then use Transformers to learn the dependencies between variables. This avoids the limitations of traditional Transformers using positional encoding to learn the dependencies between points within sequences, resulting in a significant performance improvement.

Experimental details

Dataset

In our investigation, we evaluated the PRformer model using seven widely recognized datasets in the field of time series analysis. These datasets represent diverse domains and intervals, offering a comprehensive assessment environment: The datasets used in our analysis can be categorized as follows:

- **ETT**: Electricity Transformer Temperature (ETT) is a critical parameter in long-term electric power infrastructure planning. This dataset contains data spanning two years from two separate counties in China. It has been curated for use in long-sequence time-series forecasting (LSTF) research, and different subsets have been carefully constructed to explore the granularity of this problem. These subsets include ETTh1 and ETTh2, which provide data at 1-hour intervals, as well as ETTm1, which offers data at 15-minute intervals.
- **Weather**: A comprehensive collection of 21 weather parameters, including temperature and humidity, meticulously logged at 10-minute intervals during the entire year of 2020.
- **Electricity**: This dataset chronicles the hourly electricity usage of 321 customers, covering a timespan from 2012 to 2014.
- **Traffic**: Sourced from the California Department of Transportation, this dataset offers insights into highway occupancy rates with data aggregated hourly from 862 sensors across the San Francisco Bay Area.
- **Solar-Energy**: This dataset consists of solar power production records from the year 2006, collected every 10 minutes from 137 photovoltaic (PV) plants in Alabama State. These data points provide comprehensive insights into the power output of hypothetical solar plants across the United States.

For our analysis, we divided these datasets into training, validation, and test segments in line with the protocols recommended in recent literature (Wu et al. 2021; Zeng et al. 2023; Nie et al. 2023). Specifically, the ETT datasets were split in a 6:2:2 ratio, consistent with these guidelines. For the other datasets, a 7:1:2 division was employed. To ensure uniformity in our approach, we set the length of historical data sequences at 720 time steps, with the forecast horizon varying among the options of 96, 192, 336, and 720 time steps.

Confiuration

All experiments were implemented in PyTorch (Paszke et al. 2019) and conducted on a dedicated NVIDIA TITAN XP 11GB GPU. We employed the Adam optimizer (Kingma and Ba 2014) for training the model over 30 epochs, with an initial learning rate decayed exponentially starting from the fourth epoch, using a decay factor of 0.9. The early stopping mechanism was configured with a patience value of 10.

The specific parameters utilized by PRformer on different datasets are outlined in Table 6. The meaning of each parameter in the table is elucidated as follows:

- **lookback**: Length of the historical lookback window.
- **pyramidal_windows**: Sets the list of period lengths for the pyramidal convolution layers. The pyramidal convolution layers are generated based on this list.
- **e_layers**: Number of Transformer encoder layers set.
- **d_model**: The dimension of the embedding for univariate time series PRE.
- **dropout**: Dropout rate.
- **batch_size**: Batch size used for training.
- **l rate (learning rate)**: The initial learning rate used in the optimization process.

Table 6: Parameter Configuration Table of PRformer Across Different Datasets.

Datasets	lookback	pyramidal_windows	e_layers	d_model	dropout	batch_size	lr
ETTh1	720	24 48 72 144	5	720	0.1	256	0.001
ETTh2	720	24 48 72 144	5	720	0.1	256	0.0002
ETTm1	720	4 16 32 96	5	720	0.1	256	0.0002
ETTm2	720	4 16 32 96	5	720	0.1	256	0.0001
Weather	720	6 24 48 144	3	720	0.1	64	0.0001
Electricity	720	24 48 72 96 144	3	660	0.1	16	0.0005
Traffic	720	24 48 72 144	4	520	0.1	8	0.001

Implementation Details of PRE

The pyramidal convolution structure in PRE is generated according to the set list of period lengths parameters *pyramidal_windows*. Each number in the *pyramidal_windows* list represents a period, corresponding to a convolutional recurrent neural network chain. The convolutional pyramid structure, composed of multiple chains, represents a period at each layer, with the period length increasing by multiples from the bottom to the top layer. Each chain generates a time series embedding representation for one period, and the embedding dimension of each chain is evenly distributed according to the number of pyramid layers, being $d\_model|layer\_num$. Finally, the results from all chains are weighted concatenated and transformed through a linear layer similar to a multi-head attention mechanism to obtain the PRE embedding representation. Both the input and output dimensions of the linear layer are d_model.

Full PRE promotion results

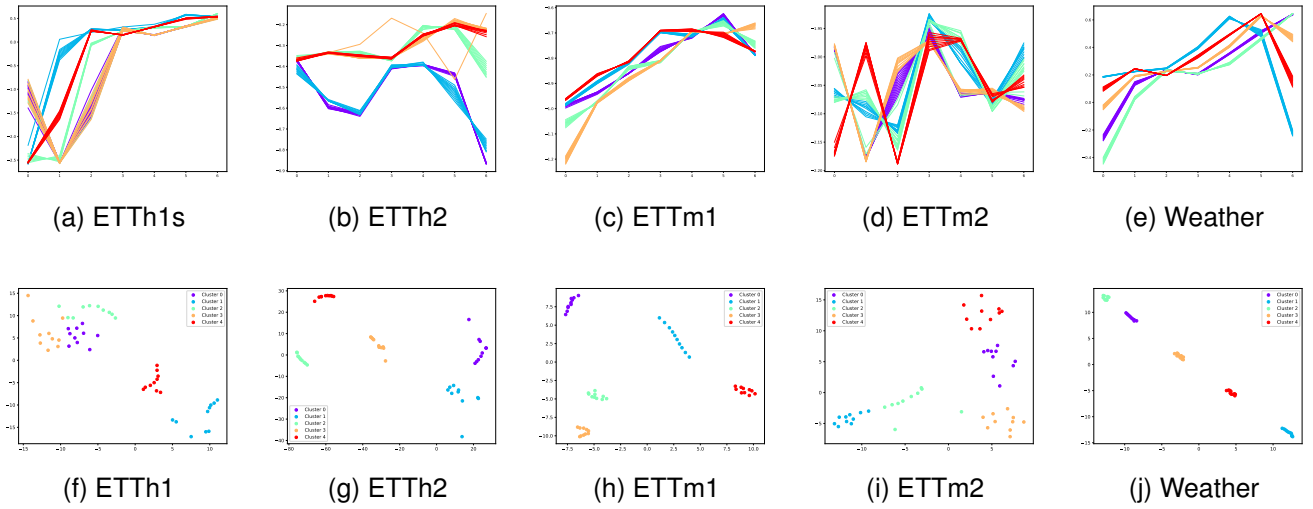
To validate the effectiveness and generalizability of the PRE in enhancing the Transformer's ability to learn temporal sequence representations, we applied PRE to the Transformer and its variants: Transformer(Vaswani et al. 2017), Reformer (Kitaev, Kaiser, and Levskaya 2020), Informer(Zhou et al. 2021),and Flowformer (Wu et al. 2022b). Due to space constraints, Table 3 presents the average results, while the complete predictive outcomes are detailed in Table 7. The results demonstrate that PRE consistently improves the performance of these Transformers, thereby affirming its efficacy and generalization capabilities.

Univariate Time Series Representation Learning

PRE provides a novel approach for time series prediction with Transformer. Initially, single-variable time series are represented using PRE. Subsequently, the Transformer attention mechanism is employed to learn intricate dependencies among multiple variables and make predictions. To analyze PRE's learned representations, 50 embeddings were randomly chosen from the test set of multiple datasets, post-PRE training. We then applied t-SNE(Van der Maaten and Hinton 2008) for dimensionality reduction, forming five distinct clusters in the embedding space. This is visualized in Figure 5.From the figures, we observe that similar representations correspond to original sequences with similar shapes and patterns, while distant representations exhibit larger shape differences. This observation underscores PRE's ability to generate meaningful embeddings for single variables, aiding the Transformer in constructing dependencies among multiple variables and subsequent prediction tasks.

Table 7: Full performance promotion results of Transformers with PRE.

Models			Transformer (2017)		Reformer (2020)		Informer (2021)		Flowformer (2022)	
Metric			MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Electricity	Original	96	0.260	0.358	0.312	0.402	0.274	0.368	0.215	0.320
		192	0.266	0.367	0.348	0.433	0.296	0.386	0.259	0.355
		336	0.280	0.375	0.350	0.433	0.300	0.394	0.296	0.383
		720	0.302	0.386	0.340	0.420	0.373	0.439	0.296	0.380
		Avg	0.277	0.372	0.338	0.422	0.311	0.397	0.267	0.360
	+PRE	96	0.128	0.218	0.131	0.227	0.130	0.226	0.132	0.228
		192	0.148	0.238	0.149	0.244	0.149	0.243	0.150	0.244
		336	0.162	0.254	0.166	0.262	0.166	0.262	0.166	0.261
		720	0.184	0.276	0.206	0.297	0.203	0.295	0.197	0.289
		Avg	**0.156**	**0.247**	**0.163**	**0.258**	**0.162**	**0.257**	**0.161**	**0.256**
Traffic	Original	96	0.647	0.357	0.732	0.423	0.719	0.391	0.691	0.393
		192	0.649	0.356	0.733	0.420	0.696	0.379	0.729	0.419
		336	0.667	0.364	0.742	0.420	0.777	0.420	0.756	0.423
		720	0.697	0.376	0.755	0.432	0.864	0.472	0.825	0.449
		Avg	0.665	0.363	0.741	0.424	0.764	0.416	0.750	0.421
	+PRE	96	0.353	0.222	0.362	0.257	0.363	0.257	0.358	0.256
		192	0.372	0.233	0.381	0.265	0.380	0.265	0.378	0.265
		336	0.385	0.241	0.397	0.274	0.396	0.273	0.395	0.273
		720	0.421	0.258	0.436	0.297	0.436	0.296	0.435	0.296
		Avg	**0.383**	**0.239**	**0.394**	**0.273**	**0.394**	**0.273**	**0.392**	**0.272**
Weather	Original	96	0.395	0.427	0.689	0.596	0.300	0.384	0.182	0.233
		192	0.619	0.560	0.752	0.638	0.598	0.544	0.250	0.288
		336	0.689	0.594	0.639	0.596	0.578	0.523	0.309	0.329
		720	0.926	0.710	1.130	0.792	1.059	0.741	0.404	0.385
		Avg	0.657	0.573	0.803	0.656	0.634	0.548	0.286	0.309
	+PRE	96	0.144	0.187	0.147	0.198	0.146	0.200	0.149	0.203
		192	0.188	0.233	0.191	0.241	0.197	0.245	0.199	0.249
		336	0.241	0.274	0.243	0.283	0.256	0.291	0.243	0.284
		720	0.326	0.332	0.314	0.334	0.324	0.342	0.316	0.337
		Avg	**0.225**	**0.257**	**0.224**	**0.264**	**0.231**	**0.269**	**0.227**	**0.268**



(a) ETTh1s (b) ETTh2 (c) ETTm1 (d) ETTm2 (e) Weather

(f) ETTh1 (g) ETTh2 (h) ETTm1 (i) ETTm2 (j) Weather

Figure 5: Correlation of origin time series with their respective PRE embeddings across multiple datasets. Figures a-e represent the original time series, while figures f-j depict the corresponding t-SNE clustering results of PRE embeddings.Similar representations correspond to original sequences with analogous shapes and patterns, whereas distant representations showcase more substantial deviations in shape, thereby acquiring meaningful time series representations.