# MLAFormer: Multi-scale Transformer with Local Convolutional Auto-Correlation and Pre-training for Time Series Forecasting

Ying Jin[1], Ming Gao[2,3,*], Jiafu Tang[2], Weiguo Fan[4], Yanwu Yang[1], and Jingmin An[2]

**Abstract**
Recent advancements in time series forecasting have focused on Transformer and its variants, which have the potential of long sequence representation and generation. However, capturing their versatile long-term dependencies and intricate temporal patterns remains challenging. Current Transformer-based methods simply model time series from single scale, or solely rely on the self-attention mechanisms to capture temporal dependencies, which may overlook the rich information available in different scales and corresponding frequency domains. There is also a paucity of systematical investigation in the effectiveness of generative pre-training for time series. To address these limitations, we propose MLAFormer with the following highlights: (1) Multi-scale encoder-decoder discovers multi-periodicities of time series, which empowers MLAFormer in capturing both local short-term dependencies and global long-term dependencies. (2) Local Convolutional Auto-Correlation utilizes causal convolution to perceive local context and frequency domain transformation to discover the sub-series similarity. (3) The progressive pre-series decomposition preserves the intrinsic variations of time series, facilitating the integration of temporal information with frequency-domain information captured by the attention mechanism. (4) Asymmetric self-supervised pre-training enhances the sequence representation and further improves the prediction performance. Extensive experiments show that MLAFormer outperforms existing Transformers, reducing the prediction error by 39.94% on average and MLAFormer with pre-training achieves a further 4.49% improvement.
**Keywords** Multi-scale, Pre-training, Time Series Forecasting, Transformer

## 1 Introduction

Time series forecasting is ubiquitous across many fields, such as economic fields encompassing GDPs and stock prices, and meteorological fields involving temperature and humidity measurement (Au et al., 2008; Bouteska et al., 2023; Wu et al., 2021). Accurately predicting the future based on existing information can support decision-making and respond to developing trends reasonably. Many models have been developed for time series forecasting and achieved a progressive breakthrough, including classical models, e.g., ARIMA (Box et al., 1971), and deep learning models, e.g., RNN (Rumelhart et al., 1986).

In recent years, a powerful neural network framework, Transformer (Vaswani et al., 2017) has shown remarkable performances in various applications, such as natural language processing (Dai et al., 2019; Yang et al., 2022), computer vision (Ma et al., 2023; Xu et al., 2022), and audio processing (Hu et al., 2021; Kenton and Toutanova, 2019; Song et al., 2023). Particularly in time series forecasting, it is arguably advantageous in sequence modeling benefiting from unparalleled architecture design and key self-attention mechanism. It eliminates recursive and convolutional operations and relies entirely on the attention mechanism to establish sequence dependencies. However, it encounters the problem of quadratic computational complexity and memory complexity (Lin et al., 2022). Consequently, many scholars have improved Transformer, such as Informer (Zhou et al., 2021), LogSparse Transformer (Li et al., 2019), etc. However, long-term time series forecasting remains challenging due to its intricate nature (Ak et al., 2015). Most Transformer-based models fail to fully consider knowledge about the intrinsic structure of time series and harness frequency domain information. Neglecting this information, modeling complex temporal variations and identifying periodic patterns become difficult and lack interpretability (Wang et al., 2022b; Wu et al., 2021). What's more, dependencies between individual observations tend to weaken over time and future values are highly dependent on recent observations. However, most attention mechanisms are insensitive to recent changes (Li et al., 2019). Recent studies such as ETSFormer (Woo et al., 2022) have explored this idea by simplifying the attention with exponential smoothing; however, it may lead to over-stationarization and undermine the extreme points. More importantly, observational variations of different

This work is supported by the National Natural Science Foundation of China (72293563, 71831003), Dalian Science and Technology Talent Innovation Support Plan (2022RG17) and the Basic Scientific Research Project of Liaoning Provincial Department of Education under Grant (JYTZD2023050).
* Corresponding author, gm@dufe.edu.cn. All the authors contributed equally.
1 School of Management, Huazhong University of Science and Technology, Wuhan, 430074, China
2 School of Management Science and Engineering, Key Laboratory of Big Data Management Optimization and Decision of Liaoning Province, Dongbei University of Finance of Economics, Dalian, 116025, China
3 Center for Post-doctoral Studies of Computer Science, Northeastern University, Shenyang, 110819, China
4 Department of Business Analytics, Tippie College of Business, University of Iowa, Iowa City, IA 52242, USA

granularities are interrelated. That means in a time series, the value of a specific moment is not solely determined by the changes that occur within a single day, but could also be related to larger patterns or trends that exist over longer periods, such as weeks or months. For instance, an increase in sales on a particular day could be due to an ongoing promotional campaign, or it could be influenced by seasonal trends such as holiday shopping periods or back-to-school seasons that significantly impact daily sales over a span of months. Existing Transformers typically model time series from single scale, rarely pay attention to temporal features at different periodicities.

To obtain accurate and reliable forecasting results, we propose multi-scale Transformer with Local Convolutional Auto-Correlation (MLAFormer) to overcome the above limitations. First, we employ multi-scale representation to discover multi-periodicities of time series and capture intricate variations. Multi-scale feature fusion is commonly used in image recognition (Lin et al., 2017). In general, fine-grained features can capture local short-term dependencies and coarse-grained features can capture global long-term dependencies. More importantly, considering greater influence of local information on future values, Local Convolutional Auto-Correlation incorporates causal convolution into frequency-enhanced auto-correlation. Moreover, progressive pre-series decomposition architecture preserves the basic property of time series and empowers the Transformer architecture with the fusion of time-domain and frequency-domain information. Furthermore, we design efficient pre-training strategies to generate enhanced sequence representation. Specifically, we develop asymmetric self-supervised pre-training methods, in which positional information is learned automatically. The encoder-decoder pre-trained model is loaded and fine-tuned to refine the predictive capability of MLAFormer.

Our contributions are summarized as follows:

- We have summarized the limitations of existing Transformer-based time series models. Based on these observations, we optimize the architecture of Transformer and propose MLAFormer for time series forecasting, which introduces multi-scale representations to mine intricate temporal patterns. Our asymmetric self-supervised pre-training strategies achieve a further boost to the accuracy and efficiency of MLAFormer.
- We propose progressive pre-series decomposition architecture and Local Convolutional Auto-Correlation with better local temporal perception. This design seamlessly integrates time-domain and frequency-domain information. We also propose multi-scale positional encoding to enhance the model's learning of order information. We conduct thorough ablation studies to validate the improvements of each component in MLAFormer.
- Extensive and systematical experiments show that MLAFormer outperforms the existing state-of-the-art methods. On average, it achieves a 39.94% improvement on MSE without pre-training and a further 4.49% enhancement with pre-training implemented.

## 2 Related work

We work on improving Transformer and designing pre-training methods for time series forecasting. Thus, we conduct a literature review from the following aspects: time series forecasting, Transformer-based time series models, and self-supervised pre-training.

### 2.1 Time Series Forecasting

Many models have been applied to time series forecasting. These models can be roughly divided into two categories: classic models and deep learning models (Liu et al., 2022; Zhou et al., 2021). Among classic statistical models, ARMA and ARIMA (Box et al., 1971) are commonly employed in the time series field. ARIMA is a general form of ARMA. In dealing with nonstationary time series, stabilization is achieved through performing differences. However, these models are easily affected by data quality, and have limitations in handling nonlinear data and long-term prediction (Valipour et al., 2013).

Real-world problems often involve large and intricate datasets. Deep learning algorithms have demonstrated improved performance over traditional statistical models in time series forecasting, especially Recurrent Neural Network-based methods (Lara-Benítez et al., 2021; Rumelhart et al., 1986). For instance, Qin et al.(2017) proposed a dual-stage attention-based recurrent neural network model (DA-RNN), in which the encoder uses the input attention mechanism to select the corresponding input features at each time step, and the decoder uses the temporal attention mechanism to select the corresponding hidden layer state in the entire time step, which can capture the temporal dependencies of time series to a certain extent. RNN has a memory function and shows advantages in capturing sequence dependencies (Elman, 1990), but due to gradient disappearance and gradient explosion, it fails to learn long-term dependencies (Li et al., 2018; Zheng et al., 2019). Although its variants such as LSTM and Gated Recurrent Unit (GRU) (Cho et al., 2014) can improve RNN, the issue is still unresolved (Hochreiter and Schmidhuber, 1997; Nelson et al., 2017) . Some research shows promising directions for unraveling intricate temporal patterns, such as LightTS (Zhang et al., 2022) based on MLP architectures and TimesNet (Wu et al., 2023) modeling 2D variations.

### 2.2 Transformer-based Time Series Models

Transformer has shown potential for time series forecasting (Zhang et al., 2023). However, the quadratic complexity ( $O(L^2)$ ) issue limits the long-sequence prediction. Therefore, many studies have emerged to improve Transformer. We have summarized and compared these models (Table 1). LogSparse Transformer (LogTrans) (Li et al., 2019) proposes local attention and LogSparse attention. Local attention enhances the model's ability to perceive local information. LogSparse attention calculates the attention

of the exponential position in the historical time step, which reduces the memory to $O(L \log L)$. Furthermore, Informer (Zhou et al., 2021) proposes ProbSparse self-attention, which also reduces the complexity to $O(L \log L)$. It achieves long-sequence output in the decoder, validating the promising predictive effect of Transformer-based time series models.

The models above implement point-wise computational attention mechanisms. Autoformer (Wu et al., 2021) proposes Auto-Correlation based on the inherent periodicity of time series, which uses a more efficient sequence-level connection. It incorporates time series decomposition as an inner block of the model to enhance the predictive capability. ETSFormer (Woo et al.) also incorporates series decomposition in modeling time series and introduces exponential smoothing attention (ESA) and frequency attention (FA). ESA gives higher weights to neighboring observations. FA uses the Fourier transform to select *K* Fourier bases with the largest amplitude in the frequency domain to extract the main seasonal components.

In addition to the above-mentioned, some models compute attention on segmented time series. Preformer (Du et al., 2023) introduces Segment-Correlation, which divides time series into segments and computes segment-wise attention. The number of segments is much smaller than that of points, so it greatly reduces the calculation amount and allows for better exploration of neighboring points. The selection of the segment length ( $L_{seg}$ ) is crucial. A shorter length increases the calculation amount, while a longer length ignores fine-grained information. Preformer avoids the adaptive problem and instead aggregates the attention under different lengths. PatchTST (Nie et al., 2023) introduces subseries-level patches as input tokens and applies a vanilla Transformer encoder to model channel-independence signals. The patching design enhances local information, reduces computation and memory usage, and allows for a longer look-back window.

Similar to MLAFormer, other models also pay attention to the multi-resolution representation of time series. Informer (Zhou et al., 2021) feeds the final encoder's downsampling representation into the decoder. Yformer (Madhusudhanan et al., 2023) improves upon the Informer architecture with a Y-shaped encoder-decoder. It combines a coupled scaling mechanism with sparse attention and feeds each encoder's downsampling representation at different resolutions into the decoder. Pyraformer (Liu et al., 2022) proposes a pyramidal attention mechanism that leverages a bottom-up multi-resolution representation to establish inter-scale and intra-scale connections and reduces the time complexity and space complexity to $O(L)$.

What's more, Aliformer (Qi et al., 2021) is proposed for merchandise sales prediction, which has demonstrated satisfactory performance in the Tmall business. The study points out that Transformer does not consider the impact of known future knowledge on prediction, such as the impact of known product and platform promotion information on future sales. Its Ali attention layer adds a branch of future information, which can reduce the interference caused by masked future data. Bidirectional attention provides better utilization of future information. However, this model is more suitable for the scenario of mastering domain knowledge.

Different from these models, our MLAFormer explores downscaling and upscaling in both the encoder and decoder to realize multi-scale transformation, and captures both time-domain and frequency-domain information within the architecture of progressive decomposition.

**Table 1** Transformer-based time series models

| Method | Attention | Train | | Test Step | Multi -scale |
|---|---|---|---|---|---|
| | | Time | Memory | | |
| Transformer | Full Attention | $O(L^2)$ | $O(L^2)$ | $L$ | |
| LogTrans | Local Attention & LogSparse Attention | $O(L \log L)$ | $O(L^2)$ | 1 | |
| Informer | ProbSparse Attention | $O(L \log L)$ | $O(L \log L)$ | 1 | ✓ |
| Autoformer | Auto-Correlation | $O(L \log L)$ | $O(L \log L)$ | 1 | |
| ETSFormer | ESA & FA | $O(L \log L)$ | $O(L \log L)$ | 1 | |
| Preformer | Segment-Correlation | $O(L^2 / \log L_{seg})$ | $O(L^2 / \log L_{seg})$ | 1 | ✓ |
| Yformer | ProbSparse Attention | $O(L \log L)$ | $O(L \log L)$ | 1 | ✓ |
| Pyraformer | Pyramidal Attention | $L$ | $L$ | 1 | ✓ |
| Aliformer | AliAttention | $O(L^2)$ | $O(L^2)$ | 1 | |
| PatchTST | Full Attention | $O(L^2)$ | $O(L^2)$ | 1 | |
| **MLAFormer (Ours)** | Local Convolutional Attention | $O(L \log L)$ | $O(L \log L)$ | 1 | ✓ |

### 2.3 Self-supervised Pre-training

Self-supervised learning can mine information from large-scale data utilizing auxiliary tasks. Pre-training through self-supervised learning has been effectively applied in multiple fields such as speech (Siniscalchi and Salerno, 2016), images (Wang et al., 2022a), and natural language (Li et al., 2018). For example, in the field of natural language processing, GPT (Radford et al., 2018) performs unidirectional modeling based on Transformer with masked self-attention and fine-tunes the pre-trained model for downstream tasks. Similarly, BERT (Kenton and Toutanova, 2019) achieves deep bidirectional representation by randomly masking some tokens and predicting these masked tokens in the pre-training stage, delivering top performance on multiple tasks. Pre-training has also been applied to computer vision, where MAE (He et al., 2022) designs an asymmetric encoder-decoder architecture. It reconstructs masked pixels to perform self-supervised tasks. Nevertheless, pre-training in the field of time series is rarely explored (Nie et al., 2023). InterpoMAE (Mengyue Zha et al., 2022) suggests that time series generation may go on a self-supervised learning

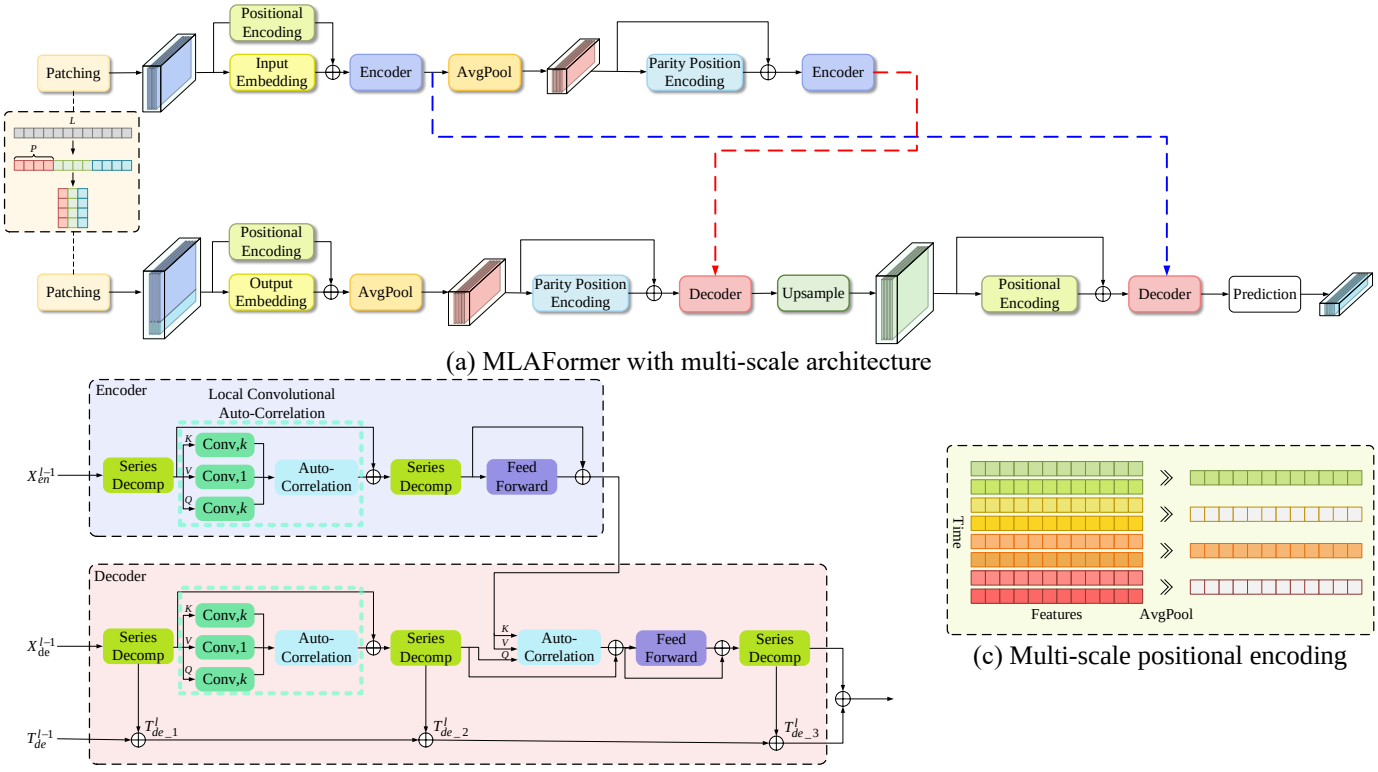
track. It recovers the missing parts through fully connected layers and avoids masked tokens. It has promising scalability in downstream tasks such as time series classification, prediction, and imputation with a simple design idea.

Existing Transformers do not fully utilize the temporal features of different periodicities and overlook that pre-training can facilitate the model to learn reinforced representation of time series. In contrast, our model exploits multi-scale fusion to discover multi-periodicities to capture both local short-term and global long-term dependency. Moreover, we develop asymmetric self-supervised pre-training methods to learn temporal patterns effectively.

## 3 Methodology

### 3.1 Problem Definition

Time series forecasting aims to fit historical data with models and use them to predict future values. $x_t \in \mathbb{R}^M$ denotes the observation at time *t*. *M* is the number of features. Given the historical time series $X_{1:L} = \{x_1, x_2, \cdots, x_L\}$, the task is to predict the corresponding future series $X_{L+1:L+H} = \{x_{L+1}, x_{L+2}, \cdots, x_{L+H}\}$. $\hat{X}_{L+1:L+H}$ is the prediction of $X_{L+1:L+H}$. We aim to learn a function $\hat{X}_{L+1:L+H} = f(X_{L+1:L+H})$. For the long-sequence time series forecasting, the prediction length *H* is longer.




(a) MLAFormer with multi-scale architecture

(b) Encoder-Decoder architecture

(c) Multi-scale positional encoding

**Fig. 1** Overall architecture of MLAFormer. (a) Time series is segmented into multiple patches and embedded by multi-scale positional encoding. The multi-scale architecture uses average pooling and upsampling operations to learn multi-resolution representations of time series. (b) The encoder and decoder block consist of main components: pre-series decomposition and Local Convolutional Auto-Correlation. The decoder block extracts trend information and seasonal information from the series progressively. (c) Multi-scale positional encoding includes conventional positional encoding (left) and parity positional encoding (right). Conventional positional encoding denotes all the positions are encoded. Parity position encoding denotes that odd or even positions are encoded (colorful), while the remaining positions are not encoded (grey).

### 3.2 MLAFormer

The MLAFormer architecture is built on channel-independence time series patches. Fig. 1 (a) shows the process of MLAFormer's cross multi-scale representation through pooling and upsampling. Multi-scale positional encoding is performed before each encoder and decoder to enhance the model's learning of order information. Fig. 1 (b) shows the encoder-decoder architecture. The input goes through the pre-series decomposition block to decompose the trend and seasonal components. The output then undergoes Local Convolutional Auto-Correlation to discover dependency.

### 3.3 Instance Normalization

The input of the model is first processed by the normalization technique RevIN (Kim et al., 2021), which is proposed to address the discrepancy between training and test data distribution. Each univariate series is normalized using its instance-specific mean and standard deviation. The model output is denormalized symmetrically to return to the original distribution information.

### 3.4 Patching

In a multivariate time series, on the one hand, there are multiple channels, and each input token can represent information from a single channel; on the other hand, individual data points do not provide sufficient information. Considering low density and multi-channel relationships of time series, we adopt the patching and channel-independent strategies in (Nie et al., 2023). The encoder takes historical sequences as input. Each channel-independent univariate time series $X^i \in \mathbb{R}^L$, $i \in \{1, \cdots, M\}$ is divided into $I$ patches with the length of $P$, then a sequence of patches $X_P^i \in \mathbb{R}^{I \times P}$ is generated. The patches can be overlapped or not. Denote the non-overlapping region between two consecutive patches as stride $S$. The number of patches $I$ can be formulated as

$$I = \left\lfloor \frac{(L-P)}{S} \right\rfloor + 2 . \tag{1}$$

Thus, the number of input tokens can be reduced from $L$ to $I$, which means less memory usage and computational complexity. Subsequently, each patch is projected to the input tokens $X_{en}^i \in \mathbb{R}^{I \times D}$ through a learnable linear layer, where $D$ is the embedding dimension of the token.

The decoder takes historical data and masked future data as input. Similarly, the future uses $O$ placeholders to represent and yields the tokens $X_{de}^i \in \mathbb{R}^{(I+O) \times D}$.

### 3.5 Multi-scale Encoder-Decoder

In the fields of computer vision and statistical signal processing, multi-resolution representation has proven advantageous in remote interaction modeling (Liu et al., 2022). Inspired by feature pyramid networks, we propose a multi-scale encoder-decoder architecture with different resolutions from bottom to top and top to bottom. Upsampling and downsampling enable time series transformations, combining low-level features with high resolution and high-level features with rich temporal information. Fine-grained features corresponding to the original time series' time points (e.g., hourly periodicity) can capture local short-term dependencies, while the coarse-grained features corresponding to features with lower resolution (e.g., weekly periodicity) can capture global long-term dependencies.

We design the cross multi-scale encoder-decoder architecture (Fig. 1 (a)) to explore multi-scale feature representation for time series forecasting. The cross multi-scale encoder-decoder crosses the multi-scale sequences from the encoder into the decoder to perform cross-attention. Average pooling and upsampling combine the features with varying resolutions and achieve multi-feature fusion. Average pooling takes the average of subsets of contiguous data points. Upsampling increases the number of data points in a time series by inserting interpolated values between existing data points. The calculation process is as follows.

$$\begin{aligned} X_{en\_o}^i &= Encoder(X_{en}^i), \\ X_{en\_p}^i &= Encoder(AvgPool(X_{en\_o}^i)), \\ X_{de\_p}^i &= Decoder(AvgPool(X_{de}^i), X_{en\_p}^i), \\ X_{de\_u}^i &= Decoder(Upsample(X_{de\_p}^i), X_{en\_o}^i), \end{aligned} \tag{2}$$

where $X_{en}^i \in \mathbb{R}^{I \times D}$, $X_{en\_o}^i \in \mathbb{R}^{I \times D}$, $X_{en\_p}^i \in \mathbb{R}^{\frac{I}{2} \times D}$, $X_{de}^i \in \mathbb{R}^{(I+O) \times D}$, $X_{de\_p}^i \in \mathbb{R}^{\frac{I+O}{2} \times D}$, $X_{de\_u} \in \mathbb{R}^{(I+O) \times D}$. $AvgPool(\cdot)$ and $Upsample(\cdot)$ denote average pooling and upsampling operation, respectively.

### 3.6 Pre-series Decomposition Block

The series decomposition block implements progressive decomposition forecasting and plays a crucial role in our architecture. The moving average operation extracts the long-term trend, with the remainder as periodic fluctuations (3).

$$\begin{aligned} X_t &= AvgPool(Padding(X)), \\ X_s &= X - X_t, \end{aligned} \tag{3}$$

where $X, X_s, X_t \in \mathbb{R}^{I \times D}$. The moving average operation with a proper filling of the sequence is performed to obtain the trend component $X_t$ and the seasonal component $X_s$, that is, $X_s, X_t = SeriesDecomp(X)$. $Padding(\cdot)$ denotes the padding operation to maintain the sequence length unchanged.

Time series data are usually noisy and not smooth. However, incorporating basic knowledge of the intrinsic variations of time series in the model can enhance its learning ability and achieve stronger interpretability (Woo et al.). To achieve this, we place the series decomposition block prior to the attention block, which enables early separation of the trend component for better preservation of nature information and facilitates the discovery of period-based auto-correlation (Fig. 1 (b)).

The *encoder* contains two sublayers, - the decomposition-attention sublayer and the decomposition-full connection sublayer. The input to the encoder is decomposed by the series decomposition block with the seasonal component entering the Local Convolutional Auto-Correlation block. After the residual connection, the series decomposition is performed again. The calculation of the *l*-th encoder layer is as follows.

$$
\begin{aligned}
S_{en\_1}^{l}, T_{en\_1}^{l} &= SeriesDecomp(X_{en}^{l-1}), \\
S_{en\_2}^{l}, T_{en\_2}^{l} &= SeriesDecomp(Auto\text{-}Correlation(S_{en\_1\_q}^{l}, S_{en\_1\_k}^{l}, S_{en\_1\_v}^{l}) + S_{en\_1}^{l}), \\
X_{en}^{l} &= FeedForward(S_{en\_2}^{l}) + S_{en\_2}^{l},
\end{aligned} \tag{4}
$$

where $S_{en\_1\_q}^{l}$, $S_{en\_1\_k}^{l}$, and $S_{en\_1\_v}^{l}$ denote *Q*, *K*, and *V* after the convolutional transformation of $S_{en\_1}^{l}$ respectively. $Auto-Correlation(\cdot)$ will be explained in the next section. $FeedForward(\cdot)$ denotes the full connection layer.

In the *decoder* block, the series decomposition block is also pre-positioned to progressively decompose intermediate variables. The seasonal component participates in the self-attention of the decoder and the cross attention between the encoder-decoder. The trend component is accumulated with that of the previous decoder after linear transformation. The calculation of the *l*-th decoder layer is as follows.

$$
\begin{aligned}
S_{de\_1}^{l}, T_{de\_1}^{l} &= SeriesDecomp(X_{\mathrm{de}}^{l-1}), \\
S_{de\_2}^{l}, T_{de\_2}^{l} &= SeriesDecomp(Auto\text{-}Correlation(S_{de\_1\_q}^{l}, S_{de\_1\_k}^{l}, S_{de\_1\_v}^{l}) + S_{de\_1}^{l}), \\
S_{de\_3}^{l}, T_{de\_3}^{l} &= SeriesDecomp(FeedForward(Auto\text{-}Correlation(S_{de\_2}^{l}, X_{en}^{l}) + S_{de\_2}^{l}) + \\
&\quad Auto\text{-}Correlation(S_{de\_2}^{l}, X_{en}^{l}) + S_{de\_2}^{l}), \\
T_{de}^{l} &= T_{de}^{l-1} + W_1^{l} * T_{de\_1}^{l} + W_2^{l} * T_{de\_2}^{l} + W_3^{l} * T_{de\_3}^{l},
\end{aligned} \tag{5}
$$

where $S_{de\_1\_q}^{l}$, $S_{de\_1\_k}^{l}$, and $S_{de\_1\_v}^{l}$ denote *Q*, *K*, and *V* with the convolutional transformation of $S_{de\_1}^{l}$. $W_i^{l}$, $i \in \{1,2,3\}$ denotes the linear transformation for the trend component. Finally, the sum of refined seasonal and trend components is fed to a linear flatten layer to output the prediction results.

### 3.7 Local Convolutional Auto-Correlation

We improve the existing attention mechanisms and propose Local Convolutional Auto-Correlation, which involves applying a causal convolution operation on queries (*Q*) and keys (*K*) followed by frequency-enhanced Auto-Correlation (Wu et al., 2021). Local Convolutional Auto-Correlation can be aware of local context when exploring sequence-level auto-correlation, so that the model not only preserves the long-term dependencies of time series but also perceives recent changes that contribute more to prediction.

We regard the matrix operation of *Q* and *K* in full attention as a one-dimensional convolution operation with a kernel size of 1 and perform a one-dimensional convolution operation with a kernel size of *k* instead.

$$
\begin{aligned}
S_{en\_1\_q}^{l} &= Conv1d(Padding(S_{en\_1}^{l}), k), \\
S_{en\_1\_k}^{l} &= Conv1d(Padding(S_{en\_1}^{l}), k), \\
S_{en\_1\_v}^{l} &= Conv1d(S_{en\_1}^{l}, 1),
\end{aligned} \tag{6}
$$

where $Conv1d(\cdot)$ denotes performing a one-dimensional convolution operation of kernel size *k* on *Q* and *K* with paddings.

Instead of relying on point-wise calculation, Auto-Correlation uses a more efficient sequence-level connection. Auto-Correlation makes better use of time-series information and realizes $O(L \log L)$ complexity. It discovers the period-based dependencies by calculating the auto-correlation coefficient of the sequence (Wu et al., 2021).

As shown in (7), $R_{XX}$ represents the similarity between the sequence $X_t$ and its delayed sequence $X_{t-\tau}$. $R_{Q,K}$ denotes auto-correlation between *Q* and *K*, which can be obtained by Fast Fourier Transforms (8). Perform Fast Fourier Transforms on *Q* and *K*, followed by a conjugate operation on *K*. The delay aggregation operation $argTopk(\cdot)$ is to determine the most probable period $\tau$. The roll operation $Roll(\cdot)$ aligns similar subsequences. Finally, information aggregation of the selected *k* periods is carried out.

$$
\begin{aligned}
R_{XX}(\tau) &= \lim_{L\to\infty} \frac{1}{L} \sum_{t=1}^{L} X_t X_{t-\tau} \\
R_{Q,K}(\tau) &= F^{-1}(F(Q)F^{*}(K)) \\
\tau_1, \cdots, \tau_k &= \underset{\tau\in\{1,\cdots,L\}}{argTopk}(R_{Q,K}(\tau)) \\
R_{Q,K}(\tau_1), \cdots, R_{Q,K}(\tau_k) &= Softmax(R_{Q,K}(\tau_1), \cdots, R_{Q,K}(\tau_k)) \\
Auto-Correlation(Q,K,V) &= \sum_{i=1}^{k} Roll(V, \tau_i) R_{Q,K}(\tau_i)
\end{aligned} \tag{7}
$$

$$
\begin{aligned}
S_{XX}(f) &= F(X_t)F^{*}(X_t) = \int_{-\infty}^{\infty} X_t e^{-i2\pi tf} dt \overline{\int_{-\infty}^{\infty} X_t e^{-i2\pi tf} dt} \\
R_{XX}(\tau) &= F^{-1}(S_{XX}(f)) = \int_{-\infty}^{\infty} S_{XX}(f) e^{i2\pi f\tau} df
\end{aligned} \tag{8}
$$

### 3.8 Multi-scale Positional Encoding

We perform multi-scale positional (Fig. 1 (c)) encoding with conventional positional encoding in Transformer and parity positional encoding before each encoder and decoder. This design enhances the learning of order information. In parity positional encoding, only odd or even positions are encoded, which helps the model discriminate the relative position after pooling and upsampling.

## 4 Pre-training and Downstream Task

Pre-training has proven effective for image and natural language processing tasks, as exemplified by advanced models such as MAE and BERT. To make better use of time series information and improve accuracy, we perform pre-training for time series forecasting. We mask a portion of the data during the pre-training stage and then predict the masked data to obtain an enhanced representation of time series by self-supervised learning. We then initialize the prediction model with the weights of the pre-trained model and fine-tune the parameters to predict future data. Although pre-training increases training costs, it is performed only once to improve prediction performance and accelerate convergence (Howard and Ruder, 2018).

### 4.1 Pre-training Architecture

As shown in Fig. 2, the architecture consists of two stages - the first being the self-supervised pre-training stage, and the second being the fine-tuning stage. We develop four pre-training strategies: random masked (RM) pre-training, variable-length masked (VM) pre-training, random masked auto-insertion (RMAI) pre-training, and variable-length masked auto-insertion (VMAI) pre-training. MLAFormer performs sequence reconstruction on the masked data. Subsequently, the downstream task loads the parameters of the pre-trained model and fine-tunes the parameters to predict the unknown sequences. It is notable that following the principle of simplicity and efficiency, we adopt a complete structure during pre-training and an encoder-only structure during prediction.

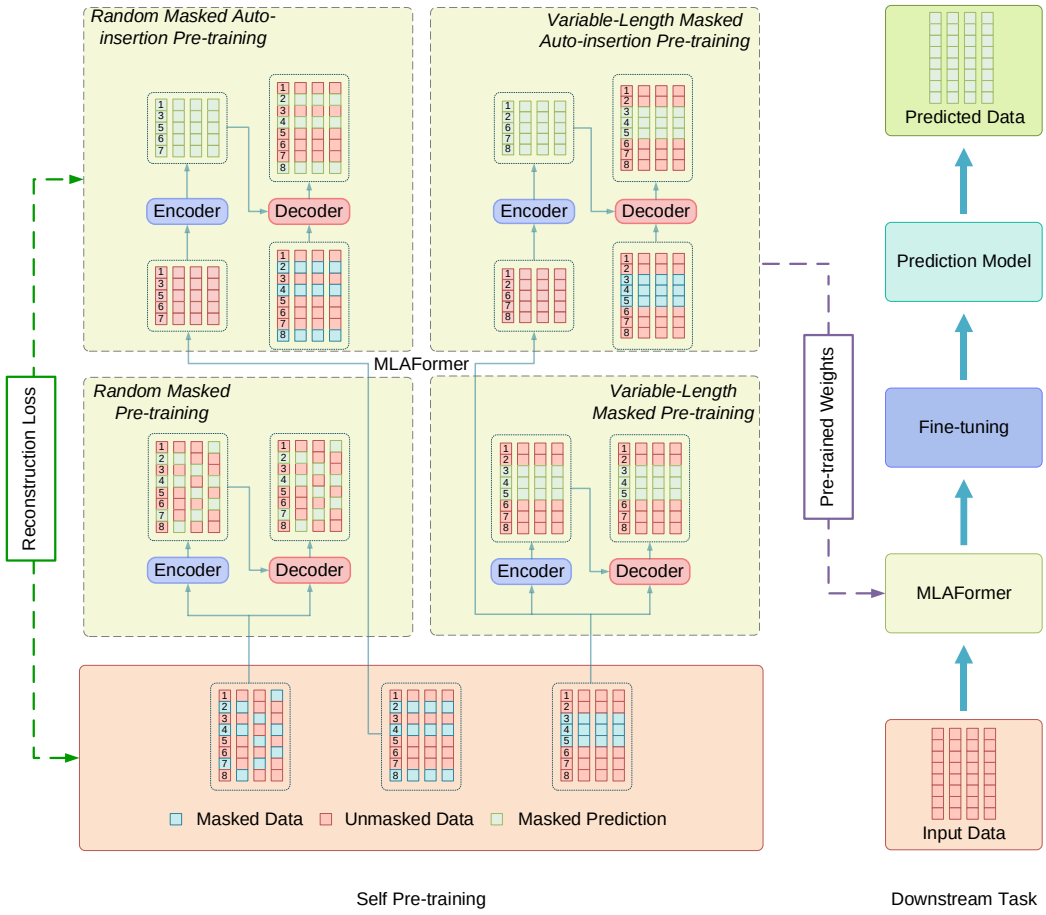



**Fig. 2** Pre-training architecture. In different pre-training strategies, part of the sequence is masked. The downstream task loads and fine-tunes weights of the pre-training model.

### 4.2 Masking

Different from the prediction task where future time series are padded, the self-supervised pre-training allows for the existence of various masking strategies. Primarily, we adopt a cloze strategy, where a certain portion of the data is masked randomly and filled in using the mean value. For time series $X \in R^{L \times M}$ , $mr$ is its masking ratio. It can be expressed as follows.

$$X\_mr = Padding(Masking(X)), \tag{9}$$

where $X\_mr \in \mathbb{R}^{L\times M}$ is the filled data after being masked. $Masking(\cdot)$ indicates the masking operation on the original data. $Padding(\cdot)$ indicates the mean fill operation for the masked data.

Based on this, we also design a masking strategy with variable-length sub-sequences. The input sequence of each batch is of a different length. For time series $X \in R^{L\times M}$, a consecutive subsequence with length *vl* is masked and the remaining part is recovered to a fixed length using the mean value. This process can be expressed as follows.

$$\begin{aligned} X\_mean &= Mean(X_{vl+1:L}), \\ X\_vl &= Concat(X\_mean, X_{vl+1:L}), \end{aligned} \tag{10}$$

where $X\_mean \in \mathbb{R}^{vl\times M}$ is the mean of $X_{vl+1:L}$. $X\_vl$ is the sequence after filling the variable-length sub-sequence to a fixed length. The value of *vl* is for random generation. $Mean(\cdot)$ represents the operation of taking the mean, and $Concat(\cdot)$ represents the concatenation operation.

### 4.3 Self Encoder-Decoder Pre-training

During the pre-training stage, MLAFormer reconstructs the input sequences. The encoder maps the masked data to the latent feature space. The decoder maps the latent features back to the original feature space to generate an output close to the ground truth. As shown in Fig. 2, we propose asymmetric self pre-training, which achieves both inserting masked data and recovering positional encoding. In the encoder, masked sequences are removed, leaving only visible positions for encoding (e.g., 1, 3, 5, 6, 7). In the decoder, missing sequences are replaced with the mean value of unmasked sequences. Thus, the model can learn positional information of missing sequences automatically and significantly reduce the computational cost as the length of the input data decreases.

### 4.4 Reconstruction Target

Fine-tuning uses the knowledge learned during the pre-training stage to set the model parameters of the downstream task at a better starting position (Li et al., 2020). Pre-training makes further optimization and improves prediction accuracy. The parameters do not have to be learned from scratch, saving training time and space. After fine-tuning, the decoder outputs the reconstructed time series. To evaluate the pre-trained model's learning, we calculate the mean absolute error (MAE) between the original and reconstructed time series. Current pre-trained models solely calculate the prediction error of the masked data, overly pursuing the local prediction performance while ignoring the global correlation (Zhou et al., 2020). Thus, to enhance self-supervised learning signals and enable better reconstruction, we consider both masked data prediction and unmasked data restoration.

## 5 Experiments

### 5.1 Datasets

To evaluate the effectiveness of MLAFormer in time series forecasting, we conduct experiments on the following public datasets.

**ETT**[1] records the oil temperature data of power transformers from July 2016 to July 2018. The time granularity of ETTm1 and ETTm2 is 15 minutes. The time granularity of ETTh1 and ETTh2 is 1 hour. The ratio of the training/validation/test is 12 months/4 months/4 months.

**Weather**[2] records weather conditions such as air temperature and air pressure of weather stations in Germany.

**Illness**[3] records weekly data on patients with influenza-like illness from the US Centers for Disease Control and Prevention from 2002 to 2020. The Weather and Illness datasets are divided into training, validation, and test sets with a ratio of 7:1:2.

### 5.2 Experimental settings

For the Illness dataset, the input length is 36 and the prediction lengths are 36, 45, 54, and 60. For the other datasets, the input length is 96 and the prediction lengths are 96, 192, 336, and 720. All the predictions in this paper are multivariate predictions. The batch size is set to 32. Each encoder has 2 layers. Each decoder has 1 layer. The embedding dimension is 512. The model is trained with the AdamW Optimizer with L1 loss. The initial learning rate is $10^{-3}$. The probability of dropout is 0.05. We choose $MAE = \frac{1}{n}\sum_{i=1}^{n}|y-\hat{y}|$ and $MSE = \frac{1}{n}\sum_{i=1}^{n}(y-\hat{y})^2$ as evaluation metrics. The experiments are repeated five times and the mean value is taken as the final result. The computing environment is shown in Table 2.

**Table 2** Computing environment.

| Language/Environment | Version/Type |
|---|---|
| GPU | NVIDIA Quadro RTX 8000 (48GB) |
| CPU | AMD Ryzen 9 5900X 12-core Processor |

[1] https://github.com/zhouhaoyi/ETDataset
[2] https://www.bgc-jena.mpg.de/wetter/
[3] https://gis.cdc.gov/grasp/fluview/fluportaldashboard.html

| | |
|---|---|
| RAM | 128 GB |
| CUDA | 11.7 |
| Python | 3.7 |
| Pytorch | 1.13.0 |

### 5.3 Baselines

We compare MLAFormer with the following state-of-the-art baselines:

1) **Transformer-based models**

Transformer (Vaswani et al., 2017) is a neural network architecture, consisting of multi-head self-attention layers and fully connected layers. Self-attention maps a query and a set of key-value pairs to compute sequence representation.

LogTrans (Li et al., 2019) pays attention to sparsity in self-attention and reduces space complexity. Compared to canonical Transformer, it produces equivalent or superior outcomes with less memory usage.

Informer (Zhou et al., 2021) employs ProbSparse self-attention mechanism to capture long-term dependencies and achieves long-sequence outputs. It uses the distilling operation to focus on dominant features in the stacked layers.

Autoformer (Wu et al., 2021) is a progressive decomposition architecture, which separates seasonal and trend components. Its Auto-Correlation utilizes the sub-series similarity to discover dependencies.

ETSformer (Woo et al.) introduces inductive biases of time-series structures and proposes novel Exponential Smoothing Attention (ESA) and Frequency Attention (FA). ESA assigns higher weights to recent observations using exponential decay, while FA utilizes Fourier transformation to grasp the prevailing periodic fluctuations.

Pyraformer (Liu et al., 2022) introduces the pyramidal attention module to model multi-range dependencies, where inter-scale connections represent features with different resolutions and intra-scale connections link neighboring points together.

Yformer (Madhusudhanan et al., 2023) is a U-Net inspired architecture, which connects the downscaled encoder layer to the corresponding upsampled decoder layer.

Preformer (Du et al., 2023) introduces segment-wise attention mechanism, which reduces the computational cost. In addition, it designs a predictive paradigm PreMSSC, using one segment delayed values and the corresponding keys for calculation.

PatchTST (Nie et al., 2023) introduces channel-independent patches as input tokens to Transformer, which enhance the learning of local semantic information and reduce computation and memory usage.

2) **Non-Transformer-based models**

TS2Vec (Yue et al., 2022) is a unified architecture for time series representation with contrastive learning. Hierarchical contrasting leverages max pooling to perform sequence representation, with instance-wise contrastive losses and temporal contrastive losses complementing each other.

LightTS (Zhang et al., 2022) is a simple and lightweight architecture based on MLP structures, which adaptively learns effective information. It uses continuous and interval sampling to capture both short-term and long-term patterns.

TimesNet (Wu et al., 2023) is a task-general time series model, which transforms 1D time series into 2D tensors and models both intraperiod- and interperiod-variations.

### 5.4 Model Comparison

As shown in Table 3, MLAFormer achieves the state-of-the-art performance. Compared with the most recent advanced model PatchTST, the MAE of MLAFormer is reduced by 3.57% (0.411→0.396) in ETTh1, 4.55% (0.355→0.339) in ETTh2, 6.34% (0.368→0.344) in ETTm1, 4.72% (0.266→0.253) in ETTm2, 3.39% (0.209→0.202) in Weather, and 3.70% (0.821→0.791) in Illness for the output length of 96. The MSE of MLAFormer is reduced by 4.38% (0.402→0.385) in ETTh1, 3.46% (0.308→0.297) in ETTh2, 3.32% (0.311→0.320) in ETTm1, 1.79% (181→0.178) in ETTm2, 7.00% (0.176→0.164) in Weather, and 8.07% (1.788→1.644) in Illness for the output length of 96. On average, MLAFormer achieves the overall 32.59% and 39.94% improvements on MAE and MSE, respectively. We also observe that as the prediction length increases, MLAFormer has a consistent prediction advantage without error surges on long sequences, showing its competitiveness in long-term time-series forecasting.

### 5.5 Pre-training Analysis

We perform pre-training based on MLAFormer. As shown in Table 4, our pre-training outperforms training from scratch for the six datasets. RM, VM, RMAI, and VMAI pre-training further achieve an overall 4.31%, 4.16%, 3.59% and 3.98% reduction on MSE, respectively. Although pre-training increases the cost, fine-tuning the parameters of the pre-training model can significantly improve prediction performance and accelerate convergence.

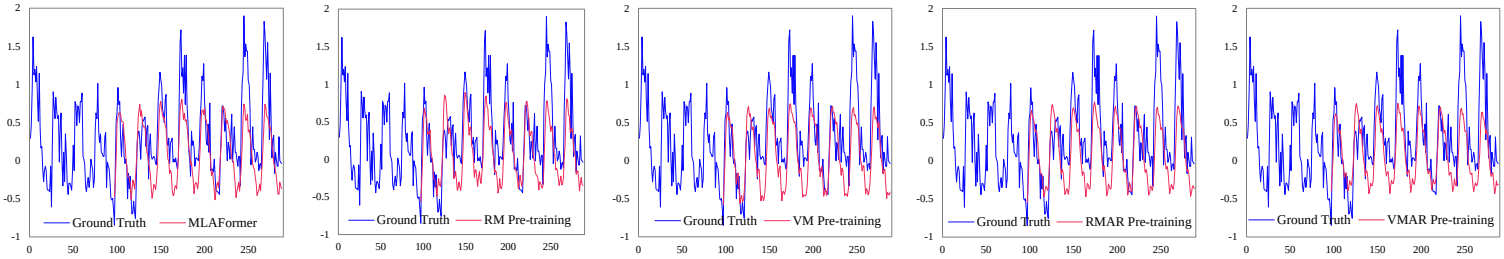
Fig. 3 shows the prediction slice of MLAFormer and pre-training methods. They demonstrate the capability to capture the periodicity and trend. The variations between these methods are slight, but not negligible. Pre-training methods fit overall variations better and provide more accurate learning when the values of the time series are at inflection points. RMAI pre-training and VMAI pre-training remove masked sequences in the encoder, which are already comparable with training with the entire series. This design saves the training cost and is better for large-scale datasets.

**Table 3** Results with different prediction lengths. In this paper, the best result and the second-best one are emphasized in **bold** and underlined.

| Models | | Transformer (2017) | | LogTrans (2019) | | Informer (2021) | | Autoformer (2021) | | ETSformer (2022) | | Pyraformer (2022) | | Yformer (2023) | | Preformer (2023) | | TS2Vec (2022) | | LightTS (2022) | | TimesNet (2023) | | PatchTST (2023) | | **MLAFormer（Ours）** | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Metrics | | MAE | MSE | MAE | MSE | MAE | MSE | MAE | MSE | MAE | MSE | MAE | MSE | MAE | MSE | MAE | MSE | MAE | MSE | MAE | MSE | MAE | MSE | MAE | MSE | MAE | MSE |
| ETTh1 | 96 | 0.893 | 1.525 | 0.876 | 1.174 | 0.894 | 1.280 | 0.482 | 0.470 | 0.476 | 0.490 | 0.665 | 0.799 | 0.777 | 0.940 | 0.456 | 0.435 | 0.611 | 0.704 | 0.454 | 0.454 | 0.443 | 0.441 | 0.411 | 0.402 | **0.396** | **0.385** |
| | 192 | 0.759 | 0.986 | 0.798 | 1.057 | 0.733 | 0.959 | 0.504 | 0.505 | 0.499 | 0.538 | 0.731 | 0.922 | 0.790 | 1.002 | 0.480 | 0.467 | 0.665 | 0.809 | 0.482 | 0.502 | 0.481 | 0.502 | 0.445 | 0.456 | **0.429** | **0.444** |
| | 336 | 0.832 | 1.124 | 0.808 | 1.098 | 0.807 | 1.099 | 0.504 | 0.527 | 0.516 | 0.573 | 0.772 | 1.016 | 0.837 | 1.126 | 0.494 | 0.486 | 0.745 | 0.960 | 0.513 | 0.555 | 0.504 | 0.545 | 0.456 | 0.495 | **0.447** | **0.479** |
| | 720 | 0.953 | 1.322 | 0.819 | 1.110 | 0.910 | 1.267 | 0.555 | 0.620 | 0.545 | 0.584 | 0.818 | 1.140 | 0.977 | 1.420 | 0.540 | 0.542 | 0.818 | 1.106 | 0.572 | 0.619 | 0.541 | 0.585 | 0.485 | 0.523 | **0.470** | **0.488** |
| ETTh2 | 96 | 1.090 | 1.865 | 1.183 | 2.202 | 1.304 | 2.570 | 0.474 | 0.459 | 0.384 | 0.334 | 1.066 | 1.736 | 1.037 | 1.627 | 0.409 | 0.367 | 0.782 | 1.006 | 0.445 | 0.408 | 0.385 | 0.355 | 0.355 | 0.308 | **0.339** | **0.297** |
| | 192 | 1.315 | 2.474 | 1.528 | 3.429 | 1.511 | 3.151 | 0.496 | 0.508 | 0.433 | 0.428 | 1.499 | 3.514 | 1.330 | 2.741 | 0.470 | 0.472 | 1.120 | 2.065 | 0.522 | 0.553 | 0.444 | 0.462 | 0.404 | 0.388 | **0.386** | **0.370** |
| | 336 | 1.336 | 2.573 | 1.576 | 3.556 | 1.450 | 2.890 | 0.492 | 0.488 | 0.468 | 0.476 | 1.340 | 2.825 | 1.350 | 2.796 | 0.515 | 0.521 | 1.296 | 2.724 | 0.565 | 0.635 | 0.465 | 0.480 | 0.432 | 0.418 | **0.416** | **0.401** |
| | 720 | 1.362 | 2.498 | 1.520 | 3.316 | 1.491 | 2.948 | 0.495 | 0.489 | 0.497 | 0.503 | 1.335 | 2.859 | 1.258 | 2.228 | 0.554 | 0.568 | 1.312 | 2.495 | 0.701 | 0.966 | 0.489 | 0.509 | 0.445 | 0.434 | **0.435** | **0.419** |
| ETTm1 | 96 | 0.571 | 0.646 | 0.545 | 0.600 | 0.546 | 0.604 | 0.469 | 0.476 | 0.465 | 0.451 | 0.556 | 0.598 | 0.543 | 0.620 | 0.431 | 0.398 | 0.557 | 0.624 | 0.410 | 0.378 | 0.381 | 0.347 | 0.368 | 0.331 | **0.344** | **0.320** |
| | 192 | 0.601 | 0.700 | 0.604 | 0.721 | 0.579 | 0.638 | 0.493 | 0.513 | 0.486 | 0.498 | 0.576 | 0.639 | 0.689 | 0.843 | 0.453 | 0.433 | 0.590 | 0.688 | 0.427 | 0.414 | 0.415 | 0.403 | 0.395 | 0.380 | **0.375** | **0.374** |
| | 336 | 0.692 | 0.873 | 0.713 | 0.917 | 0.713 | 0.850 | 0.514 | 0.542 | 0.515 | 0.541 | 0.649 | 0.784 | 0.764 | 0.927 | 0.468 | 0.455 | 0.624 | 0.726 | 0.457 | 0.456 | 0.451 | 0.460 | 0.413 | **0.402** | **0.398** | 0.404 |
| | 720 | 0.829 | 1.191 | 0.762 | 0.997 | 0.757 | 0.973 | 0.526 | 0.567 | 0.563 | 0.623 | 0.727 | 0.945 | 0.776 | 0.984 | 0.487 | 0.484 | 0.670 | 0.806 | 0.509 | 0.536 | 0.491 | 0.544 | 0.441 | 0.475 | **0.436** | **0.474** |
| ETTm2 | 96 | 0.461 | 0.392 | 0.753 | 0.928 | 0.439 | 0.360 | 0.341 | 0.264 | 0.300 | 0.204 | 0.550 | 0.508 | 0.660 | 0.787 | 0.301 | 0.210 | 0.492 | 0.440 | 0.345 | 0.253 | 0.275 | 0.200 | 0.266 | 0.181 | **0.253** | **0.178** |
| | 192 | 0.602 | 0.594 | 0.702 | 0.775 | 0.629 | 0.649 | 0.356 | 0.294 | 0.346 | 0.271 | 0.778 | 0.979 | 0.763 | 1.023 | 0.356 | 0.292 | 0.597 | 0.618 | 0.410 | 0.346 | 0.319 | 0.268 | 0.313 | 0.252 | **0.303** | **0.249** |
| | 336 | 0.818 | 1.080 | 0.889 | 1.239 | 0.864 | 1.178 | 0.401 | 0.367 | 0.411 | 0.366 | 0.930 | 1.396 | 1.179 | 2.226 | 0.388 | 0.340 | 0.793 | 1.029 | 0.476 | 0.463 | 0.361 | 0.334 | 0.353 | 0.316 | **0.337** | **0.304** |
| | 720 | 1.394 | 3.300 | 1.477 | 2.979 | 1.493 | 3.438 | 0.470 | 0.494 | 0.493 | 0.507 | 1.371 | 3.300 | 1.520 | 3.623 | 0.459 | 0.464 | 1.084 | 1.898 | 0.618 | 0.743 | 0.442 | 0.483 | 0.409 | 0.414 | **0.396** | **0.406** |
| Weather | 96 | 0.232 | 0.174 | 0.286 | 0.221 | 0.234 | 0.174 | 0.261 | 0.196 | 0.260 | 0.199 | 0.537 | 0.564 | 0.249 | 0.196 | 0.238 | 0.178 | 0.523 | 0.606 | 0.227 | 0.189 | 0.209 | 0.171 | 0.209 | 0.176 | **0.202** | **0.164** |
| | 192 | 0.286 | 0.230 | 0.303 | 0.249 | 0.297 | 0.239 | 0.307 | 0.254 | 0.293 | 0.246 | 0.497 | 0.530 | 0.334 | 0.310 | 0.297 | 0.244 | 0.545 | 0.629 | 0.265 | 0.230 | 0.256 | 0.222 | 0.250 | 0.221 | **0.240** | **0.207** |
| | 336 | 0.349 | 0.306 | 0.338 | 0.301 | 0.349 | 0.307 | 0.345 | 0.307 | 0.338 | 0.309 | 0.510 | 0.558 | 0.370 | 0.358 | 0.351 | 0.318 | 0.599 | 0.737 | 0.302 | 0.277 | 0.299 | 0.283 | 0.291 | 0.279 | **0.283** | **0.260** |
| | 720 | 0.401 | 0.377 | 0.386 | 0.382 | 0.402 | 0.376 | 0.395 | 0.386 | 0.385 | 0.379 | 0.557 | 0.627 | 0.526 | 0.619 | 0.414 | 0.415 | 0.770 | 1.106 | 0.356 | 0.346 | 0.349 | 0.357 | 0.341 | 0.355 | **0.339** | **0.341** |
| Illness | 36 | 1.371 | 4.401 | 1.779 | 6.325 | 1.461 | 4.808 | 1.120 | 2.818 | 1.525 | 4.834 | 1.343 | 4.180 | 1.516 | 5.067 | 1.153 | 3.381 | 1.221 | 3.439 | 1.931 | 7.008 | 0.862 | 2.135 | 0.821 | 1.788 | **0.791** | **1.644** |
| | 45 | 1.480 | 4.753 | 1.806 | 6.502 | 1.489 | 4.863 | 1.243 | 3.214 | 1.432 | 4.274 | 1.370 | 4.269 | 1.577 | 5.435 | 1.147 | 3.228 | 1.304 | 3.826 | 1.923 | 6.952 | 0.877 | 2.110 | 0.843 | 1.798 | **0.804** | **1.716** |
| | 54 | 1.473 | 4.741 | 1.850 | 6.769 | 1.503 | 4.931 | 1.254 | 3.277 | 1.402 | 4.038 | 1.384 | 4.302 | 1.515 | 5.174 | 1.132 | 3.132 | 1.303 | 3.893 | 1.906 | 6.832 | 0.898 | 2.158 | 0.872 | 1.863 | **0.802** | **1.690** |
| | 60 | 1.549 | 5.046 | 1.837 | 6.694 | 1.539 | 5.070 | 1.289 | 3.426 | 1.411 | 4.096 | 1.404 | 4.375 | 1.554 | 5.309 | 1.155 | 3.208 | 1.286 | 3.817 | 1.921 | 6.841 | 0.942 | 2.345 | 0.876 | 1.958 | **0.806** | **1.673** |

**Table 4** Pre-training results with different prediction lengths.

| Method | | Metric | ETTh1 | | | | ETTh2 | | | | ETTm1 | | | | ETTm2 | | | | Weather | | | | Illness | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | 96 | 192 | 336 | 720 | 96 | 192 | 336 | 720 | 96 | 192 | 336 | 720 | 96 | 192 | 336 | 720 | 96 | 192 | 336 | 720 | 36 | 45 | 54 | 60 |
| MLAFormer | | MAE | 0.396 | 0.429 | 0.447 | 0.470 | 0.339 | 0.386 | 0.416 | 0.435 | 0.344 | 0.375 | 0.398 | 0.436 | 0.253 | 0.303 | 0.337 | 0.396 | 0.202 | 0.240 | 0.283 | 0.339 | 0.791 | 0.804 | 0.802 | 0.806 |
| | | MSE | 0.385 | 0.444 | 0.479 | 0.488 | 0.297 | 0.370 | 0.401 | 0.419 | 0.320 | 0.374 | 0.404 | 0.474 | 0.178 | 0.249 | 0.304 | 0.406 | 0.164 | 0.207 | 0.260 | 0.341 | 1.644 | 1.716 | 1.690 | 1.673 |
| RM | Pre-training | MAE | 0.391 | 0.422 | 0.438 | 0.455 | 0.337 | 0.383 | 0.409 | 0.430 | **0.337** | **0.367** | 0.392 | 0.431 | 0.251 | **0.296** | 0.335 | **0.391** | **0.192** | **0.235** | **0.277** | 0.333 | **0.689** | 0.734 | 0.758 | 0.807 |
| | | MSE | 0.379 | **0.431** | 0.468 | **0.462** | 0.293 | 0.366 | 0.383 | 0.408 | **0.309** | **0.362** | 0.396 | **0.463** | 0.174 | **0.241** | 0.303 | **0.397** | **0.157** | **0.201** | **0.256** | 0.336 | **1.296** | 1.387 | **1.467** | 1.685 |
| VM | Pre-training | MAE | **0.391** | **0.419** | **0.436** | **0.455** | **0.332** | 0.379 | 0.410 | 0.429 | 0.338 | 0.369 | 0.392 | **0.430** | **0.250** | 0.296 | **0.335** | 0.392 | 0.196 | 0.235 | 0.281 | 0.335 | 0.719 | 0.725 | **0.756** | **0.786** |
| | | MSE | **0.378** | 0.432 | 0.471 | 0.468 | 0.288 | 0.362 | 0.374 | 0.407 | 0.311 | 0.366 | 0.395 | 0.464 | **0.174** | 0.242 | **0.300** | 0.397 | 0.159 | 0.202 | 0.258 | 0.336 | 1.408 | 1.374 | 1.518 | **1.591** |
| RMAI | Pre-training | MAE | 0.396 | 0.424 | 0.448 | 0.455 | 0.333 | **0.379** | 0.408 | **0.428** | 0.339 | 0.368 | **0.386** | 0.430 | 0.252 | 0.298 | 0.337 | 0.394 | 0.199 | 0.239 | 0.280 | **0.332** | 0.740 | **0.709** | 0.770 | 0.824 |
| | | MSE | 0.384 | 0.438 | 0.448 | 0.466 | **0.284** | **0.36** | 0.391 | **0.406** | 0.312 | 0.366 | **0.372** | 0.465 | 0.176 | 0.244 | 0.304 | 0.400 | 0.162 | 0.204 | 0.259 | **0.336** | 1.495 | **1.335** | 1.518 | 1.753 |
| VMAI | Pre-training | MAE | 0.396 | 0.423 | 0.440 | 0.456 | 0.334 | 0.381 | **0.405** | 0.431 | 0.340 | 0.370 | 0.391 | 0.431 | 0.252 | 0.297 | 0.337 | 0.394 | 0.196 | 0.238 | 0.279 | 0.333 | 0.744 | 0.739 | 0.766 | 0.798 |
| | | MSE | 0.385 | 0.435 | **0.438** | 0.465 | 0.288 | 0.360 | **0.365** | 0.410 | 0.314 | 0.367 | 0.378 | 0.464 | 0.175 | 0.243 | 0.303 | 0.400 | 0.159 | 0.204 | 0.258 | 0.337 | 1.455 | 1.451 | 1.498 | 1.663 |




(a) MLAFormer (b) RM Pre-training (c) VM Pre-training (d) RMAI Pre-training (e) VMAI Pre-training

**Fig. 3** Prediction (length =192) on the ETTh1 dataset

### 5.6 Ablation Study

To quantify the individual improvement of each block, we perform extensive ablation experiments. We remove each improvement and quantify the increased error versus our original architecture.

1) **Multi-scale Encoder-Decoder**

In this work, we propose to utilize average pooling and upsampling to perform cross multi-scale encoder-decoder. We validate the effectiveness of multi-scale representation in MLAFormer. As shown in Table 5, compared with MLAFormer without multi-scale, the complete method gives 3.18% average MAE reduction and 5.28% average MSE reduction in ETTh1, and 1.73% average MAE reduction and 1.98% average MSE reduction in ETTm2. The results demonstrate that multi-scale representation of time series can capture dependencies at different resolutions, which is critical for forecasting. Time series data often exhibit patterns and trends at different scales. By incorporating multi-scale representation, the model can better identify and leverage these patterns.

**Table 5** Multi-scale ablation study on MLAFormer. "-ms" denotes MLAFormer without multi-scale.

| Models | | RM$^{-ms}$ Pre-training | | RM Pre-training | | VM$^{-ms}$ Pre-training | | VM Pre-training | | RMAI$^{-ms}$ Pre-training | | RMAI Pre-training | | VMAI$^{-ms}$ Pre-training | | VMAI Pre-training | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Metrics | | MAE | MSE | MAE | MSE | MAE | MSE | MAE | MSE | MAE | MSE | MAE | MSE | MAE | MSE | MAE | MSE |
| ETTh1 | 96 | 0.407 | 0.400 | 0.391 | 0.379 | 0.407 | 0.401 | 0.391 | 0.378 | 0.411 | 0.403 | 0.396 | 0.384 | 0.417 | 0.415 | 0.396 | 0.385 |
| | 192 | 0.430 | 0.449 | 0.422 | 0.431 | 0.434 | 0.453 | 0.419 | 0.432 | 0.438 | 0.456 | 0.424 | 0.438 | 0.444 | 0.463 | 0.423 | 0.435 |
| | 336 | 0.447 | 0.485 | 0.438 | 0.468 | 0.454 | 0.494 | 0.436 | 0.471 | 0.453 | 0.489 | 0.448 | 0.448 | 0.458 | 0.497 | 0.440 | 0.438 |
| | 720 | 0.468 | 0.483 | 0.455 | 0.462 | 0.461 | 0.474 | 0.455 | 0.468 | 0.470 | 0.483 | 0.455 | 0.466 | 0.473 | 0.490 | 0.456 | 0.465 |
| ETTm2 | 96 | 0.257 | 0.179 | 0.251 | 0.174 | 0.262 | 0.184 | 0.250 | 0.174 | 0.257 | 0.179 | 0.252 | 0.176 | 0.257 | 0.181 | 0.252 | 0.175 |
| | 192 | 0.299 | 0.245 | 0.296 | 0.241 | 0.301 | 0.245 | 0.296 | 0.242 | 0.303 | 0.248 | 0.298 | 0.244 | 0.301 | 0.245 | 0.297 | 0.243 |
| | 336 | 0.343 | 0.310 | 0.335 | 0.303 | 0.337 | 0.304 | 0.335 | 0.300 | 0.339 | 0.307 | 0.337 | 0.304 | 0.344 | 0.310 | 0.337 | 0.303 |
| | 720 | 0.398 | 0.406 | 0.391 | 0.397 | 0.402 | 0.412 | 0.392 | 0.397 | 0.396 | 0.402 | 0.394 | 0.400 | 0.396 | 0.404 | 0.394 | 0.400 |

2) **Local Convolutional Auto-Correlation**

The architecture of MLAFormer has good compatibility with attention mechanisms. Local Convolutional Auto-Correlation in MLAFormer is replaced with other attention mechanisms. As seen in Table 6, our attention achieves the best results. Our attention mechanism is capable of perceiving local variations and integrating frequency-domain learning. It also reflects the superiority of MLAFormer, where the same attention yields better results than the original architecture.

3) **Pre-series Decomposition Block**

We attempt to verify the effectiveness of different time series decomposition structures. In MLAformer, the series is decomposed before the attention block. As shown in Table 7, for the dataset ETTh1, the average MAE and MSE of MLAFormer are reduced by 2.74% and 4.68%, respectively. For the dataset ETTm2, the average MAE and MSE of MLAFormer are reduced by 2.65% and 2.81%, respectively. The pre-series decomposition block preserves the nature of time series and facilitates the model to discover seasonal and trend variations.

**Table 6** Results of attention mechanisms in MLAFormer.

| Models | | Full Attention | | LogSparse Attention | | ProbSparse Attention | | Auto-Correlation | | Local Convolutional Auto-Correlation | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Metrics | | MAE | MSE | MAE | MSE | MAE | MSE | MAE | MSE | MAE | MSE |
| ETTh1 | 96 | 0.414 | 0.417 | 0.407 | 0.398 | 0.412 | 0.409 | 0.412 | 0.404 | **0.396** | **0.385** |
| | 192 | 0.439 | 0.463 | 0.436 | 0.456 | 0.443 | 0.471 | 0.439 | 0.456 | **0.429** | **0.444** |
| | 336 | 0.455 | 0.500 | 0.453 | 0.489 | 0.456 | 0.497 | 0.455 | 0.494 | **0.447** | **0.479** |
| | 720 | 0.491 | 0.530 | 0.472 | 0.490 | 0.476 | 0.494 | 0.478 | 0.497 | **0.470** | **0.488** |
| ETTm2 | 96 | 0.259 | 0.182 | 0.259 | 0.180 | 0.259 | 0.182 | 0.270 | 0.193 | **0.253** | **0.178** |
| | 192 | 0.305 | 0.253 | 0.327 | 0.276 | 0.306 | 0.254 | 0.307 | 0.253 | **0.303** | **0.249** |
| | 336 | 0.342 | 0.308 | 0.342 | 0.312 | 0.343 | 0.309 | 0.345 | 0.315 | **0.337** | **0.304** |
| | 720 | 0.399 | 0.413 | 0.411 | 0.425 | 0.400 | 0.409 | 0.400 | 0.412 | **0.396** | **0.406** |

**Table 7** Results of series decomposition blocks. "MLAFormer-sd" denotes that the series decomposition block is after attention in MLAFormer.

| Models | | MLAFormer$^{-sd}$ | | MLAFormer | |
|---|---|---|---|---|---|
| Metrics | | MAE | MSE | MAE | MSE |
| ETTh1 | 96 | 0.410 | 0.404 | 0.396 | 0.385 |
| | 192 | 0.445 | 0.469 | 0.429 | 0.444 |
| | 336 | 0.452 | 0.496 | 0.447 | 0.479 |
| | 720 | 0.483 | 0.514 | 0.470 | 0.488 |
| ETTm2 | 96 | 0.258 | 0.180 | 0.253 | 0.178 |
| | 192 | 0.318 | 0.266 | 0.303 | 0.249 |
| | 336 | 0.344 | 0.310 | 0.337 | 0.304 |
| | 720 | 0.403 | 0.414 | 0.396 | 0.406 |

4) **Multi-scale Positional Encoding**

Considering that the sequence aggregation process already contains the sequence information, positional encoding is omitted in Autoformer. In this paper, we devise multi-scale positional encoding to enhance order information. As shown in Table 8, models added with positional encoding yield better results. For the dataset ETTh1, the average MSE of MLAFormer is 4.10%, 2.65%, and 2.96% lower than that of MLAFORMER$^{-p0}$, MLAFORMER$^{-p1}$ , and MLAFORMER$^{-p2}$, respectively. The results highlight that multi-scale positional encoding is effective in preserving order information and distinguishing relative positional information within the multi-scale representation.

**Table 8** Results of positional encoding. "MLAFormer$^{-p0}$" denotes MLAFormer without positional encoding. "MLAFormer$^{-p1}$" denotes MLAFormer without positional encoding in the input embedding. "MLAFormer$^{-p2}$" denotes MLAFormer without parity positional encoding.

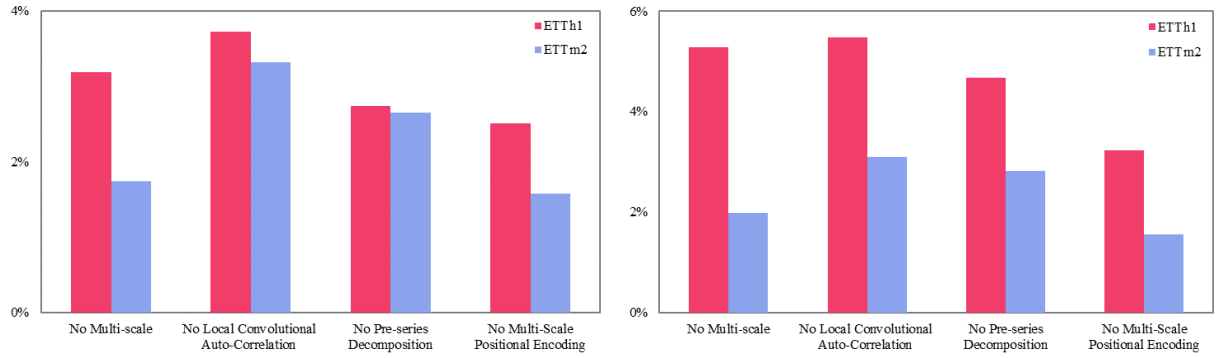
| Models | | MLAFormer$^{-p0}$ | | MLAFormer$^{-p1}$ | | MLAFormer$^{-p2}$ | | MLAFormer | |
|---|---|---|---|---|---|---|---|---|---|
| Metrics | | MAE | MSE | MAE | MSE | MAE | MSE | MAE | MSE |
| ETTh1 | 96 | 0.414 | 0.409 | 0.411 | 0.398 | 0.413 | 0.402 | 0.396 | 0.385 |
| | 192 | 0.441 | 0.460 | 0.438 | 0.456 | 0.440 | 0.458 | 0.429 | 0.444 |
| | 336 | 0.458 | 0.497 | 0.451 | 0.489 | 0.455 | 0.496 | 0.447 | 0.479 |
| | 720 | 0.485 | 0.504 | 0.478 | 0.499 | 0.477 | 0.493 | 0.470 | 0.488 |
| ETTm2 | 96 | 0.261 | 0.184 | 0.262 | 0.184 | 0.257 | 0.180 | 0.253 | 0.178 |
| | 192 | 0.305 | 0.250 | 0.304 | 0.251 | 0.305 | 0.250 | 0.303 | 0.249 |
| | 336 | 0.345 | 0.313 | 0.341 | 0.308 | 0.343 | 0.310 | 0.337 | 0.304 |
| | 720 | 0.403 | 0.414 | 0.402 | 0.414 | 0.399 | 0.409 | 0.396 | 0.406 |

**Table 9** Computational cost and ranking in prediction accuracy. The dataset is ETTh1 with the prediction length of 192. A lower ranking indicates higher accuracy.

| Model | Parameter (M) | GPU Memory (MiB) | Training Time (s/epoch) | Inference Time (s/iter) | Ranking |
|---|---|---|---|---|---|
| MLAFormer$^{-ms}$ | 5.75 | 1646 | 9.73 | 0.013 | — |
| Full Attention | 3.65 | 1426 | 15.24 | 0.014 | — |
| LogSparse Attention | 3.65 | 1432 | 15.29 | 0.016 | — |
| ProbSparse Attention | 3.65 | 1442 | 18.43 | 0.019 | — |
| Auto-Correlation | 3.65 | 1704 | 22.90 | 0.024 | — |
| MLAFormer | 5.75 | 1764 | 23.56 | 0.023 | 1 |
| PatchTST | 6.05 | 1568 | 4.79 | 0.005 | 2 |
| TimesNet | 9.22 | 3854 | 61.92 | 0.075 | 3 |
| LightTS | 0.14 | 838 | 2.51 | 0.002 | 7 |
| TS2Vec | 33.42 | 2120 | 24.48 | 0.015 | 8 |
| Preformer | 4.63 | 2466 | 23.68 | 0.029 | 4 |
| Yformer | 13.84 | 2454 | 22.92 | 0.022 | 12 |
| Pyraformer | 6.25 | 1568 | 4.75 | 0.005 | 10 |
| ETSformer | 1.35 | 1728 | 9.27 | 0.012 | 6 |
| Autoformer | 9.62 | 2738 | 31.43 | 0.069 | 5 |
| Informer | 5.42 | 1592 | 9.32 | 0.011 | 11 |
| LogTrans | 30.38 | 3366 | 29.74 | 0.050 | 12 |
| Transformer | 44.15 | 3486 | 2114.44 | 8.059 | 9 |

5) **Ablation Study Analysis**

The above ablation study shows that the optimization of each block contributes to prediction accuracy. As shown in Fig. 4, multi-scale representation, Local Convolutional Auto-Correlation, pre-series decomposition, and multi-scale positional encoding reduce the MAE by 3.18%, 3.72%, 2.74%, and 2.50% for the dataset ETTh1, and reduce the MSE by 1.98%, 3.10%, 2.81%, and 1.56% for the dataset ETTm2, respectively. Each block is effective, especially multi-scale representation and Local Convolutional Auto-Correlation yielding the largest improvement.




(a) MAE reduction of each block (b) MSE reduction of each block

**Fig. 4** Results of ablation analysis. The datasets are ETTh1 and ETTm2. The prediction lengths are 96, 192, 336, 720.

### 5.7 Computational Cost

We compare computational cost in Table 9, including baseline models, MLAFormer and its ablation models. We observe that for MLAFormer, the multi-scale transformation is the main time cost and these variants have no significant differences in memory and time usage. Comparing Transformer and its variants, MLAFormer achieves relatively competent performance in memory usage, training time, and inference time. Although the pure linear model like LightTS needs less memory and time consumption, MLAFormer achieves better accuracy. Overall, MLAFormer is acceptable in memory and time cost. It balances efficiency and

accuracy, which is essential in long-sequence modeling.

## 6 Conclusions and Future Work

In this paper, MLAFormer is proposed with multi-scale encoder-decoder architecture to capture long-term dependencies in different scales and Local Convolutional Auto-Correlation to perceive local context in each scale for time series forecasting jointly, which outperforms the existing state-of-the-art methods. Inspired by NLP and vision, we also design a set of prominent generative pre-training schemes to further improve the prediction performance. In addition, we find the proposed progressive series decomposition is essential for preserving or generating intrinsic variations of time series, and the multi-scale positional encoding helps to indicate order information for the masks reconstruction in pre-training. Our research provides new insight into neural architecture design and pre-training for long-term time series forcasting.

In the future, we will explore more effective multi-scale representation learning framework to trade off the effectiveness and efficiency between global dependencies and local context. We will further focus on the adaptability of the proposed pre-training with other architectures, extending it to non-Transformer models and decoder-only models. Moreover, we consider imitating symbolic modeling as in NLP, and utilizing prompt/instruct learning with random prediction lengths or discrete time points for both short-term and long-term prediction (Kenton and Toutanova, 2019; Ouyang et al., 2022). Eventually, achieving a multi-task time series foundation model is possible to support forecasting, imputation, and anomaly detection.

## Statements and Declarations

**Competing Interests:** On behalf of all authors, the corresponding author states that there is no conflict of interest.
**Compliance with Ethical Standards:** This article does not contain any studies with human participants or animals performed by any of the authors.

## References


Ak, R., Vitelli, V., and Zio, E. (2015). An interval-valued neural network approach for uncertainty quantification in short-term wind speed prediction. IEEE transactions on neural networks and learning systems, 26(11), 2787-2800.

Au, K.-F., Choi, T.-M., and Yu, Y. (2008). Fashion retail forecasting by evolutionary neural networks. International Journal of Production Economics, 114(2), 615-630.

Bouteska, A., Seranto, M. L., Hajek, P., and Abedin, M. Z. (2023). Data-driven decadal climate forecasting using Wasserstein time-series generative adversarial networks. Annals of Operations Research, 1-19.

Box, G. E. P., Reinsel, G. C., Jenkins, G. M., and Ljung, G. M. (1971). Time series analysis, forecasting and control.

Cho, K., Van Merriënboer, B., 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, Doha, Qatar. Association for Computational Linguistics, 1724-1734.

Dai, Z., Yang, Z., Yang, Y., Carbonell, J., Le, Q., and Salakhutdinov, R. (2019). Transformer-XL: Attentive language models beyond a fixed-length context. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, Florence, Italy. Association for Computational Linguistics, 2978-2988.

Du, D., Su, B., and Wei, Z. (2023). Preformer: predictive Transformer with multi-scale segment-wise correlations for long-term time series forecasting. In ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing, Rhodes Island, Greece, 1-5.

Elman, J. L. (1990). Finding structure in time. Cognitive science, 14(2), 179-211.

He, K., Chen, X., Xie, S., Li, Y., Dollár, P., and Girshick, R. (2022). Masked autoencoders are scalable vision learners. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, New Orleans, USA, 16000-16009.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. Neural computation, 9(8), 1735-1780.

Howard, J. and Ruder, S. (2018). Universal language model fine-tuning for text classification. In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Melbourne, Australia. Association for Computational Linguistics, 328-339.

Hu, S. X., Arefin, M. R., Nguyen, V.-N., Dipani, A., Pitkow, X., and Tolias, A. S. (2021). AvaTr: One-shot speaker extraction with Transformers. In Interspeech, Brno, Czechia.

Kenton, J. D. M.-W. C. and Toutanova, L. K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Minneapolis, Minnesota, 2.

Kim, T., Kim, J., Tae, Y., Park, C., Choi, J.-H., and Choo, J. (2021). Reversible instance normalization for accurate time-series forecasting against distribution shift. In International Conference on Learning Representations, Austria.

Lara-Benítez, P., Carranza-García, M., and Riquelme, J. C. (2021). An experimental review on deep learning architectures for time series forecasting. International Journal of Neural Systems, 31(03), 2130001.

Li, S., Jin, X., Xuan, Y., Zhou, X., Chen, W., Wang, Y.-X., and Yan, X. (2019). Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting. Advances in neural information processing systems, 32.

Li, S., Li, L., Hong, Q., and Liu, L. (2020). Improving Transformer-based speech recognition with unsupervised pre-training and multi-task semantic knowledge learning. In Interspeech, Shanghai, China, 5006-5010.

Li, Y., Yu, R., Shahabi, C., and Liu}, Y. (2018). Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. In International Conference on Learning Representation, Vancouver, Canada, 678-685.

Lin, T.-Y., Dollár, P., Girshick, R., He, K., Hariharan, B., and Belongie, S. (2017). Feature pyramid networks for object detection. In Proceedings of the IEEE conference on computer vision and pattern recognition, Hawaii, USA, 2117-2125.

Lin, T., Wang, Y., Liu, X., and Qiu, X. (2022). A survey of Transformers. AI Open, 3, 111-132.

Liu, S., Yu, H., Liao, C., Li, J., Lin, W., Liu, A. X., and Dustdar, S. (2022). Pyraformer: Low-complexity pyramidal attention for long-range time series modeling and forecasting. In International conference on learning representations, Austria.

Ma, J., Liu, J., Lin, Q., Wu, B., Wang, Y., and You, Y. (2023). Multitask learning for visual question answering. IEEE transactions on neural networks and learning systems, 34(3), 1380-1394.

Madhusudhanan, K., Burchert, J., Duong-Trung, N., Born, S., and Schmidt-Thieme, L. (2023). U-Net inspired Transformer architecture for far horizon time series forecasting. In Joint European Conference on Machine Learning and Knowledge Discovery in Databases. Springer, 36-52.

Mengyue Zha, Wong, S., Liu, M., Zhang, T., and Chen, K. (2022). Time Series Generation with Masked Autoencoder,. arXiv preprint arXiv:2109.08381.

Nelson, D. M., Pereira, A. C., and De Oliveira, R. A. (2017). Stock market's price movement prediction with LSTM neural networks. In 2017 International joint conference on neural networks (IJCNN). Ieee, 1419-1426.

Nie, Y., Nguyen, N. H., Sinthong, P., and Kalagnanam, J. (2023). A time series is worth 64 words: Long-term forecasting with Transformers. In the Eleventh International Conference on Learning Representations, Kigali, Rwanda.

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., and Ray, A. (2022). Training language models to follow instructions with human feedback. Advances in neural information processing systems, 35, 27730-27744.

Qi, X., Hou, K., Liu, T., Yu, Z., Hu, S., and Ou, W. (2021). From known to unknown: Knowledge-guided Transformer for time-series sales forecasting in alibaba. arXiv preprint arXiv:2109.08381.

Qin, Y., Song, D., Cheng, H., Cheng, W., Jiang, G., and Cottrell, G. W. (2017). A dual-stage attention-based recurrent neural network for time series prediction. In Proceedings of the 26th International Joint Conference on Artificial Intelligence, Melbourne, Australia. AAAI Press, 2627–2633.

Radford, A., Narasimhan, K., Salimans, T., and Sutskever, I. (2018). Improving language understanding by generative pre-training.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. Nature, 323(6088), 533-536.

Siniscalchi, S. M. and Salerno, V. M. (2016). Adaptation to new microphones using artificial neural networks with trainable activation functions. IEEE transactions on neural networks and learning systems, 28(8), 1959-1965.

Song, Q., Sun, B., and Li, S. (2023). Multimodal sparse Transformer network for audio-visual speech recognition. IEEE transactions on neural networks and learning systems, 34(12), 10028-10038.

Valipour, M., Banihabib, M. E., and Behbahani, S. M. R. (2013). Comparison of the ARMA, ARIMA, and the autoregressive artificial neural network models in forecasting the monthly inflow of Dez dam reservoir. Journal of hydrology, 476, 433-441.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. In Proceedings of the 31st International Conference on Neural Information Processing Systems, California, USA, 6000–6010.

Wang, G., Lin, L., Chen, R., Wang, G., and Zhang, J. (2022a). Joint Learning of Neural Transfer and Architecture Adaptation for Image Recognition. IEEE transactions on neural networks and learning systems, 33(10), 5401-5415.

Wang, Z., Xu, X., Zhang, W., Trajcevski, G., Zhong, T., and Zhou, F. (2022b). Learning latent seasonal-trend representations for time series forecasting. Advances in neural information processing systems, 35, 38775-38787.

Woo, G., Liu, C., Sahoo, D., Kumar, A., and Hoi, S. (2022). ETSFormer: Exponential smoothing Transformers for time-series forecasting. arXiv preprint arXiv:2202.01381.

Wu, H., Hu, T., Liu, Y., Zhou, H., Wang, J., and Long, M. (2023). Timesnet: Temporal 2d-variation modeling for general time series analysis. In the eleventh international conference on learning representations, Kigali, Rwanda.

Wu, H., Xu, J., Wang, J., and Long, M. (2021). Autoformer: decomposition transformers with auto-correlation for long-term series forecasting. In Proceedings of the 35th International Conference on Neural Information Processing Systems. Curran Associates Inc., Article 1717.

Xu, P., Joshi, C. K., and Bresson, X. (2022). Multigraph Transformer for free-hand sketch recognition. IEEE transactions on neural networks and learning systems, 33(10), 5150-5161.

Yang, Z., Ma, J., Chen, H., Zhang, J., and Chang, Y. (2022). Context-aware attentive multilevel feature fusion for named entity recognition. IEEE transactions on neural networks and learning systems.

Yue, Z., Wang, Y., Duan, J., Yang, T., Huang, C., Tong, Y., and Xu, B. (2022). TS2Vec: Towards universal representation of time series. In Proceedings of the AAAI Conference on Artificial Intelligence, Vancouver, Canada, 8980-8987.

Zhang, G. P., Xia, Y., and Xie, M. (2023). Intermittent demand forecasting with transformer neural networks. Annals of Operations Research.

Zhang, T., Zhang, Y., Cao, W., Bian, J., Yi, X., Zheng, S., and Li, J. (2022). Less is more: Fast multivariate time series forecasting with light sampling-oriented mlp structures. arXiv preprint arXiv:2207.01186.

Zheng, C., Wang, S., Liu, Y., Liu, C., Xie, W., Fang, C., and Liu, S. (2019). A novel equivalent model of active distribution networks based on LSTM. IEEE transactions on neural networks and learning systems, 30(9), 2611-2624.

Zhou, H., Zhang, S., Peng, J., Zhang, S., Li, J., Xiong, H., and Zhang, W. (2021). Informer: Beyond efficient Transformer for long sequence time-series forecasting. In Proceedings of the AAAI conference on artificial intelligence, Vancouver, Canada, 11106-11115.

Zhou, K., Wang, H., Zhao, W. X., Zhu, Y., Wang, S., Zhang, F., Wang, Z., and Wen, J.-R. (2020). S3-Rec: Self-supervised learning for sequential recommendation with mutual information maximization. In Proceedings of the 29th ACM International Conference on Information & Knowledge Management, New York, USA. Association for Computing Machinery, 1893–1902.