Multi-resolution Time-Series Transformer for Long-term Forecasting

Yitian Zhang[1†] **Liheng Ma**[1†] **Soumyasundar Pal**[2] **Yingxue Zhang**[2] **Mark Coates**[1]
[1]McGill University, Mila, and ILLS [2]Huawei Noah's Ark Lab

Abstract

The performance of transformers for time-series forecasting has improved significantly. Recent architectures learn complex temporal patterns by segmenting a time-series into patches and using the patches as tokens. The patch size controls the ability of transformers to learn the temporal patterns at different frequencies: shorter patches are effective for learning localized, high-frequency patterns, whereas mining long-term seasonalities and trends requires longer patches. Inspired by this observation, we propose a novel framework, **M**ulti-resolution **T**ime-**S**eries **T**ransformer (MTST), which consists of a multi-branch architecture for simultaneous modeling of diverse temporal patterns at different resolutions. In contrast to many existing time-series transformers, we employ relative positional encoding, which is better suited for extracting periodic components at different scales. Extensive experiments on several real-world datasets demonstrate the effectiveness of MTST in comparison to *state-of-the-art* forecasting techniques.

1 INTRODUCTION

Time-series forecasting has ubiquitous applications in various domains including but not limited to quantitative finance, weather prediction, and electricity management. Building on the success of transformers in diverse fields, transformer architectures were recently introduced in multivariate time-series forecasting, viewing each timestamp as a token (Zhou et al., 2021; Wu et al., 2021; Zhou et al., 2022). However, the effectiveness of time-series transformers (TSTs) is disputed: most TSTs are not sensitive to the temporal order of tokens and can even be outperformed by simple linear models (Zeng et al., 2023). Recently, the work of Zhang and Yan (2023) and Nie et al. (2023) revealed that timestamp-level tokenization prevents attention mechanisms from effectively capturing temporal patterns. In contrast, patch-level tokenization (where a *patch* is a window of timesteps) allows attention mechanisms to model temporal patterns within each patch and learn the relationships between patches.

Despite the promising performance demonstrated by patch-based TSTs, these methods fail to explicitly incorporate multi-scale analysis, which has proved effective in many time-series modeling domains. Therefore, in this work, we endow patch-based TSTs with the ability to learn multi-scale features with attention mechanisms via multi-resolution representations, and propose a novel architecture, **M**ulti-resolution **T**ime-**S**eries **T**ransformer (MTST) [1]. Unlike previous works that rely on subsampling, MTST constructs a multi-resolution representation by simply adjusting the patch-level tokenization of a time-series: a large number of small-sized patches leads to high-resolution feature maps; a small number of large-sized patches results in low-resolution feature maps. By constructing multiple sets of tokens with different patch-sizes, each MTST layer can model the temporal patterns of different frequencies simultaneously with multi-branch self-attentions. As shown in an example from the Electricity dataset (Figure 1), the role of the branch with larger-size patches is mostly to capture the lower-frequency and coarser temporal patterns; the branch with smaller-size patches contributes to modeling the higher-frequency and finer temporal patterns. By processing the signals with a multi-resolution multi-branch architecture, MTST can model complex temporal signals that contain multiple seasonalities. Furthermore, in order to overcome the weak sensitivity to the ordering of time-series that is exhibited by many TSTs, instead of employing learned/fixed absolute po-

Proceedings of the 27[th] International Conference on Artificial Intelligence and Statistics (AISTATS) 2024, Valencia, Spain. PMLR: Volume 238.

[†]Equal contribution, the work was partially done when YZ and LM were interns at Huawei Noah's Ark Lab.

[1]Our code is publicly available at https://github.com/networkslab/MTST.

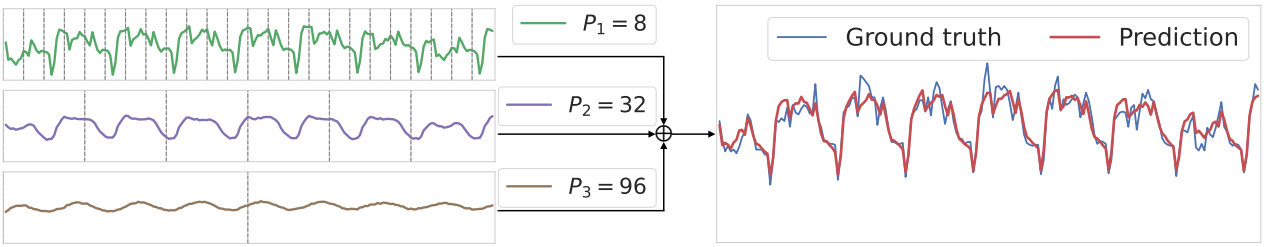


Figure 1: An example from *Electricity* dataset: MTST learns multi-scale temporal patterns in different branches, where P_i stands for the patch size in i-th branch.

sitional encoding, we utilize *relative* positional encoding, which is naturally aligned with capturing periodic temporal patterns.

To provide support for our motivation and hypotheses, we conduct extensive experiments on multiple time-series forecasting benchmarks. Our proposed MTST demonstrates *state-of-the-art* performance in comparison with diverse forecasting methods, reaching the lowest mean squared error on 28 out of 28 test settings. Comprehensive ablation studies and analysis experiments justify the effectiveness of proposed designs and the consequent advantage over previous techniques.

2 PROBLEM STATEMENT

We focus on long-term forecasting of multivariate time-series from historical data. Let $\mathbf{x}_t{=}[x_{t,1}, x_{t,2}, \cdots, x_{t,M}]^\intercal \in \mathbb{R}^{M\times 1}$ be a multivariate signal, where $x_{t,m}$ denotes m-th variate at time t, for $1{\leqslant}m{\leqslant}M$. The goal is to learn a model which can forecast the future T timestamps from the recent history of L timestamps. Here, L and T are termed the look-back window and prediction horizon, respectively. In other words, for any arbitrary time offset t_0, the model processes $\mathbf{x}_{t_0+1:t_0+L}$ as input and provides an estimate of $\mathbf{x}_{t_0+L+1:t_0+L+T}$ as its output. The estimate of $\mathbf{x}_t$ is denoted by $\widehat{\mathbf{x}}_t$. We drop the time offset t_0 in all subsequent discussions for simplicity.

A training dataset $\mathcal{D}_{trn}$ is assumed to be available for learning the model parameters. Usually, the time-series is spliced to construct the training set. The k-th example in the training set is denoted by $(\mathbf{x}^{(k)}_{1:L}, \mathbf{x}^{(k)}_{L+1:L+T})$. While $\mathbf{x}_{1:L+T}$ is known in the training set, $\mathbf{x}_{L+1:L+T}$ is unknown and must be estimated in the test set.

The forecasting performance of the model is assessed by computing the mean squared error (MSE) and the mean absolute error (MAE) between the prediction and the ground truth on the test set $\mathcal{D}_{test}$. These metrics are defined as:

$$\text{MSE} = \frac{1}{MT|\mathcal{D}_{test}|} \sum_{k\in\mathcal{D}_{test}} \sum_{t=L+1}^{L+T} ||\mathbf{x}_t^{(k)} - \widehat{\mathbf{x}}_t^{(k)}||_2^2 , \quad (1)$$

$$\text{MAE} = \frac{1}{MT|\mathcal{D}_{test}|} \sum_{k\in\mathcal{D}_{test}} \sum_{t=L+1}^{L+T} ||\mathbf{x}_t^{(k)} - \widehat{\mathbf{x}}_t^{(k)}||_1 . \quad (2)$$

3 METHODOLOGY

The proposed MTST consists of N layers, as shown in Fig. 2. Each MTST layer has multiple branches, with B_n branches at the n-th layer. The multi-branch architecture allows us to learn representations of the time-series at different scales simultaneously. Each branch contains a tokenizer (with different patch-size), which converts the input representation into patches. These patches are further processed as tokens by self-attention (SA) with relative positional encoding (RPE). The branch outputs are fused together to form a single embedding to be fed to the next MTST layer. We delve into a detailed discussion of each of these components in this section.

Following the protocol in the previous works (Zeng et al., 2023; Nie et al., 2023), MTST processes each $\mathbf{x}_{1:L,m}$ independently to generate the output $\widehat{\mathbf{x}}_{L+1:L+T,m}$, and subsequently combine them to form a multivariate forecast. This technique is termed *channel-independence* and we omit the variate index m in order to simplify the notation in the following sections. However, MTST is not limited by channel-independence and is readily extended to other patch-based TSTs that model dependencies among variates, such as the Crossformer (Zhang and Yan, 2023).

3.1 Branch specific tokenization

Temporal patches consisting of multiple successive timestamps are essential for learning effective representations for forecasting (Nie et al., 2023). Let $\mathbf{y}^{(n-1)}{\in}\mathbb{R}^{d_{n-1}\times 1}$ be the d_{n-1}-dimensional output representation of a univariate time-series at the $(n{-}1)$-th

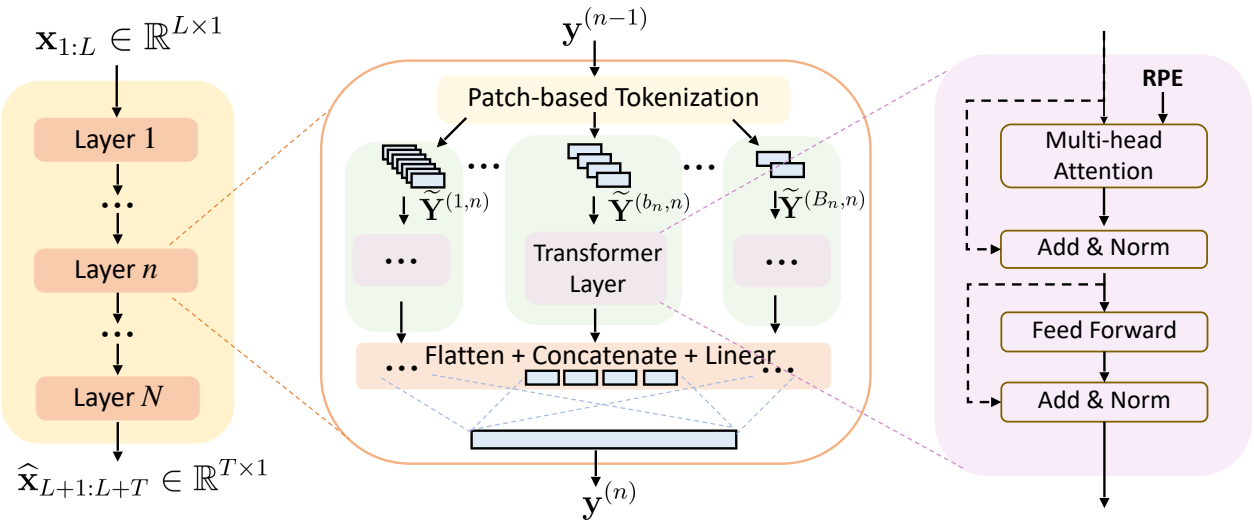
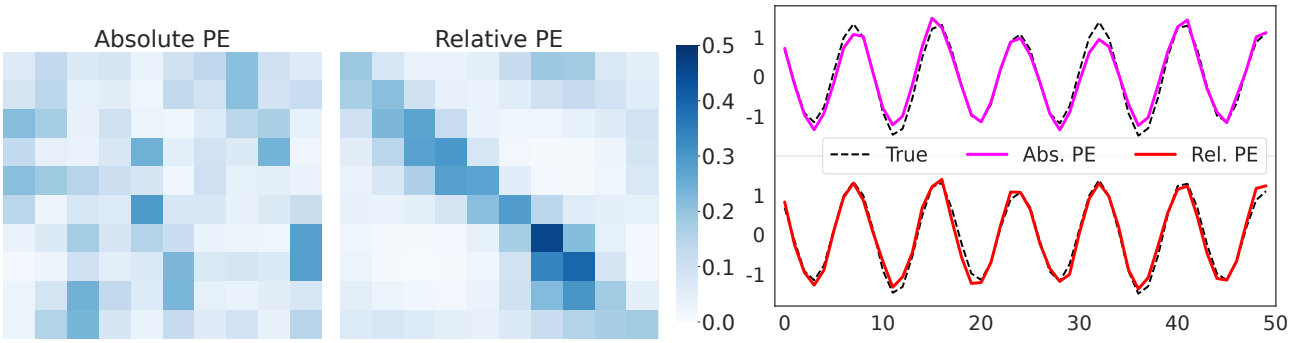


Figure 2: Multi-resolution Time-Series Transformer (MTST) Architecture.



(a) (b)

Figure 3: (a) Heatmaps of attention scores for absolute and relative positional encoding, (b) obtained forecasts using a transformer with absolute and relative positional encoding.

layer of MTST. We have $d_0{=}L$ and $\mathbf{y}^{(0)}{=}[x_1,\cdots,x_L]^\intercal$. Similarly, at the last layer, we have $d_N{=}T$ and $\mathbf{y}^{(N)}{=}[\widehat{x}_{L+1},\cdots,\widehat{x}_{L+T}]^\intercal$.

Denote by P_{b_n} the patch size and by S_{b_n} the stride (length of the non-overlapping region between two successive patches) at the b_n-th branch. Then the tokenizer $\mathcal{T}_{b_n} : \mathbb{R}^{d_n\times 1}{\to}\mathbb{R}^{J_{b_n}\times P_{b_n}}$ converts $\mathbf{y}^{(n-1)}$ into $J_{b_n}{=}\lceil (d_{n-1}{-}P_{b_n})/S_{b_n}\rceil + 1$ overlapping patches in a sliding-window fashion. Specifically, we compute $\widetilde{\mathbf{Y}}^{(b_n,n)}{=}[\widetilde{\mathbf{y}}_1^{(b_n,n)},\cdots,\widetilde{\mathbf{y}}_{J_{b_n}}^{(b_n,n)}]^\intercal{=}\mathcal{T}_{b_n}(\mathbf{y}^{(n-1)})$ as follows:

$$\widetilde{\mathbf{y}}_j^{(b_n,n)}{=}\mathbf{y}^{(n-1)}_{[(j-1)S_{b_n}+1:(j-1)S_{b_n}+P_{b_n}]}\,, \tag{3}$$

for $j=1,\cdots,J_{b_n}$. Here $\widetilde{\mathbf{Y}}^{(b_n,n)}$ denotes the output of the tokenizer and $\widetilde{\mathbf{y}}_j^{(b_n,n)^\intercal}$ is its j-th row (token). In order to make sure that the number of tokens is an integer, we pad the last dimension of $\mathbf{y}^{(n-1)}$ at the end of the last token $(J_{b_n}{-}1)S_{b_n}{+}P_{b_n}{-}d_{n-1}$ times, if $(d_{n-1}{-}P_{b_n})$ is not divisible by S_{b_n}.

A high value of J_{b_n} (equivalently, a low value of P_{b_n}) allows the b_n-th branch to focus on shorter patches, resulting in higher-resolution modeling of short-term temporal features. By contrast, longer patches facilitate learning of longer-term seasonalities and trends.

3.2 Self-attention

The self-attention in MTST is performed on the patch-level tokens in each branch independently. We omit the layer index n and branch index b_n temporarily to simplify notation. $\texttt{Attn} : \mathbb{R}^{J\times P}{\to}\mathbb{R}^{J\times D}$ is applied to the tokens to capture the relationships between them. Specifically, we compute $\widetilde{\mathbf{Z}}{=}[\widetilde{\mathbf{z}}_1,\cdots,\widetilde{\mathbf{z}}_J]^\intercal{=}\texttt{Attn}(\widetilde{\mathbf{Y}})$ as follows:

$$\alpha_{ij}{=}\texttt{Softmax}_j\Big(\frac{(\mathbf{W}_Q\widetilde{\mathbf{y}}_i)^T(\mathbf{W}_K\widetilde{\mathbf{y}}_j)}{\sqrt{D}} + \mathbf{w}_{pos}^T\mathbf{p}_{ij}\Big)\,, \tag{4}$$

$$\widetilde{\mathbf{z}}_i{=}\sum_j \alpha_{ij}\mathbf{W}_V\widetilde{\mathbf{y}}_j\,. \tag{5}$$

Here, $\mathbf{p}_{ij} \in \mathbb{R}^{D_{\text{pos}} \times 1}$ and $\alpha_{ij} \in [0,1]$ represent the relative positional encoding and the attention score between the i-th and j-th tokens, respectively; $\mathbf{W}_{\text{Q}}, \mathbf{W}_{\text{K}}, \mathbf{W}_{\text{V}} \in \mathbb{R}^{D \times P}$ and $\mathbf{w}_{\text{pos}} \in \mathbb{R}^{D_{\text{pos}} \times 1}$ are learnable weights. As in (Vaswani et al., 2017), we set $D=P$ and use a multi-head variant of this attention mechanism with different sets of weight matrices in each head. Moreover, similar to other transformer layers, the output of the self-attention operation is processed with a particular combination of residual connections, normalizations, and a feed-forward network. These details are shown schematically in Figure 2, but are deferred to the Appendix (Section A.1) to avoid notational clutter.

3.2.1 Relative positional encoding

Instead of absolute PE, employed in most previous TSTs, we introduce *relative* PE (RPE), which encodes the distances between each pair of tokens. Figure 3 demonstrates that the use of relative PE enables each token to identify similar tokens on a synthetic dataset with strong periodic patterns with considerably better accuracy. This results in improved forecasting.

Our relative PE is defined as $\mathbf{p}_{ij} := \texttt{sign}(i-j)\texttt{PE}(|i-j|)$, where $\texttt{sign}$ denotes the sign function and the function $\texttt{PE} : \mathbb{Z}^+ \to \mathbb{R}^{D_{pos}}$ is defined as

$$\begin{aligned} \texttt{PE}_{2t}(i) &:= \sin(i/10000^{2t/D_{pos}}) \,, \\ \texttt{PE}_{2t+1}(i) &:= \cos(i/10000^{2t/D_{pos}}) \,, \end{aligned} \tag{6}$$

for $t \in \{1, \ldots, D_{pos}/2\}$.

3.3 Fusing representations from all branches

At each layer, the token representations obtained from all branches are fused to form a single embedding. This allows sharing of information across scales, which helps in learning expressive representations of the time-series. The fusing is carried out by successive application of flattening, concatenation, and a linear transformation. First each $\widetilde{Z}^{(b_n,n)} \in \mathbb{R}^{J_{b_n} \times P_{b_n}}$ is flattened to a row vector of length $J_{b_n} \cdot P_{b_n}$. Next, the flattened vectors are concatenated to form $\mathbf{z}^{(n)} = [\texttt{Flatten}(\widetilde{Z}^{(1,n)}), \cdots, \texttt{Flatten}(\widetilde{Z}^{(B_n,n)})]^T \in \mathbb{R}^{\left(\sum_{b_n=1}^{B_n} J_{b_n} \cdot P_{b_n}\right) \times 1}$. Finally a linear layer, with weight $\mathbf{W}^{(n)} \in \mathbb{R}^{d_n \times \left(\sum_{b_n=1}^{B_n} J_{b_n} \cdot P_{b_n}\right)}$ and bias $\mathbf{b}^{(n)} \in \mathbb{R}^{d_n \times 1}$, is used to obtain $\mathbf{y}^{(n)}$, i.e.,

$$\mathbf{y}^{(n)} = \mathbf{W}^{(n)} \mathbf{z}^{(n)} + \mathbf{b}^{(n)} \,. \tag{7}$$

We refer to the combination of operations as $\texttt{Fuse}$. The n-th layer of the MTST can be summarized as:

$$\mathbf{y}^{(n)} = \texttt{Fuse}\Big(\texttt{Attn}\big(\mathcal{T}_1(\mathbf{y}^{(n-1)})\big), \ldots, \texttt{Attn}\big(\mathcal{T}_{B_n}(\mathbf{y}^{(n-1)})\big)\Big) \,. \tag{8}$$

4 RELATED WORK

4.1 Long-horizon Time-Series Forecasting

Multivariate time-series forecasting has been an active research area for decades. Until recently, statistical modeling based algorithms were the most effective, but many deep learning techniques have emerged and achieved impressive forecasting (Oreshkin et al., 2020; Salinas et al., 2020; Sen et al., 2019). However, convolutional and recurrent architectures fail to capture long-range dependency, leading to poor long-term forecasting. In order to address this, transformer-based models have been proposed. Earlier versions treated each timestamp as a token and performed forecasting in a sequence-to-sequence manner (Li et al., 2019; Zhou et al., 2021). Several time-series transformers (TSTs), such as Autoformer (Wu et al., 2021) and FEDformer (Zhou et al., 2022), incorporated inductive biases such as trends and seasonalities.

Zeng et al. (2023) demonstrated that timestamp-level tokenization prevents models from capturing temporal patterns. A simple linear model outperforms most timestamp-level TSTs. Crossformer (Zhang and Yan, 2023) and PatchTST (Nie et al., 2023) address this by using patches (windows of multiple timesteps) as tokens, inspired by the patch-based transformer for images in (Dosovitskiy et al., 2020).

Although previous patch-based methods achieve excellent forecasting, they cannot disentangle multi-scale features. In contrast, our proposed MTST is designed to perform multi-resolution decomposition to naturally extract multiple periodicities.

4.2 Multi-scale Feature Learning

Several recent time-series forecasting techniques have incorporated multi-scale analysis. NHiTS (Challu et al., 2022) introduces multi-rate signal sampling and hierarchical interpolation to model features at multiple granularities. MICN (Wang et al., 2023) incorporates convolution with different kernel sizes to learn multi-scale features. TimesNet (Wu et al., 2023) strives to capture the multi-periodicity in time-series by converting 1D sequences to a set of 2D tensors. Among transformer-based models, Pyraformer (Liu et al., 2021a) forms a multi-resolution representation via pyramidal attention. Scaleformer (Shabani et al., 2023) proposes a multi-resolution representation framework for existing timestamp-based TSTs via down/up-sampling. However, migrating existing multiscale techniques to token-based TSTs is not trivial: the subsampling/downsampling techniques in previous work result in sub-optimal representations for patches, because the methods are not cognizant of the chrono-

logical order at the timestamp level (Marin et al., 2023).

Although these methods have tried to incorporate multi-resolution analysis, the attempts have proved ineffective. Pyraformer and Scaleformer's forecasts are dramatically inferior to those of PatchTST, which is incapable of forming explicit multi-scale decompositions.

4.3 Positional Encoding

To enable transformers to sense token positions, learned or fixed sinusoidal absolute positional encoding (PE) is usually injected into tokens before the encoders (Vaswani et al., 2017). Huang et al. (2019) observe that relative PE (Shaw et al., 2018) can better model periodic patterns, which is crucial in time-series. In contrast to all previous TSTs that use absolute PE, MTST employs a flexible relative PE (Dai et al., 2019), whose design provides useful inductive bias for forecasting.

5 EXPERIMENTS

5.1 Benchmarking MTST

5.1.1 Datasets

We evaluate the performance of our proposed MTST on seven widely-used public benchmark datasets, including Weather, Traffic, Electricity and four ETT datasets (ETTh1, ETTh2, ETTm1, ETTm2). *Weather* is a collection of 2020 weather data from 21 meteorological indicators, including air temperature and humidity, provided by the Max-Planck Institute for Biogeochemistry[2]. *Traffic* is a dataset provided by Caltrans Performance Measurement System (PeMS), collecting hourly data of the road occupancy rates measured by different sensors on San Francisco Bay area freeways from California Department of Transportation[3]. *Electricity* contains hourly time-series of the electricity consumption of 321 customers from 2012 to 2014 (Trindade, 2015; Wu et al., 2021)[4]. *ETT* datasets are composed of a series of sensor measurements, including load and oil temperature, collected from electricity transformers between 2016 and 2018, provided by Zhou et al. (2021). Following the standard pipelines, the datasets are split into training, validation, and test sets with the ratio of 6:2:2 for four ETT datasets and 7:1:2 for the remaining datasets. Detailed statistics of the datasets are summarized in Table 1.

Datasets	ETTh1/2	ETTm1/2	Traffic	Electricity	Weather
Variates	7	7	862	321	21
Timesteps	17,420	69,680	17,544	26,304	52,696
Granularity	1 hour	15 min	1 hour	1 hour	10 min

Table 1: The statistics of datasets used in the experiments

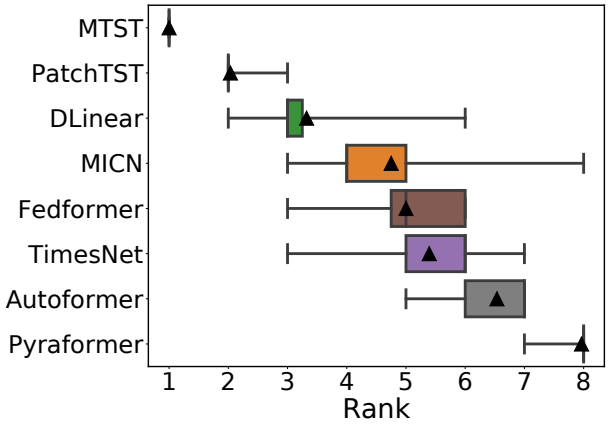


Figure 4: Boxplot for ranks of the algorithms (based on their MSE) across seven datasets and four prediction horizons. The medians and means of the ranks are shown by the vertical lines and the black triangles respectively; whiskers extend to the minimum and maximum ranks.

5.1.2 Baselines and Experimental Setup

We evaluate our proposed model in comparison with the *state-of-the-art* (SOTA) time-series transformer – PatchTST (Nie et al., 2023); three multiscale time-series models – MICN (Wang et al., 2023), TimesNet (Wu et al., 2023), and Pyraformer (Liu et al., 2021a); as well as a simple and competitive linear baseline – DLinear (Zeng et al., 2023). We also include two earlier timestamp-based time-series transformers for reference – Fedformer (Zhou et al., 2022) and Autoformer (Wu et al., 2021).

For fair comparison, we follow the experimental setup in Nie et al. (2023) so that the prediction horizon $T \in \{96, 192, 336, 720\}$ and the look-back window L=336. We report the baseline results from Nie et al. (2023)[5] except for TimesNet and MICN, for which we reproduce their results with L=336 based on the officially released code. Since Scaleformer (Shabani et al., 2023) has multiple variants, and the best-performing variant differs according to the dataset under study, we

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

[3]https://pems.dot.ca.gov

[4]Wu et al. (2021) selected 321 of 370 customers from the original dataset in Trindade (2015). This version is widely used in the follow-up works.

[5]The results for Fedformer, Autoformer and Pyraformer are the best on each dataset from multiple look-back windows $L \in \{24, 48, 96, 192, 336, 720\}$.

Table 2: Multivariate long-term forecasting results with $L = 336$ and $T \in \{96, 192, 336, 720\}$. **Bold** and underlined denote the best and second-best results respectively. * indicates statistically significant difference between the top-2 results.

Models		MTST		PatchTST		DLinear		MICN		TimesNet		Fedformer		Autoformer		Pyraformer	
Dataset	T	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Traffic	96	**0.356***	**0.244***	0.367	0.251	0.410	0.282	0.473	0.293	0.595	0.318	0.576	0.359	0.597	0.371	2.085	0.468
	192	**0.375***	**0.251***	0.385	0.259	0.423	0.287	0.483	0.298	0.615	0.326	0.610	0.380	0.607	0.382	0.867	0.467
	336	**0.386***	**0.256***	0.398	0.265	0.436	0.296	0.491	0.303	0.616	0.326	0.608	0.375	0.623	0.387	0.869	0.469
	720	**0.425***	**0.279***	0.434	0.287	0.466	0.315	0.559	0.327	0.655	0.353	0.621	0.375	0.639	0.395	0.881	0.473
Electricity	96	**0.127***	**0.222***	0.130	0.222	0.140	0.237	0.157	0.266	0.178	0.284	0.186	0.302	0.196	0.313	0.386	0.449
	192	**0.144***	**0.238***	0.148	0.240	0.153	0.249	0.175	0.287	0.187	0.289	0.197	0.311	0.211	0.324	0.386	0.443
	336	**0.162***	**0.256***	0.167	0.261	0.169	0.267	0.200	0.308	0.208	0.307	0.213	0.328	0.214	0.327	0.378	0.443
	720	**0.199***	**0.289***	0.202	0.291	0.203	0.301	0.228	0.338	0.245	0.321	0.233	0.344	0.236	0.342	0.376	0.445
Weather	96	**0.150***	**0.199***	0.152	0.199	0.176	0.237	0.178	0.249	0.163	0.219	0.238	0.314	0.249	0.329	0.896	0.556
	192	**0.194***	**0.240***	0.197	0.243	0.220	0.282	0.243	0.269	0.211	0.259	0.275	0.329	0.325	0.370	0.622	0.624
	336	**0.246***	**0.281***	0.249	0.283	0.265	0.319	0.278	0.338	0.286	0.311	0.339	0.377	0.351	0.391	0.739	0.753
	720	**0.319***	**0.333***	0.320	0.335	0.323	0.362	0.320	0.360	0.359	0.363	0.389	0.409	0.415	0.426	1.004	0.934
ETTh1	96	**0.358***	**0.390***	0.375	0.399	0.375	0.399	0.413	0.442	0.421	0.440	0.376	0.415	0.435	0.446	0.664	0.612
	192	**0.396***	**0.414***	0.414	0.421	0.405	0.416	0.451	0.462	0.511	0.498	0.423	0.446	0.456	0.457	0.790	0.681
	336	**0.391***	**0.420***	0.431	0.436	0.439	0.443	0.556	0.528	0.484	0.478	0.444	0.462	0.486	0.487	0.891	0.738
	720	**0.430***	**0.457***	0.449	0.466	0.472	0.490	0.658	0.607	0.554	0.527	0.469	0.492	0.515	0.517	0.963	0.782
ETTh2	96	**0.257***	**0.326***	0.274	0.336	0.289	0.353	0.303	0.364	0.366	0.417	0.332	0.374	0.332	0.368	0.645	0.597
	192	**0.309***	**0.361***	0.339	0.379	0.383	0.418	0.403	0.446	0.426	0.447	0.407	0.446	0.426	0.434	0.788	0.683
	336	**0.302***	**0.366***	0.331	0.380	0.448	0.465	0.603	0.550	0.406	0.435	0.400	0.447	0.477	0.479	0.907	0.747
	720	**0.372***	**0.416***	0.379	0.422	0.605	0.551	1.106	0.852	0.427	0.457	0.412	0.469	0.453	0.490	0.963	0.783
ETTm1	96	**0.286***	**0.338***	0.290	0.342	0.299	0.343	0.308	0.360	0.356	0.385	0.326	0.390	0.510	0.492	0.543	0.510
	192	**0.327***	0.366	0.332	0.369	0.335	**0.365**	0.343	0.384	0.452	0.428	0.365	0.415	0.514	0.495	0.557	0.537
	336	**0.362***	0.389	0.366	0.392	0.369	**0.386**	0.395	0.411	0.419	0.425	0.392	0.425	0.510	0.492	0.754	0.655
	720	**0.414***	**0.421**	0.420	0.424	0.425	**0.421**	0.427	0.434	0.452	0.451	0.446	0.458	0.527	0.493	0.908	0.724
ETTm2	96	**0.162***	**0.251***	0.165	0.255	0.167	0.260	0.169	0.268	0.188	0.276	0.180	0.271	0.205	0.293	0.435	0.507
	192	**0.220**	**0.291**	**0.220**	0.292	0.224	0.303	0.247	0.333	0.242	0.310	0.252	0.318	0.278	0.336	0.730	0.673
	336	**0.272***	**0.326***	0.278	0.329	0.281	0.342	0.290	0.351	0.300	0.346	0.324	0.364	0.343	0.379	1.201	0.845
	720	**0.358***	**0.379***	0.367	0.385	0.397	0.421	0.417	0.434	0.391	0.403	0.410	0.420	0.414	0.419	3.625	1.451
#Rank-1st (total=28)		28	26	1	0	0	3	0	0	0	0	0	0	0	0	0	0

do not include it in the comparison. We provide a comparison with Scaleformer in the Appendix (Table 8). Note that in all experiments, models are trained and evaluated for each prediction horizon independently.

5.1.3 Hyperparameters

In all our experiments, we use instance-normalization and denormalization (Kim et al., 2022) on the input and prediction, respectively. In each MTST layer, the transformer layer contains scaled dot-product attention and batch normalization (Ioffe and Szegedy, 2015), with the use of relative PE. The model is trained using the Adam optimizer (Kingma and Ba, 2015) to minimize the MSE loss over the training set. The detailed hyperparameter configuration of MTST associated with the model architecture and the training process for each dataset/horizon is provided in Section B.2 of the Appendix.

5.1.4 Experimental Results

Table 2 reports the experimental results for seven benchmarks. The performance is measured by MSE and MAE; the best and second-best results for each case (dataset, horizon, and metric) are highlighted in bold and underlined, respectively. We conduct the Wilcoxon signed rank test (Wilcoxon, 1945) with bootstrap sampling with significance level 5% on the top-2 results; we use * to denote a significant difference.

Our proposed model, MTST, achieves SOTA performances in all cases (7 datasets, 4 horizons, and 2 metrics). MTST outperforms, with statistical significance, the previous SOTA patch-based TST – PatchTST – on 27 out of 28 cases for the MSE metric.

We rank the algorithms in Table 2 based on their MSE and order them based on their average rank across seven datasets and four prediction horizons. Figure 4 shows the boxplot of rank. We observe that the proposed MTST achieves the best average rank and lowest variability across all settings.

As a multi-scale convolution-based approach, MICN demonstrates competitive performance for small prediction horizons T, but encounters more severe performance degradation for larger T. This matches the observations in previous works that convolution-based models struggle to capture long-range dependencies. This motivates the development of transformer-based models for long-term forecasting. Notably, DLinear, a

Table 3: Ablation study on multi-branch architecture.

Models		MTST (Base)		w/o Low-RES.		w/o High-RES.	
Dataset	T	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	96	**0.358**	**0.390**	0.373	0.402	0.372	0.400
	192	**0.396**	**0.414**	0.397	0.418	0.399	0.424
	336	**0.391**	**0.420**	0.397	0.428	0.399	0.424
	720	**0.430**	**0.457**	0.435	0.460	**0.430**	**0.457**
ETTh2	96	**0.257**	**0.326**	0.260	0.329	0.266	0.335
	192	**0.309**	**0.361**	0.311	0.364	0.317	0.370
	336	**0.302**	**0.366**	0.304	0.369	0.311	0.376
	720	**0.372**	**0.416**	0.373	0.417	0.380	0.422
ETTm1	96	0.286	**0.338**	0.290	0.341	**0.285**	**0.338**
	192	0.327	0.366	0.334	0.372	**0.322**	**0.364**
	336	**0.362**	0.389	0.367	0.389	0.365	**0.386**
	720	**0.414**	**0.421**	0.422	**0.421**	0.419	0.423
Weather	96	**0.150**	**0.199**	0.151	**0.199**	0.151	**0.199**
	192	**0.194**	**0.240**	0.196	**0.240**	0.196	0.243
	336	**0.246**	0.281	**0.246**	**0.280**	0.247	**0.280**
	720	**0.319**	**0.333**	0.322	0.335	0.324	0.335
Traffic	96	**0.356**	**0.244**	0.357	**0.244**	0.362	0.249
	192	**0.375**	**0.251**	0.379	0.253	0.378	0.253
	336	**0.386**	0.258	0.388	**0.257**	0.391	0.261
	720	**0.425**	0.281	**0.425**	**0.279**	0.427	0.283
#Rank-1 (total=20)		18	15	2	7	3	6

Table 4: Ablation study on positional encoding.

MTST w/		RPE (ours)		SinAPE		LenaredAPE	
Dataset	T	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	96	**0.358**	**0.390**	0.362	0.392	0.363	0.394
	192	**0.396**	**0.414**	0.400	0.417	0.400	0.420
	336	0.391	0.420	**0.389**	**0.419**	0.390	**0.419**
	720	**0.430**	**0.457**	0.440	0.464	0.435	0.462
ETTh2	96	**0.257**	**0.326**	0.258	0.328	0.258	0.328
	192	**0.309**	**0.361**	**0.309**	0.363	0.311	0.365
	336	**0.302**	**0.366**	0.305	0.371	0.306	0.373
	720	**0.372**	**0.416**	0.377	0.422	0.375	0.420
ETTm1	96	**0.286**	**0.338**	0.297	0.346	0.290	0.342
	192	**0.327**	**0.366**	0.332	0.370	0.330	0.370
	336	**0.362**	0.389	0.369	0.390	0.366	**0.388**
	720	**0.414**	**0.421**	0.419	0.423	0.420	0.425
Weather	96	**0.150**	0.199	**0.150**	**0.197**	**0.150**	0.198
	192	**0.194**	**0.240**	**0.194**	**0.240**	**0.194**	**0.240**
	336	**0.246**	**0.281**	0.247	**0.281**	0.247	0.282
	720	**0.319**	**0.333**	0.326	0.339	0.320	**0.333**
Traffic	96	**0.356**	**0.244**	0.361	0.247	0.360	0.246
	192	**0.375**	**0.251**	0.381	0.257	0.381	0.256
	336	**0.386**	**0.258**	0.393	0.263	0.397	0.266
	720	**0.425**	**0.281**	0.432	0.288	0.431	0.286
#Rank-1 (total=20)		19	17	4	4	2	4

simple linear model, outperforms earlier timestamp-based TSTs (Fedformer, Autoformer, Pyraformer) with a noticeable performance gap, indicating the ineffectiveness of the timestamp-based TSTs and the need to develop patch-based TSTs.

5.2 Ablation Study and Analysis Experiment

5.2.1 Ablation: Multi-Resolution

We conduct an ablation experiment to study the usefulness of the multi-resolution representation on 5 datasets: ETTh1, ETTh2, ETTm1, Weather, and Traffic, under the same setup of the main experiment. To verify the importance of each resolution, we compare MTST against two variants: *w/o Low-RES.* stands for the variant without the lowest-resolution representations (i.e., the branch with the largest-sized patch), which mainly contributes to low-frequency temporal patterns; *w/o High-RES.* denotes the variant removing the highest-resolution representations (i.e., the branch with the smallest-sized patches), which mainly aims to capture high-frequency temporal patterns.

From Table 3, we can see that removing either resolution generally leads to performance degradation. However, removing the highest-resolution branch results in slightly better performance in a few cases for small datasets (e.g., ETTm1). This is potentially due to overfitting, which usually focuses on high-frequency components.

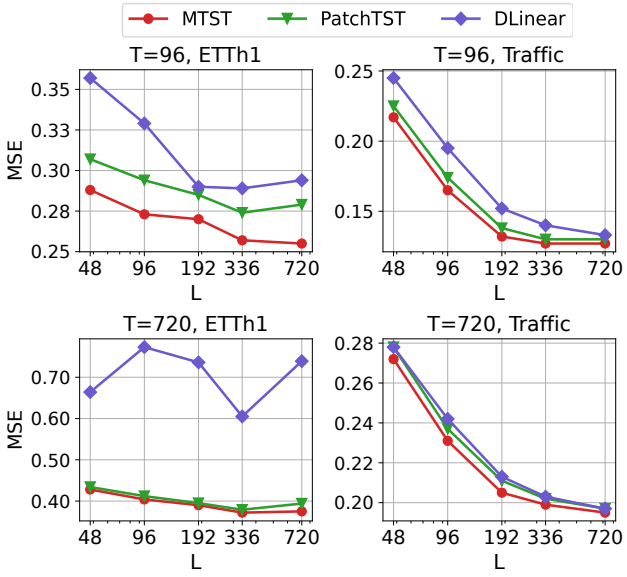
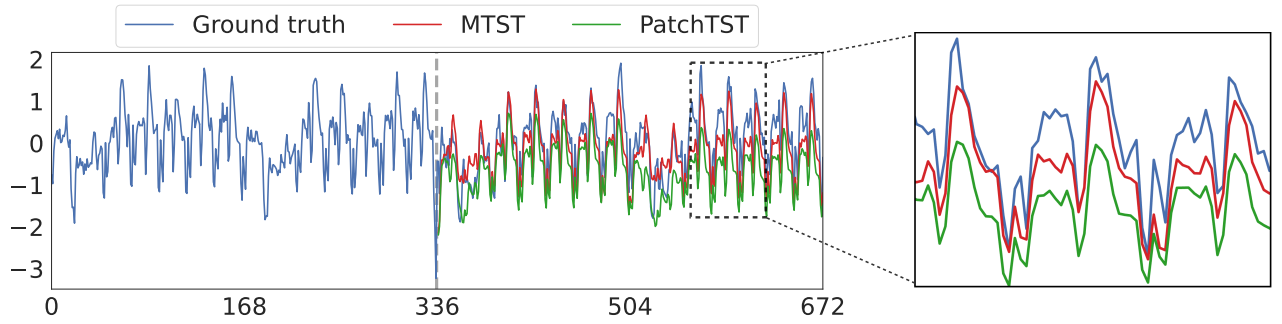


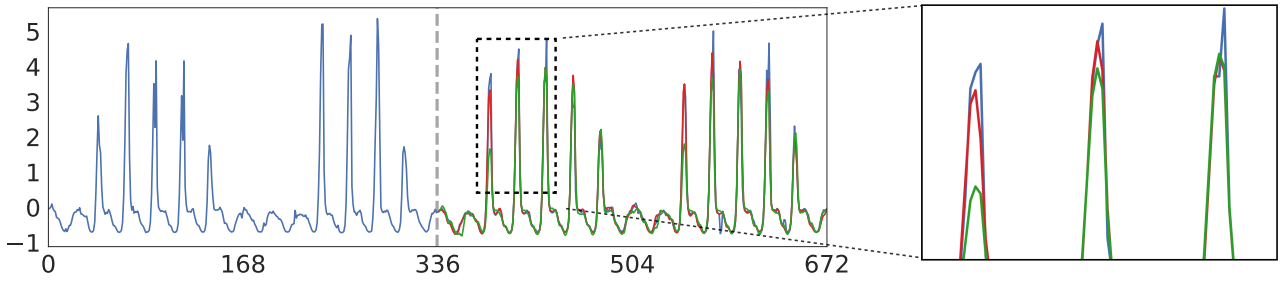
Figure 5: MSE at different look-back windows on ETTh1 and Traffic datasets, $L \in \{48, 96, 192, 336, 720\}$ and T=96 and 720.

5.2.2 Ablation: Positional Encoding

Positional encoding is known to be an important component in transformer design. However, previous TSTs employ absolute positional encoding by default. Therefore, we conduct an ablation experiment to study whether the choice of positional encoding for time-series forecasting is important. Using our design of MTST with relative positional encoding (RPE)



(a) 318-th variate on 4646-th example on Electricity.



(b) 4-th variate on 807-th example on Traffic.

Figure 6: Visualization of forecasts for PatchTST and MTST (L=336 and T=336).

as the base model, we compare with the widely used fixed absolute positional encoding (SinAPE), introduced by Vaswani et al. (2017), as well as the learned absolute positional encoding (LearnedAPE) used in PatchTST (Nie et al., 2023). The APE is injected into the token representations after patch-based tokenization in each branch of every layer.

From the results of the PE ablation study (shown in Table 4), the incorporation of the relative PE is beneficial to the outcome of forecasting for most cases. The observation also matches the finding from previous works in other domains that RPE performs better than APE when periodicities are important (Huang et al., 2019).

5.2.3 Analysis: Look-back Window

In the main experiment (Table 2), MTST reaches state-of-the-art performance with $L = 336$. We now conduct a more in-depth study into MTST's behavior for different look-back windows. For the ETTh1 and Traffic datasets, we train and evaluate our model with $L \in \{48, 96, 192, 336, 720\}$ independently, and compare with the second and third best models (PatchTST and DLinear).

We visualize the results with $T = 96$ and 720 in Figure 5. Although all algorithms show improved performance with increasing the look-back window in most cases, the advantages of MTST over PatchTST and DLinear are retained over nearly all look-back windows for $T = 96$ and 720.

5.2.4 Analysis: Qualitative Comparisons

To complement the quantitative results, Figure 6 provides visualizations of two randomly chosen test examples from the Electricity and Traffic datasets. We plot the ground truth in the look-back window $L = 336$ and prediction horizon $T = 336$, as well as the predictions from MTST. We include PatchTST as a representative single-scale patch-based TST. More visualizations are included in the Appendix (Section E). From Figure 6, we observe that although both PatchTST and MTST can capture the overall temporal patterns, MTST's forecasts are superior to that of PatchTST. Specifically, in Figure 6a, MTST is less impacted by the abnormally deep trough near the end of the look-back window, potentially owing to the low-resolution branch, and consequently performs better on predicting the overall trend. In Figure 6b, PatchTST is more impacted by the low amplitude of the first high-peak in the second weekly cycle in the look-back window, while MTST can forecast the amplitude of the first high peak better, potentially due to the finer prediction of the high-resolution branch.

6 CONCLUSION

In this paper, we incorporate multi-scale analysis into patch-based time-series transformers, and accordingly propose a novel framework, termed Multi-resolution Time-Series Transformer (MTST), for forecasting. Contrary to up/down-sampling techniques used in previous work, we use multiple patch-based tokenizations with different patch-sizes together with the multi-branch transformer architecture in each MTST layer, which enables flexibly modeling of temporal patterns of different scales. Additionally, we propose to incorporate relative positional encoding in our design, which is well-matched to the seasonality characteristics present in many real-world datasets. Extensive experimental results on seven multivariate time-series datasets demonstrate that the proposed MTST outperforms the previous *state-of-the-art* approaches. The example visualization of the prediction from each branch on real-world datasets also provides validation for our hypothesis on multi-scale learning. Furthermore, we have conducted ablation studies on multiple resolutions and positional encoding, which justify our design choices.

7 ACKNOWLEDGEMENT

We acknowledge the support of the Natural Sciences and Engineering Research Council of Canada (NSERC), [funding reference number 260250].
Cette recherche a été financée par le Conseil de recherches en sciences naturelles et en génie du Canada (CRSNG), [numéro de référence 260250].

References

Bai, S., Kolter, J. Z., and Koltun, V. (2018). An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv e-prints: arXiv 1803.01271*.

Bolya, D., Fu, C.-Y., Dai, X., Zhang, P., Feichtenhofer, C., and Hoffman, J. (2023). Token merging: Your ViT but faster. In *Proc. Int. Conf. Learn. Representations*.

Challu, C., Olivares, K. G., Oreshkin, B. N., Garza, F., Mergenthaler-Canseco, M., and Dubrawski, A. (2022). N-HiTS: Neural hierarchical interpolation for time series forecasting. In *Proc. AAAI Conf. Artif. Intell.*

Chen, L.-C., Papandreou, G., Kokkinos, I., Murphy, K., and Yuille, A. L. (2018). DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs. *IEEE Trans. Pattern Anal. Mach. Intell.*, 40(4):834–848.

Chen, Z., Ma, Q., and Lin, Z. (2021). Time-aware multi-scale RNNs for time series modeling. In *Proc. Int. Joint Conf. Artif. Intell.*

Cui, Z., Chen, W., and Chen, Y. (2016). Multi-scale convolutional neural networks for time series classification. *arXiv preprint: arXiv 1603.06995*.

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 *Proc. Annu. Meeting Assoc. Comput. Linguist.*

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. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. In *Proc. Int. Conf. Learn. Representations*.

Holt, C. C. (2004). Forecasting seasonals and trends by exponentially weighted moving averages. *Int. J. Forecasting*, 20(1):5–10.

Huang, C.-Z. A., Vaswani, A., Uszkoreit, J., Simon, I., Hawthorne, C., Shazeer, N., Dai, A. M., Hoffman, M. D., Dinculescu, M., and Eck, D. (2019). Music transformer: Generating music with long-term structure. In *Proc. Int. Conf. Learn. Representations*.

Hyndman, R., Koehler, A. B., Ord, J. K., and Snyder, R. D. (2008). *Forecasting with exponential smoothing: the state space approach*. Springer Science & Business Media.

Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by ueducing internal covariate shift. In *Proc. Int. Conf. Mach. Learn.*

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 *Proc. Int. Conf. Learn. Representations*.

Kingma, D. P. and Ba, J. (2015). Adam: A method for stochastic optimization. In *Proc. Int. Conf. Learn. Representations*.

Lai, G., Chang, W.-C., Yang, Y., and Liu, H. (2018). Modeling long-and short-term temporal patterns with deep neural networks. In *Int. ACM SIGIR Conf. Research Development Inf. Retr.*

Li, S., Jin, X., Xuan, Y., Zhou, X., Chen, W., Wang, Y.-X., and Yan, X. (2019). Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting. In *Adv. Neural Inf. Process. Syst.*

Lim, B., Arık, S. Ö., Loeff, N., and Pfister, T. (2021). Temporal fusion transformers for inter-

pretable multi-horizon time series forecasting. *Int. J. Forecast.*, 37(4):1748–1764.

Lin, T.-Y., Dollar, P., Girshick, R., He, K., Hariharan, B., and Belongie, S. (2017). Feature pyramid networks for object detection. In *Proc. IEEE Conf. Comput. Vis. and Pattern Recog.*

Liu, S., Yu, H., Liao, C., Li, J., Lin, W., Liu, A. X., and Dustdar, S. (2021a). Pyraformer: Low-complexity pyramidal attention for long-range time series modeling and forecasting. In *Proc. Int. Conf. Learn. Representations*.

Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., and Guo, B. (2021b). Swin transformer: Hierarchical vision transformer using shifted windows. In *Proc. IEEE/CVF Int. Conf. Comput. Vis.*

Makridakis, S. and Hibon, M. (1997). ARMA models and the Box–Jenkins methodology. *J. Forecast.*, 16(3):147–163.

Marin, D., Chang, J.-H. R., Ranjan, A., Prabhu, A., Rastegari, M., and Tuzel, O. (2023). Token pooling in vision transformers for image classification. In *Proc. IEEE/CVF Winter Conf. Appl. Comput. Vis.*

Nie, Y., Nguyen, N. H., Sinthong, P., and Kalagnanam, J. (2023). A time series is worth 64 words: Long-term forecasting with transformers. In *Proc. Int. Conf. Learn. Representations*.

Oreshkin, B. N., Carpov, D., Chapados, N., and Bengio, Y. (2020). N-BEATS: Neural basis expansion analysis for interpretable time series forecasting. In *Proc. Int. Conf. Learn. Representations*.

Ronneberger, O., Fischer, P., and Brox, T. (2015). U-Net: Convolutional networks for biomedical image segmentation. In *Proc. Med. Image. Comput. Comput. Assist. Interv.*

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

Sen, R., Yu, H.-F., and Dhillon, I. S. (2019). Think globally, act locally: A deep neural network approach to high-dimensional time series forecasting. In *Adv. Neural Inf. Process. Syst.*

Shabani, M. A., Abdi, A. H., Meng, L., and Sylvain, T. (2023). Scaleformer: Iterative multi-scale refining transformers for time series forecasting. In *Proc. Int. Conf. Learn. Representations*.

Shaw, P., Uszkoreit, J., and Vaswani, A. (2018). Self-attention with relative position representations. In *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics: Human Language Technologies*.

Smyl, S. (2020). A hybrid method of exponential smoothing and recurrent neural networks for time series forecasting. *Int. J. Forecast.*, 36(1):75–85.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*

Trindade, A. (2015). ElectricityLoadDiagrams20112014. UCI Machine Learning Repository.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Aidan N Gomez, Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. In *Adv. Neural Inf. Process. Syst.*

Vijay, E., Jati, A., Nguyen, N., Sinthong, G., and Kalagnanam, J. (2023). TSMixer: Lightweight MLP-mixer model for multivariate time series forecasting. In *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*.

Wang, H., Peng, J., Huang, F., Wang, J., Chen, J., and Xiao, Y. (2023). MICN: Multi-scale local and global context modeling for long-term series forecasting. In *Proc. Int. Conf. Learn. Representations*.

Wang, W., Xie, E., Li, X., Fan, D.-P., Song, K., Liang, D., Lu, T., Luo, P., and Shao, L. (2021). Pyramid Vision Transformer: A Versatile Backbone for Dense Prediction without Convolutions. In *Proc. IEEE Int. Conf. Comput. Vis.*

Wen, R., Torkkola, K., Narayanaswamy, B., and Madeka, D. (2017). A multi-horizon quantile recurrent forecaster. *arXiv preprint: arXiv 1711.11053*.

Wilcoxon, F. (1945). Individual comparisons by ranking methods. *Biometrics Bulletin*, 1(6):80–83.

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 *Proc. Int. Conf. Learn. Representations*.

Wu, H., Xu, J., Wang, J., and Long, M. (2021). Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. In *Adv. Neural Inf. Process. Syst.*

Yu, F. and Koltun, V. (2016). Multi-scale context aggregation by dilated convolutions. In *Proc. Int. Conf. Learn. Representations*.

Zeng, A., Chen, M., Zhang, L., and Xu, Q. (2023). Are transformers effective for time series forecasting? In *Proc. AAAI Conf. Artif. Intell.*

Zhang, Y. and Yan, J. (2023). Crossformer: Transformer utilizing cross-dimension dependency for multivariate time series forecasting. In *Proc. Int. Conf. Learn. 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 *Proc. AAAI Conf. Artif. Intell.*

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 *Proc. Int. Conf. Mach. Learn.*

SUPPLEMENTARY MATERIAL

A IMPLEMENTATION DETAILS

A.1 Detailed Architecture of Transformer Layers in MTST

As written in Section 3.2 of the main paper, here we describe the full architectural details of the transformer layer. The output of the self-attention operation $\widetilde{\mathbf{Z}}$=$\texttt{Attn}(\widetilde{\mathbf{Y}})$ is further processed with a particular combination of residual connections, batch normalizations (Ioffe and Szegedy, 2015), and a two-layer feed-forward network $\texttt{FFN}_\theta : \mathbb{R}^D \to \mathbb{R}^D$ as follows:

$$\widetilde{\mathbf{Z}}' = \texttt{BN}\big(\widetilde{\mathbf{Y}} + \texttt{Attn}(\widetilde{\mathbf{Y}})\big)\,, \tag{9}$$

$$\widetilde{\mathbf{Z}}'' = \texttt{BN}\big(\widetilde{\mathbf{Z}}' + \texttt{FFN}_\theta(\widetilde{\mathbf{Z}}')\big)\,. \tag{10}$$

Here, $\texttt{BN}(\cdot)$ stands for batch-normalization, and θ denotes the learnable weights and biases inside the $\texttt{FFN}$. Subsequently, $\widetilde{\mathbf{Z}}''$-s from different branches are fused together as described in Section 3.3.

A.2 Computational Complexity

As a transformer-based model, the asymptotic computational complexity of the operations in the n-th layer of MTST is $O(J_{b'_n}^2)$, where $J_{b'_n}$=$\lceil (d_{n-1}-P_{b'_n})/S_{b'_n} \rceil + 1$ is the number of tokens in the branch with the finest resolution. In other words, MTST has the same asymptotic computational complexity as PatchTST when $J_{b'_n}$ equals the number of tokens in PatchTST. As $J_{b'_n} << L$, the computational requirement of MTST scales favorably compared to most timestamp-based TSTs with $O(L^2)$ complexity.

From the hyperparameter search, we empirically observe that MTST reaches its optimal performance with shallower architectures compared to PatchTST, as shown in Table 6. Therefore, despite having multiple branches, the training time of MTST is similar to that of PatchTST. Examples from 2 datasets with a single NVIDIA V100 GPU and 72 threads of Intel(R) Xeon(R) Gold 6140 CPU @ 2.30GHz are shown in Table 5. All our experiments were conducted with these hardware devices.

Table 5: Training time (L=336, T=96).

Dataset	Traffic		ETTm2	
Model	MTST	PatchTST	MTST	PatchTST
Time(s) / Epoch	513.916	412.918	69.871	62.876
# Epoch	68	77	28	28
Total time(h)	9.707	8.832	0.543	0.489

B EXPERIMENTAL DETAILS

B.1 Baseline settings

The default look-back windows for different baseline models, as identified in the original papers and the code, can differ from each other. For Transformer-based models, the default look-back window is $L = 96$. TimesNet and MICN follow the same setting for a fair comparison.

However, this can possibly lead to a poorer performance than if the algorithms are provided with a longer horizon. A short 96 time-stamp look-back window is often inadequate for forecasting over longer horizons of more than 300 timestamps. For example, MICN shows that the overall prediction performance gradually improves as the size of the look-back window increases. Therefore, we re-run TimesNet and MICN with a look back window $L = 336$, which is the default experimental setting used by PatchTST and DLinear. We also conduct the experiments for MTST under the same look-back window for fair comparison.

In MICN, the principal hyperparameters are *stride=kernel*. We use the same setting when increasing the L from 96 to 336, selecting the stride and kernel from $\{12, 16\}$. TimesNet selects the top-k amplitude values and obtains the most significant frequencies $\{f_1, ..., f_k\}$ with amplitudes $\{A_{f_1}, ..., A_{f_k}\}$. Based on the calculation, the input is derived using k different periods. We follow the setting for L=96 with k=5, indicating 5 different scales.

B.2 Hyperparameters

For fair comparison, most hyperparameter values are borrowed from the experimental study reported by Nie et al. (2023). At each branch in each layer, the transformer layer consists of an attention mechanism with 16 heads, and a 2-layer feed-forward network with a hidden dimension of 256 and an output dimension of 128. We only perform grid search for the number of layers, $N \in \{1, 2, 3\}$, the number of branches, $B \in \{2, 3, 4\}$, and the patch-sizes in different branches $P_b \in \{4, 8, 16, 24, 36, 48, 64, 96\}$. The stride length is set to a default value of $S = P/2$. The detailed hyperparameter configurations of MTST for all datasets are shown in Table 6. The model parameters are learned by minimizing the MSE of the forecasts on the training set using the Adam optimizer (Kingma and Ba, 2015).

Table 6: Hyperparameters of MTST

Dataset		Traffic	ELC	Weather	ETTh1	ETTh2	ETTm1	ETTm2
Layer number N		1	1	2	2	1	2	2
Branch Number B		3	3	2	2	2	2	2
Patch length P	T = {96, 192}	[8,16,48]	[8,16,48]	[24,96]	[8,16]	[16,96]	[16,96]	[16,96]
	T = {336, 720}	[8,32,96]	[8,32,96]	[16,96]				
Stride length S	T = {96, 192}	[4,8,24]	[4,8,24]	[12,48]	[4,8]	[8,48]	[8,48]	[8,48]
	T = {336, 720}	[4,16,48]	[4,16,48]	[8,48]				
Feed forward Dropout		0.2	0.2	0.2	0.3	0.3	0.2	0.2
Fusing layer Dropout		0	0	0	0.1	0.3	0	0
Batch size		10	32	128	256	256	128	128
Initial leraning rate		$5e^{-4}$	$5e^{-4}$	$5e^{-4}$	$1e^{-4}$	$1e^{-4}$	$1e^{-4}$	$1e^{-4}$

C ADDITIONAL EXPERIMENTAL RESULTS

C.1 Forecasting Results with Standard Deviations

To quantify the variability of our experimental results in Table 2, we repeat each experiment three times with different random seeds (2021, 2022, and 2023), resulting in different initializations of the model parameters in each run. The results shown in Table 2 of the main paper is obtained from the trained model, which was initialized by setting the random seed to 2021. The mean and standard derivation of the MSE of MAE across multiple trials are reported in Table 7. The variances are small, which indicates that using a different random seed has minimal impact on the forecast.

C.2 Results for a Shorter Look-back Window (L=96)

For a fair comparison, we also conduct the experiments under the L=96 setting that is the default for MICN, TimesNet, and other transformer-based models, except PatchTST. For the shorter look-back window, we slightly modify the patch length for each branch since we need $P<L$ to obtain the tokens. By default, MTST has 2 layers and 2 branches across all datasets and horizons. At each layer, the patch length is set to P=[12, 16]. Other hyper-parameters follows the $L = 336$ setting in Table 2.

As shown in Table 8, MTST secures Rank 1 in 19 cases out of 28 cases in terms of MSE and 25 cases in terms of MAE. As a strong contender, PatchTST claims top performance on ETTm1 dataset. This matches our observations in the ablation study of the usefulness of multi-resolution, which illustrates that the one branch patch-based TST could have slightly better performance for small datasets because of the overfitting on high-resolution components. MTST's consistent success across various metrics and look-back windows highlights its adaptability and reliability in addressing time-series forecasting challenges, making it a compelling choice for real-world applications.

Table 7: Multivariate forecasting results with different random seeds in MTST and PatchTST at L=336.

Model		MTST		PatchTST	
Dataset	T	MSE	MAE	MSE	MAE
Traffic	96	0.355 ± 0.0005	0.243 ± 0.0009	0.367 ± 0.0006	0.250 ± 0.0007
	192	0.375 ± 0.0008	0.252 ± 0.0005	0.386 ± 0.0004	0.259 ± 0.0004
	336	0.387 ± 0.0005	0.257 ± 0.0009	0.399 ± 0.0010	0.267 ± 0.0016
	720	0.423 ± 0.0012	0.281 ± 0.0008	0.438 ± 0.0097	0.291 ± 0.0104
Electricity	96	0.127 ± 0.0003	0.221 ± 0.0006	0.130 ± 0.0006	0.223 ± 0.0006
	192	0.145 ± 0.0000	0.238 ± 0.0003	0.148 ± 0.0002	0.240 ± 0.0002
	336	0.162 ± 0.0003	0.257 ± 0.0003	0.166 ± 0.0006	0.260 ± 0.0006
	720	0.198 ± 0.0007	0.289 ± 0.0002	0.202 ± 0.0006	0.292 ± 0.0006
Weather	96	0.151 ± 0.0008	0.199 ± 0.0016	0.153 ± 0.0024	0.200 ± 0.0023
	192	0.194 ± 0.0005	0.240 ± 0.0005	0.198 ± 0.0015	0.243 ± 0.0010
	336	0.247 ± 0.0008	0.282 ± 0.0005	0.249 ± 0.0012	0.284 ± 0.0014
	720	0.321 ± 0.0012	0.335 ± 0.0009	0.319 ± 0.0002	0.335 ± 0.0003
ETTh1	96	0.360 ± 0.0017	0.391 ± 0.0012	0.375 ± 0.0008	0.400 ± 0.0004
	192	0.397 ± 0.0022	0.415 ± 0.0009	0.413 ± 0.0012	0.421 ± 0.0006
	336	0.391 ± 0.0016	0.419 ± 0.0009	0.428 ± 0.0033	0.433 ± 0.0028
	720	0.433 ± 0.0021	0.460 ± 0.0021	0.446 ± 0.0035	0.464 ± 0.0027
ETTh2	96	0.257 ± 0.0000	0.325 ± 0.0000	0.275 ± 0.0005	0.336 ± 0.0006
	192	0.310 ± 0.0005	0.361 ± 0.0009	0.339 ± 0.0010	0.379 ± 0.0014
	336	0.303 ± 0.0008	0.367 ± 0.0008	0.329 ± 0.0010	0.382 ± 0.0027
	720	0.372 ± 0.0005	0.417 ± 0.0008	0.378 ± 0.0010	0.421 ± 0.0009
ETTm1	96	0.287 ± 0.0009	0.340 ± 0.0005	0.289 ± 0.0009	0.342 ± 0.0007
	192	0.331 ± 0.0025	0.368 ± 0.0021	0332 ± 0.0008	0.370 ± 0.0007
	336	0.364 ± 0.0014	0.391 ± 0.0008	0.366 ± 0.0022	0.391 ± 0.0012
	720	0.416 ± 0.0017	0.422 ± 0.0008	0.420 ± 0.0056	0.424 ± 0.0033
ETTm2	96	0.162 ± 0.0000	0.251 ± 0.0005	0.164 ± 0.0011	0.254 ± 0.0010
	192	0.220 ± 0.0005	0.291 ± 0.0005	0.222 ± 0.0008	0.294 ± 0.0014
	336	0.272 ± 0.0005	0.326 ± 0.0005	0.278 ± 0.0020	0.330 ± 0.0010
	720	0.358 ± 0.0000	0.380 ± 0.0005	0.365 ± 0.0024	0.383 ± 0.0010

Table 8: Multivariate long-term forecasting results with MTST. Look back window L=96 and prediction lengths $T \in \{96, 192, 336, 720\}$. The best results are in bold and the second best are nderlined. The baseline results are from the original papers except for DLinear, which is reproduced for L=96.

Models		MTST		PatchTST		Dlinear		MICN		TimesNet		Crossformer		Scaleformer		Fedformer		Autoformer		Pyformer	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Traffic	96	**0.422**	**0.271**	0.477	0.305	0.650	0.397	0.519	0.309	0.593	0.321	-	-	0.564	0.351	0.587	0.366	0.613	0.388	2.085	0.468
	192	**0.437**	**0.281**	0.471	0.299	0.600	0.372	0.537	0.315	0.617	0.336	-	-	0.570	0.349	0.604	0.373	0.616	0.382	0.867	0.467
	336	**0.451**	**0.285**	0.485	0.305	0.606	0.374	0.534	0.313	0.629	0.336	0.530	0.300	0.576	0.349	0.621	0.383	0.622	0.337	0.869	0.469
	720	**0.490**	**0.309**	0.518	0.325	0.646	0.395	0.577	0.325	0.640	0.350	0.573	0.313	0.602	0.360	0.626	0.382	0.660	0.408	0.881	0.473
Electricity	96	**0.160**	**0.248**	0.174	0.259	0.194	0.277	0.164	0.269	0.168	0.272	-	-	0.182	0.297	0.193	0.308	0.201	0.317	0.386	0.449
	192	**0.171**	**0.263**	0.178	0.265	0.193	0.280	0.177	0.285	0.184	0.289	-	-	0.188	0.300	0.201	0.315	0.222	0.334	0.386	0.443
	336	**0.188**	**0.281**	0.196	0.282	0.207	0.296	0.193	0.304	0.198	0.300	0.404	0.423	0.210	0.324	0.214	0.329	0.231	0.338	0.378	0.443
	720	0.230	**0.315**	0.237	0.316	0.242	0.328	**0.212**	0.321	0.220	0.320	0.433	0.438	0.232	0.339	0.246	0.355	0.254	0.361	0.376	0.445
Weather	96	0.175	**0.216**	0.178	0.219	0.197	0.255	**0.161**	0.229	0.172	0.220	-	-	0.163	0.226	0.217	0.296	0.266	0.336	0.896	0.556
	192	**0.219**	**0.255**	0.224	0.259	0.239	0.297	0.220	0.281	**0.219**	0.261	-	-	0.221	0.290	0.276	0.336	0.307	0.367	0.622	0.624
	336	**0.276**	**0.296**	0.278	0.298	0.284	0.332	0.278	0.331	0.280	0.306	0.495	0.515	0.282	0.340	0.339	0.380	0.359	0.395	0.739	0.753
	720	0.351	**0.346**	0.350	**0.346**	0.348	0.385	**0.311**	0.356	0.365	0.359	0.526	0.542	0.369	0.396	0.403	0.428	0.419	0.428	1.004	0.934
ETTh1	96	**0.376**	**0.393**	0.393	0.408	0.383	0.396	0.421	0.431	0.384	0.402	-	-	-	-	**0.376**	0.419	0.449	0.459	0.664	0.612
	192	0.429	**0.422**	0.445	0.434	0.433	0.426	0.474	0.487	0.436	0.429	-	-	-	-	**0.420**	0.448	0.500	0.482	0.790	0.681
	336	0.444	**0.436**	0.484	0.451	0.491	0.467	0.569	0.551	0.491	0.469	**0.440**	0.461	-	-	0.459	0.465	0.521	0.496	0.891	0.738
	720	**0.469**	**0.466**	0.480	0.471	0.527	0.518	0.770	0.672	0.521	0.500	0.519	0.524	-	-	0.506	0.507	0.514	0.512	0.963	0.782
ETTh2	96	**0.276**	**0.333**	0.294	0.343	0.329	0.380	0.299	0.364	0.340	0.374	-	-	-	-	0.346	0.388	0.358	0.397	0.645	0.597
	192	**0.353**	**0.382**	0.377	0.393	0.431	0.443	0.441	0.454	0.402	0.414	-	-	-	-	0.429	0.439	0.456	0.452	0.788	0.683
	336	**0.357**	**0.395**	0.381	0.409	0.455	0.460	0.654	0.567	0.452	0.452	-	-	-	-	0.496	0.487	0.482	0.486	0.907	0.747
	720	**0.406**	**0.430**	0.412	0.433	0.773	0.631	0.956	0.716	0.462	0.468	-	-	-	-	0.463	0.474	0.515	0.511	0.963	0.783
ETTm1	96	0.323	**0.360**	0.321	**0.360**	0.346	0.374	**0.316**	0.362	0.338	0.375	-	-	-	-	0.379	0.419	0.505	0.475	0.543	0.510
	192	0.363	0.386	**0.362**	**0.384**	0.383	0.393	0.363	0.390	0.374	0.387	-	-	-	-	0.426	0.441	0.553	0.496	0.557	0.537
	336	0.393	0.406	**0.392**	**0.402**	0.417	0.418	0.408	0.426	0.410	0.411	0.404	0.427	-	-	0.445	0.459	0.621	0.537	0.754	0.655
	720	0.453	0.441	**0.450**	**0.435**	0.479	0.457	0.481	0.476	0.478	0.450	0.569	0.528	-	-	0.543	0.490	0.671	0.561	0.908	0.724
ETTm2	96	**0.174**	**0.256**	0.178	0.260	0.187	0.281	0.179	0.275	0.187	0.267	-	-	-	-	0.203	0.287	0.255	0.339	0.435	0.507
	192	**0.243**	**0.302**	0.249	0.307	0.272	0.349	0.307	0.376	0.249	0.309	-	-	-	-	0.269	0.328	0.281	0.340	0.730	0.673
	336	**0.301**	**0.340**	0.313	0.346	0.344	0.395	0.325	0.388	0.321	0.351	-	-	-	-	0.325	0.366	0.339	0.372	1.201	0.845
	720	**0.397**	**0.395**	0.400	0.398	0.438	0.444	0.502	0.490	0.408	0.403	-	-	-	-	0.421	0.415	0.433	0.432	3.625	1.451
# Rank 1		19	25	3	5	0	0	4	0	1	0	1	0	0	0	2	0	0	0	0	0

C.3 Results on ILI Dataset

The influenza-like illness (ILI) dataset[6] contains the weekly time-series of ratio of patients seen with ILI and the total number of the patients in the United States between 2002 and 2021. It has 7 variates and 966 timestamps, and is thus a smaller dataset compared to the others we study. Therefore, we use a different setting, following the experimental setup in Nie et al. (2023), so that the prediction horizon $T \in \{24, 36, 48, 60\}$ and the look-back window L=104. We report the baseline results from Nie et al. (2023) except for TimesNet and MICN, for which we reproduce their results with L=104 based on the officially released code. Note that the results for Fedformer and Autoformer are the best from multiple look-back windows $L \in \{24, 36, 48, 60, 104, 144\}$.

For the hyperparameters for this dataset, we have N=1 layer with B=2 branches. At each branch, the transformer consists of an attention mechanism with 4 heads, and the 2-layer feed-forward network with hidden dimension of 128 and output dimension of 32. The patch length for each branch $P = [12, 24]$ and stride length $S = [2, 4]$. The findings presented in Table 9 indicate that MTST surpasses all baseline models across all horizons in terms of the MSE metric.

Table 9: Multivariate long-term forecasting results for the ILI dataset with L=104 and $T \in \{24, 36, 48, 60\}$. **Bold** and underlined denote the best and second-best results respectively. * indicates statistically significant difference between the top-2 results.

Models		MTST		PatchTST		DLinear		MICN		TimesNet		Fedformer		Autoformer	
Dataset	T	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ILI	24	**1.499***	**0.790***	1.522	0.814	2.215	1.081	2.345	1.043	2.157	0.978	2.624	1.095	2.906	1.182
	36	**1.413***	**0.789***	1.430	0.834	1.963	0.963	2.330	1.001	2.318	1.031	2.516	1.021	2.585	1.038
	48	**1.605***	0.877	1.673	**0.854***	2.130	1.024	2.386	1.051	2.121	1.005	2.505	1.041	3.024	1.145
	60	**1.499***	**0.814***	1.529	0.862	2.368	1.096	2.616	1.131	1.975	0.975	2.742	1.122	2.761	1.114

D RELATED WORK

D.1 Long-term Time-Series Forecasting

Multivariate time-series forecasting has been an essential research focus for decades, being developed from various conventional statistical models (Makridakis and Hibon, 1997; Hyndman et al., 2008; Holt, 2004) to diverse deep-learning techniques. Viewing time-series as sequential data and inspired by early auto-regressive statistical models, earlier deep learning methods propose to model time-series with recurrent neural networks (RNNs) (Salinas et al., 2020; Smyl, 2020; Lai et al., 2018; Lim et al., 2021; Wen et al., 2017). Besides RNNs, convolution neural networks (CNNs) have also been widely used to extract features from time-series (Bai et al., 2018; Wang et al., 2023; Cui et al., 2016) However, most RNN and CNN-based approaches struggle to capture long-term dependency, which is crucial for long-term time-series forecasting. Recently, several works have proposed model architectures based on multiple-layer perceptrons (MLPs) (Oreshkin et al., 2020; Challu et al., 2022; Vijay et al., 2023) and demonstrated satisfactory performance for long-term forecasting.

Driven by the same motivation, numerous transformer-based models have been introduced to the time-series domain to better capture long-range dependency. Due to the high computational cost of attention mechanisms, earlier works in time-series transformers (TSTs) focus on the efficiency of learning long time-series sequences (Li et al., 2019; Zhou et al., 2021). Another line of work, concentrating on prediction performance, incorporates various inductive biases of time-series into achitecture designs: trend-seasonal decomposition and auto-correlation in Autoformer (Wu et al., 2021), multi-scale feature representation in Pyraformer (Liu et al., 2021a) and ScaleFormer (Shabani et al., 2023) as well as signal patterns of multiple frequencies in FEDformer (Zhou et al., 2022).

The aforementioned TSTs are all based on timestamp-level tokenization. A recent study argued that these TSTs are incapable of learning temporal patterns and can be outperformed by a simple linear model (Zeng et al., 2023). Subsequently, PatchTST (Nie et al., 2023) introduced patch-level tokenization (Dosovitskiy et al., 2020) and reached *state-of-the-art* performance for various time-series benchmarks. Another related approach is CrossFormer (Zhang and Yan, 2023), which aims to capture both cross-time and cross-variate dependencies

[6] https://gis.cdc.gov/grasp/fluview/fluportaldashboard.html

using attention. However, due to the non-cognizance of the original chronological order from patching, how to incorporate the inductive biases of time-series into TSTs with patch-level tokenization remains an open question. According to the best of our knowledge, our proposed method is one of the first to incorporate multi-scale features as an inductive bias into patch-based time-series transformers.

D.2 Multi-scale Feature Learning

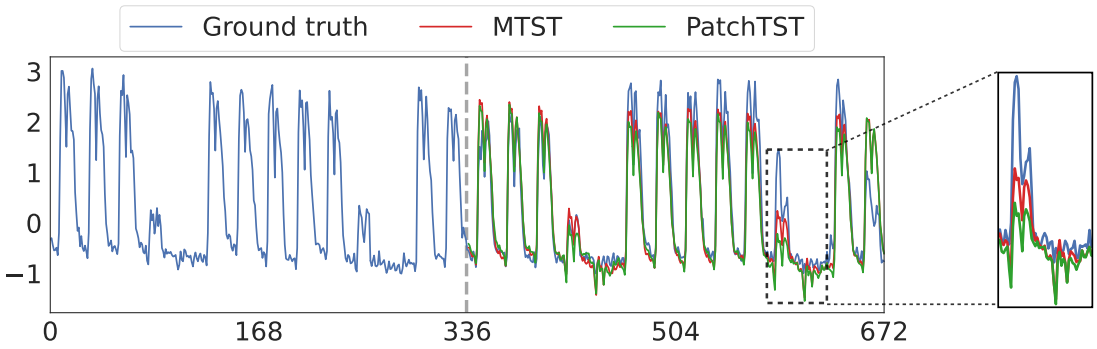
Combining multiple scales of contextual features has proven beneficial in processing complex, information-rich signals such as images, videos, and time-series. For example, several works proposed learning of multi-scale features in the image domain, including learning with convolution of different scales (Yu and Koltun, 2016; Chen et al., 2018; Szegedy et al., 2016), and constructing multi-resolution representations via subsampling (Ronneberger et al., 2015; Lin et al., 2017). Notably, following the introduction of patch-based transformers in computer vision, several works have constructed hierarchical representations via merging patches instead of subsampling (Wang et al., 2021; Liu et al., 2021b; Bolya et al., 2023).

Similar techniques have been introduced for modeling time-series in recent years. For non-transformer-based models, several works propose to capture multi-scale features with different operators in a multi-branch architecture: MCNN (Cui et al., 2016) utilizes different downsampling transformations followed by convolution and max-pooling; MICN (Wang et al., 2023) incorporates convolution with different kernel sizes in each branch; TAMS-RNNs (Chen et al., 2021) propose multi-scale RNNs with multiple hidden states. NHits (Challu et al., 2022) introduces multi-rate signal sampling schemes to the NBeats architecture to model multi-granularity features.

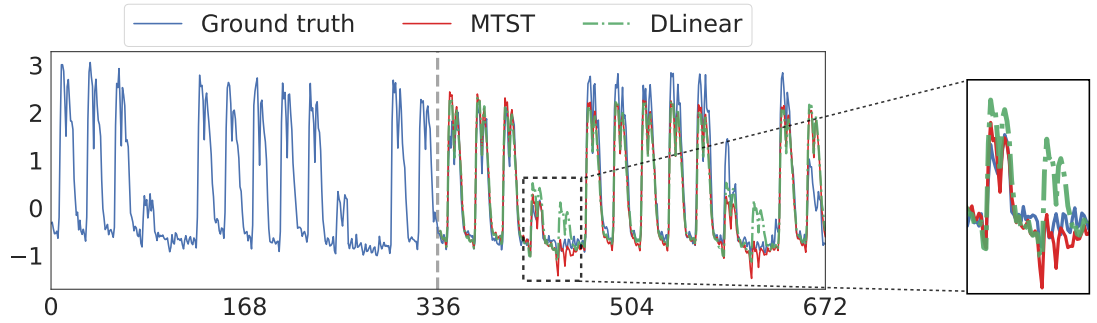
Among transformer-based models, Pyraformer (Liu et al., 2021a) proposes pyramidal attention with inter-scale and intra-scale connections to build multi-resolution representations with a focus on reducing computational complexity. Scaleformer (Shabani et al., 2023) uses mean-pooling for downsampling to reach multi-resolution representations for timestamp-level tokens. However, pooling techniques typically result in sub-optimal representation when applied to patch-level tokens, due to non-cognizance of the original chronological order (Marin et al., 2023). Therefore, we explore the alternative method to construct multi-resolution representations with adjusting patch sizes.

E FORECAST VISUALIZATION

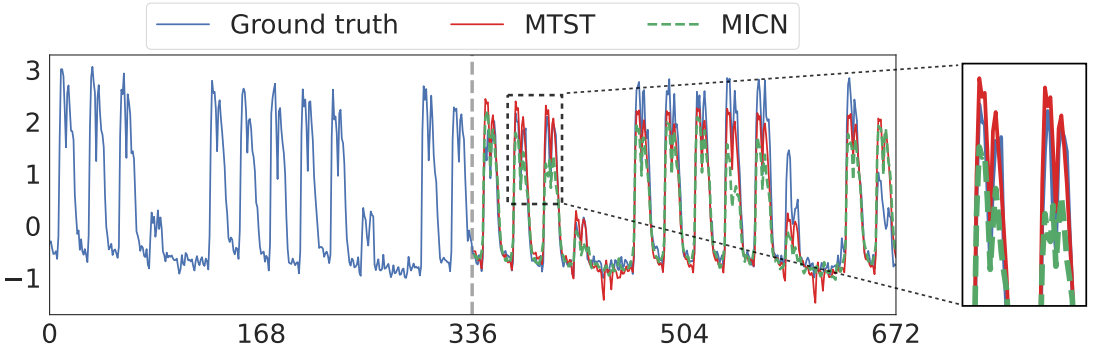
We visualize the long-term forecasting results of MTST and other baselines in Figure 7 and Figure 8. Here we show the results of predicting 336 steps ahead on Electricity and Traffic dataset. In comparison to PatchTST and DLinear, as shown in Figure 7a and 7b, MTST exhibits superior performance in capturing the weekend patterns within the data specifically. This proficiency is attributed to the low-resolution component, which captures long-term seasonality and trends. Moreover, from Figure 7c and Figure 8c, we can see that MTST outperforms in scale prediction compared to MICN. Furthermore, MTST demonstrates superior prediction accuracy, particularly when it comes to forecasting peaks in the Traffic dataset (Figure 8). This enhanced performance can potentially be attributed to the finer predictions generated by the high-resolution branch of MTST. To validate this hypothesis, we conducted an experiment where the high-resolution branch was removed. The results, as illustrated in Figure 9, clearly indicate a noticeable decline in performance of MTST w/o HighRes, emphasizing the significant contribution of the high-resolution branch in enabling finer predictions.



(a) Comparison of MTST and PatchTST

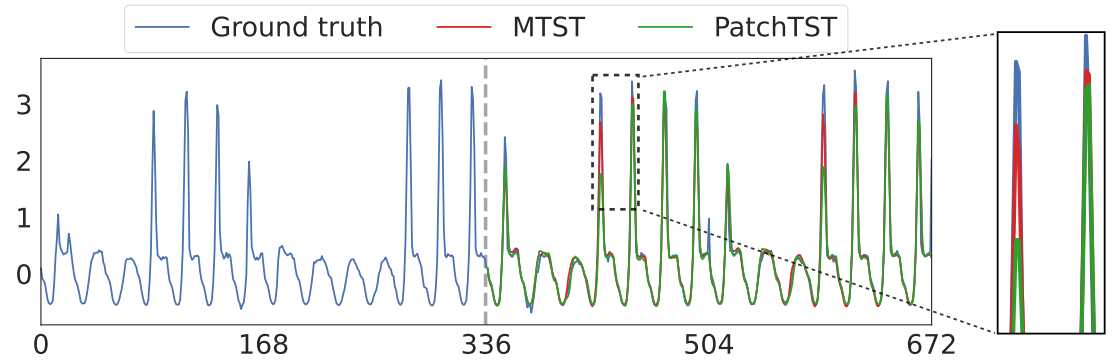


(b) Comparison of MTST and DLinear

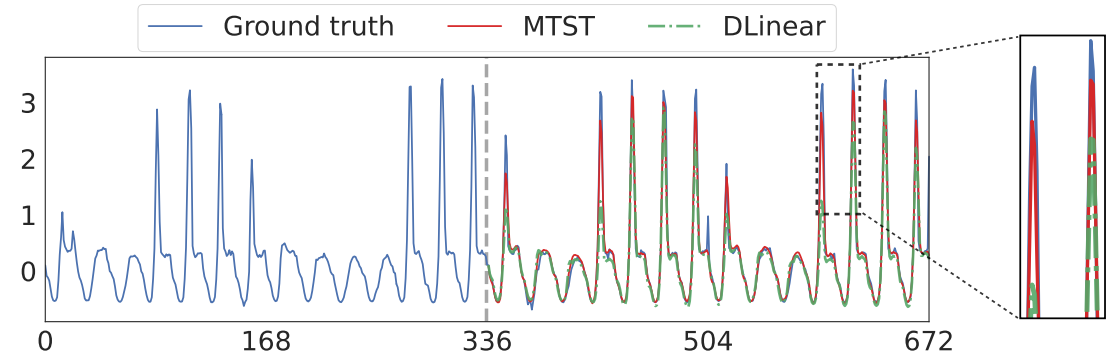


(c) Comparison of MTST and MICN.

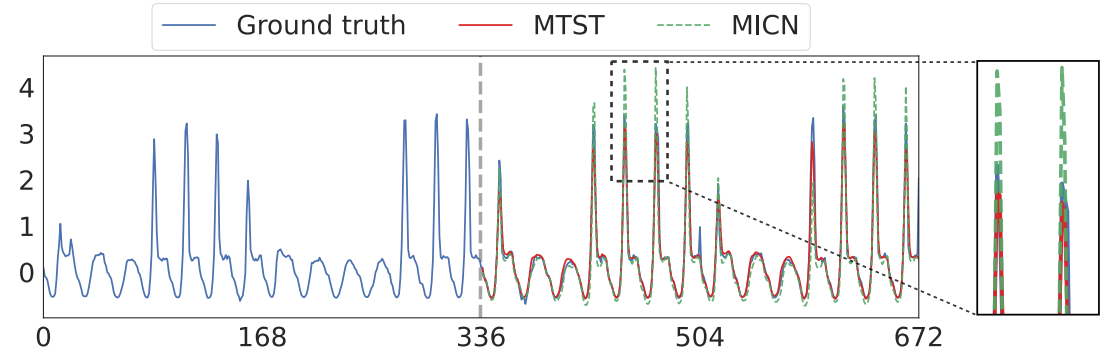
Figure 7: 11-th variate on 28-th test sample on Electricity.



(a) Comparison of MTST and PatchTST



(b) Comparison of MTST and DLinear



(c) Comparison of MTST and MICN.

Figure 8: 11-th variate on 765-th test sample on Traffic.

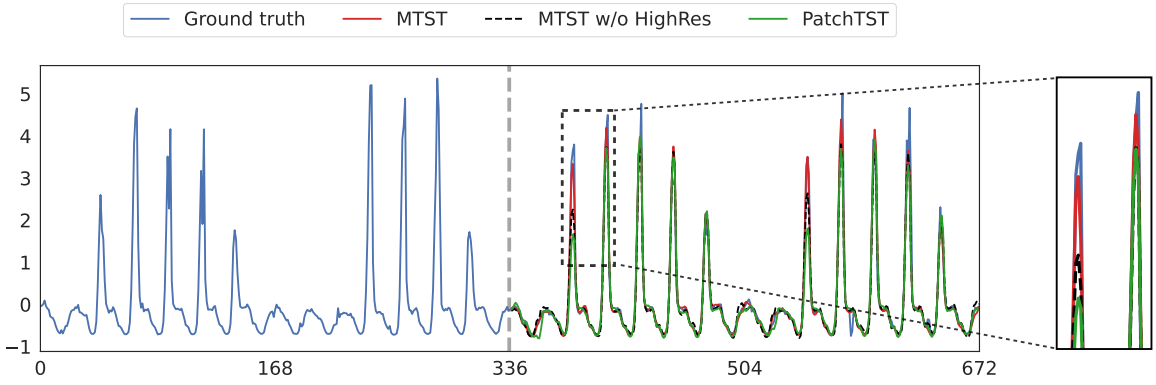


Figure 9: 4-th variate on 804-th test sample on Traffic.