IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 28 December 2023, accepted 19 January 2024, date of publication 23 January 2024, date of current version 1 February 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3357693

RESEARCH ARTICLE

Multi-Scale Transformer Pyramid Networks for Multivariate Time Series Forecasting

YIFAN ZHANG[1], RUI WU[2], (Member, IEEE), SERGIU M. DASCALU[1], (Member, IEEE), AND FREDERICK C. HARRIS JR.[1]

[1]Department of Computer Science and Engineering, University of Nevada at Reno, Reno, NV 89557, USA
[2]Department of Computer Science, East Carolina University, Greenville, NC 27858, USA

Corresponding author: Yifan Zhang (yfzhang@nevada.unr.edu)

This work was supported by the National Science Foundation under NSF Award 2142428, NSF Award 2142360, NSF Award OIA-2019609, and NSF Award OIA-2148788.

ABSTRACT Multivariate Time Series (MTS) forecasting entails the intricate process of modeling temporal dependencies within historical data records. Transformers have demonstrated remarkable performance in MTS forecasting due to their capability to capture long-term dependencies. However, prior work has been confined to modeling temporal dependencies at either a fixed scale or multiple scales that exponentially increase (most with base 2). This limitation impedes their capacity to effectively capture diverse seasonalities. In our study, we present a dimension-invariant embedding technique designed to capture short-term temporal dependencies. This procedure projects MTS data into a higher-dimensional space while preserving the original time steps and variable dimensions. Furthermore, we present a novel Multi-scale Transformer Pyramid Network (MTPNet), specifically designed to capture temporal dependencies at multiple unconstrained scales effectively. The predictions are inferred from multi-scale latent representations obtained from transformers at various scales. Extensive experiments on nine benchmark datasets demonstrate that the proposed MTPNet outperforms recent state-of-the-art methods. This enhancement in performance is particularly pronounced in datasets rich in fine-scale information, as it enables MTPNet to effectively capture a wide spectrum of temporal dependencies, ranging from fine to coarse scales. This finding highlights MTPNet's notable potential in analyzing MTS data sampled at the minute level. Code is available at github.com/MTPNet.

INDEX TERMS Time series forecasting, transformer, multi-scale feature pyramid, value embedding.

I. INTRODUCTION

Multivariate time series (MTS) data, which captures multiple variables over time, is of critical importance in various fields including finance, climate, and energy. MTS forecasting, a critical machine learning task, aims to predict the future values of multiple variables based on their historical records. The MTS data inherently demonstrates low semantic characteristics, necessitating its analysis as a collection of multiple values. For instance, this involves analyzing values of multiple variables at a single time step or values of multiple time steps for a single variable. This approach facilitates the extraction of two types of information from the MTS data: correlations among variables (spatial dependencies) and correlations across time steps (temporal dependencies).

In recent years, machine learning models [1], [2], notably transformers [3], have significantly advanced the exploration of MTS forecasting problems. The pioneering work by [4] introduced transformers to MTS forecasting, highlighting their potential in adeptly capturing temporal dependencies. Consequently, numerous transformer-based methods have been introduced for the task of MTS forecasting [5], [6], [7]. A common technique for extracting spatial dependencies utilizes a linear layer to project the MTS data into a higher-dimensional space along the spatial dimension. As a result, the values of variables at a single time step are represented as a vector in this higher-dimensional space. As for temporal dependencies, their scales are critical in

The associate editor coordinating the review of this manuscript and approving it for publication was Dost Muhammad Khan.

achieving accurate MTS forecasting. However, most existing methods are confined to capturing temporal dependencies solely at a single scale. For example, Informer [5] aims to model temporal dependency between individual time steps in the time series sequence. Autoformer [6] proposes an auto-correlation mechanism to capture temporal dependencies among sub-series. PatchTST [7] and Crossformer [8] introduce a patch procedure that divides each series within the MTS data into patches of a specific length, enabling the use of a canonical transformer to model temporal dependencies at the sub-series level.

Few methods have been proposed to model multi-scale temporal dependencies. SCINet [9] leverages an SCI-Block, which employs downsampling techniques to divide the input sequence into two sub-sequences. This division enables the extraction of distinctive temporal relations at a coarser scale by utilizing two convolution neural network (CNN) kernels. By arranging multiple SCI-Blocks in a hierarchical tree structure across multiple levels, SCINet models temporal relations at various scales. MICN [10] also incorporates downsampling techniques, involving the reduction of the original MTS data's resolution. Then, a multi-scale isometric convolution layer, comprising multiple branches of the local-global module, processes downsampled MTS data of varying scale sizes. Pyraformer [11] introduces a pyramidal graph for modeling MTS data at different resolutions. This pyramidal graph embodies a tree structure in which each parent node has several child nodes. The parent nodes summarize the sub-series of all child nodes. Thus the scale increases exponentially at the base of the number of child nodes. Crossformer [8] comprises three transformer encoder-decoder pairs that aim to capture temporal dependencies within MTS data at three scales. Crossformer merges the latent representations from the lower level as input for the next level, leading to a doubling of the scale as the levels progress.

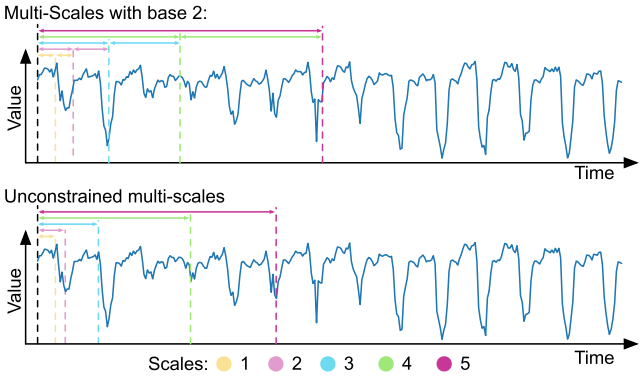

FIGURE 1. Illustration of the multi-scale mechanisms in baseline methods and MTPNet.

However, all those methods suffer from a limitation wherein their multiple scales increase exponentially (most with base 2) as shown in Figure 1. For instance, beginning with a seasonality of 1 hour, current methods scale up in increments of 2 hours, 4 hours, 8 hours, 16 hours, 32 hours, and so on. Consequently, this approach overlooks the crucial daily seasonality and potentially other seasonalities that fall within the one to 24-hour range. Consequently, they may fail to capture certain scales of temporal dependencies in MTS data that are crucial for accurate forecasting tasks. This limitation underscores the importance of developing more flexible and adaptable approaches capable of effectively modeling temporal dependencies across a wider range of arbitrary scales within the MTS data. To address the aforementioned limitations, we propose Multi-scale Transformer Pyramid Networks (MTPNet) that effectively model temporal dependencies at multiple unconstrained scales as shown in Figure 1. The MTPNet facilitates the setting of scales tailored to the specific characteristics of MTS data. The contributions of our work are summarized as follows:

- We propose a dimension invariant (DI) embedding mechanism that captures short-term temporal dependencies and projects the MTS data into a high-dimensional space. Notably, this DI embedding technique preserves both the spatial and temporal dimensions of the MTS data. The DI embedding technique further partitions the embedded feature maps along the temporal dimension. Consequently, it enables the transformer to model temporal dependencies at a designated scale.
- We propose a multi-scale transformer-based pyramid that effectively models temporal dependencies across multiple unconstrained scales, thereby offering the versatility to capture temporal patterns at various resolutions. The MTPNet comprises multiple transformers, each employing the DI embedding with varying patch sizes, to effectively model multi-scale temporal dependencies.
- We evaluate the proposed MTPNet using nine real-world datasets, and the experimental results demonstrate its superior performance compared to recent state-of-the-art methods.

II. RELATED WORKS

A. MTS FORECASTING

The primary objective of the MTS forecasting task is to establish an accurate inference between historical observations $X \in \mathbb{R}^{I \times D}$ of D variables within a look-back window of I time steps and future H time steps' values of $X_{pred} \in \mathbb{R}^{H \times D}$. Traditional statistical methods like ARIMA [12] and exponential smoothing [13] are confined to forecast univariate time series. While VAR and VARMA [14] can be extended to multivariate time series (MTS) data, their performance diminishes as the prediction length H increases. Advances in deep learning have greatly enhanced the development of MTS forecasting. LSTNet [15] and TPA-LSTM [16] combine CNN and RNN to capture short-term and long-term temporal dependencies. MTGNN [17] introduces a graph neural network framework explicitly designed to model spatial dependencies among variables in MTS data. While these methods are based on various neural network architectures, their common objective is to discover forecasting inferences through iterative weight

adjustments that minimize the discrepancies between the forecasts and the ground truth.

B. TRANSFORMERS

Transformers were first developed for natural language processing [18], [19], [20] and soon achieved great success in computer vision [21], [22] and MTS forecasting [4], [5], [6], [7], [8]. The canonical transformer architecture includes a self-attention mechanism and a feed-forward network. To enhance the training process, it employs residual connections [23] and layer normalization [24].

Early studies that applied transformers in MTS forecasting focused on modeling temporal dependencies at individual time step resolution. These approaches encountered quadratic time complexity issues, which impose limitations on the input length—a crucial factor for MTS forecasting. Informer [5] proposed ProbSparse self-attention reduced time complexity to $\mathcal{O}(n \log n)$ by only calculating a subset of queries. Fedformer [25] enhanced transformers with Fourier transforms and Wavelet transforms and achieved linear computational complexity and memory cost. Several recent studies [7], [8] have adopted the patch mechanism introduced in ViT [21] to partition the MTS data into patches, thus facilitating the transformer's efficacy in managing extended input sequences and capturing temporal dependencies at the sub-series level.

Despite the successful application of transformers in MTS forecasting, existing methods have limitations in capturing temporal dependencies at various constrained scales. This can hinder their ability to effectively capture seasonality patterns at arbitrary scales.

III. METHOD

A. DECOMPOSITION

We decompose the MTS data into seasonal and trend-cyclical components following [6], [25], [26]. Given MTS input $X \in \mathbb{R}^{I \times D}$, the decomposition procedure is as follows:

$$\begin{aligned} \mathbf{X}_t &= \text{mean}\left(\sum_{i=1}^{n} \text{MovingAvg}\left(\text{Padding}\left(\mathbf{X}\right)\right)_i\right) \\ \mathbf{X}_s &= \mathbf{X} - \mathbf{X}_t \end{aligned} \tag{1}$$

where $\mathbf{X}_s \in \mathbb{R}^{I \times D}$ and $\mathbf{X}_t \in \mathbb{R}^{I \times D}$ are seasonal and trend-cyclical components, respectively.

Figure 2 illustrates our proposed framework, incorporating seasonal and trend models to learn and forecast the seasonal and trend-cyclical components, respectively. The MTPNet functions as the seasonal model, while a simple linear layer is employed as the trend model to infer predictions directly from historical records. In scenarios where the MTS data lacks distinct seasonality and trend, we use MTPNet as the trend model to effectively learn intricate trend-cyclical components. Finally, the predictions from both the seasonal and trend models are summed elementwise to derive the final MTS predictions.

B. TRANSFORMER FEATURE PYRAMID

To address the limitations of a transformer that captures temporal dependencies solely at a single scale, we propose a multi-scale transformer pyramid network, as depicted in Figure 4. The primary objective of the MTPNet is to capture temporal dependencies across diverse unconstrained scales, ranging from fine to coarse resolutions. Notably, the total number of levels, denoted by K, is not fixed but depends on the array of available patch sizes, where $k = 1, \cdots, K$. This hierarchical architecture empowers MTPNet to model multi-scale representations of the complex temporal dependencies within the input sequence.

As illustrated in Figure 4, transformers at all levels take the MTS sequence as input, which is referred to as all-scale inputs. Note that the decoder inputs are omitted in Figure 4 for brevity and details are discussed later. The DI embedding components are distinctive at each level, as they partition input MTS data into patches of unconstrained lengths of $p_k \in \{p_1, \cdots, p_K\}$. Consequently, the multi-level transformers focus on capturing the temporal dependencies at scales from fine to coarse.

The inter-scale connections facilitate information flow between transformers at different levels within the pyramid architecture. Encoders and decoders are symmetrically structured, with encoders adopting a bottom-up approach and decoders following a top-down pattern. This design allows encoders to progressively learn latent representations from fine to coarse scales, while decoders generate fine-scale representations guided by coarse-scale levels. This yields K latent representations from the feature pyramid. Finally, a 1-layer CNN generates predictions from the concatenated K latent representations.

C. DIMENSION INVARIANT EMBEDDING

This section introduces the DI embedding technique and emphasizes the significance of maintaining both spatial and temporal dimensions intact. Due to the inherent lack of semantic information in MTS data compared to words or images, transformer-based methods for MTS forecasting commonly group values either along the spatial dimension (variables) or the temporal dimension (time steps) for further analysis.

Figure 3 shows the workflow of spatial, temporal, and DI embedding techniques. The spatial embedding [5] employs a linear layer to project the values of all variables at a single time step into an alternately dimensional space (e.g., 64, 128) while preserving the temporal dimension invariant. The temporal embedding [7], [8] preserves the spatial dimension's invariance while employing a linear layer to embed values of a variable at multiple time steps into a higher-dimensional space. Both embedding techniques break one dimension of the MTS data: spatial embedding mixes spatial information, while temporal embedding restricts the temporal scale.

To avoid these disadvantages, we introduce the DI embedding technique which utilizes a 1-layer CNN with a

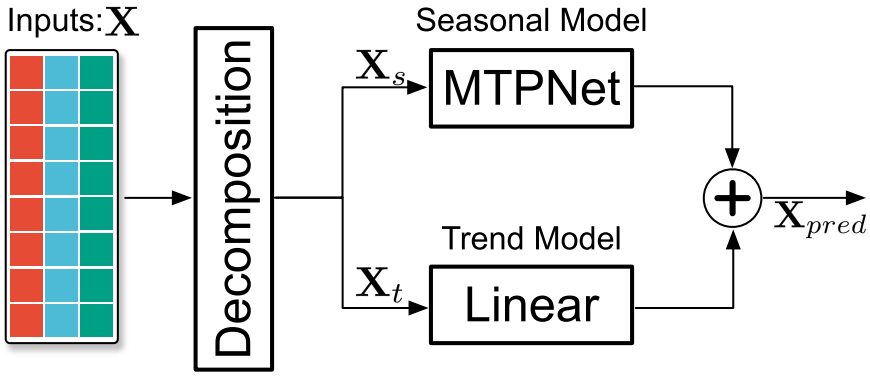

FIGURE 2. Illustration of the overall framework: Decomposition of MTS data into seasonal and trend-cyclical components, employing Multi-scale Transformer Pyramid Networks (MTPNet) as the seasonal model and a linear layer as the trend model. The seasonal and trend predictions are summed to obtain the final predictions.

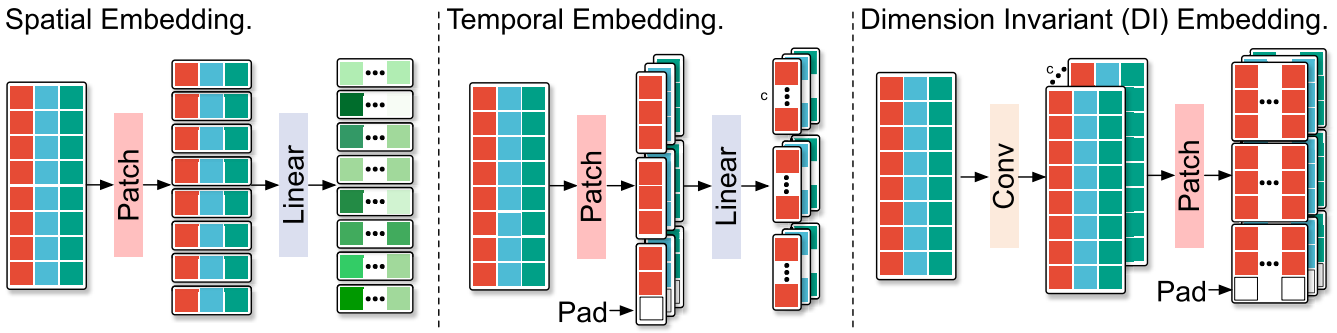

FIGURE 3. Illustration of spatial, temporal, and dimension invariant embedding techniques.

kernel size of 3×1 to embed the MTS data into feature maps while preserving both spatial and temporal dimensions invariant as follows:

$$\mathbf{X}_{emb} = \text{Conv}\left(\mathbf{X}_{input}\right) \tag{2}$$

where the $\mathbf{X}_{input} \in \mathbb{R}^{1\times I\times D}$ is either $\mathbf{X}_s$ (seasonal) or $\mathbf{X}_t$ (trend), $\mathbf{X}_{emb} \in \mathbb{R}^{c\times I\times D}$ is the embedded feature maps. The 3×1 CNN kernel captures local temporal dependencies while keeping variables independent. The Conv also captures short-term temporal dependencies.

The DI embedding then applies the Patch procedure to the embedded inputs, generating patched inputs at scale p as follows:

$$\mathbf{X}_{emb} = \text{Patch}\left(\mathbf{X}_{emb}, \mathbf{X}_{\mathbf{0}}, p\right) \tag{3}$$

where $\mathbf{X}_{\mathbf{0}}$ is zero-padding if the length of the time series is not divisible by the patch size p. The Patch procedure divides the time series into $N = \lceil I/p \rceil$ non-overlapping patches of size p, yielding $\mathbf{X}_{emb} \in \mathbb{R}^{c\times N\times p\times D}$, as shown in Figure 3. In contrast to Vision Transformers [27], which partition an image across height and width dimensions, MTS data require division solely along the time step dimension. Consequently, the embedded feature maps transformed to $N \times p$ along the time step dimension, diverging from the original total of I time steps. It is important to note that the dimensions of the feature map and variables remain unchanged. The DI embedding enhances forecasting accuracy by enabling the model to learn temporal dependencies at multiple unconstrained sub-series levels. This advancement is driven by two primary factors: firstly, it allows MTS data to be analyzed independently of variables; secondly, it empowers the model to capture dependencies among time series patches of varying sizes. In contrast, spatial embedding techniques limit the model's ability to capture temporal dependencies at the individual time step level and require combining multiple variables. Temporal embedding typically generates patches of a fixed size, thus constraining the machine learning model to operate at either a single scale or at multiple scales that grow exponentially.

D. TRANSFORMER ENCODER AND DECODER

The MTPNet comprises multiple transformer encoder-decoder pairs, designed to learn temporal dependencies at multiple unconstrained scales. Figure 4 illustrates the detailed computation procedure of one level of the MTPNet. The multiple transformer pyramid architecture in our design overcomes the constraints observed in Informer and PatchTST, where a single transformer is employed to model

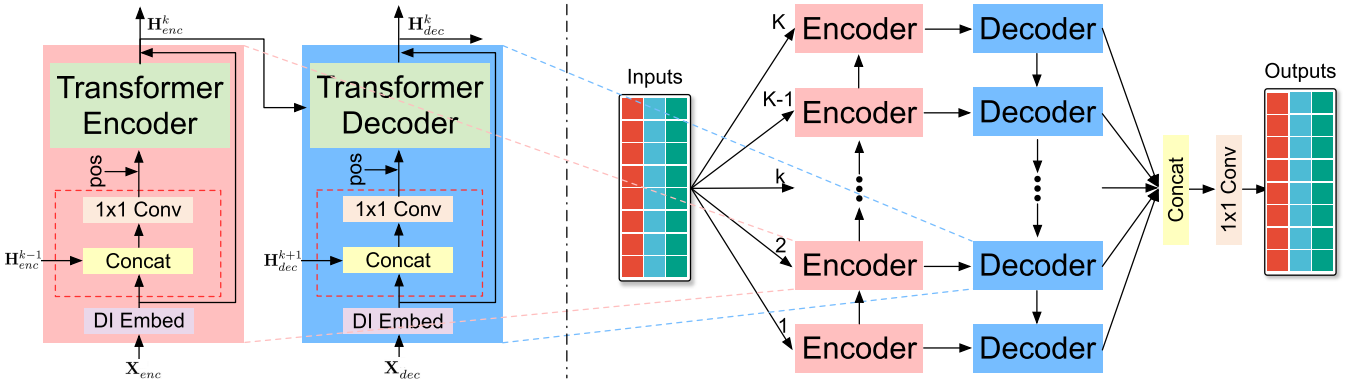

FIGURE 4. **Left: The workflow of a single-level transformer-based encoder-decoder pair. Right: Illustration of the proposed multi-scale transformer pyramid network (MTPNet).**

a singular scale. Existing methods like Crossformer, MICN, and Pyraformer aim to model scales that grow exponentially through the concatenation of fine-scale patches to generate coarse-scale patches. Our proposed MTPNet concatenates embeddings along the feature dimension, and then each transformer within the pyramid utilizes DI embedding to partition the feature embeddings to a designated scale. This ensures that the scale at each level remains independent, enhancing the model's versatility and effectiveness.

We take the k-th level of MTPNet as an illustrative example to elaborate on the detailed computation process of the transformer encoder-decoder pair.

1) ENCODER

The input $\mathbf{X}_{enc} \in \mathbb{R}^{1\times I\times D}$ are the same for the encoder at any level in the MTPNet. The DI embedding at k-th level takes input MTS data and patch size p_k as follows:

$$\mathbf{X}_{di}^{k} = \mathrm{DI}\left(\mathbf{X}_{enc}, p_k\right) \tag{4}$$

where $\mathbf{X}_{di}^{k} \in \mathbb{R}^{c\times N_{enc}^{k}\times p_k\times D}$ represents the embeded and patched k-th level encoder's input. Then, the inter-scale connections concatenate and fuse input embedding $\mathbf{X}_{di}^{k}$ with lower level encoder's output $\mathbf{H}_{enc}^{k-1}$ as follows:

$$\mathbf{X}_{emb}^{k} = \begin{cases} \mathbf{X}_{di}^{k} & \text{if } k = 1, \\ \mathrm{Conv}\left(\mathrm{Concat}\left(\mathbf{X}_{di}^{k}, \mathbf{H}_{enc}^{k-1}\right)\right) & \text{if } k > 1. \end{cases} \tag{5}$$

where Concat denote concatenate process along the feature map dimension (c) and Conv represents 1-layer CNN with a kernel size of 1×1 to fuse and reduce the concatenated feature map dimension from $2c$ to c. Note that the numbers of patches N_{enc}^{k} and $N_{enc}^{k} - 1$ differ. To address this, we apply an inverse patch operation to reassemble the patches into a complete sequence before concatenating them. Subsequently, we partition the concatenated embeddings into patches of size p_k. To incorporate positional information, we add a learnable position embedding $\mathbf{W}_{pos}^{k}$ (denoted pos) to input embedding as follows:

$$\mathbf{X}_{emb}^{k} = \mathbf{X}_{emb}^{k} + \mathbf{W}_{pos}^{k} \tag{6}$$

This step is critical because the inherent nature of the Transformer architecture is order-agnostic. Therefore, positional information needs to be incorporated to capture the temporal dependencies within the input sequence. The input embedding $\mathbf{X}_{emb}^{k}$ is split into univariate embeddings. Therefore, we obtain the transformer encoder's input $\mathbf{X}_{emb}^{k,d} \in \mathbb{R}^{c\times N_{enc}^{k}\times p_k\times 1}$ which represents the d-th variable.

We employ the canonical transformer encoder [18], utilizing the scaled dot-product attention mechanism as follows:

$$\begin{aligned} Q, K, V &= \mathrm{Linear}\left(\mathbf{X}_{emb}^{k,d}\right) \\ \mathrm{Attention}\left(Q, K, V\right) &= \mathrm{Softmax}\left(QK^{T}/\sqrt{d_k}\right)V \end{aligned} \tag{7}$$

where the Q, K, and V are query, keys, and values embedded from the input sequence of d-th series of $\mathbf{X}_{p}^{k,d} \in \mathbb{R}^{c\times N_{enc}^{k}\times p_k}$. Note that we flatten the feature map and patch size dimension of $\mathbf{X}_{p}^{k,d}$ so that $\mathbf{X}_{p}^{k,d} \in \mathbb{R}^{N_{enc}^{k}\times(c\times p_k)}$ represents the latent representations of patches of size p_k. We also utilize the canonical multi-head attention as follows:

$$\begin{aligned} Q_h, K_h, V_h &= \mathrm{Linear}\left(Q, K, V\right)_h \\ \mathbf{H}_{h}^{k,d} &= \mathrm{Attention}\left(Q_h, K_h, V_h\right) \\ \mathbf{H}_{enc}^{k,d} &= \mathrm{Linear}\left(\mathrm{Concat}\left(\mathbf{H}_{1}^{k}, \cdots, \mathbf{H}_{h}^{k}, \cdots\right)\right) \end{aligned} \tag{8}$$

where the $\mathbf{H}_{enc}^{k,d} \in \mathbb{R}^{c\times N_{enc}^{k}\times p_k}$ is the output of transformer encoder at level k for d-th series, and the subscript h indicates the h-th head of multi-head attention. By applying the encoder to all D series in the MTS data, we obtained the k-th level's encoder output $\mathbf{H}_{enc}^{k} \in \mathbb{R}^{c\times N_{enc}^{k}\times p_k\times D}$. The transformer encoder also includes normalization layers, a feed-forward network, and residual connections, details are available in [18].

The last step of the encoder is skip-connection as follows:

$$\mathbf{H}_{enc}^{k} = \mathbf{H}_{enc}^{k} + \mathbf{X}_{di}^{k} \tag{9}$$

where $\mathbf{H}_{enc}^{k} \in \mathbb{R}^{c\times N_{enc}^{k}\times p_k\times D}$ is the output of k-th level encoder.

2) DECODER

The decoder's input $X_{dec} \in \mathbb{R}^{(L+H)\times D}$ is the concatenation of the historical records (L time steps) and zero-padding (H future time steps). Similar to the encoder's workflow presented in Equations 4, 5, and 6, the decoder's input goes through DI embedding and inter-scale connections to obtain a patched embedding for the transformer decoder. It is worth mentioning that the decoder's inter-scale connections follow a top-down order, thus the decoder's output latent representations flow from coarse-scale to fine-scale. Then a learnable position embedding (denoted pos) is added to the input embedding. Consequently, we obtain the transformer decoder's input, denoted $\mathbf{X}^k_{dec} \in \mathbb{R}^{c\times N^k_{dec}\times p_k\times D}$. The decoder's number of patches is $N^k_{dec} = \lceil (L+H)/p_k \rceil$.

We also employ the canonical transformer decoder [18], utilizing the scaled dot-product attention and multi-head attention presented in Equations 7 and 8. The decoder also includes normalization layers, a feed-forward network, and residual connections, as described in [18]. The last step of the decoder is also a skip connection as presented in Equation 9.

The output of the decoder at k-th level is $\mathbf{H}^k_{dec} \in \mathbb{R}^{c\times N^k_{dec}\times p_k\times D}$. To conduct the MTS forecasting task, we only need the latent representations of future time steps. Therefore, we only need the last $N^k_{pred} = \lceil H/p_k \rceil$ patches that presenting future time steps, denoted $\mathbf{H}^k_{dec} \in \mathbb{R}^{c\times N^k_{pred}\times p_k\times D}$. At each level, the latent representations represent the predicted value using a vector of length c, containing temporal dependencies at a scale of patch size p_k.

E. MULTI-SCALE PREDICTION

Lastly, we concatenate latent representations of all K levels and generate predictions using a Conv layer as follows:

$$\begin{aligned} \mathbf{H} &= \text{Concat}\left(\mathbf{H}^1_{dec}, \cdots, \mathbf{H}^K_{dec}\right) \\ \mathbf{X}_{pred} &= \text{Conv}\left(\mathbf{H}\right) \end{aligned} \tag{10}$$

To concatenate latent representations at different scales, we apply an inverse patch operation to reassemble patches into the complete sequence. Subsequently, all K latent representations are concatenated, resulting in $\mathbf{H} \in \mathbb{R}^{(K\times c)\times H\times D}$. Each predicting future value is represented as a vector of length $K \times c$, which captures the temporal dependencies at scales ranging from p_1 to p_K. The Conv layer project each vector of length $K \times c$ into predicting value, generating the predictions $\mathbf{X}_{pred} \in \mathbb{R}^{H\times D}$. The process of inferring predictions from multi-scale latent representations enables the effective utilization of temporal dependencies across a wide range of unconstrained scales, ultimately enhancing forecasting accuracy.

IV. EXPERIMENTS

A. EXPERIMENTAL SETTINGS

1) DATA

In our experiments, we employed nine benchmark datasets [6]. These datasets are ETTh1, ETTh2, ETTm1, ETTm2, Weather, Traffic, Electricity, Exchange-Rate, and ILI, all of which are publicly accessible. Each dataset encompasses a range of variables, exhibiting diverse features. The sampling frequency of these datasets varies substantially, ranging from every 10 minutes to weekly intervals. This variation introduces differing degrees of temporal dependencies within each dataset. A summary of the characteristics of these nine benchmark datasets, integral for our methodological evaluation, is presented below:

- ETT[1]: The Electricity Transformer Temperature (ETT) is essential for long-term electric power infrastructure planning. The Informer study [5] compiled data from two Chinese counties' electricity transformers, focusing on seven indicators such as oil temperature and load. The ETTh1 and ETTh2 datasets feature 17,420 hourly samples each, while ETTm1 and ETTm2 have 69,680 samples recorded every 15 minutes.
- Traffic[2]: This dataset, obtained from the California Department of Transportation, includes road occupancy rates (0 to 1) from 862 freeway sites in the San Francisco Bay area, covering over a decade. Lai [15] compiled hourly data for 48 months (2015-2016), yielding 17,544 samples.
- Electricity[3]: This dataset from the UCI Machine Learning Repository features hourly electricity consumption data for 321 clients from 2012 to 2014, amounting to 26,304 samples.
- Weather[4]: This dataset includes 21 meteorological indicators recorded every 10 minutes over a year in Germany, totaling 52,696 samples.
- Exchange-Rate[5]: This dataset covers 27 years (1990-2016) and contains daily exchange rates for eight major economies: Australia, Britain, Canada, Switzerland, China, Japan, New Zealand, and Singapore, totaling 7,588 samples.
- ILI[6]: This dataset includes patient data with seven indicators from 2002 to 2021, sampled weekly, totaling 966 samples, and is distinguished by its unique forecasting horizon setting.

We divided each dataset into training, validation, and test subsets using a 0.6:0.2:0.2 ratio for the four ETT datasets and a 0.7:0.1:0.2 ratio for the other five datasets.

2) IMPLEMENTATION DETAILS

For training, we utilize the Adam optimizer and Cosine Annealing scheduler with an initial learning rate ranging between $1e-5$ and $1e-3$ and set the batch size to 32 while using L1 loss. The patch sizes of MTPNet are selected from $\{4, 6, 8, 12, 24, 32, 48, 96\}$ via grid search.

[1] https://github.com/zhouhaoyi/ETDataset
[2] http://pems.dot.ca.gov
[3] https://archive.ics.uci.edu/ml/datasets/ElectricityLoadDiagrams20112014
[4] https://www.bgc-jena.mpg.de/wetter/
[5] https://github.com/laiguokun/multivariate-time-series-data
[6] https://gis.cdc.gov/grasp/fluview/fluportaldashboard.html

The look-back window sizes of MTPNet are selected from {96, 192, 336, 720} through grid search, except for the ILI dataset, where it was set to 104. The transformers of MTPNet consist of 2 encoder layers and 1 decoder layer. The multi-head attention number is set to 4. The MTPNet is implemented with PyTorch and runs with an NVIDIA GeForce RTX 3090 GPU and an NVIDIA RTX A6000 GPU. The main results are averaged across six runs with distinct seeds: 1, 2022, 2023, 2024, 2025, and 2026. Additionally, ablation studies were conducted using seed 1.

3) EVALUATION AND BASELINES

We selected seven state-of-the-art (SOTA) baseline methods as follows:

- Transformer-based methods: Informer [5], Pyraformer [11], FEDformer [25], Crossformer [8], PatchTST [7].
- Linear methods: DLinear [26].
- CNN methods: MICN [10].

We obtained the results of DLinear, Pyraformer, Fedformer, and Informer from Dlinear [26]. The results for PatchTST and Crossformer were obtained from the original paper. In cases where results are not available, we conducted experiments using the optimal hyperparameters as presented in the original papers. In particular, for a fair comparison, we did not fix the input length to 96, as recent studies [7], [8], [26] have highlighted the significance of optimal input lengths for method performance.

We employed Mean Squared Error (MSE) and Mean Absolute Error (MAE) as quantitative metrics for assessing forecasting accuracy. These measures are consistently used across all baseline methods, providing a standardized basis for comparison and evaluation. Consequently, the results of the baseline methods presented in the experiments represent their respective best performances.

B. MAIN RESULTS

Table 1 shows the main experimental results of all methods for nine datasets on MSE and MAE, where the best and second-best results for each case (dataset, horizon, and metric) are highlighted in bold and underlined, respectively. The MTPNet outperforms SOTA baseline methods, achieving 45 best results and 19 second-best results out of 72 cases. MTPNet achieves a modest enhancement in accuracy when compared with the best existing method, PatchTST. Compared to DLinear, which raised questions about the effectiveness of transformers in MTS forecasting, MTPNet demonstrates a reduction of 7.04% in MSE and 8.56% in MAE. Notably, PatchTST and Dlinear are limited to modeling temporal dependencies at a fixed scale. In contrast, the implementation of a multi-scale transformer pyramid architecture in our proposed MTPNet has enhanced forecasting accuracy. This highlights the efficacy of modeling multi-scale temporal dependencies in predictive tasks.

Compared to methods that model temporal dependencies across multiple scales that exponentially increase, MTPNet significantly enhances forecasting accuracy. In comparison with MICN, MTPNet exhibited a performance enhancement of 19.53% in MSE and 16.72% in MAE. Against Crossformer, it achieved noteworthy reductions in MSE and MAE, averaging 39.84% and 30.32%, respectively. Additionally, MTPNet demonstrates a substantial decrease in MSE and MAE by 64.35% and 51.63%, showcasing its efficacy. These findings highlight the critical role of effectively modeling unconstrained multi-scale temporal dependencies in forecasting tasks.

The MTPNet, designed to model temporal dependencies from fine to coarse scales, exhibits increased proficiency with datasets containing high-frequency information (e.g., sampled every 10 minutes). Consequently, MTPNet achieved the best results in 22 out of 24 cases for datasets like ETTm1 (15 minutes), ETTm2 (15 minutes), and Weather (10 minutes). Conversely, MTPNet's performance was marginally surpassed by PatchTST and Dlinear in the Exchange (1 day) and ILI (1 week) datasets. This is attributed to the lack of fine-scale information in these particular datasets. We conclude that MTPNet exhibits advantages in forecasting datasets that encompass rich information spanning from fine to coarse scales. It is worth noting that the average standard deviations of MTPNet across all cases, with six different random seeds, are 0.036 and 0.009 for MSE and MAE, respectively, demonstrating its robustness against randomness.

C. ABLATION STUDY

1) HOW IMPORTANT IS DI EMBEDDING?

Table 2 presents a comparison of DI embedding with spatial embedding and temporal embedding. The mechanisms of spatial and temporal embedding are illustrated in Figure 3. The DI embedding consistently outperformed the spatial embedding mechanism. For the horizon of 96, the MSE and MAE values of the spatial embedding were 9.09% and 5.6% higher, respectively, compared to the DI embedding. The performance gap increased further for the horizon of 720, with the spatial embedding showing 32.11% higher MSE and 12.95% higher MAE compared to the DI embedding. The temporal embedding slightly degrades the MSE and MAE values for ETTh1, ETTm1, and Weather datasets by 3.9% and 3.8%, respectively. Notably, the temporal embedding achieved the best performance for the Exchange-Rate dataset. We conjecture that this is because the Exchange-Rate dataset inherently exhibits less temporal dependence. In conclusion, our findings demonstrate that DI embedding outperforms both spatial and temporal embeddings. Furthermore, breaking the dimensionality of MTS data leads to a degradation in performance.

2) HOW IMPORTANT IS MULTI-SCALE TEMPORAL DEPENDENCY LEARNING?

To evaluate the effectiveness of multi-scale temporal dependency learning, we present experimental results of

TABLE 1. Quantitative evaluation (MSE/MAE) of state-of-the-art multivariate time series forecasting methods on nine datasets. The forecasting horizons include 24, 36, 48, 96 for the ILI dataset, and 96, 192, 336, 720 for the others. Bold results indicate the best performance while underlined results represent the second-best performance.

Methods		MTPNet		PatchTST/64		DLinear		Crossformer		MICN		Pyraformer		Fedformer		Informer	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
$ETTh_1$	96	**0.364**	**0.385**	0.370	0.400	0.375	0.399	0.431	0.458	0.421	0.431	0.664	0.612	0.376	0.419	0.865	0.713
	192	**0.404**	**0.410**	0.413	0.429	0.405	0.416	0.420	0.448	0.474	0.487	0.790	0.681	0.420	0.448	1.008	0.792
	336	0.431	**0.432**	**0.422**	0.440	0.439	0.443	0.440	0.461	0.569	0.551	0.891	0.738	0.459	0.465	1.107	0.809
	720	0.453	**0.463**	**0.447**	0.468	0.472	0.490	0.519	0.524	0.770	0.672	0.963	0.782	0.506	0.507	1.181	0.865
$ETTh_2$	96	0.278	**0.335**	**0.274**	0.337	0.289	0.353	1.177	0.757	0.299	0.364	0.645	0.597	0.358	0.397	3.755	1.525
	168	**0.340**	**0.376**	0.341	0.382	0.383	0.418	1.206	0.796	0.441	0.454	0.788	0.683	0.429	0.439	5.602	1.931
	336	0.365	0.403	**0.329**	**0.384**	0.448	0.465	1.452	0.883	0.654	0.567	0.907	0.747	0.496	0.487	4.721	1.835
	720	0.400	0.435	**0.379**	**0.422**	0.605	0.551	2.040	1.121	0.956	0.716	0.963	0.783	0.463	0.474	3.647	1.625
$ETTm_1$	96	**0.291**	**0.332**	0.293	0.346	0.299	0.343	0.320	0.373	0.316	0.362	0.543	0.510	0.379	0.419	0.672	0.571
	192	**0.332**	**0.355**	0.333	0.370	0.335	0.365	0.400	0.432	0.363	0.390	0.557	0.537	0.426	0.441	0.795	0.669
	336	**0.367**	**0.376**	0.369	0.392	0.369	0.386	0.408	0.428	0.408	0.426	0.754	0.655	0.445	0.459	1.212	0.871
	720	0.425	**0.410**	**0.416**	0.420	0.425	0.421	0.582	0.537	0.481	0.476	0.908	0.724	0.543	0.490	1.166	0.823
$ETTm_2$	96	**0.164**	**0.248**	0.166	0.256	0.167	0.260	0.444	0.463	0.179	0.275	0.435	0.507	0.203	0.287	0.365	0.453
	192	0.223	**0.291**	**0.223**	0.296	0.224	0.303	0.833	0.657	0.307	0.376	0.730	0.673	0.269	0.328	0.533	0.563
	336	**0.273**	**0.325**	0.274	0.329	0.281	0.342	0.766	0.620	0.325	0.388	1.201	0.845	0.325	0.366	1.363	0.887
	720	**0.356**	**0.380**	0.362	0.385	0.397	0.421	0.959	0.752	0.502	0.490	3.625	1.451	0.421	0.415	3.379	1.338
Weather	96	**0.146**	**0.189**	0.149	0.198	0.176	0.237	0.158	0.231	0.161	0.229	0.622	0.556	0.217	0.296	0.300	0.384
	192	**0.188**	**0.230**	0.194	0.241	0.220	0.282	0.194	0.262	0.220	0.281	0.739	0.624	0.276	0.336	0.598	0.544
	336	**0.238**	**0.271**	0.245	0.282	0.265	0.319	0.495	0.515	0.278	0.331	1.004	0.753	0.339	0.380	0.578	0.523
	720	**0.310**	**0.322**	0.314	0.334	0.323	0.362	0.526	0.542	0.311	0.356	1.420	0.934	0.403	0.428	1.059	0.741
Traffic	96	0.401	**0.234**	**0.360**	0.249	0.410	0.282	0.538	0.300	0.519	0.309	0.867	0.468	0.587	0.366	0.719	0.391
	192	0.431	**0.247**	**0.379**	0.256	0.423	0.287	0.515	0.288	0.537	0.315	0.869	0.467	0.604	0.373	0.696	0.379
	336	0.453	**0.259**	**0.392**	0.264	0.436	0.296	0.530	0.300	0.534	0.313	0.881	0.469	0.621	0.383	0.777	0.420
	720	0.491	**0.284**	**0.432**	0.286	0.466	0.315	0.573	0.313	0.577	0.325	0.896	0.473	0.626	0.382	0.864	0.472
Electricity	96	**0.128**	**0.219**	0.129	0.222	0.140	0.237	0.141	0.240	0.164	0.269	0.386	0.449	0.193	0.308	0.274	0.368
	192	**0.146**	**0.237**	0.147	0.240	0.153	0.249	0.166	0.265	0.177	0.177	0.378	0.443	0.201	0.315	0.296	0.386
	336	0.164	**0.256**	**0.163**	0.259	0.169	0.267	0.323	0.369	0.193	0.304	0.376	0.443	0.214	0.329	0.300	0.394
	720	0.203	0.293	**0.197**	**0.290**	0.203	0.301	0.404	0.423	0.212	0.321	0.376	0.445	0.246	0.355	0.373	0.439
Exchange	96	0.091	0.215	0.896	0.209	**0.081**	**0.203**	0.323	0.425	0.102	0.235	1.748	1.105	0.148	0.278	0.847	0.752
	192	0.175	0.301	0.187	0.308	**0.157**	**0.293**	0.448	0.506	0.172	0.316	1.874	1.151	0.271	0.380	1.204	0.895
	336	0.280	**0.389**	0.349	0.432	0.305	0.414	0.840	0.718	**0.272**	0.407	1.943	1.172	0.460	0.500	1.672	1.036
	720	**0.613**	0.606	0.900	0.715	0.643	**0.601**	1.416	0.959	0.714	0.658	2.085	1.206	1.195	0.841	2.478	1.310
ILI	24	1.602	0.837	**1.319**	**0.754**	2.215	1.081	3.041	1.186	2.684	1.112	7.394	2.012	3.228	1.260	5.764	1.677
	36	**1.371**	**0.761**	1.579	0.870	1.963	0.963	3.406	1.232	2.667	1.068	7.551	2.031	2.679	1.080	4.755	1.467
	48	**1.371**	0.822	1.553	**0.815**	2.130	1.024	3.459	1.221	2.558	1.052	7.662	2.057	2.622	1.078	4.763	1.469
	60	1.696	0.884	**1.470**	**0.788**	2.368	1.096	3.640	1.305	2.747	1.110	7.931	2.100	2.857	1.157	5.264	1.564

TABLE 2. Multivariate time series forecasting results of MTPNet with three embedding mechanisms: dimension invariant embedding, spatial embedding, and temporal embedding.

Methods		DI		Spatial		Temporal	
Metric		MSE	MAE	MSE	MAE	MSE	MAE
$ETTh_1$	96	**0.365**	**0.384**	0.371	0.396	0.369	0.391
	720	**0.455**	**0.464**	0.554	0.506	0.494	0.499
$ETTm_1$	96	**0.292**	**0.333**	0.302	0.339	0.302	0.342
	720	**0.424**	**0.409**	0.431	0.410	0.428	0.416
Weather	96	**0.145**	**0.187**	0.149	0.193	0.155	0.200
	720	**0.312**	**0.324**	0.316	0.326	0.320	0.331
Exchange	96	0.090	0.214	0.116	0.244	**0.085**	**0.206**
	720	0.581	0.592	1.185	0.840	**0.523**	**0.548**

single-scale MTPNet of coarse (large patch size) and fine (small patch size) in Table 3. Both modifications performed worse than the multi-scale design. Specifically, MTPNet-Fine exhibited a more substantial performance drop than MTPNet-Coarse, showing the challenge of capturing meaningful temporal dependencies from a small number of time steps due to time series data's naturally low semantic characteristics. Furthermore, the multi-scale transformer pyramid architecture consistently outperformed individual fixed scales. This observation emphasizes the critical importance of the multi-scale transformer pyramid design in the context of MTS forecasting.

3) HOW IMPORTANT ARE INTER-SCALE CONNECTIONS?

We modified the MTPNet by removing the inter-scale connections, resulting in no information flow between multiple levels. Each transformer level now receives input solely from the embedded input sequence patches of fixed scale. The results are presented in Table 3 as "w/o inter-scale." Surprisingly, the performance of MTPNet without inter-scale connections was even improved by a trivial amount. We conjecture that this improvement may be due to time series data's naturally low semantic characteristics. As a result, MTS forecasting doesn't require latent representation flow between levels to extract high semantic information.

4) HOW IMPORTANT ARE INPUTS FOR ALL SCALES?

The "w/o inter-scale" column in Table 3 presents the results of MTPNet without the inputs for all levels. In this configuration, transformers, except for the first level, only take the latent representation from the previous level as input. The performance of MTPNet without all scale inputs dropped

TABLE 3. The ablation study results of MTPNet's pyramid structure.

Methods		MTPNet		w/o inter-scale		w/o all-scale		bottom-up		Fine		Coarse	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
$ETTh_1$	96	0.365	0.384	0.365	0.385	0.364	0.386	**0.362**	**0.383**	0.400	0.423	0.375	0.390
	720	0.455	0.464	0.450	**0.460**	0.462	0.462	**0.449**	0.462	0.515	0.509	0.461	0.462
$ETTm_1$	96	0.292	0.333	**0.291**	**0.332**	0.292	0.333	0.292	0.333	0.307	0.346	0.301	0.339
	720	0.424	0.409	**0.423**	**0.409**	0.428	0.411	0.424	0.409	0.432	0.417	0.428	0.414
Weather	96	**0.145**	**0.187**	0.148	0.187	0.152	0.191	0.149	0.195	0.158	0.203	0.148	0.190
	720	0.312	0.324	**0.309**	**0.321**	0.315	0.329	0.311	0.324	0.322	0.331	0.313	0.326
Exchange	96	0.090	0.214	**0.087**	**0.208**	0.100	0.223	0.096	0.220	0.093	0.218	0.093	0.217
	720	0.581	**0.592**	**0.579**	0.597	0.648	0.619	0.601	0.596	0.635	0.612	0.648	0.617

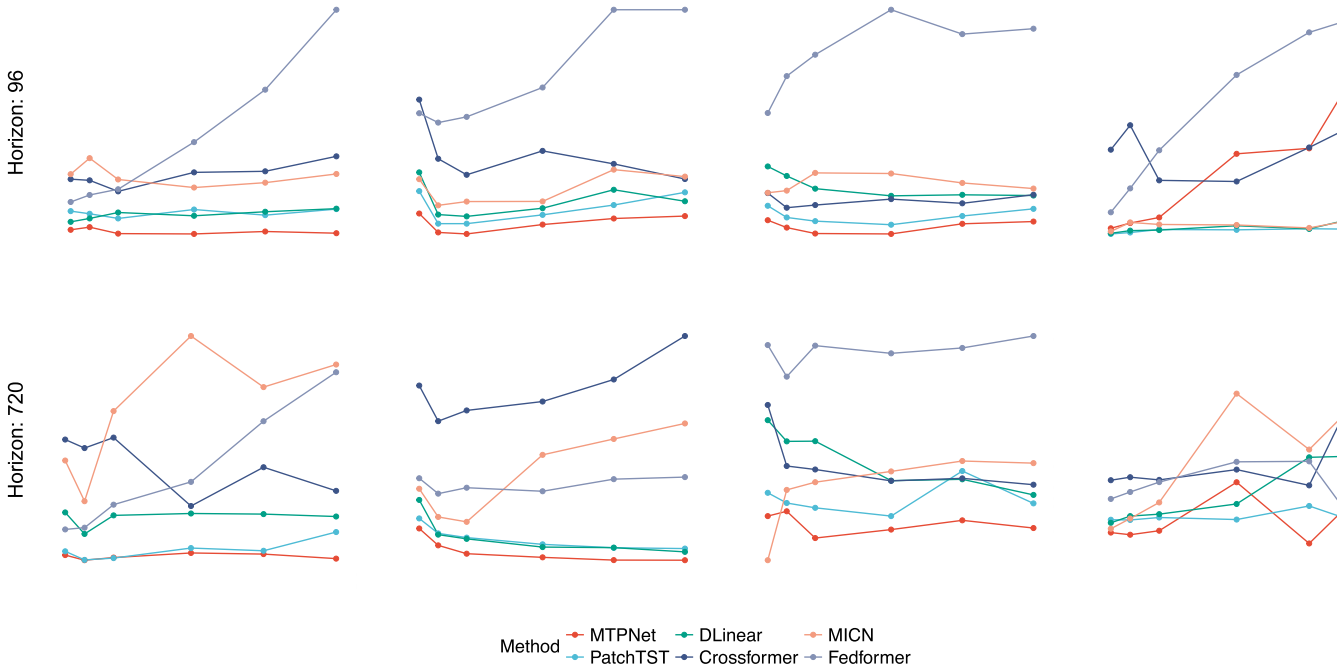

FIGURE 5. The forecasting results in terms of MAE for different look-back window sizes at horizons 96 and 720.

for ETTh1 and ETTm1 datasets when the forecasting horizon was 720 and for both horizons of Weather and Exchange-Rate datasets. The latent representations from the lower level are grounded in a different temporal scale. In contrast, the direct MTS data input integrates a DI embedding component, patching input at the specific scale of the current level. As the forecasting horizon increases, the task grows more challenging, emphasizing the greater significance of direct input. From the experiments, we conclude that direct MTS data input is critical for more challenging forecasting scenarios.

5) TOP-DOWN VS. BOTTOM-UP

The "bottom-up" column in Table 3 shows the results of MTPNet with a modification where the top-down latent representation flow in decoders is replaced with a bottom-up approach. This reversal in information flow means that each transformer decoder's input is a fusion of the input sequence and the latent representation of the lower layer (finer scale). This modification only degrades the performance of the Exchange-Rate dataset. It is worth noting that MTPNet generates predictions by utilizing a Conv layer to project from the latent representations of all K levels. Thus, the information from all temporal scales is utilized when generating predictions. This finding indicates that both information flow from coarse to fine and from fine to coarse can enhance forecasting accuracy.

6) HOW DOES THE LOOK-BACK WINDOW LENGTH AFFECT THE PERFORMANCE?

The size of the input sequence plays a crucial role in MTS forecasting as it determines the amount of historical information that can be utilized. Theoretically, an extended

input sequence is expected to enhance forecasting accuracy as it encompasses a greater volume of information. However, recent studies [7], [26] have shown that this assumption does not always hold. Figure 5 illustrates the effect of the input sequence length on the forecasting accuracy. For both horizons, no method consistently benefits from a longer input sequence across all four datasets. In most cases, a method reaches an optimal input sequence length, and its performance degrades when the input sequence becomes longer. Notably, for the Exchange-Rate dataset, a shorter input sequence appears to be optimal for most methods. We attribute this to the inherently low temporal dependency in this dataset. In conclusion, the optimal input sequence length varies depending on the dataset and forecasting horizon.

V. CONCLUSION

Time series data often exhibit various scales of seasonality, and these temporal dependencies are crucial for accurate forecasting tasks. In this study, we proposed a multi-scale transformer-based pyramid network for MTS forecasting. The proposed MTPNet tackles the complexity of modeling temporal dependencies across either a fixed scale or constrained multi-scales. The overall framework initially decomposes MTS data into seasonal and trend components. A linear layer is employed to directly generate predictions for the trend component from its historical data. In parallel, MTPNet is employed to model temporal dependencies and generate predictions for the seasonal component. The DI embedding procedure is utilized to segment the time series sequence into patches, where the size of each patch varies according to the level. Subsequently, MTPNet leverages multiple transformers to capture temporal dependencies across a range of unconstrained scales. These multi-scale latent representations are subsequently concatenated, followed by the application of a CNN layer to generate predictions for the seasonal component. The final predictions are derived by summing the predictions of the seasonal and trend components in an element-wise manner. Extensive experimental results demonstrate that MTPNet outperforms existing state-of-the-art methods, particularly those that aim to address the multi-scale temporal dependency issue. Moving forward, we intend to further develop a sparse attention mechanism as a substitute for the canonical attention mechanism. We aim to decrease the computational complexity from quadratic to linear by enabling each query to attend to a limited number of highly correlated keys. We are committed to further developing an application that employs our proposed methods to visualize meteorological MTS data in the Lake Tahoe region. This application aims to forecast critical events, including snowstorms, icy road conditions, and wildfires.

ACKNOWLEDGMENT

Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

REFERENCES

[1] W. Zheng and J. Hu, "Multivariate time series prediction based on temporal change information learning method," *IEEE Trans. Neural Netw. Learn. Syst.*, pp. 1–15, Jan. 2022.
[2] T. Y. Lin, P. Dollàr, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jul. 2017, pp. 2117–2125.
[3] Q. Wen, T. Zhou, C. Zhang, W. Chen, Z. Ma, J. Yan, and L. Sun, "Transformers in time series: A survey," in *Proc. 32nd Int. Joint Conf. Artif. Intell.*, Aug. 2023, pp. 1–8.
[4] S. Li, X. Jin, Y. Xuan, X. Zhou, W. Chen, Y.-X. Wang, and X. Yan, *Enhancing Locality Breaking Memory Bottleneck Transformer Time Ser. Forecasting*. Red Hook, NY, USA: Curran Associates, 2019.
[5] H. Zhou, S. Zhang, J. Peng, S. Zhang, J. Li, H. Xiong, and W. Zhang, "Informer: Beyond efficient transformer for long sequence time-series forecasting," in *Proc. AAAI Conf. Artif. Intell.*, vol. 35, no. 12, 2021, pp. 11106–11115.
[6] H. Wu, J. Xu, J. Wang, and M. Long, "Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting," in *Proc. Adv. Neural Inf. Process. Syst.*, 2021.
[7] Y. Nie, N. H. Nguyen, P. Sinthong, and J. Kalagnanam, "A time series is worth 64 words: Long-term forecasting with transformers," in *Int. Conf. Learn. Represent.*, 2023.
[8] Y. Zhang and J. Yan, "Crossformer: Transformer utilizing cross-dimension dependency for multivariate time series forecasting," in *Proc. Int. Conf. Learn. Represent.*, 2023.
[9] M. Liu, A. Zeng, M. Chen, Z. Xu, Q. Lai, L. Ma, and Q. Xu, "SCINet: Time series modeling and forecasting with sample convolution and interaction," in *Proc. 36th Conf. Neural Inf. Process. Syst. (NeurIPS)*, 2022.
[10] H. Wang, J. Peng, F. Huang, J. Wang, J. Chen, and Y. Xiao, "MICN: Multi-scale local and global context modeling for long-term series forecasting," Tech. Rep., 2023.
[11] S. Liu, H. Yu, C. Liao, J. Li, W. Lin, A. X. Liu, and S. Dustdar, "Pyraformer: Low-complexity pyramidal attention for long-range time series modeling and forecasting," in *Proc. Int. Conf. Learn. Represent.*, 2022.
[12] G. E. Box and G. M. Jenkins, "Some recent advances in forecasting and control," *J. Roy. Stat. Soc., Ser. C, Appl. Statist.*, vol. 17, no. 2, pp. 91–109, 1968.
[13] R. Hyndman, A. B. Koehler, J. K. Ord, and R. D. Snyder, *Forecasting With Exponential Smoothing: State Space Approach*. Cham, Switzerland: Springer, 2008.
[14] L. Kilian and H. Lütkepohl, *Structural Vector Autoregressive Analysis*. Cambridge, U.K.: Cambridge Univ. Press, 2017.
[15] G. Lai, W.-C. Chang, Y. Yang, and H. Liu, "Modeling long-and short-term temporal patterns with deep neural networks," in *Proc. Int. ACM Conf. Res. Devel. Inf. Retrieval (SIGIR)*, 2018, pp. 95–104.
[16] S.-Y. Shih, F.-K. Sun, and H.-Y. Lee, "Temporal pattern attention for multivariate time series forecasting," *Mach. Learn.*, vol. 108, nos. 8–9, pp. 1421–1441, Sep. 2019.
[17] Z. Wu, S. Pan, G. Long, J. Jiang, X. Chang, and C. Zhang, "Connecting the dots: Multivariate time series forecasting with graph neural networks," in *Proc. 26th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2020.
[18] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. U. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. Red Hook, NY, USA: Curran Associates, 2017, pp. 1–15. [Online]. Available: https://proceedings.neurips.cc/paperfiles/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf
[19] S. Bano and S. Khalid, "BERT-based extractive text summarization of scholarly articles: A novel architecture," in *Proc. Int. Conf. Artif. Intell. Things (ICAIoT)*, Dec. 2022, pp. 1–5.
[20] S. Bano, S. Khalid, N. M. Tairan, H. Shah, and H. A. Khattak, "Summarization of scholarly articles using BERT and BiGRU: Deep learning-based extractive approach," *J. King Saud Univ. Comput. Inf. Sci.*, vol. 35, no. 9, Oct. 2023, Art. no. 101739.
[21] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16×16 words: Transformers for image recognition at scale," in *Proc. ICLR*, 2021.

[22] W. Wang, "Pyramid vision transformer: A versatile backbone for dense prediction without convolutions," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 568–578.
[23] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.
[24] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," Tech. Rep., 2016.
[25] T. Zhou, Z. Ma, Q. Wen, X. Wang, L. Sun, and R. Jin, "FEDformer: Frequency enhanced decomposed transformer for long-term series forecasting," in *Proc. 39th Int. Conf. Mach. Learn. (ICML)*, 2022.
[26] A. Zeng, M. Chen, L. Zhang, and Q. Xu, "Are transformers effective for time series forecasting?" Tech. Rep., 2023.
[27] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16×16 words: Transformers for image recognition at scale," 2020, *arXiv:2010.11929*.

YIFAN ZHANG received the bachelor's degree in automation from Huainan Normal University, China, in 2016, and the master's degree in control science and engineering from the Nanjing University of Aeronautics and Astronautics, China, in 2019. He is currently pursuing the Ph.D. degree with the Department of Computer Science and Engineering, University of Nevada at Reno, Reno, NV, USA. His research interests include machine learning, time series data analysis, and computer vision.

RUI WU (Member, IEEE) received the bachelor's degree in computer science and technology from Jilin University, China, in 2013, and the master's and Ph.D. degrees in computer science and engineering from the University of Nevada at Reno, Reno, NV, USA, in 2015 and 2018, respectively. He is currently an Assistant Professor with the Department of Computer Science, East Carolina University, and collaborates with geological and hydrological scientists to protect ecological systems. His main research interests include machine learning and data visualization using AR/VR devices.

SERGIU M. DASCALU (Member, IEEE) received the master's degree in automatic control and computers from the Politehnica University of Bucharest, Romania, in 1982, and the Ph.D. degree in computer science from Dalhousie University, Canada, in 2001. In July 2022, he joined the Department of Computer Science and Engineering, University of Nevada at Reno (UNR), Reno, NV, USA, as a Professor. At UNR, he is also the Director of the Software Engineering Laboratory (SOELA) and the Co-Director of the Cyberinfrastructure Laboratory (CIL). Since joining UNR, he has worked on research projects funded by federal agencies (NSF, NASA, DoD-ONR) as well as the industry. He has advised 11 Ph.D. and more than 50 master's students. He is a Senior Member of ACM. He received several awards, including the 2009 Nevada Center for Entrepreneurship Faculty Advisor Award, the 2011 UNR Outstanding Undergraduate Research Faculty Mentor Award, the 2011 UNR Donald Tibbitts Distinguished Teacher of the Year Award, the 2014 CoEN Faculty Excellence Award, and the 2019 UNR Vada Trimble Outstanding Graduate Mentor Award.

FREDERICK C. HARRIS JR. received the B.S. and M.S. degrees in mathematics and educational administration from Bob Jones University, Greenville, SC, USA, in 1986 and 1988, respectively, and the M.S. and Ph.D. degrees in computer science from Clemson University, Clemson, SC, USA, in 1991 and 1994, respectively.

He is currently the Associate Dean of Research with the College of Engineering, a Foundation Professor with the Department of Computer Science and Engineering, and the Director of the High Performance Computation and Visualization Laboratory, University of Nevada at Reno (UNR). Since joining UNR, he has worked on research projects funded by federal agencies (NSF, NASA, DARPA, ONR, DoD) as well as industry. He is also the Nevada State EPSCoR Director and the Project Director of Nevada NSF EPSCoR. He has published more than 300 peer-reviewed journals and conference papers along with several book chapters. He edited or co-edited 15 books. He has 14 Ph.D. students and 84 M.S. thesis students finished under his supervision. His research interests include parallel computation, simulation, computer graphics, and virtual reality. He is a Senior Member of ACM and the International Society for Computers and their Applications (ISCA).