A Multi-Scale Decomposition MLP-Mixer for Time Series Analysis

Shuhan Zhong
Department of Computer Science and Engineering
The Hong Kong University of Science and Technology
szhongaj@cse.ust.hk

Sizhe Song
Department of Computer Science and Engineering
The Hong Kong University of Science and Technology
ssongad@cse.ust.hk

Weipeng Zhuo
Guangdong Provincial Key Laboratory IRADS and Department of Computer Science
BNU-HKBU United International College
weipengzhuo@uic.edu.cn

Guanyao Li
Guangdong Enterprise Key Laboratory for Urban Sensing, Monitoring and Early Warning
Guangzhou Urban Planning and Design Survey Research Institute
gyli@gzpi.com.cn

Yang Liu
Guangdong Enterprise Key Laboratory for Urban Sensing, Monitoring and Early Warning
Guangzhou Urban Planning and Design Survey Research Institute
liuyang@gzpi.com.cn

S.-H. Gary Chan
Department of Computer Science and Engineering
The Hong Kong University of Science and Technology
gchan@cse.ust.hk

ABSTRACT

Time series data, including univariate and multivariate ones, are characterized by unique composition and complex multi-scale temporal variations. They often require special consideration of decomposition and multi-scale modeling to analyze. Existing deep learning methods on this best fit to univariate time series only, and have not sufficiently considered sub-series modeling and decomposition completeness. To address these challenges, we propose MSD-Mixer, a **M**ulti-**S**cale **D**ecomposition MLP-**Mixer**, which learns to explicitly decompose and represent the input time series in its different layers. To handle the multi-scale temporal patterns and multivariate dependencies, we propose a novel temporal patching approach to model the time series as multi-scale patches, and employ MLPs to capture intra- and inter-patch variations and channel-wise correlations. In addition, we propose a novel loss function to constrain both the mean and the autocorrelation of the decomposition residual for better decomposition completeness. Through extensive experiments on various real-world datasets for five common time series analysis tasks, we demonstrate that MSD-Mixer consistently and significantly outperforms other state-of-the-art algorithms with better efficiency.

PVLDB Reference Format:
Shuhan Zhong, Sizhe Song, Weipeng Zhuo, Guanyao Li, Yang Liu, and S.-H. Gary Chan. A Multi-Scale Decomposition MLP-Mixer for Time Series Analysis. PVLDB, 17(7): 1723 - 1736, 2024.
doi:10.14778/3654621.3654637

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/zshhans/MSD-Mixer.

Proceedings of the VLDB Endowment, Vol. 17, No. 7 ISSN 2150-8097.
doi:10.14778/3654621.3654637

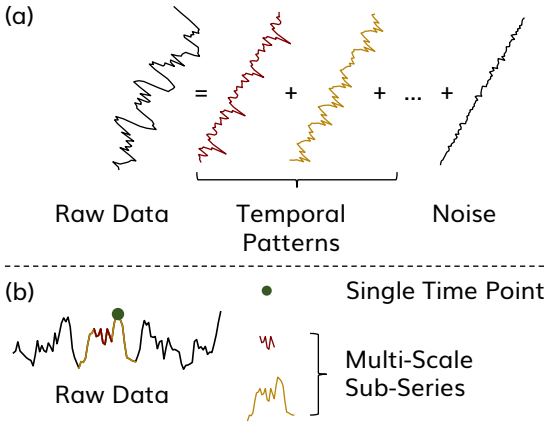

Figure 1: (a) Decomposition of time series. (b) Comparison of single time points and multi-scale sub-series.

1 INTRODUCTION

A time series is a sequence of data points indexed in time order. It typically consists of successive numerical observations obtained in a fixed time interval. In multivariate time series, each observation contains more than one variable, which forms the "channel" dimension. With the fast development of sensing and data storage technologies, time series data is now becoming omnipresent in our lives, from weather conditions and urban traffic flow to personal health data monitored by smart wearable devices. The analysis of time series data, such as forecasting [5, 10, 15, 40], missing data imputation [4, 21, 30], anomaly detection [33–36], and classification [7, 8, 25], is therefore facilitating more and more real-world applications, and attracts increasing research interest from both the academia and the industry.

In contrast to images and natural languages, time series data is characterized by its special composition and complex temporal patterns or correlations. Specifically, each data point in a time series is actually a superposition of various underlying temporal patterns

plus noise at that time (Figure 1(a)). To better model and analyze the data, it is hence important to decompose the data into disentangled components corresponding to different temporal patterns [18, 45]. Furthermore, time series data carries the semantic information of the temporal patterns in local consecutive data points termed *sub-series*, rather than individual data point [31, 42, 48, 51]. The temporal patterns are usually in multiple timescales, thus making it important to extract sub-series features and model their changes in multiple time scales (Figure 1(b)). To make the problem even more complex, multivariate time series may involve intricate correlations between different channels [16, 55]. These characteristics all make time series analysis a challenging problem.

Recent approaches for time series analysis take advantage of the strong expressiveness of deep learning, especially the Transformer architecture [41], and have achieved significant performance in various tasks [22, 49, 56, 57]. However, it is recently pointed out that the Transformer is actually no better than multi-layer perceptrons (MLPs) in time series modeling, since Transformers are designed to embed information of single time points and model their pair-wise correlations, while time series data carries information in its multi-scale sub-series instead of single time points. At the same time, most approaches consider no or merely simple decomposition of temporal patterns [49, 57], which makes it hard for them to handle various intricate temporal patterns in time series data [3, 44]. Considering the composition of time series, some methods adopt a deep learning architecture that learns to decompose temporal patterns from the input for forecasting [6, 32, 47]. However, without considering inter-channel dependency, [32] and [6] are best applicable to univariate rather than multivariate time series. Furthermore, the aforementioned works did not investigate into the residual of the decomposition, which may lead to *incomplete* decomposition, i.e., meaningful temporal patterns may be left in the residual and not utilized by the model.

In this work, we address these problems by proposing MSD-Mixer, a novel **M**ulti-**S**cale **D**ecomposition MLP-**Mixer** to analyze both univariate and multivariate time series. MSD-Mixer is based exclusively on MLPs, which is simple but effective for time series modeling. To account for the special composition of time series data, MSD-Mixer explicitly decomposes the input time series into different components by generating their latent representations in different layers, and accomplishes the analysis task based on such representations. In MSD-Mixer we propose a novel *multi-scale temporal patching* approach that divides the input time series into non-overlapping patches along the temporal dimension in each layer for sub-series modeling. Different layers have different patch sizes such that they can focus on different time scales. To better model multi-scale temporal patterns and inter-channel dependencies, MSD-Mixer employs MLPs along different dimensions to learn intra- and inter-patch variations as well as channel-wise correlations. In addition, to enhance the learning of the decomposition process, we propose a novel loss function to constrain both the mean and the autocorrelation of the decomposition residual during training. Used together with the loss function from the target analysis task, it helps MSD-Mixer to decompose the time series data more thoroughly for better analysis results.

Empowered by the above-mentioned decomposition and multi-scale modeling features, MSD-Mixer distinguishes itself as a *task-general* backbone that can be adapted for various time series analysis tasks. Through extensive experiments on various real-world datasets, we demonstrate that MSD-Mixer consistently outperforms both task-general and task-specific state-of-the-art approaches by a wide margin across five common time series analysis tasks, namely long-term forecasting (up to 9.8% in MSE), short-term forecasting (up to 5.6% in OWA), imputation (up to 46.1% in MSE), anomaly detection (up to 33.1% in F1-score) and classification (up to 36.3% in Mean Rank).

To summarize, we make the following contributions in this paper:

- *A novel task-general backbone MSD-Mixer* that is well designed to analyze time series data by learning to explicitly decompose and represent the temporal patterns.
- *A multi-scale temporal patching approach* in MSD-Mixer that facilitates modeling the time series data as multi-scale patches with MLPs, to better account for the multi-scale temporal patterns in the data.
- *A residual loss* for MSD-Mixer to constrain both the mean and the autocorrelation of the decomposition residual for better decomposition completeness.
- *Extensive experiments* on 26 datasets for five common time series analysis tasks to validate the effectiveness of MSD-Mixer.

The remainder of this paper is organized as follows: We first review related works in Section 2, and elaborate on MSD-Mixer and its modules in Section 3. We show the experimental results in Section 4 and conclude in Section 5.

2 RELATED WORKS

2.1 Classical Methods

As one of the fundamental data modalities, time series has been well studied for long in various science and engineering domains that rely on temporal measurements, and has been mostly discussed for forecasting [14, 18]. Regarding the special composition and the complex temporal patterns of time series data, early approaches employ manually designed rules or function models to decompose the time series data, such that the temporal patterns can be disentangled and modeled separately [9, 11, 13, 17, 39, 43, 46]. The decomposition usually consists of several components representing different temporal patterns, plus a residual which is supposed to be noise with no useful information. These approaches usually require considerable domain knowledge and manual effort to be adapted to specific domains, and are less expressive and scalable considering nowadays large multivariate time series datasets with complex temporal patterns and channel-wise correlations.

2.2 Deep Models without Decomposition

In recent years, deep learning has been widely applied in time series analysis for its strong expressiveness and scalability on large and complex datasets. The deep learning based approaches either apply MLP [53], convolutional neural network (CNN) [19, 27, 48], recurrent neural network (RNN) [2], Transformer [7, 31, 50], or their

combination [4, 20, 42] to model the time series data for specific tasks. Among them, RNNs have been pointed out for their deficiency in modeling long sequences which are common in time series analysis tasks. Its difficulty with parallelized training also greatly affects their efficiency. CNNs, instead, usually require special attention to make the trade-off between the number of layers and the effective receptive fields, or consider the design of dilation or pooling rate when applied for time series analysis [24, 26]. Transformer-based models are taking the lead in many time series analysis tasks due to the powerful capability of attention mechanism to capture long-sequence dependencies. Many works have been done to further improve the efficiency [22, 56] and effectiveness [37, 49, 57] of the Transformer for time series data. However, it is recently shown that the Transformer, which relies on point embedding and their pair-wise correlations, is not a promising choice for time series data, since the semantic information is embedded in the sub-series level variations instead of single time points [51]. In light of this, PatchTST [31] and TimesNet [48] are proposed to combine patch modeling with Transformer and CNN for time series data. Despite the above-mentioned achievements, most deep learning based approaches do not consider the decomposition of temporal patterns, or only simply consider the decomposition of very limited types and number of components [49, 57], which makes it hard for them to deal with multiple intricate temporal patterns in time series data [3].

2.3 Deep Models with Decomposition

By combining deep learning with decomposition, N-BEATS [32], N-HiTS [6], and ETSformer [47] show satisfactory results in time series forecasting. However, N-BEATS and N-HiTS do not consider the inter-channel correlation, which has been shown critical in multivariate time series analysis tasks. In addition, they are based on plain MLP on the temporal dimension while ETSformer [47] is based on self-attention for temporal modeling, all of which do not take into account the sub-series level features. Furthermore, they simply ignore the residual of the decomposition, which may lead to incomplete decomposition that meaningful temporal patterns can be left in the residual and not utilized. Besides, all these schemes have only been tested on the forecasting task, leaving other analysis tasks such as imputation, anomaly detection, and classification unexplored.

In comparison, we propose MSD-Mixer that advances them with *multi-scale temporal patching* and multi-dimensional MLP mixing for multi-scale sub-series and inter-channel modeling. Meanwhile, we propose a *residual loss* for better completeness of the decomposition process in MSD-Mixer. Furthermore, we experiment MSD-Mixer and compare it with state-of-the-art methods on various datasets across the forecasting, imputation, anomaly detection, and classification tasks to show its superior modeling ability over other methods.

3 MSD-MIXER

In this section, we first formally introduce the definition of general time series analysis problems and time series decomposition in 3.1. Then, we overview the architecture and workflow of our proposed MSD-Mixer in Section 3.2, followed by elaboration on the key designs in MSD-Mixer in Sections 3.3 to 3.5, then summarize in Section 3.6.

3.1 Problem Settings

3.1.1 Time Series Analysis. In this paper, we summarize the general learning-based time series analysis tasks, including but not limited to *forecasting*, *imputation*, *anomaly detection*, and *classification*, as the following problem: Given a dataset $\mathcal{D}$ containing sample pairs $(\boldsymbol{X}, \boldsymbol{Y})$, where $\boldsymbol{X} \in \mathbb{R}^{C \times L}$ denotes the input multivariate time series with C channels and L time steps, and $\boldsymbol{Y}$ denotes the label whose form is subject to the target time series analysis task. The time series analysis problem is to obtain an optimal function $\mathcal{F}(\cdot)$ on the dataset that maps the input $\boldsymbol{X}$ to its corresponding label as $\hat{\boldsymbol{Y}} = \mathcal{F}(\boldsymbol{X})$, such that the difference between the prediction $\hat{\boldsymbol{Y}}$ and the ground truth $\boldsymbol{Y}$ is minimized.

Different target time series analysis tasks have different forms of $\boldsymbol{Y}$ and $\hat{\boldsymbol{Y}}$. For example, $\boldsymbol{Y} \in \mathbb{R}^{C \times H}$ in a forecasting task with horizon size H, and $\boldsymbol{Y} \in \mathbb{R}^{M}$ in a classification task with M target classes. Different tasks also employ different metrics to measure the difference between $\boldsymbol{Y}$ and $\hat{\boldsymbol{Y}}$ for computing the task-specific loss, e.g., cross-entropy loss used in classification and mean square error loss used in forecasting.

3.1.2 Time Series Analysis with Decomposition. Existing deep learning approaches for time series generally learn to directly represent the input $\boldsymbol{X}$ and map it to the output, which makes it hard for them to handle multiple intricate temporal patterns. Instead, we consider the decomposition of $\boldsymbol{X}$ as:

$$\boldsymbol{X} = \sum_{i=1}^{k} \boldsymbol{S}_i + \boldsymbol{R}, \tag{1}$$

where $\boldsymbol{S}_i, \boldsymbol{R} \in \mathbb{R}^{C \times L}$ denote the i-th component ($i = 1, ..., k$) and the residual, respectively. Suppose each component $\boldsymbol{S}_i$ has a lower-dimensional representation $\boldsymbol{E}_i$, then the target $\mathcal{F}$ can be divided and conquered by a set of functions $f_i(\cdot)$ of the component representations:

$$\hat{\boldsymbol{Y}} = \mathcal{F}(\boldsymbol{X}) = \sum_{i=1}^{k} f_i(\boldsymbol{E}_i). \tag{2}$$

3.2 MSD-Mixer Overview

Figure 2 shows the overall architecture of MSD-Mixer. MSD-Mixer comprises a stack of k layers, and learns to hierarchically decompose the input $\boldsymbol{X}$ into k components $\{\boldsymbol{S}_1, ..., \boldsymbol{S}_k\}$ by generating their lower-dimensional representations $\{\boldsymbol{E}_1, ..., \boldsymbol{E}_k\}$ in the corresponding layers. The number of layers and components k is a hyperparameter in MSD-Mixer which should be determined according to the properties of the dataset. Here we define $\boldsymbol{Z}_0 = \boldsymbol{X}$, and

$$\boldsymbol{Z}_i = \boldsymbol{X} - \sum_{j=1}^{i} \boldsymbol{S}_j, (i = 1, ..., k), \tag{3}$$

such that $\boldsymbol{Z}_i$ specifies the remaining part after the first i components has been decomposed from the input $\boldsymbol{X}$, and we have

$$\boldsymbol{Z}_i = \boldsymbol{Z}_{i-1} - \boldsymbol{S}_i. \tag{4}$$

As is shown in Figure 2, the i-th layer of MSD-Mixer takes the remaining part $\boldsymbol{Z}_{i-1}$ from the previous layer as input, and learns

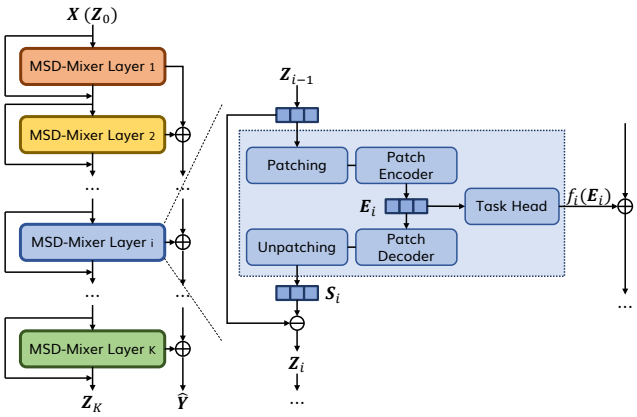

Figure 2: MSD-Mixer overview.

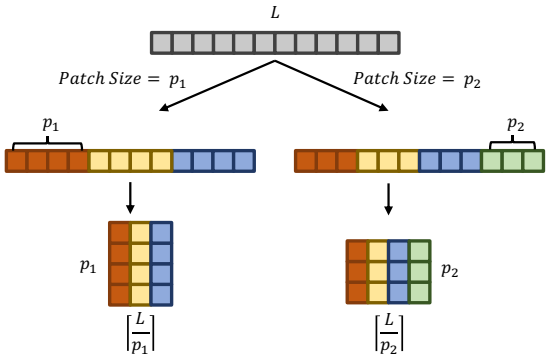

Figure 3: Examples of multi-scale temporal patching. The channel dimension is omitted for simplicity.

to represent Z_{i-1} with a lower dimensional representation E_i. The represented part is the i-th component S_i. More specifically, within each layer, Z_{i-1} is first patched in the Patching module, and then fed into the Patch Encoder module to generate the representation of i-th components as $E_i = g_i(Z_{i-1})$. The Patch Decoder module then reconstructs S_i from E_i, and the Unpatching module unpatches it to the original dimensionality. After that, S_i is subtracted from Z_{i-1} to obtain Z_i. Z_i is fed to the next layer as input for further decomposition, and E_i is projected by linear layers and summed to obtain Y following Equation 2.

3.3 Multi-Scale Temporal Patching

Considering the importance of multi-scale sub-series modeling for time series analysis, we introduce *multi-scale temporal patching* in MSD-Mixer such that different layers can focus on different sub-series features. Each layer of MSD-Mixer has a predefined patch size p_i, which is a hyperparameter to be determined or tuned for specific datasets.

We depict the patching process in Figure 3. To transform an input time series with C channels and L time steps into patches with patch size p, we first pad the time series with zeros at the beginning of the time series to ensure the length is divisible by p, and then segment the time series along the temporal dimension into non-overlapping patches with stride p. We then permute the data to create a new dimension for the patches, resulting in a high dimensional tensor of $C \times L' \times p$, where $L' = \lceil L/p \rceil$

3.4 Patch Encoder and Decoder

The Patch Encoder and Decoder modules are based exclusively on MLPs along different dimensions for feature extraction. We show the design of each MLP block in Figure 4(a), which simply consists of two fully connected layers, a GELU nonlinearity layer, and a DropPath layer [23], together with a residual connection that adds the input to the output. We use the following three types of MLP blocks in Patch Encoder and Decoder modules:

- *The channel-wise MLP block* allows communication between different channels, to capture inter-channel correlations.
- *The inter-patch MLP block* allows communication between different patches, to capture global contexts.
- *The intra-patch MLP block* allows communication between different time steps within a patch, to capture sub-series level variations.

As shown in Figure 4(b) the Patch Encoder module consists of a channel-wise MLP block, an inter-patch MLP block, an intra-patch MLP block, and a linear layer in order to produce the component representation E_i from the patched Z_{i-1}. The Patch Decoder module (4(c)) consists of the same number and type of blocks as the Patch Encoder module, but in a reversed order to reconstruct S_i from E_i.

3.5 Residual Loss

The residual of the decomposition is useful in checking whether the information in the data has been adequately captured into the components. An ideal decomposition should yield a residual with the following two properties:

- It should have zero mean. If the residual has a mean other than zero, then there can be biases left in the residual.
- It should contain no autocorrelation. The stronger the autocorrelation is in the residual, the more likely there can be temporal patterns such as trends and periodic information left in the residual.

A residual that does not satisfy these properties indicates the incompleteness of the decomposition, which means useful information has not been fully accounted for by the components.

By jointly considering the two properties of the decomposition residual, we propose a novel *residual loss* to train MSD-Mixer such that it can learn to achieve better decomposition completeness. The *residual loss* minimizes both the mean and the autocorrelation of the residual. The autocorrelation of the residual can be measured by its autocorrelation coefficients which are defined in $[-1, 1]$. A larger absolute value of the coefficient indicates a stronger correlation. It is usually expected that the autocorrelation coefficients of a successful decomposition should lie within $\pm 2/\sqrt{L}$ where L is the series length. In a MSD-Mixer with k layers, Z_k output by the last layer specifies the residual of the decomposition. We first compute the autocorrelation coefficient matrix $A = \{a_{i,j}\} \in \mathbb{R}^{C \times (L-1)}$ of

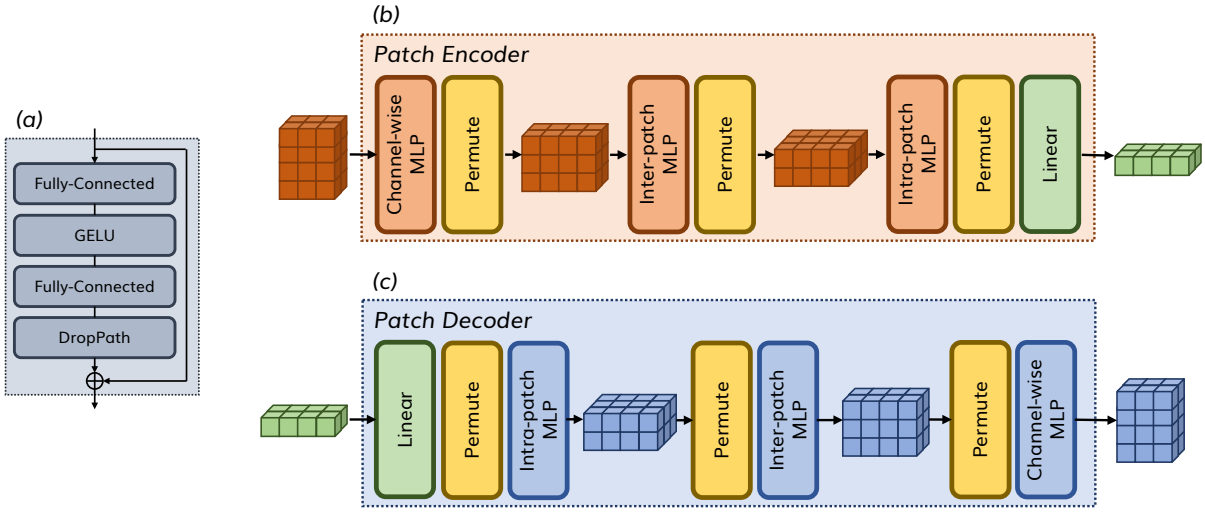

Figure 4: (a) MLP block. (b) Patch Encoder. (c) Patch Decoder.

Z_k as:

$$a_{i,j} = \frac{\sum_{t=j+1}^{L}(z_{i,t} - \bar{z}_i)(z_{i,t-j} - \bar{z}_i)}{\sum_{t=1}^{L}(z_{i,t} - \bar{z}_i)^2}, \tag{5}$$

where $z_{i,j}$ is the i-th channel and j-th time step of Z_k. We then define the *residual loss* by:

$$\mathcal{L}_r = \frac{\sum_{i,j} z_{i,j}^2}{C \times L} + \frac{\sum_{i,j}(\text{ReLU}(|a_{i,j}| - \alpha/\sqrt{L}))^2}{C \times (L-1)}. \tag{6}$$

The first term of $\mathcal{L}_r$ minimizes the mean of the residual. And the right term imposes a constraint on the autocorrelation coefficients of the residual, where α is a hyperparameter controlling the maximum tolerance of the autocorrelation coefficients. We finally train MSD-Mixer by simultaneously optimizing the weighted sum of the task-specific loss and the *residual loss*:

$$\mathcal{L} = \mathcal{L}_t + \lambda \mathcal{L}_r. \tag{7}$$

3.6 Summary

We summarize the overall training process of MSD-Mixer in Algorithm 1. Given a training dataset $\mathcal{D}$ with time series data and corresponding labels as $\mathcal{D} = \{(\boldsymbol{X}, \boldsymbol{Y})\}$, we train MSD-Mixer based on the dataset until the loss converges. During the process, we first sample $(\boldsymbol{X}, \boldsymbol{Y})$ from $\mathcal{D}$ (line 3), and then initialize $\boldsymbol{Z}_0$ to be $\boldsymbol{X}$ (line 4). After that, for each layer $i \in [1, k]$, we patch $\boldsymbol{Z}_{i-1}$ using the patch size p_i (line 6). The patched input is then fed into MLP-Mixer to generate $\boldsymbol{E}_i$ (line 7). This learned representation $\boldsymbol{E}_i$ is eventually decoded and unpatched to reconstruct the input in each layer (lines 8 – 10). The labels are predicted with loss computed for back propagation (lines 11 – 13). This whole process is repeated until convergence to return the trained MSD-Mixer model (line 14).

Algorithm 1: Training of MSD-Mixer.

Input: Training set $\mathcal{D} = \{(\boldsymbol{X}, \boldsymbol{Y})\}$, number of layers k, patch size for each layer $p_1, \cdots, p_k$.
Output: Trained MSD-Mixer.
1 Initialize MSD-Mixer with k layers of patch size $p_1, \cdots, p_k$.
2 **repeat**
3 Sample $(\boldsymbol{X}, \boldsymbol{Y})$ from $\mathcal{D}$.
4 $\boldsymbol{Z}_0 = \boldsymbol{X}$.
5 **for** $i = 1, 2, \ldots, k$ **do**
6 Patch $\boldsymbol{Z}_{i-1}$ with p_i.
7 Compute $\boldsymbol{E}_i = g_i(\boldsymbol{Z}_{i-1})$ with i-th layer's Patch Encoder.
8 Compute $\boldsymbol{S}_i = h_i(\boldsymbol{E}_i)$ with i-th layer's Patch Decoder.
9 Unpatch $\boldsymbol{S}_i$ with p_i.
10 $\boldsymbol{Z}_i = \boldsymbol{Z}_{i-1} - \boldsymbol{S}_i$.
  end
11 Compute $\hat{\boldsymbol{Y}} = \sum_{i=1}^{k} f_i(\boldsymbol{E}_i)$ with Task Head modules.
12 Computer loss according to Equation 5–7.
13 Back propagation.
  until *convergence*;
14 **return** *Trained MSD-Mixer.*

4 ILLUSTRATIVE EXPERIMENTAL RESULTS

In this section, we first overview the experiment setup and key results in Section 4.1. Then, in Section 4.2 to 4.5, we discuss in detail the experiments and results of different time series analysis tasks, followed by ablation studies on our proposed modules in Section 4.7. Lastly, we study the efficiency of MSD-Mixer by comparing the number of model parameters and training time consumption of different approaches in Section 4.8, and empirically analyze the decomposition of MSD-Mixer by example cases in Section 4.9.

4.1 Overview

In order to validate the modeling ability of MSD-Mixer, we conduct extensive experiments on a wide range of well-adopted benchmark datasets across *five* most common time series analysis tasks, including long-term forecasting, short-term forecasting, imputation, anomaly detection, and classification. The tasks and benchmark datasets are of different characteristics that we leverage them to

Table 1: Summary of Tasks, Datasets and Metrics

Tasks	Datasets [48]	Metrics
Long-Term Forecasting	ETT (4 subsets), ECL, Weather, Traffic, Exchange	Mean Square Error (MSE), Mean Absolute Error (MAE)
Short-Term Forecasting	M4 (6 subsets)	SMAPE, MASE, OWA [29]
Imputation	ETT (4 subsets), ECL, Weather	MSE, MAE
Anomaly Detection	SMD, MSL, SMAP, SWaT, PSM	F1-Score
Classification	UEA (10 subsets [7])	Accuracy

investigate on different aspects of MSD-Mixer. Table 1 summarizes the five tasks, datasets, and evaluation metrics we use in the experiments.

4.1.1 Baselines. We compare our proposed MSD-Mixer with state-of-the-art *task-general* approaches that can serve as general solutions to various time series analysis tasks, as well as *task-specific* approaches that are proposed for specific time series analysis tasks.

For *task-general* baselines, we select approaches that cover mainstream deep learning architectures, including CNN, Transformer, and MLP. Among them, TimesNet [48] and PatchTST [31] combine sub-series modeling with CNN and Transformer, respectively. Crossformer [55] is a Transformer-based approach with special designs for channel-wise correlations. ETSformer [47] is among the first to leverage Transformer for decomposition. NST [28] and FEDformer [57] are also Transformer-based approaches that consider the stationarity and frequency domain features of time series data. DLinear [51] and LightTS [53] are MLP-based light-weight approaches. LightTS further considers channel-wise and local-global features in the data.

In addition, we introduce and compare with extra state-of-the-art task-specific approaches for tasks in the corresponding sections, including long-term forecasting (Section 4.2), short-term forecasting (Section 4.3), anomaly detection (Section 4.5), and classification (Section 4.6).

4.1.2 Implementation. We follow the implementation of baselines in [48]. All models including MSD-Mixer and the baselines are implemented with PyTorch and trained with a single NVIDIA GeForce RTX 4090 GPU with 24 GB memory. We search the best number of layers from 4 to 6, and dimensions of the model from 64 to 512 for MSD-Mixer with different datasets. We set the patch size in MSD-Mixer by considering the series length and sampling interval of the dataset. For instance, the ETTm1 dataset provides two years' data of electricity transformer temperature from two separate counties in China. It contains time series with a length of 96 samples and the sampling interval is 15 minutes, i.e., two samples are collected 15 minutes apart. To model the time series efficiently, we use five layers in MSD-Mixer with patch sizes for each layer as {24, 12, 4, 2, 1}, which correspond to the sub-series of 6 hours (15min $\times$ 24), 3 hours, 1 hour, 30 minutes, and 15 minutes, respectively.

4.1.3 Overall Performance. Table 2 summarizes the overall performance of task-general schemes. As shown in the table, our proposed MSD-Mixer outperforms other state-of-the-art baselines significantly in all the benchmarks across the five tasks. Benefiting from the special design of the decomposition, multi-scale temporal patching, and the *residual loss* components, MSD-Mixer is far ahead of its CNN-based (TimesNet), Transformer-based (PatchTST, Crossformer, ETSformer, NST, FEDformer) and MLP-based (DLinear, LightTS) task-general counterparts by a large margin, demonstrating its great and comprehensive modeling ability for time series analysis.

4.2 Long-Term Forecasting

4.2.1 Task Settings. Long-term time series forecasting has always been a primary goal of time series analysis. Characterized by its extraordinary long horizon as output, the forecasting testifies the long-range modeling ability of different schemes. In this task, we evaluate the approaches on eight popular real-world datasets across energy, transportation, weather, and finance domains. Each dataset contains one long multivariate time series. Information of the datasets is summarized in Table 3.

We include Scaleformer [37] as a task-specific baseline in this experiment. Scaleformer is the latest Transformer-based approach with impressive performance in long-term forecasting by iteratively refining the forecasting result at multiple time scales.

In this task, we denote the model input as $\boldsymbol{X} \in \mathbb{R}^{C \times L}$, and the model output as $\boldsymbol{Y} \in \mathbb{R}^{C \times H}$, where C, L, and H are the number of channels, the length of input time series, and the forecasting horizon, respectively. We fix the length of input time series as 96, and then train and test each scheme with four forecasting horizons, i.e., 96, 192, 336, and 720, to evaluate the performance of different schemes under different forecasting horizons. We obtain input and output sample pairs with a sliding window over the long time series. We use the mean squared error (MSE) between the ground truth $\boldsymbol{Y}$ and the prediction $\hat{\boldsymbol{Y}}$ as the loss function to train the models, and report both MSE and mean absolute error (MAE) for performance evaluation.

4.2.2 Result Analysis. As shown in Table 4, MSD-Mixer achieves the best performance on most datasets, including different forecasting horizon settings, with 45 first and 9 second places out of 64 benchmarks in total. Furthermore, on most benchmarks, MSD-Mixer outperforms the second place by a significant margin. From the results, we can see that although ETSformer adopts the decomposition design, it is still based on the pair-wise self-attention for temporal feature extraction, which has been shown to be ineffective for sub-series modeling. Therefore it struggles to perform well. TimesNet and PatchTST consider sub-series modeling in their design. Thus these two schemes are shown to be the strongest baselines. Compared with them, MSD-Mixer still outperforms significantly by combining decomposition with multi-scale sub-series modeling in the design. We believe that the outstanding performance of MSD-Mixer fully demonstrates the efficacy of our multi-scale decomposition in time series modeling.

Table 2: Overall performance comparison with task-general baselines. Each number of a scheme in the table represents in how many benchmarks the scheme performs the best. The best results are in bold and the second bests are underlined.

Task	# of Benchmarks	MSD-Mixer (Ours)	PatchTST (2023)	Crossformer (2023)	TimesNet (2023)	DLinear (2023)	ETSformer (2022)	NST (2022)	FEDformer (2022)	LightTS (2022)
Long-Term Forecasting	64	**45**	7	8	1	3	2	0	1	1
Short-Term Forecasting	15	**15**	0	0	0	0	0	0	0	0
Imputation	48	**45**	0	0	9	0	0	0	0	0
Anomaly Detection	5	**4**	0	0	1	0	0	0	0	0
Classification	10	**5**	0	0	0	0	0	0	0	0
Total	142	**114**	7	8	11	3	2	0	1	1

Table 3: Statistics of datasets for long-term forecasting.

Dataset	Dim	Total Timesteps	Frequency
ETTm1, ETTm2	7	69680	15 mins
ETTh1, ETTh2	7	17420	1 hour
ECL	321	26304	10 mins
Traffic	862	17544	1 hour
Weather	21	52696	10 mins
Exchange	8	7588	1 day

4.3 Short-Term Forecasting

4.3.1 Task Settings. In this task, we adopt the dataset and performance measures from the well-known M4 competition [29], which focuses on the short-term forecasting of univariate time series. The dataset contains 100,000 sequences of data in total, which are further divided into 6 subsets by sampling intervals including yearly, quarterly, monthly, weekly, daily, and hourly. More information of the dataset is summarized in Table 5. Each subset contains real-life time series data from different domains, such as economics, finance, industry, demographics, etc. The analysis requires the forecasting models to learn the general temporal patterns from samples across diverse domains.

We include N-BEATS [32] and N-HiTS [6] as task-specific baselines. N-BEATS is an MLP-based model for time series decomposition. N-HiTS further enhances N-BEATS with multi-scale modeling by introducing down-sampling and interpolation. They are proposed for univariate time series forecasting only.

We evaluate with the symmetric mean absolute percentage error (SMAPE) and the mean absolute scaled error (MASE), defined as

$$\begin{aligned} \text{SMAPE} &= \frac{200}{H} \sum_{i=1}^{H} \frac{|y_i - \hat{y}_i|}{|y_i| + |\hat{y}_i|}, \\ \text{MASE} &= \frac{1}{H} \sum_{i=1}^{H} \frac{|y_i - \hat{y}_i|}{\frac{1}{H-m} \sum_{j=m+1}^{H} |y_j - y_{h-m}|}, \end{aligned} \tag{8}$$

where y_i and $\hat{y}_i$ are the ground truth and forecasting of the i-th time step in H total future time steps, m is the periodicity of the data. Intuitively, SMAPE measures the relative errors of the forecasting result. MASE is the MAE of the forecast values divided by the MAE of the in-sample one-step naive forecast. We also use the overall weighted average (OWA) of SMAPE and MASE as an evaluation metric, which is defined by the M4 competition [29].

4.3.2 Result Analysis. As shown in Table 6, MSD-Mixer again leads the board with top-1 performance in every benchmark, which demonstrates the excellent capability of MSD-Mixer on modeling short and univariate time series. Each time point in a univariate time series is a single scalar, which makes it even harder for Transformer based models to attain meaningful attention scores. Therefore, Transformer based approaches, which rely on pair-wise correlations have inferior performance in this task, especially ETSformer. Among the baselines, N-HiTS and N-BEATS are based on decomposition, and they show satisfactory performance in this task, which validates the effectiveness of decomposition for time series modeling. MSD-Mixer further advances N-BEATS and N-HiTS with *multi-scale temporal patching* and mixing, as well as the *residual loss* for better extraction of temporal patterns, which facilitates MSD-Mixer with stronger modeling ability than them, and helps MSD-Mixer achieve the best performance.

4.4 Imputation

4.4.1 Task Settings. Missing values are common in real-world continuous data systems that collect time series data from various sources. A single missing data in a time series can break down the whole downstream application since most analysis methods assume complete data, which makes missing data imputation critical for time series analysis. In this task, we experiment on the ETT, ECL, and Weather datasets which are summarized in Table 3.

In the experiments, we first obtain the data samples $\boldsymbol{X} \in \mathbb{R}^{C \times L}$ with a sliding window, whose size L is set as 96. We then create missing values $\boldsymbol{X}_{mask}$ in the data by randomly masking the $\boldsymbol{X}$ with zeros. We use $\boldsymbol{X}_{mask}$ as model input and the unmasked data $\boldsymbol{X}$ as ground truth for training and testing. To evaluate the performance under different ratios of missing data, for each dataset we train and evaluate each model with four missing data ratios, i.e., 12.5%, 25%, 37.5%, and 50%. We use the MSE between the ground truths and the predictions at the masked positions as the loss function to train the models, and report both MSE and MAE for performance evaluation. In particular, due to the missing data in the input, it is not feasible to compute the autocorrelation of the residual for MSD-Mixer. Therefore, we only compute the first term of the residual in the *residual loss* (Equation 6).

4.4.2 Result Analysis. Results in Table 7 show that MSD-Mixer also achieves the best performance on most datasets, as well as on different missing data ratios, with 45 first place out of 48 benchmarks in total. This task requires the model to learn correct temporal patterns from the data with missing values masked as zeros, which is challenging for most models. We observe that MSD-Mixer and TimesNet, which shows good performance in this task, have

Table 4: Long-term forecasting results. The best results are in bold and the second bests are underlined. (*Task-specific baseline.)

Models		MSD-Mixer (Ours)		Scaleformer* (2023)		PatchTST (2023)		Crossformer (2023)		TimesNet (2023)		DLinear (2023)		ETSformer (2022)		NST (2022)		FEDformer (2022)		LightTS (2022)	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTm1	96	**0.304**	**0.351**	0.392	0.415	0.334	0.372	0.316	0.373	0.338	0.375	0.345	0.372	0.375	0.398	0.386	0.398	0.379	0.419	0.374	0.400
	192	**0.344**	**0.375**	0.437	0.451	0.378	0.394	0.377	0.411	0.374	0.387	0.380	0.389	0.408	0.410	0.459	0.444	0.426	0.441	0.400	0.407
	336	**0.370**	**0.395**	0.499	0.478	0.406	0.414	0.431	0.442	0.410	0.411	0.413	0.413	0.435	0.428	0.495	0.464	0.445	0.459	0.438	0.438
	720	**0.427**	**0.428**	0.584	0.536	0.462	0.445	0.600	0.547	0.478	0.450	0.474	0.453	0.499	0.462	0.585	0.516	0.543	0.490	0.527	0.502
ETTm2	96	**0.169**	**0.259**	0.182	0.276	0.175	**0.259**	0.236	0.281	0.187	0.267	0.193	0.292	0.189	0.280	0.192	0.274	0.203	0.287	0.209	0.308
	192	**0.232**	**0.300**	0.252	0.319	0.240	0.302	0.294	0.349	0.249	0.309	0.284	0.362	0.253	0.319	0.280	0.339	0.269	0.328	0.311	0.382
	336	**0.292**	**0.337**	0.335	0.372	0.302	0.342	0.353	0.382	0.321	0.351	0.369	0.427	0.314	0.357	0.334	0.361	0.325	0.366	0.442	0.466
	720	**0.392**	0.398	0.460	0.446	0.399	**0.397**	0.588	0.547	0.408	0.403	0.554	0.522	0.414	0.413	0.417	0.413	0.421	0.415	0.675	0.587
ETTh1	96	0.377	**0.391**	0.404	0.441	0.444	0.438	0.386	0.429	0.384	0.402	0.386	0.400	0.494	0.479	0.513	0.491	**0.376**	0.419	0.424	0.432
	192	0.427	**0.422**	0.438	0.461	0.488	0.463	**0.419**	0.444	0.436	0.429	0.437	0.432	0.538	0.504	0.534	0.504	0.420	0.448	0.475	0.462
	336	0.469	**0.443**	0.464	0.477	0.525	0.484	**0.440**	0.461	0.491	0.469	0.481	0.459	0.574	0.521	0.588	0.535	0.459	0.465	0.518	0.488
	720	**0.485**	**0.475**	0.507	0.516	0.532	0.510	0.519	0.524	0.521	0.500	0.519	0.516	0.562	0.535	0.643	0.616	0.506	0.507	0.547	0.533
ETTh2	96	**0.284**	**0.345**	0.335	0.385	0.312	0.358	0.401	0.464	0.340	0.374	0.333	0.387	0.340	0.391	0.476	0.458	0.358	0.397	0.397	0.437
	192	**0.362**	**0.392**	0.455	0.451	0.401	0.410	0.483	0.479	0.402	0.414	0.477	0.476	0.430	0.439	0.512	0.493	0.429	0.439	0.520	0.504
	336	**0.399**	**0.428**	0.477	0.479	0.437	0.442	0.498	0.510	0.452	0.452	0.594	0.541	0.485	0.479	0.552	0.551	0.496	0.487	0.626	0.559
	720	**0.426**	0.457	0.467	0.490	0.442	**0.454**	0.556	0.527	0.462	0.468	0.831	0.657	0.500	0.497	0.562	0.560	0.463	0.474	0.863	0.672
ECL	96	**0.152**	**0.254**	0.182	0.297	0.211	0.312	0.187	0.283	0.168	0.272	0.197	0.282	0.187	0.304	0.169	0.273	0.193	0.308	0.207	0.307
	192	**0.165**	**0.263**	0.188	0.300	0.214	0.313	0.258	0.330	0.184	0.289	0.196	0.285	0.199	0.315	0.182	0.286	0.201	0.315	0.213	0.316
	336	**0.173**	**0.273**	0.210	0.324	0.230	0.328	0.323	0.369	0.198	0.300	0.209	0.301	0.212	0.329	0.200	0.304	0.214	0.329	0.230	0.333
	720	**0.201**	**0.299**	0.232	0.339	0.272	0.359	0.404	0.423	0.220	0.320	0.245	0.333	0.233	0.345	0.222	0.321	0.246	0.355	0.265	0.360
Traffic	96	**0.500**	0.324	0.564	0.351	0.579	0.388	0.512	**0.290**	0.593	0.321	0.650	0.396	0.607	0.392	0.612	0.338	0.587	0.366	0.615	0.391
	192	**0.506**	0.324	0.570	0.349	0.571	0.382	0.523	0.297	0.617	0.336	0.598	0.370	0.621	0.399	0.613	0.340	0.604	0.373	0.601	0.382
	336	**0.528**	0.341	0.576	0.349	0.582	0.385	0.530	**0.300**	0.629	0.336	0.605	0.373	0.622	0.396	0.618	0.328	0.621	0.383	0.613	0.386
	720	**0.561**	0.369	0.602	0.360	0.596	0.389	0.573	**0.313**	0.640	0.350	0.645	0.394	0.632	0.396	0.653	0.355	0.626	0.382	0.658	0.407
Weather	96	**0.148**	**0.212**	0.220	0.289	0.180	0.222	0.153	0.217	0.172	0.220	0.196	0.255	0.197	0.281	0.173	0.223	0.217	0.296	0.182	0.242
	192	0.200	0.262	0.341	0.385	0.229	**0.261**	**0.197**	0.269	0.219	**0.261**	0.237	0.296	0.237	0.312	0.245	0.285	0.276	0.336	0.227	0.287
	336	0.256	0.310	0.463	0.455	0.281	**0.298**	**0.252**	0.311	0.280	0.306	0.283	0.335	0.298	0.353	0.321	0.338	0.339	0.380	0.282	0.334
	720	0.327	0.362	0.682	0.565	0.358	0.349	**0.318**	0.363	0.365	0.359	0.345	0.381	0.352	**0.288**	0.414	0.410	0.403	0.428	0.352	0.386
Exchange	96	**0.085**	0.203	0.109	0.240	**0.085**	**0.202**	0.186	0.346	0.107	0.234	0.088	0.218	**0.085**	0.204	0.111	0.237	0.148	0.278	0.116	0.262
	192	**0.176**	**0.297**	0.241	0.353	0.180	0.301	0.467	0.522	0.226	0.344	**0.176**	0.315	0.182	0.303	0.219	0.335	0.271	0.380	0.215	0.359
	336	0.336	**0.418**	0.471	0.508	0.336	0.420	0.783	0.721	0.367	0.448	**0.313**	0.427	0.348	0.428	0.421	0.476	0.460	0.500	0.377	0.466
	720	0.953	0.738	1.259	0.865	0.881	0.710	1.367	0.943	0.964	0.746	0.839	**0.695**	1.025	0.774	1.092	0.769	1.195	0.841	**0.831**	0.699

Table 5: Statistics of datasets for short-term forecasting.

Dataset	Dim	Length	Train	Test
Yearly	1	6	23000	23000
Quarterly	1	8	24000	24000
Monthly	1	18	48000	48000
Weekly	1	13	359	359
Daily	1	14	4227	4227
Hourly	1	48	414	414

both considered sub-series modeling in multiple timescales. From this, we think sub-series modeling may help provide local context information for the estimation of missing values. Meanwhile, MSD-Mixer performs much better than TimesNet. This is because MSD-Mixer also considers multi-scale decomposition of the time series. It disentangles the temporal patterns within the data, such that they can be better modeled for the estimation of the missing value. Furthermore, the performance of other baseline methods drops quickly as the missing ratio increases, whereas the performance of our MSD-Mixer remains more stable, and consistently better than others. This also highlights the excellent capability of MSD-Mixer to model temporal patterns in complex time series data.

4.5 Anomaly Detection

4.5.1 Task Settings. Anomaly detection for time series data is of immense value in many real-time monitoring applications. It is also challenging due to the lack of labeled data. In this task, we leverage the popular paradigm of reconstruction-based unsupervised framework for anomaly detection. On this premise, a model learns to represent and reconstruct the normal data, thus abnormal data points can be detected with large reconstruction errors. Therefore, it is critical to learn high quality representations with the model. We experiment on five widely-adopted anomaly detection datasets for time series analysis, whose information is summarized in Table 8.

We include the Anomaly Transformer [50] as a task-specific baseline for comparison. Anomaly Transformer is one of the latest Transformer-based methods tailor-made for reconstruction-based unsupervised anomaly detection. It proposes a special Anomaly-Attention mechanism and a minimax strategy to learn and amplify the normal-abnormal associations.

For the experiment, we preprocess the datasets by splitting the time series into non-overlapping segments. In the training phase,

Table 6: Short-term forecasting results. The best results are in bold and the second bests are underlined. (*Task-specific baselines.)

Models		MSD-Mixer (Ours)	N-HiTS* (2023)	N-BEATS* (2020)	PatchTST (2023)	Crossformer (2023)	TimesNet (2023)	DLinear (2023)	ETSformer (2022)	NST (2022)	FEDformer (2022)	LightTS (2022)
Yr.	SMAPE	**13.191**	13.418	13.436	13.777	13.392	13.387	16.965	18.009	13.717	13.728	14.247
	MASE	**2.967**	3.045	3.043	3.056	3.001	2.996	4.283	4.487	3.078	3.048	3.109
	OWA	**0.777**	0.793	0.794	0.806	0.787	0.786	1.058	1.115	0.807	0.803	0.827
Qtr.	SMAPE	**9.971**	10.202	10.124	11.058	16.317	10.100	12.145	13.376	10.958	10.792	11.364
	MASE	**1.151**	1.194	1.169	1.321	2.197	1.182	1.520	1.906	1.325	1.283	1.328
	OWA	**0.872**	0.899	0.886	0.984	1.542	0.890	1.106	1.302	0.981	0.958	1.000
Mon.	SMAPE	**12.588**	12.791	12.677	14.433	12.924	12.670	13.514	14.588	13.917	14.260	14.014
	MASE	**0.921**	0.969	0.937	1.154	0.966	0.933	1.037	1.368	1.097	1.102	1.053
	OWA	**0.869**	0.899	0.880	1.043	0.902	0.878	0.956	1.149	0.998	1.012	0.981
Oth.	SMAPE	**4.615**	5.061	4.925	5.216	5.493	4.891	6.709	7.267	6.302	4.954	15.880
	MASE	**3.124**	3.216	3.391	3.688	3.690	3.302	4.953	5.240	4.064	3.264	11.434
	OWA	**0.978**	1.040	1.053	1.130	1.160	1.035	1.487	1.591	1.304	1.036	3.474
Avg.	SMAPE	**11.700**	11.927	11.851	13.011	13.474	11.829	13.639	14.718	12.780	12.840	13.525
	MASE	**1.557**	1.613	1.599	1.758	1.866	1.585	2.095	2.408	1.756	1.701	2.111
	OWA	**0.838**	0.861	0.855	0.939	0.985	0.851	1.051	1.172	0.930	0.918	1.051

Table 7: Imputation results. The best results are in bold and the second bests are underlined.

Models		MSD-Mixer (Ours)		PatchTST (2023)		Crossformer (2023)		TimesNet (2023)		DLinear (2023)		ETSformer (2022)		NST (2022)		FEDformer (2022)		LightTS (2022)	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTm1	12.5%	**0.019**	0.096	0.047	0.138	0.037	0.137	**0.019**	**0.092**	0.058	0.162	0.067	0.188	0.026	0.107	0.035	0.135	0.075	0.180
	25%	**0.019**	**0.092**	0.040	0.127	0.038	0.141	0.023	0.101	0.080	0.193	0.096	0.229	0.032	0.119	0.052	0.166	0.093	0.206
	37.5%	**0.024**	**0.103**	0.043	0.132	0.041	0.142	0.029	0.111	0.103	0.219	0.133	0.271	0.039	0.131	0.069	0.191	0.113	0.231
	50%	**0.027**	**0.103**	0.048	0.139	0.047	0.152	0.036	0.124	0.132	0.248	0.186	0.323	0.047	0.145	0.089	0.218	0.134	0.255
ETTm2	12.5%	**0.018**	**0.079**	0.026	0.093	0.044	0.148	**0.018**	0.080	0.062	0.166	0.108	0.239	0.021	0.088	0.056	0.159	0.034	0.127
	25%	**0.020**	**0.084**	0.026	0.094	0.047	0.151	**0.020**	0.085	0.085	0.196	0.164	0.294	0.024	0.096	0.080	0.195	0.042	0.143
	37.5%	**0.022**	**0.091**	0.033	0.110	0.044	0.145	0.023	**0.091**	0.106	0.222	0.237	0.356	0.027	0.103	0.110	0.231	0.051	0.159
	50%	**0.026**	0.100	0.033	0.106	0.047	0.150	**0.026**	**0.098**	0.131	0.247	0.323	0.421	0.030	0.108	0.156	0.276	0.059	0.174
ETTh1	12.5%	**0.031**	**0.116**	0.081	0.189	0.099	0.218	0.057	0.159	0.151	0.267	0.126	0.263	0.060	0.165	0.070	0.190	0.240	0.345
	25%	**0.041**	**0.135**	0.093	0.202	0.125	0.243	0.069	0.178	0.180	0.292	0.169	0.304	0.080	0.189	0.106	0.236	0.265	0.364
	37.5%	**0.056**	**0.157**	0.104	0.214	0.146	0.263	0.084	0.196	0.215	0.318	0.220	0.347	0.102	0.212	0.124	0.258	0.296	0.382
	50%	**0.071**	**0.179**	0.124	0.232	0.158	0.281	0.102	0.215	0.257	0.347	0.293	0.402	0.133	0.240	0.165	0.299	0.334	0.404
ETTh2	12.5%	**0.037**	**0.125**	0.059	0.152	0.103	0.220	0.040	0.130	0.100	0.216	0.187	0.319	0.042	0.133	0.095	0.212	0.101	0.231
	25%	**0.040**	**0.131**	0.059	0.154	0.110	0.229	0.046	0.141	0.127	0.247	0.279	0.390	0.049	0.147	0.137	0.258	0.115	0.246
	37.5%	**0.048**	**0.145**	0.064	0.161	0.129	0.246	0.052	0.151	0.158	0.276	0.400	0.465	0.056	0.158	0.187	0.304	0.126	0.257
	50%	**0.058**	0.163	0.070	0.170	0.148	0.265	0.060	**0.162**	0.183	0.299	0.602	0.572	0.065	0.170	0.232	0.341	0.136	0.268
ECL	12.5%	**0.048**	**0.150**	0.103	0.215	0.068	0.181	0.085	0.202	0.092	0.214	0.196	0.321	0.093	0.210	0.107	0.237	0.102	0.229
	25%	**0.059**	**0.170**	0.105	0.219	0.079	0.198	0.089	0.206	0.118	0.247	0.207	0.332	0.097	0.214	0.120	0.251	0.121	0.252
	37.5%	**0.070**	**0.184**	0.109	0.225	0.087	0.203	0.094	0.213	0.144	0.276	0.219	0.344	0.102	0.220	0.136	0.266	0.141	0.273
	50%	**0.080**	**0.197**	0.113	0.231	0.113	0.212	0.100	0.221	0.175	0.305	0.235	0.357	0.108	0.228	0.158	0.284	0.160	0.293
Weather	12.5%	**0.025**	**0.043**	0.043	0.069	0.036	0.092	**0.025**	0.045	0.039	0.084	0.057	0.141	0.027	0.051	0.041	0.107	0.047	0.101
	25%	**0.028**	**0.050**	0.041	0.065	0.035	0.088	0.029	0.052	0.048	0.103	0.065	0.155	0.029	0.056	0.064	0.163	0.052	0.111
	37.5%	**0.030**	**0.049**	0.043	0.069	0.035	0.088	0.031	0.057	0.057	0.117	0.081	0.180	0.033	0.062	0.107	0.229	0.058	0.121
	50%	**0.033**	**0.056**	0.045	0.070	0.038	0.092	0.034	0.062	0.066	0.134	0.102	0.207	0.037	0.068	0.183	0.312	0.065	0.133

Table 8: Statistics of datasets for anomaly detection.

Dataset	Dim	Length	Train	Test
SMD	38	100	708405	708420
MSL	55	100	58317	73729
SMAP	25	100	135183	427617
SWaT	51	100	495000	449919
PSM	25	100	132481	87841

we train the model to represent and reconstruct the input by minimizing reconstruction loss, which is the MSE between the model input and output. In the testing phase, we compute the difference between the test reconstruction loss of a data point and the average training reconstruction loss. If the difference is higher than a threshold, the data point is treated as an anomaly. The threshold values for different datasets are set as those in [48]. We report the point-wise precision, recall, and F1-score of the detection results, and use F1-score to compare the performance of different methods.

4.5.2 Result Analysis. As shown in Table 9, MSD-Mixer achieves the best F1-scores in 4 out of the 5 datasets. Compared with the baselines that simply learn to represent and reconstruct the time series, MSD-Mixer further learns to explicitly decompose the time

Table 9: Anomaly detection results. The best results are in bold and the second bests are underlined. (*Task-specific baseline.)

Models		MSD-Mixer (Ours)	Anomaly* (2022)	PatchTST (2023)	Crossformer (2023)	TimesNet (2023)	DLinear (2023)	ETSformer (2022)	NST (2022)	FEDformer (2022)	LightTS (2022)
SMD	Precision	88.7	88.9	87.5	83.1	88.7	83.6	87.4	88.3	88.0	87.1
	Recall	86.1	82.2	82.2	76.6	83.1	71.5	79.2	81.2	82.4	78.4
	F1-score	**87.4**	85.5	84.7	79.7	85.8	77.1	83.1	84.6	85.1	82.5
MSL	Precision	91.3	79.6	87.4	84.7	83.9	84.3	85.1	68.6	77.1	82.4
	Recall	88.4	87.4	69.5	83.7	86.4	85.4	84.9	89.1	80.1	75.8
	F1-score	**89.8**	83.3	77.4	84.2	85.2	84.9	85.0	77.5	78.6	79.0
SMAP	Precision	93.4	91.9	90.5	92.0	92.5	92.3	92.3	89.4	90.5	92.6
	Recall	96.9	58.1	56.4	55.4	58.3	55.4	55.8	59.0	58.1	55.3
	F1-score	**95.2**	71.2	69.5	69.1	71.5	69.3	69.5	71.1	70.8	69.2
SWaT	Precision	93.1	72.5	91.3	88.5	86.8	80.9	90.0	68.0	90.2	92.0
	Recall	98.3	97.3	83.2	93.5	97.3	95.3	80.4	96.8	96.4	94.7
	F1-score	**95.7**	83.1	87.1	90.9	91.7	87.5	84.9	79.9	93.2	93.3
PSM	Precision	97.4	68.4	98.9	97.2	98.2	98.3	99.3	97.8	97.3	98.4
	Recall	96.7	94.7	92.4	89.7	96.8	89.3	85.3	96.8	97.2	96.0
	F1-score	97.0	79.4	95.6	93.3	**97.5**	93.6	91.8	97.3	97.2	97.2

Table 10: Statistics of datasets for classification.

Dataset	Dim	Length	Classes	Train	Test
AWR	9	144	25	275	300
AF	2	640	3	15	15
CT	3	182	20	1,422	1,436
CR	6	1,197	12	108	72
FD	144	62	2	5,890	3,524
FM	28	50	2	316	100
MI	64	3,000	2	278	100
SCP1	6	896	2	268	293
SCP2	7	1,152	2	200	180
UWGL	3	315	8	120	320

series into components and represent each component for the reconstruction. Therefore, MSD-Mixer has a stronger representation learning ability to precisely capture the normal temporal patterns and identify the abnormal data.

4.6 Classification

4.6.1 Task Settings. The time series classification problem arises in various real-life applications such as human activity recognition and medical time series based diagnosis. In this task we consider the series-level classification problem and build predictive models that output one categorical label for each time series, which emphasizes more on discriminative modeling ability than other time series analysis tasks. We experiment on ten datasets from the well-known UEA time series classification archive [1] which is the most widely used multivariate time series classification benchmark. The ten datasets have diverse characteristics in terms of domain, series length, number of samples, and the number of classes, which helps to comprehensively examine the capabilities of different methods. The datasets have been well processed and split into train and test sets. Information of the datasets is summarized in Table 10.

We include six competitive classification methods reported in recent works [7, 8] as task-specific baselines for comparison in this task. They cover both statistical (DTWD [38], MiniRocket [12]) and deep learning-based (TARNet [8], FormerTime [7], TST [52], TapNet [54]) approaches. We use the classification accuracy, number of 1st counts and mean rank as our evaluation metrics.

4.6.2 Result Analysis. Table 11 shows the result of our classification tests. MSD-Mixer performs the best with 5 first places and 2 second places out of 10 benchmarks, which demonstrates its great discriminative power in the modeling. Moreover, the diversity of datasets in terms of size, dimension, length, and number of classes also reflects the adaptability of MSD-Mixer. Different from other tasks discussed above, the task-general baselines typically perform inferior to task-specific ones in classification. The two best baselines are task-specific approaches TARNet and TST, which are Transformer-based deep learning algorithms. It should be noted that TARNet and TST adopt extra self-supervised training in addition to the supervised training with class labels, which we think may be the reason for their good performance. In contrast, MSD-Mixer outperforms them with only supervised training, which demonstrates the modeling ability of MSD-Mixer. We also notice that the two statistical baselines DTWD and MiniRocket perform well in some datasets. These results indicate that it is challenging to design a task-general backbone for classification tasks. We believe our thorough consideration on the special composition and multi-scale nature of time series is the reason why MSD-Mixer can consistently have better performance in classification.

4.7 Ablation Study

We strongly believe that the advantages of MSD-Mixer are rooted in our proposed *multi-scale temporal patching* and *residual loss*. To validate the efficacy of the proposed modules, we implement the following variants of MSD-Mixer:

- *MSD-Mixer-I*: the inverted MSD-Mixer. We arrange the layers with their patch sizes in ascending order instead of descending.
- *MSD-Mixer-N*: the MSD-Mixer without patching. We replace the patching module with max pooling and linear interpolation layers, following the strategy in [6].
- *MSD-Mixer-U*: the MSD-Mixer without multi-scale patching. We set the patch size as the square root of the input length and use this same patch size for all layers.

Table 11: Classification results. The best results are in bold and the second bests are underlined. (*Task-specific baselines.)

Models	MSD-Mixer (Ours)	TARNet* (2022)	DTWD* (2015)	TapNet* (2020)	MiniRocket* (2021)	TST* (2021)	FormerTime* (2023)	PatchTST (2023)	Crossformer (2023)	TimesNet (2023)	DLinear (2023)	ETSformer (2022)	NST (2022)	FEDformer (2022)	LightTS (2022)
AWR	0.983	0.977	0.987	0.987	**0.993**	0.947	0.985	0.040	0.937	0.977	0.963	0.973	0.497	0.587	0.970
AF	0.600	**1.000**	0.220	0.333	0.133	0.533	0.600	0.467	0.400	0.333	0.200	0.400	0.467	0.400	0.333
CT	0.987	0.994	0.989	**0.997**	0.990	0.971	0.991	0.877	0.970	0.974	0.973	0.978	0.804	0.960	0.977
CR	**1.000**	**1.000**	**1.000**	0.958	0.986	0.847	0.981	0.083	0.846	0.847	0.861	0.861	0.736	0.472	0.847
FD	**0.698**	0.641	0.529	0.556	0.612	0.625	0.687	0.500	0.687	0.686	0.672	0.673	0.500	0.684	0.658
FM	**0.660**	0.620	0.530	0.530	0.550	0.590	0.618	0.510	0.510	0.590	0.570	0.590	0.510	0.540	0.540
MI	**0.670**	0.630	0.500	0.590	0.610	0.610	0.632	0.570	0.570	0.570	0.620	0.590	0.640	0.580	0.590
SCP1	0.949	0.816	0.775	0.652	0.915	**0.961**	0.887	0.741	0.921	0.918	0.880	0.860	0.898	0.594	0.918
SCP2	**0.639**	0.622	0.539	0.550	0.506	0.604	0.592	0.500	0.583	0.572	0.527	0.561	0.500	0.511	0.522
UWGL	0.884	0.878	0.903	0.894	0.785	**0.913**	0.888	0.213	0.853	0.853	0.812	0.825	0.703	0.453	0.831
Avg. Acc.	0.807	**0.818**	0.697	0.705	0.708	0.760	0.786	0.450	0.728	0.732	0.708	0.731	0.625	0.578	0.719
1st Count	**5**	3	1	1	1	2	0	0	0	0	0	0	0	0	0
Mean Rank	**2.5**	4.4	8.3	7.5	8.0	6.0	3.8	13.0	8.5	7.1	9.0	7.4	11.0	11.3	8.5

Table 12: Average results of MSD-Mixer variants on five tasks.

Model		MSD-Mixer	MSD-Mixer-I	MSD-Mixer-N	MSD-Mixer-U	MSD-Mixer-L
Long-Term Forecasting	MSE	0.345	0.345	0.358	0.422	0.348
	MAE	0.358	0.357	0.371	0.470	0.360
Short-Term Forecasting	SMAPE	11.700	11.699	11.814	11.869	11.780
	MASE	1.557	1.557	1.598	1.587	1.567
	OWA	0.838	0.837	0.853	0.853	0.844
Imputation	MSE	0.038	0.039	0.041	0.058	0.040
	MAE	0.117	0.130	0.122	0.149	0.119
Anomaly Detection	F1	0.930	0.925	0.918	0.847	0.897
Classification	ACC	0.807	0.803	0.732	0.729	0.768

- *MSD-Mixer-L*: the MSD-Mixer trained without the *residual loss*. We train the model with the loss function from the target task only.

We carry out the experiments for the four MSD-Mixer variants on all benchmarks in the five tasks, and report their average performance over the benchmarks in each task in Table 12. MSD-Mixer-I has very similar performance to the original MSD-Mixer for all five tasks. This result indicates that the arrangement of layers with different patch sizes does not affect the performance of MSD-Mixer. We think the reason behind is that the *multi-scale temporal patching* enforces the layer to focus on the modeling of specific timescales, such that their order has a relatively small impact on the performance. Without the patching modules, MSD-Mixer-N cannot capture the sub-series features, thus we can observe a performance drop compared with MSD-Mixer, especially in classification accuracy. Likewise, by using the same patch size in all layers, MSD-Mixer-U does not model the multi-scale patterns in different layers, which affects the performance considerably in all tasks. Lastly, by comparing MSD-Mixer-L with MSD-Mixer we find that the *residual loss* do contribute to the learning of the model in all tasks by enhancing the completeness of the decomposition.

4.8 Model Efficiency

To study the efficiency of our proposed MSD-Mixer, we compare the number of model parameters and training time consumption of different approaches in the long-term forecasting task with the ETTm2 dataset in Figure 6. From the result we can observe that MSD-Mixer achieves the best MSE among all the baselines. Comparing with PatchTST and TimesNet which are the second- and third-best models, MSD-Mixer contains less than 1/10 and 4/5 model parameters (951K vs. 10.1M and 1191K), and runs more than 1.67 and 8 times faster (10.9s/epoch vs. 18.3s/epoch and 93.2s/epoch), which demonstrates the great efficiency of MSD-Mixer. On the other hand, MLP models (MSD-Mixer, DLinear, and LightTS) generally contain fewer model parameters and consume less training time than their Transformer and CNN counterparts in this experiment. Although MSD-Mixer is larger and slower than the other two MLP models, it achieves 12% and 19% improvements on MSE with the extra model parameters and time, which also shows MSD-Mixer's advancements over the previous MLP-based methods.

4.9 Case Study

To further validate the effectiveness of our carefully designed *residual loss* in MSD-Mixer, in Figure 5, we show two examples of how the input time series is decomposed by MSD-Mixer trained with (MSD-Mixer) and without (MSD-Mixer-L) our proposed *residual loss*. The examples are from the long-term forecasting task with the ETTh1 dataset, which is well acknowledged as a challenging dataset with complex characteristics including but not limited to multiple periodic variations and channel-wise heterogeneity. The sampling rate of the data is 1 hour, and input length is set to 96. We train the MSD-Mixer which has 5 layers with patch sizes as {24, 12, 6, 2, 1}, corresponding to sub-series of 1 day, half day, 6 hours, 2 hours, and 1 hour.

First, from both input plots we observe multiple irregular temporal patterns, which cannot be simply explained by seasonal or trend-cyclic patterns as discussed in previous works [49]. Their corresponding autocorrelation function (ACF) plots also indicate high correlations in multiple temporal lags in the input data. Therefore, simply considering seasonal-trend decomposition is not enough

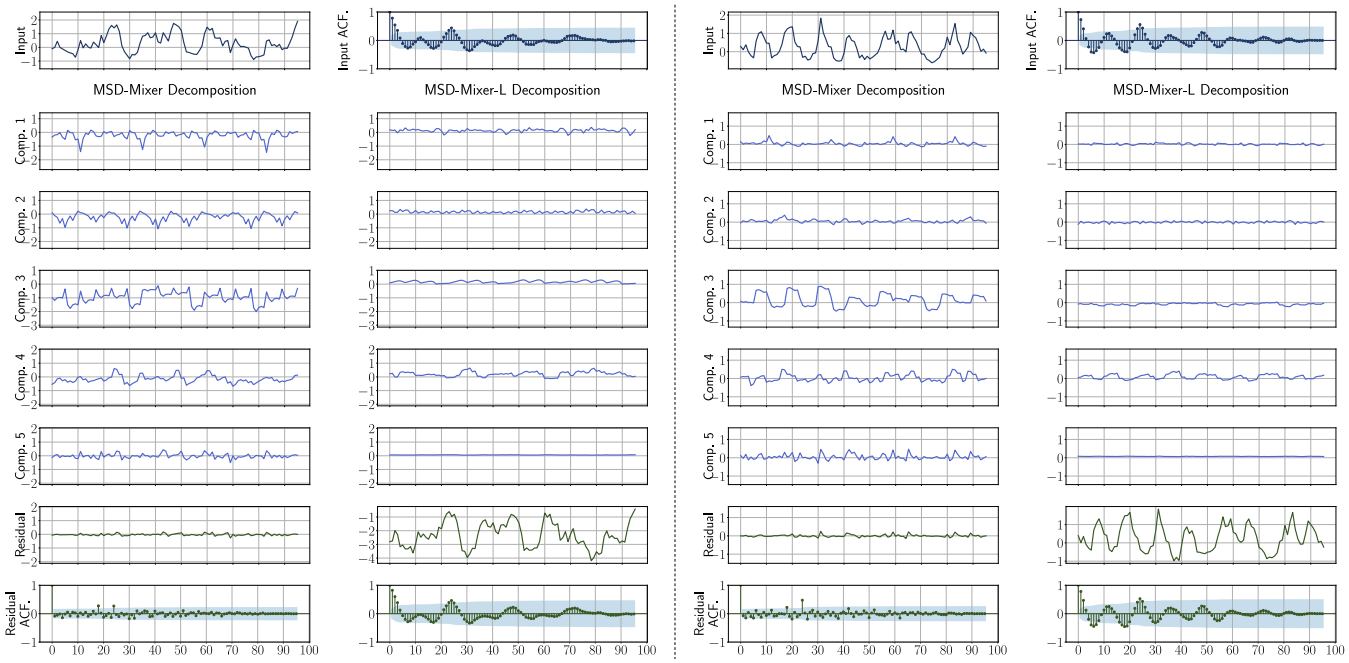

Figure 5: Examples of decomposition.

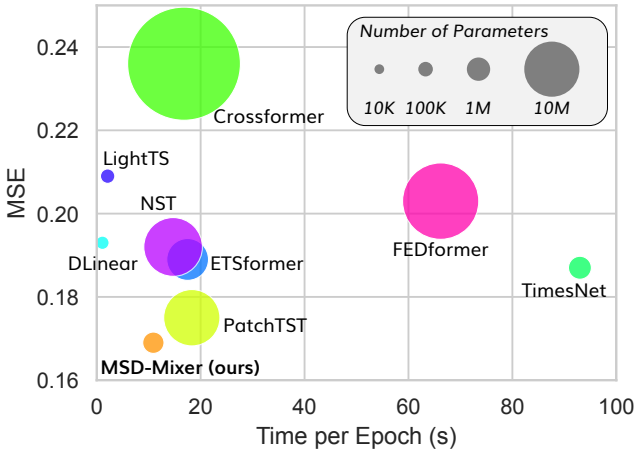

Figure 6: Model efficiency comparison.

to account for intricate temporal patterns in real-life time series data. Then, we note that the components output by MSD-Mixer are obviously more diverse, especially in terms of their timescale, compared with that of MSD-Mixer-L. It indicates that they contain different temporal patterns in the input data. We attribute it to the effectiveness of our proposed *multi-scale temporal patching*.

Furthermore, it is obvious from both examples that without our proposed *residual loss*, MSD-Mixer-L leaves most of the information from the input in the decomposition residual, while the other components contain little information. In comparison, the mean of residuals from MSD-Mixer are much smaller, and the residual ACF plots also indicate less periodic patterns. The results clearly validate the effectiveness of our proposed *residual loss* in constraining the decomposition residual. The multi-scale components also show the potential to provide interpretability on the composition of the input data and how the output is produced by MSD-Mixer.

5 CONCLUSION

In this work, we solve the time series analysis problem by considering its unique composition and complex multi-scale temporal variations, and propose MSD-Mixer, a **M**ulti-**S**cale **D**ecomposition MLP-**Mixer** which learns to explicitly decompose the input time series into different components, and represents the components in different layers. We propose a novel *multi-scale temporal patching* approach in MSD-Mixer to model the time series as multi-scale patches, and employ MLPs along different dimensions to mix intra- and inter-patch variations and channel-wise correlations. In addition, we propose a *residual loss* to constrain both the mean and the autocorrelation of the decomposition residual for decomposition completeness. Through extensive experiments on 26 real-world datasets, we demonstrate that MSD-Mixer consistently outperforms the state-of-the-art task-general and task-specific approaches by a wide margin on five common tasks, namely long-term forecasting (up to 9.8% in MSE), short-term forecasting (up to 5.6% in OWA), imputation (up to 46.1% in MSE), anomaly detection (up to 33.1% in F1-score) and classification (up to 36.3% in Mean Rank).

ACKNOWLEDGMENT

The work of Weipeng Zhuo was supported by the Guangdong Provincial Key Laboratory IRADS (2022B1212010006, R0400001-22).

REFERENCES

[1] Anthony Bagnall, Hoang Anh Dau, Jason Lines, Michael Flynn, James Large, Aaron Bostrom, Paul Southam, and Eamonn Keogh. 2018. The UEA multivariate time series classification archive, 2018. arXiv:1811.00075 [cs.LG]

[2] Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. 2018. An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling. arXiv:1803.01271 [cs.LG]

[3] Kasun Bandara, Rob J Hyndman, and Christoph Bergmeir. 2021. MSTL: A Seasonal-Trend Decomposition Algorithm for Time Series with Multiple Seasonal Patterns. arXiv:2107.13462 [stat.AP]

[4] Parikshit Bansal, Prathamesh Deshpande, and Sunita Sarawagi. 2021. Missing value imputation on multidimensional time series. *Proceedings of the VLDB Endowment* 14, 11 (2021), 2533–2545.

[5] Joos-Hendrik Böse, Valentin Flunkert, Jan Gasthaus, Tim Januschowski, Dustin Lange, David Salinas, Sebastian Schelter, Matthias Seeger, and Yuyang Wang. 2017. Probabilistic demand forecasting at scale. *Proceedings of the VLDB Endowment* 10, 12 (2017), 1694–1705.

[6] Cristian Challu, Kin G Olivares, Boris N Oreshkin, Federico Garza Ramirez, Max Mergenthaler Canseco, and Artur Dubrawski. 2023. NHITS: Neural Hierarchical Interpolation for Time Series Forecasting. , 6989–6997 pages.

[7] Mingyue Cheng, Qi Liu, Zhiding Liu, Zhi Li, Yucong Luo, and Enhong Chen. 2023. FormerTime: Hierarchical Multi-Scale Representations for Multivariate Time Series Classification. arXiv:2302.09818 [cs.LG]

[8] Ranak Roy Chowdhury, Xiyuan Zhang, Jingbo Shang, Rajesh K. Gupta, and Dezhi Hong. 2022. TARNet: Task-Aware Reconstruction for Time-Series Transformer. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining* (Washington DC, USA) *(KDD '22)*. Association for Computing Machinery, New York, NY, USA, 212–220.

[9] Robert B Cleveland, William S Cleveland, Jean E McRae, and Irma Terpenning. 1990. STL: A seasonal-trend decomposition. *J. Off. Stat* 6, 1 (1990), 3–73.

[10] Yue Cui, Kai Zheng, Dingshan Cui, Jiandong Xie, Liwei Deng, Feiteng Huang, and Xiaofang Zhou. 2021. METRO: a generic graph neural network framework for multivariate time series forecasting. *Proceedings of the VLDB Endowment* 15, 2 (2021), 224–236.

[11] Estela Bee Dagum and Silvia Bianconcini. 2016. *Seasonal adjustment methods and real time trend-cycle estimation.* Springer, New York, NY, USA.

[12] Angus Dempster, Daniel F. Schmidt, and Geoffrey I. Webb. 2021. MiniRocket: A Very Fast (Almost) Deterministic Transform for Time Series Classification. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining* (Virtual Event, Singapore) *(KDD '21)*. Association for Computing Machinery, New York, NY, USA, 248–257.

[13] Alexander Dokumentov, Rob J Hyndman, et al. 2015. STR: A seasonal-trend decomposition procedure based on regression. *Monash econometrics and business statistics working papers* 13, 15 (2015), 2015–13.

[14] Christos Faloutsos, Jan Gasthaus, Tim Januschowski, and Yuyang Wang. 2018. Forecasting big time series: old and new. *Proceedings of the VLDB Endowment* 11, 12 (2018), 2102–2105.

[15] Ziquan Fang, Lu Pan, Lu Chen, Yuntao Du, and Yunjun Gao. 2021. MDTP: A multi-source deep traffic prediction framework over spatio-temporal trajectory data. *Proceedings of the VLDB Endowment* 14, 8 (2021), 1289–1297.

[16] Lu Han, Han-Jia Ye, and De-Chuan Zhan. 2023. The Capacity and Robustness Trade-off: Revisiting the Channel Independent Strategy for Multivariate Time Series Forecasting. arXiv:2304.05206 [cs.LG]

[17] Charles C Holt. 2004. Forecasting seasonals and trends by exponentially weighted moving averages. *International journal of forecasting* 20, 1 (2004), 5–10.

[18] Rob J Hyndman and George Athanasopoulos. 2018. *Forecasting: principles and practice.* OTexts, Melbourne, Australia.

[19] Hassan Ismail Fawaz, Benjamin Lucas, Germain Forestier, Charlotte Pelletier, Daniel F Schmidt, Jonathan Weber, Geoffrey I Webb, Lhassane Idoumghar, Pierre-Alain Muller, and François Petitjean. 2020. Inceptiontime: Finding alexnet for time series classification. *Data Mining and Knowledge Discovery* 34, 6 (2020), 1936–1962.

[20] Fazle Karim, Somshubra Majumdar, Houshang Darabi, and Samuel Harford. 2019. Multivariate LSTM-FCNs for time series classification. *Neural networks* 116 (2019), 237–245.

[21] Mourad Khayati, Alberto Lerner, Zakhar Tymchenko, and Philippe Cudré-Mauroux. 2020. Mind the gap: an experimental evaluation of imputation of missing values techniques in time series. *Proceedings of the VLDB Endowment* 13, 5 (2020), 768–782.

[22] Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. 2020. Reformer: The Efficient Transformer. arXiv:2001.04451 [cs.LG]

[23] Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. 2017. FractalNet: Ultra-Deep Neural Networks without Residuals. arXiv:1605.07648 [cs.CV]

[24] Colin Lea, Michael D. Flynn, Rene Vidal, Austin Reiter, and Gregory D. Hager. 2016. Temporal Convolutional Networks for Action Segmentation and Detection. arXiv:1611.05267 [cs.CV]

[25] Jae-Gil Lee, Jiawei Han, Xiaolei Li, and Hector Gonzalez. 2008. TraClass: trajectory classification using hierarchical region-based and trajectory-based clustering. *Proceedings of the VLDB Endowment* 1, 1 (2008), 1081–1094.

[26] Yangfan Li, Kenli Li, Cen Chen, Xu Zhou, Zeng Zeng, and Keqin Li. 2021. Modeling temporal patterns with dilated convolutions for time-series forecasting. *ACM Transactions on Knowledge Discovery from Data (TKDD)* 16, 1 (2021), 1–22.

[27] Minhao Liu, Ailing Zeng, Muxi Chen, Zhijian Xu, Qiuxia Lai, Lingna Ma, and Qiang Xu. 2022. Scinet: Time series modeling and forecasting with sample convolution and interaction. *Advances in Neural Information Processing Systems* 35 (2022), 5816–5828.

[28] Yong Liu, Haixu Wu, Jianmin Wang, and Mingsheng Long. 2022. Non-stationary Transformers: Exploring the Stationarity in Time Series Forecasting. In *Advances in Neural Information Processing Systems*, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (Eds.), Vol. 35. Curran Associates, Inc., New Orleans,LA, USA, 9881–9893.

[29] Spyros Makridakis, Evangelos Spiliotis, and Vassilios Assimakopoulos. 2020. The M4 Competition: 100,000 time series and 61 forecasting methods. *International Journal of Forecasting* 36, 1 (2020), 54–74. M4 Competition.

[30] Xiaoye Miao, Yangyang Wu, Jun Wang, Yunjun Gao, Xudong Mao, and Jianwei Yin. 2021. Generative semi-supervised learning for multivariate time series imputation. , 8983–8991 pages.

[31] Yuqi Nie, Nam H. Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. 2023. A Time Series is Worth 64 Words: Long-term Forecasting with Transformers. arXiv:2211.14730 [cs.LG]

[32] Boris N. Oreshkin, Dmitri Carpov, Nicolas Chapados, and Yoshua Bengio. 2020. N-BEATS: Neural basis expansion analysis for interpretable time series forecasting.

[33] John Paparrizos, Paul Boniol, Themis Palpanas, Ruey S Tsay, Aaron Elmore, and Michael J Franklin. 2022. Volume under the surface: a new accuracy evaluation measure for time-series anomaly detection. *Proceedings of the VLDB Endowment* 15, 11 (2022), 2774–2787.

[34] John Paparrizos, Yuhao Kang, Paul Boniol, Ruey S Tsay, Themis Palpanas, and Michael J Franklin. 2022. TSB-UAD: an end-to-end benchmark suite for univariate time-series anomaly detection. *Proceedings of the VLDB Endowment* 15, 8 (2022), 1697–1711.

[35] Hansheng Ren, Bixiong Xu, Yujing Wang, Chao Yi, Congrui Huang, Xiaoyu Kou, Tony Xing, Mao Yang, Jie Tong, and Qi Zhang. 2019. Time-Series Anomaly Detection Service at Microsoft. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (Anchorage, AK, USA) *(KDD '19)*. Association for Computing Machinery, New York, NY, USA, 3009–3017. https://doi.org/10.1145/3292500.3330680

[36] Sebastian Schmidl, Phillip Wenig, and Thorsten Papenbrock. 2022. Anomaly detection in time series: a comprehensive evaluation. *Proceedings of the VLDB Endowment* 15, 9 (2022), 1779–1797.

[37] Amin Shabani, Amir Abdi, Lili Meng, and Tristan Sylvain. 2023. Scaleformer: Iterative Multi-scale Refining Transformers for Time Series Forecasting. arXiv:2206.04038 [cs.LG]

[38] Mohammad Shokoohi-Yekta, Jun Wang, and Eamonn Keogh. 2015. On the Non-Trivial Generalization of Dynamic Time Warping to the Multi-Dimensional Case. In *Proceedings of the 2015 SIAM International Conference on Data Mining (SDM)*. SIAM, Vancouver, BC, Canada, 289–297.

[39] Marina Theodosiou. 2011. Forecasting monthly and quarterly time series using STL decomposition. *International Journal of Forecasting* 27, 4 (2011), 1178–1195.

[40] Luan Tran, Min Y Mun, Matthew Lim, Jonah Yamato, Nathan Huh, and Cyrus Shahabi. 2020. DeepTRANS: a deep learning system for public bus travel time estimation using traffic forecasting. *Proceedings of the VLDB Endowment* 13, 12 (2020), 2957–2960.

[41] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. arXiv:1706.03762 [cs.CL]

[42] Huiqiang Wang, Jian Peng, Feihu Huang, Jince Wang, Junhui Chen, and Yifei Xiao. 2023. Micn: Multi-scale local and global context modeling for long-term series forecasting.

[43] Qingsong Wen, Jingkun Gao, Xiaomin Song, Liang Sun, Huan Xu, and Shenghuo Zhu. 2018. RobustSTL: A Robust Seasonal-Trend Decomposition Algorithm for Long Time Series. arXiv:1812.01767 [cs.LG]

[44] Qingsong Wen, Zhe Zhang, Yan Li, and Liang Sun. 2020. Fast RobustSTL: Efficient and Robust Seasonal-Trend Decomposition for Time Series with Complex Patterns. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (Virtual Event, CA, USA) *(KDD '20)*. Association for Computing Machinery, New York, NY, USA, 2203–2213.

[45] Mike West. 1997. Time series decomposition. *Biometrika* 84, 2 (1997), 489–494.

[46] Peter R Winters. 1960. Forecasting sales by exponentially weighted moving averages. *Management science* 6, 3 (1960), 324–342.

[47] Gerald Woo, Chenghao Liu, Doyen Sahoo, Akshat Kumar, and Steven Hoi. 2022. ETSformer: Exponential Smoothing Transformers for Time-series Forecasting. arXiv:2202.01381

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

Analysis. arXiv:2210.02186 [cs.LG]

[49] Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. 2021. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. *Advances in Neural Information Processing Systems* 34 (2021), 22419–22430.

[50] Jiehui Xu, Haixu Wu, Jianmin Wang, and Mingsheng Long. 2022. Anomaly Transformer: Time Series Anomaly Detection with Association Discrepancy. arXiv:2110.02642 [cs.LG]

[51] Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. 2022. Are Transformers Effective for Time Series Forecasting? arXiv:2205.13504 [cs.AI]

[52] George Zerveas, Srideepika Jayaraman, Dhaval Patel, Anuradha Bhamidipaty, and Carsten Eickhoff. 2021. A Transformer-Based Framework for Multivariate Time Series Representation Learning. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining* (Virtual Event, Singapore) *(KDD '21)*. Association for Computing Machinery, New York, NY, USA, 2114–2124.

[53] Tianping Zhang, Yizhuo Zhang, Wei Cao, Jiang Bian, Xiaohan Yi, Shun Zheng, and Jian Li. 2022. Less Is More: Fast Multivariate Time Series Forecasting with Light Sampling-oriented MLP Structures. arXiv:2207.01186 [cs.LG]

[54] Xuchao Zhang, Yifeng Gao, Jessica Lin, and Chang-Tien Lu. 2020. TapNet: Multivariate Time Series Classification with Attentional Prototypical Network. *Proceedings of the AAAI Conference on Artificial Intelligence* 34, 04 (Apr. 2020), 6845–6852.

[55] Yunhao Zhang and Junchi Yan. 2023. Crossformer: Transformer Utilizing Cross-Dimension Dependency for Multivariate Time Series Forecasting.

[56] Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. 2021. Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting. arXiv:2012.07436 [cs.LG]

[57] Tian Zhou, Ziqing Ma, Qingsong Wen, Xue Wang, Liang Sun, and Rong Jin. 2022. FEDformer: Frequency Enhanced Decomposed Transformer for Long-term Series Forecasting. arXiv:2201.12740 [cs.LG]