FormerTime: Hierarchical Multi-Scale Representations for Multivariate Time Series Classification

Mingyue Cheng
Anhui Province Key Laboratory of Big Data Analysis and Application, University of Science and Technology of China & State Key Laboratory of Cognitive Intelligence, Hefei, China
mycheng@mail.ustc.edu.cn

Qi Liu*
Anhui Province Key Laboratory of Big Data Analysis and Application, University of Science and Technology of China & State Key Laboratory of Cognitive Intelligence, Hefei, China
qiliuql@ustc.edu.cn

Zhiding Liu
Anhui Province Key Laboratory of Big Data Analysis and Application, University of Science and Technology of China & State Key Laboratory of Cognitive Intelligence, Hefei, China
doge@mail.ustc.edu.cn

Zhi Li
Shenzhen International Graduate School, Tsinghua University, Shenzhen, China
zhilizl@sz.tsinghua.edu.cn

Yucong Luo
Anhui Province Key Laboratory of Big Data Analysis and Application, University of Science and Technology of China & State Key Laboratory of Cognitive Intelligence, Hefei, China
prime666@mail.ustc.edu.cn

Enhong Chen
Anhui Province Key Laboratory of Big Data Analysis and Application, University of Science and Technology of China & State Key Laboratory of Cognitive Intelligence, Hefei, China
cheneh@ustc.edu.cn

ABSTRACT

Deep learning-based algorithms, e.g., convolutional networks, have significantly facilitated multivariate time series classification (MTSC) task. Nevertheless, they suffer from the limitation in modeling long-range dependence due to the nature of convolution operations. Recent advancements have shown the potential of transformers to capture long-range dependence. However, it would incur severe issues, such as fixed scale representations, temporal-invariant and quadratic time complexity, with transformers directly applicable to the MTSC task because of the distinct properties of time series data. To tackle these issues, we propose FormerTime, an hierarchical representation model for improving the classification capacity for the MTSC task. In the proposed FormerTime, we employ a hierarchical network architecture to perform multi-scale feature maps. Besides, a novel transformer encoder is further designed, in which an efficient temporal reduction attention layer and a well-informed contextual positional encoding generating strategy are developed. To sum up, FormerTime exhibits three aspects of merits: (1) learning hierarchical multi-scale representations from time series data, (2) inheriting the strength of both transformers and convolutional networks, and (3) tacking the efficiency challenges incurred by the self-attention mechanism. Extensive experiments performed on 10 publicly available datasets from UEA archive verify the superiorities of the FormerTime compared to previous competitive baselines.

*Qi Liu is corresponding author.

Our experiment codes are available at https://github.com/Mingyue-Cheng/FormerTime

WWW '23, April 30–May 04, 2023, Austin, TX, USA

ACM ISBN 978-1-4503-9416-1/23/04…$15.00
https://doi.org/10.1145/3543507.3583205

CCS CONCEPTS

• **Computing methodologies** → **Neural networks**; • **Mathematics of computing** → Time series analysis.

KEYWORDS

Multivariate time series classification, Time series representations, Self-attention mdoels

ACM Reference Format:
Mingyue Cheng, Qi Liu, Zhiding Liu, Zhi Li, Yucong Luo, and Enhong Chen. 2023. FormerTime: Hierarchical Multi-Scale Representations for Multivariate Time Series Classification. In *Proceedings of the ACM Web Conference 2023 (WWW '23), April 30–May 04, 2023, Austin, TX, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3543507.3583205

1 INTRODUCTION

Multivariate time series [16, 20, 50] are ubiquitous for many web applications [35, 47], which are sequences of events acquired longitudinally and each event is constituted by observations recorded over multiple attributes. For example, the electrocardiogram (ECG) signals [33] in electronic health records (EHRs) can be formulated as multivariate time series data since they can be obtained over time and multiple sensors. Comprehensive analysis of such data can facilitate decision-making in real applications [27, 30], such as human activity recognition, healthcare monitoring, and industry detection. Particularly, multivariate time series classification (MTSC) tasks, as one fundamental problem of time series analysis, received significant attention in both academia and industry.

Accordingly, numerous efforts [36, 49] have been devoted to the MTSC problem over the last decades. In general, most of these current works can be divided into two categories: pattern-based and feature-based models. The former type usually extracts useful bag-of-patterns or shapelet patterns from the whole time series, and then transforms these extracted patterns into features to be used as inputs for the classifier. Since they are generated in raw time series, the corresponding patterns are often interpretable. The main concern is the often-incurred expensive computation cost

during the process of pattern extraction. In contrast, feature-based methods can be very efficient and can be scaled to large-scale time series data but their classification capacity greatly depends on the effectiveness of labor-intensive features based on domain experts.

Hence, researchers begin to explore more expressive feature maps for improving classification capacity. Deep learning-based models [6, 17, 22, 24, 39, 49, 50] have achieved remarkable success and have become ever-increasingly prevalent over past advancements. The main reason is that discriminative features related to time series can be learned in an end-to-end manner, which significantly saves manual feature engineering efforts. Among them, convolutional-based methods nearly have become the dominant approach due to the strong representation capacity of convolution operations [6, 31]. In general, the strengths of convolutional models in performing time series classification can be summarized as follows: (1) convolutional networks can easily learn multi-scale representations by controlling the strides of convolutional kernels [42], and preserve the capacity of temporal-invariant via the weight-sharing mechanism, which are of great significance for MTSC tasks; (2) Very deep convolutional networks can be stacked by employing residual connections, enabling larger receptive fields for capturing the sequence dependence; (3) Convolution operations can be efficiently computed without suffering from limitations of sequence length or instance number, and thus can be easily scaled to massive datasets. Despite their effectiveness, we argue that the classification performance is still restricted in failing to capture global contexts in convolution operation.

Recently proposed transformer architecture [40, 44] have shown promising capacity in capturing global contexts for language modeling tasks. Motivated by this, we seek to transfer the powerful capacity of transformers from language domain to time series. However, it would easily incur severe issues in applying the transformer to the MTSC problem. First, in language transformers, only fixed-scale representations can be learned since word tokens serve as the basic elements. By contrast, the information density of a single object in time series is too small to reflect helpful patterns related to class labels. Taking an example in ECG classification, informative patterns are typically characterized by a series of continuous points or various sub-series instead of a single point. As such, multi-scale feature maps [37] are necessary and can take a significant influence on the classification capacity. Second, the capacity of temporal-invariant is largely weakened in vanilla transformers since self-attention is permutation-variant, which may also restrict the final performance [29]. Last, the sequence length of time series usually can be much longer than language sentences, which inevitably incurs an expensive computation burden since the time and memory complexity of the self-attention mechanism in the transformer architecture is quadratic to the sequence length input. Though some pioneering efforts [26, 38, 48] based on transformers have been devoted before, the problems discussed above are still under exploration in the MTSC task.

To tackle these severe issues, in this work, we present FormerTime, a hierarchical transformer network for the MTSC task. Specifically, we design a hierarchical structure by dividing the whole network into several different stages, with different levels of scales as input. We also develop a novel transformer encoder to perform hidden transformation with two distinct characteristics: (1) we replace the standard self-attention mechanism with our newly designed temporal reduction version to save computation cost; (2) we design a context-aware positional encoding generator, which is designed to not only preserve the order of sequence input but also enhance the capacity of temporal-invariant of the whole model. In general, our FormerTime exhibits the following merits. First, in contrast to convolutional-based classifiers, FormerTime always yields a global reception field, which is useful for capturing the long-range dependence and interaction of the whole time series. Second, FormerTime can conveniently learn time series representations on various scales via its hierarchical architecture. Third, in the FormerTime, the inductive bias of temporal-invariant is well enhanced by leveraging the contextual positional generators. More importantly, the computation consumption of FormerTime is largely saved, and could be acceptable even for very long sequences.

To evaluate the effectiveness of the FormerTime, we conduct extensive empirical studies on 10 public datasets from the UEA archive [2]. The experimental results clearly show that FormerTime can yield strong classification performance in average. It significantly outperforms compared strong baselines. In summary, we initially demonstrate the potential of transformers in the MTSC problem. To the best of our knowledge, we first the few attempts of transformer-based models in breaking the efficiency bottleneck and achieving great performance improvements. We hope our work can facilitate the study of applying transformers to the MTSC problem.

2 RELATED WORK

2.1 Time Series Classification

Tremendous efforts have been devoted to time series classification. Generally speaking, previous works can be roughly divided into two types: pattern-based and feature-based methods. Pattern-based methods typically first extract bag-of-patterns [34] or shapelet patterns [46], and then feed them into a classifier. For example, shapelet-based methods usually extract some useful subsequence, which is distinctive for different classes. Then, distance metrics are employed to generate features for classification. DTW has been proved as an effective distance measurement in time series classification. Recently proposed work [34] uses symbolic Fourier approximation to generate discrete units for classification. The strength of these methods is that the generative patterns are usually interpretable [10] while the largest weakness [3, 49] is inevitablely incurring expensive computation in producing discriminative patter features. A series of methods have been proposed to overcome this weakness by either speeding up the computation of distance or constructing the dictionary [46]. In contrast to pattern-based methods, feature-based methods can be more efficient by depending on hand-crafted static features based on domain experts. However, it is difficult to design good features to capture intrinsic properties embedded in various time series data. Therefore, the accuracy of feature-based methods is usually worse than that of sequence distance-based ones. Recently, extracting time series feature with deep neural networks [22, 24, 50] gradually become prevalent in time series classification tasks. As a pioneering attempt, multi-channel convolutional networks [50] have been proposed to deal to capacity of classification for the MTSC problem. Later, a series of convolutional-based methods, like ResNet [19], InceptionTime [23] have also proposed to achieve remarkable success due to their powerful representation ability in extracting time series features. ROCKET [12] employs random convolution kernels to train linear classifiers and has been recognized as a powerful method in recent empirical studies [31]. Other than pure convolutional based

methods, [24] perform time series classification by designing a hybrid network, in which both the LSTM layer and stacked convolutional layer along with a squeeze-and-excitation block are simultaneously used to extract features. Besides, mining graph structure among time series with graph convolutional networks for comprehensive analysis have also attracted some researchers [15, 45].

2.2 Transformers in Time Series.

Designed for sequence modeling, the transformer network [40] recently achieves great success in nature language processing (NLP). Among multiple advantages of transformers, the ability to capture long-range dependencies and interactions is especially attractive for time series modeling. Hence, a large body of transformer-based methods [43, 51] has been proposed by attempting leveraging transformers for time series forecasting or regression. For time series classification, recent advancements still lie in the early stage and mainly focus on multivariate time series classification. [32] studies the transformer for raw optical satellite time series classification and obtains the latest results comparing with convolution-based solutions. GTN [26] explores an extension of the transformer network by modeling both channel-wise and step-wise correlations, simultaneously. Besides, pre-trained transformers are also investigated in time series classification tasks. For example, TST [48] employs the transformer network to learn unsupervised representations of time series so as to alleviate the data sparsity issue. Despite their effectiveness, both the efficiency issues incurred by the self-attention and the properties of time series classification task are largely ignored in these current works. Although some prior works focusing on improving the efficiency of self-attention were developed [38], these works mainly use heuristic strategies to perform sparse attention computation without global context modeling. In this work, we aim to tackle the efficiency bottleneck and achieve performance improvements in the proposed FormerTime.

3 THE FORMERTIME MODEL

In this section, we first formally introduce the definition of multivariate time series classification (MTSC) problem. Then, we introduce the overall architecture of our designed FormerTime. After that, we elaborate the FormerTime via respectively introducing two aspects of key designs, i.e., the hierarchical architecture and the designed transformer encoder. Finally, we demonstrate the difference between the FormerTime and other relative methods.

3.1 Problem Definitions

We introduce the definitions and notations used in the following. $\mathbb{D} = (X^1, y^1), (X^2, y^2), ..., (X^n, y^n)$ is a dataset containing a collection of pairs (X^i, y^i), in which n denotes the number of examples and X^i denote a multivariate time series with its corresponding label denoted by y^i . Each multivariate time series $X = [x_1, x_2, ..., x_l]$ contains l ordered elements with m dimensions in each time step. The task of multivariate time series classification is to learn a classifier on $\mathbb{D}$ so as to map from the space of inputs X to a probability distribution over the class y.

3.2 Model Architecture Overview

An overview of the FormerTime is depicted in Figure 1. The input of the whole model is a set of multivariate time series, involving multiple dimensions (a.k.a. channels). For each dimension, the time series share the same sequence length. To produce multi-scale representations for time series data, we adopt a hierarchical architecture. Specifically, we divide the whole deep network architecture into multiple different stages so as to generate feature maps on various time scales. For simplicity, all stages share a similar architecture, which is composed of a temporal slice partition processing operation and successive L_i our designed transformer encoder layers. Relying upon such hierarchical architecture, the time series representation on various scales can be effectively extracted. Meanwhile, we use the mean pooling operation over the representation of each temporal point to denote the time series representations. The model's output is the predicted distribution of each possible label class for the given input time series. The FormerTime model can be trained end-to-end by minimizing the cross-entropy loss [7], and the loss is propagated back from the prediction output across the entire network. The following sections would mainly introduce several key designs in the FormerTime.

3.3 Multi-scale Time Series Representations

Integrating information on different time scales [5] is essential to the classification capacity in the MTSC task. However, the vanilla transformer model only can produce fixed-scale representations of the sequence input. Thus, we devise a hierarchical architecture for learning time series representations on various time scales. The key idea is that the whole model is divided into several stages so as to hierarchically performing feature maps. Here, we vary the time scale input for different stages by leveraging a temporal slice partition strategy, i.e., aggregating successive neighborhood points with a window slicing operation.

Specifically, suppose that we have sequence input $X = \{x_1, x_2, ...x_l\}$ in stage j, and the window slicing size is s_j, which denotes the scale size of the processed time series. Every s_j successive points will be grouped into a new temporal slice. Considering the semantic gap issue, we then feed these temporal slices to a trainable linear projection layer to project this raw feature to a new dimension C_j. Notably, the weights of this linear projection layer is weight-shared, in which we use d_j to denote the stride of projection operations. To some extent, these temporal slices can be regarded as the "tokens" of new time series, analogous to the relationship between word and whole speech input. Assume that the whole model is divided into three stages, and the fine-grained raw time series can be processed into a new granularity version, which contains $\frac{l}{s_1}$ slices and each size is $s_1 \times m$. Then, the linear projection layer project it into a new dimension C_1, and the output is reshaped to size of $\mathbf{F}_1 \in \mathbb{R}^{\frac{l}{s_1} \times C_1}$. After that, the normalized embeddings of each temporal slice along with its positional embeddings are fed into the transformer encoder with L_1 layers. In the same manner, by using the feature maps of the previous stage's output as the input of the next stage, different scales of time series representations, denoted by $\mathbf{F}_j$, can be effectively learned by stage-wise propagation.

The primary motivation is the information density difference between time series and language domain. Unlike the tokens in language data, which are human-generated signals and highly semantic and information-intensive, the time series are naturally redundant, e.g., a missing point can be easily recovered from its neighbors. The benefits of adopting temporal slice operation are two-fold. First, the time scales of sequence data can be flexibly

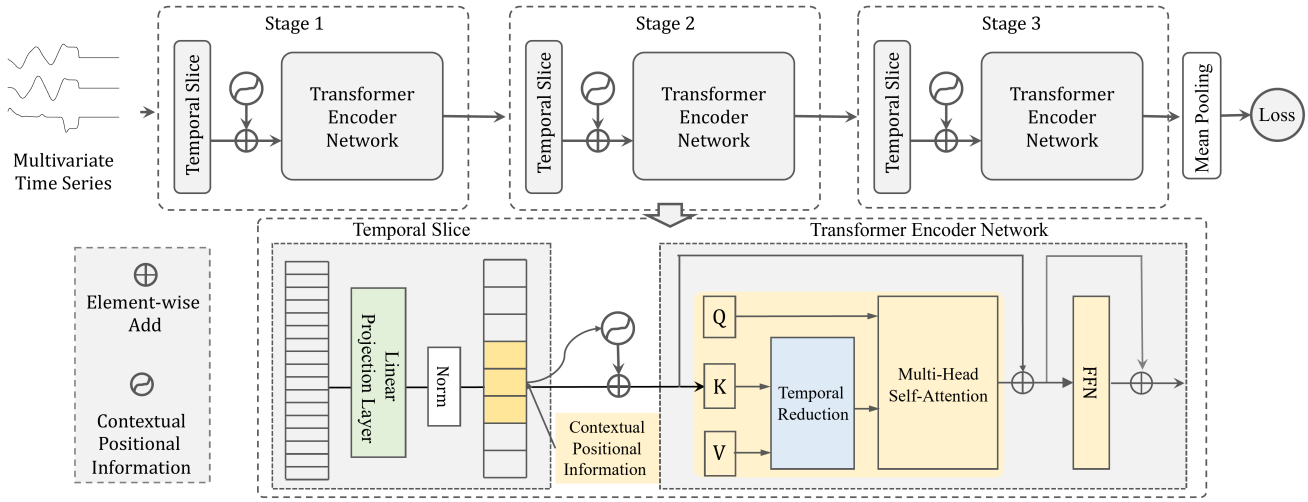

Figure 1: Illustration of the FormerTime, i.e., a efficient hierarchical transformer architecture for the MTSC task.

transformed, naturally forcing the network to generate hierarchical feature maps. Second, with this partitioning strategy, the sequence length of the whole time series can be largely reduced before sent into the encoder, saving an amount of computation consumption.

3.4 The Transformer Encoder Network

In this subsection, we will introduce our designed encoder to extract the robust global contexts of the whole sequence input.

3.4.1 Temporal Reduction Attention. To capture the global contexts of the whole time series, we would like to benefit from regarding the transformer network as the encoder to perform non-linear hidden representation transformation. One of the core designs of the vanilla transformer network is to employ a multi-head self-attention mechanism. Each temporal point needs to be computed the attention scores among all other sequence points with the inner product so as to capture the long-range dependence. However, the computation complexity incurred by the attention operation can be very heavy, growing quadratically in the sequence length of the input sequence. Unlike language data, the sequence length of time series data regularly can be very long to uncover the event of classification tasks. Directly leveraging the standard attention computation strategy is memory-expensive, making it hard for transformers to be applicable in time series.

To solve this dilemma, inspired by recent works [41], we present a novel attention computation strategy named temporal reduction attention (TRA) mechanism to replace the vanilla self-attention strategy. The core idea of TRA is to compute the attention with a sub-sampled version of all input points. To be more specific, similar to standard self-attention, our TRA receives a query Q, a key K, and a value V as input, and outputs a refined feature. Details of TRA operation in stage j can be formulated as follows:

$$\mathrm{TRA}(\mathbf{Q},\mathbf{K},\mathbf{V}) = \mathrm{Concat}(\mathrm{head}_0,...,\mathrm{head}_{\mathrm{N_j}})\mathbf{W}^{\mathrm{O}}, \tag{1}$$

in which the Concat denotes the concatenation operation, and N_i is the number of heads in the attention layer.

$$\mathrm{head_j} = \mathrm{Attention}(\mathbf{Q}\mathbf{W}_{\mathrm{j}}^{\mathrm{Q}}, \mathrm{TR}(\mathbf{K})\mathbf{W}_{\mathrm{j}}^{\mathrm{K}}, \mathrm{TR}(\mathbf{V})\mathbf{W}_{\mathrm{j}}^{\mathrm{V}}), \tag{2}$$

where $\mathbf{W}_{\mathrm{j}}^{\mathrm{Q}} \in \mathbb{R}^{\mathrm{C_j}\times \mathrm{d_{head}}}$, $\mathbf{W}_{\mathrm{j}}^{\mathrm{K}} \in \mathbb{R}^{\mathbf{C}_{\mathrm{j}}\times \mathrm{d_{head}}}$, $\mathbf{W}_{\mathrm{j}}^{\mathrm{V}} \in \mathbb{R}^{\mathbf{C}_{\mathrm{j}}\times \mathrm{d_{head}}}$ are linear projection parameters. In this way, the dimension of each head is equal to $\frac{\mathbf{C}_j}{N_j}$. Here, TR indicates the temporal reduction on K and V, which can be written as

$$\mathrm{TR(x)} = \mathrm{Norm}(\mathrm{Reshape}(\mathrm{x}, \mathrm{R_j})\mathbf{W}^{\mathrm{T}}), \tag{3}$$

where $\mathbf{x} \in \mathbb{R}^{l_j \times C_j}$ represents a input sequence, and R_i denotes the reduction ratio of the attention layers in stage j. $\mathrm{Reshape}(\mathrm{x}, \mathrm{R_j})$ is an operation of reshaping the input sequence x to a sequence of size $\frac{L_j}{R_j}$ and $\mathbf{W}^T$ is a linear projection that reduces the dimension of the input sequence to C_j. Here, we employ layer normalization operation [40] to implement Norm. Like the original attention computation mechanism, our Attention$(\cdot)$ can be represented as

$$\mathrm{Attention}(\mathrm{q},\mathrm{k},\mathrm{v}) = \mathrm{softmax}(\frac{\mathrm{qk}^{\intercal}}{\sqrt{\mathrm{d_{head}}}})\mathrm{v}. \tag{4}$$

In this way, the computational cost of our attention operation is $\frac{1}{R_i}$ of standard self-attention, so our TRA can handle longer input feature maps/sequences without requiring too many resources. The main difference between our TRA and the prior version is that our TRA reduces the temporal scale of $\mathbf{K}$ and value $\mathbf{V}$ before the attention operation, largely reducing the computational/memory overhead. Note that the capacity of global context modeling is still well-remained in our attention layer.

3.4.2 Contextual Positional Encoding. As claimed in [9, 40], positional information is a key operation in the success of the transformer network. The main reason is that the self-attention operation can be used to preserve the order of the sequence property of input data. Two types of position information in transformers have been widely adopted, including absolute and relative encodings, which respectively denote static and learnable embeddings. For the former type, absolute temporal information provides helpful cues for whenever the object would appear in the whole time series. Despite its effectiveness, it severely weakens the capacity of temporal-invariant since each temporal slice is added with a unique positional encoding. In fact, the temporal invariance plays a significant role in time series classification tasks since we hope the model to release the same response whenever the discriminative

pattern appears in the time series. Though relative positional encodings can greatly alleviate the aforementioned issues, the relative encodings lack absolute position information, which is important in classification tasks [9, 21]. Previous works have uncovered that one of the main merits of such positional information is that absolute information can be added for enhancing classification performance. Besides, we also argue that these two types of positional information actually model each temporal slice individually and only achieve sub-optimal performance because that extracted patterns from time series evolve in time and highly depend on their surrounding points.

Based on the above analysis, we hold that the well-informed positional information should possess two aspects of characteristics. First, making the input sequence permutation-variant but temporal-invariant is a necessity for time series classification. Second, having the ability to provide absolute information also matters. To fulfill the two demands, we find that characterizing the contextual information among neighboring temporal slices can be sufficient. As shown in Figure 1, we use 1-D convolutional kernel size k along with $\frac{k}{2}$ zero paddings to extract the localized contextual information as positional encodings. Note that the zero padding is vital to make the model aware of the absolute position information.

3.4.3 Entire Transformer Encoder Network. Based on the designed temporal reduction attention mechanism and context positional encodings, we organize them together to form a novel transformer encoder block for learning the time series representations. We give a sketch of the newly designed encoder at the bottom of Figure 1. On the basis of the design of standard transformer encoder [40], the entire encoder is composed of successive layers of the TRA layer and followed by a feed-forward neural network (FFN) layer. These two layers are further wrapped with a residual connection to avoid the the vanishing gradient problem. Particularly, we set a trainable parameter α, initialized with zero, according to the previous work [1]. Such a simple trick can further help the FormerTime converge more stable.

3.5 Summary and Remarks

In the following, we summarize the characteristics of FormerTime and discuss its relations to transformer-based and convolutional-based approaches in the multivariate time series classification.

Relation to Transformer-based Models. Transformer architecture has shown its superiority in global sequence modeling tasks. However, the dot-product computation in self-attention easily incurs quadratic computation and memory consumption on sequence input. Hence, the efficiency of the transformer architecture becomes the bottleneck of applying them to time series classification tasks. Besides, the vanilla transformer model lacks some key designs of inductive bias, such as multi-scale representation and temporal invariance, which can greatly benefit the time series classification task. Although some prior works of efficient transformer variants [38] have been proposed, they fail to solve these two aspects of limitations, simultaneously. In contrast, the proposed FormerTime not only breaks the bottleneck of efficiency but also achieves performance improvements in the MTSC task.

Relation to Convolutional-based Models. Recently, researchers have demonstrated the extremely expressive capacity of convolutional models in MTSC tasks. In fact, convolutional networks can yield several aspects of strength: 1) their memory and time complexity in feature extraction are not constrained by the sequence length of the sequence input, 2) they can easily learn time series with various scales of feature maps with varying strides and preserve the prior capacity of temporal-invariant with the weight-sharing mechanism for the classification task. However, convolution operation cannot achieve a global receptive field of whole sequence input while the global context information is vital for the classification capacity. Our proposed FormerTime not only absorbs the strength of convolutional models but also meets the demands of long-range dependence modeling.

Table 1: Statics of datasets in the experiments.

Dataset	Train Size	Test Size	Dimensions	Length	Classes
AWR	275	300	9	144	25
AF	15	15	2	640	3
CT	1,422	1,436	3	182	20
CR	108	72	6	1,197	12
FD	5,890	3,524	144	62	2
FM	316	100	28	50	2
MI	278	100	64	3,000	2
SRS1	268	293	6	896	2
SRS2	200	180	7	1,152	2
UWG	120	320	3	315	8

4 EXPERIMENTS

4.1 Experimental Setup

4.1.1 Datasets. We perform all experiments by conducting experiments on ten public datasets, which are selected from the well-known UEA multivariate time series classification (MTSC) archive. In reality, the UEA archive has become nearly the most widely used multivariate time series benchmarks. We select a set of 10 multivariate datasets from the UEA archive [2] with diverse characteristics in terms of the number, and the length of time series samples, as well as the number of classes. Specifically, we choose: ArticularyWordRecognition (AWR), Atrial Fibrillation (AF), CharacterTrajectories (CT), Cricket, FaceDetection (FD), FingerMovements (FM), MotorImagery(MI), SelfRegulationSCP1 (SRS1), SelfRegulationSCP2 (SRS2), UWaveGestureLibrary (UW). In these original dataset, training and testing set have been well processed. We do not take any processing for these datasets for a fair comparison. We summarize the main characteristics of dataset in Table 1.

4.1.2 Compared Baselines. For comprehensive evaluation, we choose the following prevalent baseline methods for evaluation: Shapelet Transformation (ST) [25], Learning Shapelet (LS) [18], TST [48], GTN [26], Informer [51], MCDCNN [50], MCNN [11], ResNet [19], TCN [4], InceptionTime (IT) [23], MiniROCKET (MR) [13]. Among them, ST and LS are two shapelet-based methods. TST and GTN are two transformer-based models proposed for time series classification. Though Informer is originally proposed for time series forecasting, we also treat it as a competitive baseline to verify the effectiveness of our FormerTime. In addition, the remaining compared baselines are convolutional-based models applied to the MTSC problem. Note that some traditional classifiers [28] are not considered here, since it is difficult to construct hand-crafted features for all time series. Also, we do not choose well-known distance-based

Table 2: Detailed Hyper-parameter settings of the proposed FormerTime.

Datasets	Stage 1		Stage 2		Stage 3	
	Temporal Slice	Encoder	Temporal Slice	Encoder	Temporal Slice	Encoder
AWR	s_1=2,C_1=64,d_1=2		s_2=2,C_2=64,d_2=2		s_3=2,C_3=64,d_3=2	
AF	s_1=16,C_1=64,d_1=8		s_2=2,C_2=64,d_2=2		s_3=2,C_3=64,d_3=2	
CT	s_1=16,C_1=64,d_1=8		s_2=2,C_2=64,d_2=2		s_3=2,C_3=64,d_3=2	
CR	s_1=8,C_1=64,d_1=8		s_2=2,C_2=64,d_2=2		s_3=2,C_3=64,d_3=2	
FD	s_1=2,C_1=64,d_1=2	L_1=6	s_2=2,C_2=64,d_2=2	L_2=6	s_3=2,C_3=64,d_3=2	L_3=6
FM	s_1=4,C_1=64,d_1=4	R_1=2	s_2=2,C_2=64,d_2=2	R_2=2	s_3=2,C_3=64,d_3=2	R_3=1
MI	s_1=8,C_1=64,d_1=8	N_1=4	s_2=2,C_2=64,d_2=2	N_2=4	s_3=2,C_3=64,d_3=2	N_3=4
SRS1	s_1=2,C_1=64,d_1=2		s_2=2,C_2=64,d_2=2		s_3=2,C_3=64,d_3=2	
SRS2	s_1=16,C_1=64,d_1=8		s_2=2,C_2=64,d_2=2		s_3=2,C_3=64,d_3=2	
UWG	s_1=8,C_1=64,d_1=8		s_2=2,C_2=64,d_2=2		s_3=2,C_3=64,d_3=2	

Table 3: Classification performance of compared methods in ten datasets. Bold numbers represent the best results.

Datasets	IT	LS	ST	MCDCNN	TCN	MCNN	ResNet	MR	TST	GTN	Informer	Ours
AWR	0.9827	0.9127	0.8700	0.7800	0.9467	0.8200	0.9827	0.9720	0.9789	0.9767	0.9820	**0.9847**
AF	0.4400	0.2533	0.2667	0.3733	0.4933	0.3467	0.4000	0.3333	0.4000	0.4000	0.4267	**0.6000**
CT	**0.9983**	0.9866	0.7224	0.8826	0.9915	0.9238	0.9965	0.9876	0.9882	0.9783	0.9862	0.9914
CR	0.9889	0.9639	0.9722	0.6278	0.9083	0.9167	**0.9972**	0.9806	0.9583	0.7917	0.9778	0.9806
FD	0.6820	0.5129	0.5085	0.5000	0.6801	0.6747	0.5760	0.6065	0.6005	0.5542	0.5265	**0.6872**
FM	0.6000	0.4840	0.4940	0.5920	0.5880	0.5920	0.6080	**0.6380**	0.5900	0.5350	0.6120	0.6180
MI	0.5860	0.5180	0.6100	0.5000	0.6040	0.5980	0.5780	0.5640	N/A	N/A	0.6240	**0.6320**
SRS1	0.8942	0.7038	0.6724	0.9079	0.9031	0.8949	0.8730	**0.9352**	0.8771	0.8019	0.9188	0.8867
SRS2	0.5689	0.5111	0.5300	0.5256	0.5978	**0.5989**	0.5622	0.5411	0.5796	0.5611	0.5767	0.5922
UWG	0.8869	0.8031	0.7769	0.8438	0.7981	0.8044	0.7994	**0.9075**	0.8271	0.8406	0.8363	0.8881
Average	0.7628	0.6649	0.6423	0.6533	0.7511	0.7170	0.7373	0.7466	0.7555	0.7155	0.7467	**0.7861**
MACs (M)	89	-	-	263	283	929	132	-	408	1,565	141	98

methods, like HIVE-COTE [28], as baseline due to their expensive computation consumption. We adopt accuracy as the metric.

4.1.3 Implement Details. For learning shapelet (LS), we adopt the codes in [1] while adopting the publicly available codes [2] to run shapelet transformation (ST). To implement GTN, we use the source code provided by the corresponding authors [3]. We implement TST by strictly following the network architecture settings of original works using PyTorch. We replace the decoder in Informer[4] with a linear classifier layer so as to adapt it for MTSC tasks. The remaining baselines' code consistently leverages the codes in [5]. For full reproducibility of the experiments, we release our codes and make it available [6]. The specific hyper-parameters of our FormerTime are listed as follows:

- s_j: the temporal slice size of stage j;
- C_j: the hidden size of the output in stage j;
- d_j: the stride size of window slicing operation of stage j;
- L_j: the number of transformer encoders in stage j;
- R_j: the temporal reduction rate in stage j;
- N_j: the number of attention heads of temporal reduction attention in stage j.

[1]https://tslearn.readthedocs.io/
[2]https://pyts.readthedocs.io/
[3]https://github.com/ZZUFaceBookDL/GTN
[4]https://github.com/zhouhaoyi/Informer2020
[5]https://timeseriesai.github.io/tsai/
[6]https://anonymous.4open.science/r/FormerTime-A17E/

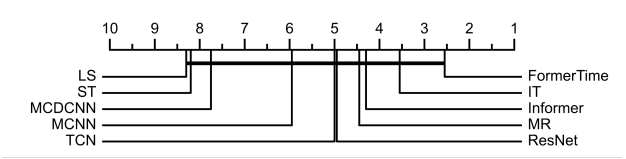

Figure 2: Critical difference diagram over the mean ranks of FormerTime, baseline methods.

More details of hyper-parameter settings in FormerTime for specific datasets can be found in Table 2. For common hyper-parameters of all models, we set the embedding size as 64. The initialized learning rate is set to 1×10^{-3} without additional processing, and we employ Adam optimizer to guide all model training. All other hyper-parameters and initialization strategies either follow the suggestions from the original works' authors or are tuned on testing datasets. We report the results of each baseline under its optimal hyper-parameter settings. For a fair comparison, all models are trained on the training set and report the accuracy score on the testing set. All models are trained until achieving the best results. **All experiments in our work are repeated for 5 times with 5 different seeds, and we reported the mean value score.**

4.2 Experimental Results

4.2.1 Classification Performance Evaluation. Table 3 summarizes the classification accuracy of all compared methods while Figure 2 reports the critical difference diagram as presented in [14]. The results of "N/A" indicates that the corresponding results cannot

Table 4: Experimental results w.r.t. studying the hyper-parameter sensitivity with varying stages.

Datasets	1	2	3	4
AWR	**0.9811**	**0.9811**	0.9720	0.9767
AF	0.4222	0.4667	**0.6000**	0.5778
CT	0.9907	0.9909	**0.9914**	0.9902
CR	**0.9861**	0.9815	0.9806	0.9769
FD	0.6750	**0.6793**	0.6776	0.6748
FM	**0.6200**	0.6033	0.6140	0.6067
MI	0.6200	0.6267	**0.6280**	0.6133
SRS1	0.8760	0.8692	0.8771	**0.8840**
SRS2	0.5722	0.5815	**0.5922**	0.5889
UWG	**0.9021**	0.8948	0.8844	0.8844
Averge	0.7645	0.7675	**0.7817**	0.7774

be run due to the out-of-memory issue. Overall, the accuracy of our proposed FormerTime could outperform previous classifiers on average. Such results demonstrate the success of FormerTime in enhancing the classification capacity in the MTSC problem. For each dataset, the classification performance of FormerTime is either the most accurate one or very close to the best one. These existing proposed models typically cannot always achieve the most distinct results. One may wonder whether the FormerTime can be effective enough. However, the experimental results are largely consistent with previous empirical studies [3, 31], i.e., one single model cannot always achieve superior performances in all scenarios. In particular, we observe that FormerTime could surpass other baselines to a large margin in datasets of AF and MI, in which the sequence length of these two datasets is very long. We guess that our temporal slice setting can be very robust for these two datasets.

For these baseline approaches, we observe that convolutional-based methods, like MR, exhibit strong classification performances in some datasets, which is analogous to the experimental results of recent empirical studies [31]. We hold the characteristics of multi-scale representation and temporal invariance of the convolution operations make a great contribution. Besides, in MR, the feature of PPV, denoting the proportion of positive values of extracted deep representations, also matters. However, for transformer-based classifiers, it seems that the performance cannot always outperform convolutional algorithms. We guess the main reason behind the performance is that: (1) the plain transformer architecture fails to learn hierarchical feature maps from time series data, and (2) the naive positional information might not be suitable for modeling time series since the semantic information of one single temporal point individually modeled. Besides, compared to these deep learning-based methods, shapelet-based methods exhibit the worst classification performance due to a lack of poor representation capacity. However, in shapelet-based approaches, interpretable sub-sequence patterns can be extracted to make the model more understandable, which is vital in some applications.

In time series classification tasks, model efficiency has always been an important concern. Here, we also show the computation cost by recording MACs[7] of compared methods. Note that only methods trained with end-to-end manner are reported. Though we adopt a self-attention operation, the computation cost of our methods can be very economical. Particularly, compared to the standard transformer network, our proposed FormerTime could significantly save computation costs. The main reason could be attributed to two aspects: 1) we model the raw time series with hierarchical architecture, which significantly shorten the sequence length, and 2) we develop a temporal reduction layer to ensure each input point can attend to all other data points.

[7] https://github.com/Lyken17/pytorch-OpCounter

Table 5: Experimental results w.r.t. studying the hyper-parameter sensitivity w.r.t. temporal slice size.

Datasets	[16,32,64]	[8,16,32]	[4,8,16]	[2,4,8]
AWR	0.9720	0.9740	0.9820	**0.9847**
AF	**0.6000**	0.5600	0.4267	0.4400
CT	**0.9914**	0.9886	0.9868	0.9873
CR	**0.9806**	0.9806	0.9778	0.9667
FD	0.6776	**0.6794**	0.6823	0.6872
FM	0.6140	0.6080	**0.6180**	0.6040
MI	**0.6280**	**0.6280**	0.6160	0.6180
SRS1	0.8771	0.8826	0.8710	**0.8867**
SRS2	**0.5922**	0.5811	0.5856	0.5600
UWG	0.8844	**0.8881**	0.8781	0.8775
Averge	**0.7817**	0.7770	0.7624	0.7612

Table 6: Experimental results w.r.t. studying the effectiveness of contextual positional embeddings.

Datasets	None	Static	Learnable	Ours
AWR	0.9433	0.9822	0.9811	**0.9720**
AF	0.4667	0.5111	0.5556	**0.6000**
CT	0.9821	0.9902	0.9863	**0.9914**
CR	**0.9815**	0.9676	0.9769	0.9806
FD	0.6740	**0.6804**	0.6774	0.6776
FM	0.5900	0.5867	**0.6200**	0.6140
MI	0.6233	0.5833	0.6167	**0.6280**
SRS1	0.8635	**0.8817**	0.8749	0.8771
SRS2	0.5704	0.5759	**0.6018**	0.5922
UWG	0.8479	0.8729	0.8677	**0.8844**
Averge	0.7543	0.7632	0.7758	**0.7817**

4.2.2 Study of Multi-scale Representations. In this part, we decide to study the effectiveness of hierarchical feature maps in FormerTime by setting different types of model variants w.r.t. the different number of stages. Specifically, we report the average classification experimental results of ten datasets in Table 4, varying the number of stages from 1 to 4. Note that the total number of layers is consistent in these model variants to eliminate the other influence factors. From the reported results, we observe that feature maps at various scales can indeed perform much better than the single-scale representation versions. Such results demonstrate the importance of multi-scale representation in time series classification tasks. Furthermore, we also empirically analyze the effectiveness of hierarchical structure by performing hyper-parameter sensitivity analysis on the size of window slicing in temporal slice partition. The average accuracy of ten datasets is recorded in Table 5. An attractive experimental phenomenon is that FormerTime equipped with a large slice size yields more promising results. We guess that the semantic information of a smaller temporal slice is too small to characterize discriminative patterns of distinguishing other examples.

4.2.3 Study of Positional Information Encoding. In the following, we would like to empirically verify the effectiveness of contextual

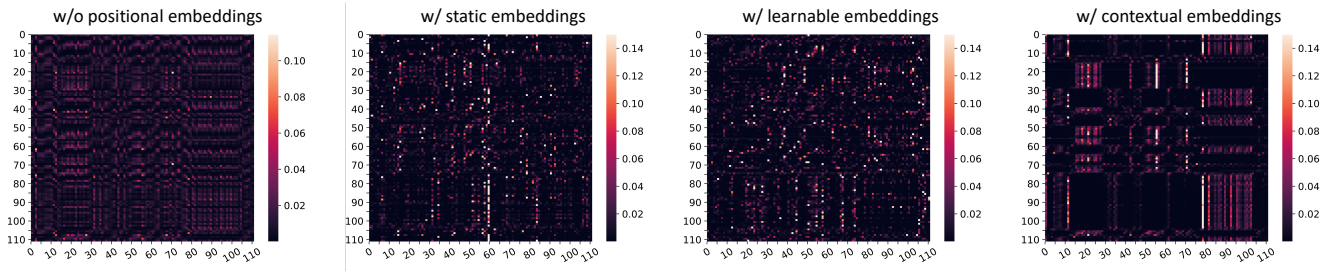

Figure 3: Normalized attention score from the first encoder block of the first stage in FormerTime: (1) without taking positional information into account, (2) using static embeddings, (3) using learnable vectors, (4) using our contextual embeddings.

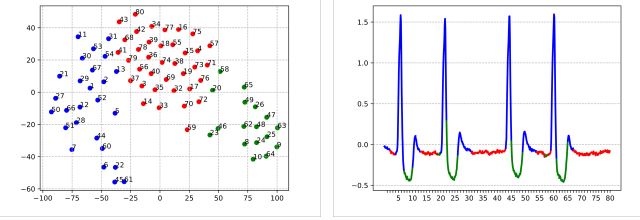

Figure 4: Left plot: Visualization of the t-SNE result of the embedding layer output on the AF dataset. Right plot: visualization of sub-sequences on raw time series data.

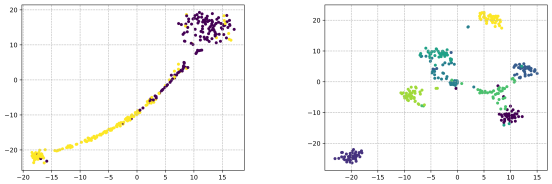

Figure 5: Visualization of the representation of whole time series on the SRS1 (left plot) and UW (right plot) datasets, extracted by pooling operation from the last hidden layer.

positional information. A natural choice is to replace the contextual embedding with static or learnable version [40], respectively. Moreover, we also evaluate the results of FormerTime without leveraging any forms of positional embedding information. The average experimental results conducted on ten datasets are shown in Table 6. From these results, we notice that FormerTime equipped with contextual positional information could surpass all other model variants, verifying the effectiveness of extracting contextual information as positional encodings. Also, FormerTime's performance dramatically degrades while absolutely discarding the positional information. We believe this is reasonable because self-attention computation is permutation-variant, whose performance would dramatically degrade if discarding the positional information. To further deeply understand the scheme of several types of positional encoding, we choose one sample from SRS1 data to visualize the attention weights of corresponding model variants. As shown in Figure 3, unlike the widely adopted static and learnable positional embeddings, more specific attention map patterns could be well learned by our contextual positional information generating strategies. Also, it seems that most of the temporal slices would produce uniform attention weights if we remove the positional information. We believe these visualized cases further demonstrate the effectiveness of our contextual information-generating strategies.

4.2.4 Analyzing Semantic of Time Series Slice Representations. In this part, we aim to analyze the semantic descriptions of constructed temporal slices encoded by the embedding layer. To achieve this goal, we apply t-SNE to reduce the embedding of each temporal slice on an example selected from AF datasets. For better understanding, we further map each temporal slice to the raw time series. As shown in Figure 4, we find that the similarity of raw time series data is well-maintained in projected vector space. Such results reflect that the semantics of time series can be accurately represented by their corresponding latent representations. We also observe that the temporal slice nearby in the vector space forms some successive subsequences (a.k.a. shapelets) in raw time series. This phenomenon indicates that the embedding learned by the FormerTime retains the potential of preserving the strength of shapelet-based methods.

4.2.5 Visualization of the Extracted Embeddings. As shown in Figure 5, we visualize the extracted feature vector from FormerTime by applying t-SNE to reduce the dimension. Here, we randomly choose examples from SRS1 and UWG datasets. In this figure, each point denotes an example and the same color denotes the corresponding original class labels. This figure suggests that the proposed FormerTime is able to project the data into an easily separable space to ensure good classification results.

5 CONCLUSION

In this work, instead of employing the prevalent convolutional architecture as the main backbones, we proposed FormerTime, a hierarchical transformer network for MTSC tasks. In FormerTime, both the strengths of transformers and CNNs were well absorbed in a unified model for further improving the classification capacity. Specifically, two aspects of vital inductive bias, including multi-scale time series representations and temporal-invariant capacity, are incorporated for enhancing the classification capacity in the MTSC task. Moreover, the terrible computation dilemma incurred by the self-attention mechanism was largely overcome in the proposed FormerTime, whose computation costs are acceptable for very long-sequence time series and large-scale data. Extensive experiments conducted on 10 publicly available datasets from the UEA archive demonstrated that FormerTime could surpass previous strong baseline methods. In the future, we hope to empower the transferability of FormerTime [8].

ACKNOWLEDGMENTS

This research was partially supported by grants from the National Key Research and Development Program of China (No. 2021YFF0901003)

REFERENCES

[1] Thomas Bachlechner, Bodhisattwa Prasad Majumder, Henry Mao, Gary Cottrell, and Julian McAuley. 2021. Rezero is all you need: Fast convergence at large depth. In *Uncertainty in Artificial Intelligence*. PMLR, 1352–1361.

[2] 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 preprint arXiv:1811.00075* (2018).

[3] Anthony Bagnall, Jason Lines, Aaron Bostrom, James Large, and Eamonn Keogh. 2017. The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances. *Data mining and knowledge discovery* 31, 3 (2017), 606–660.

[4] Shaojie Bai, J Zico Kolter, and Vladlen Koltun. 2018. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271* (2018).

[5] Wei Chen and Ke Shi. 2021. Multi-scale Attention Convolutional Neural Network for time series classification. *Neural Networks* 136 (2021), 126–140.

[6] Mingyue Cheng, Zhiding Liu, Qi Liu, Shenyang Ge, and Enhong Chen. 2022. Towards Automatic Discovering of Deep Hybrid Network Architecture for Sequential Recommendation. In *Proceedings of the ACM Web Conference 2022*. 1923–1932.

[7] Mingyue Cheng, Fajie Yuan, Qi Liu, Shenyang Ge, Zhi Li, Runlong Yu, Defu Lian, Senchao Yuan, and Enhong Chen. 2021. Learning Recommender Systems with Implicit Feedback via Soft Target Enhancement. *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval* (2021).

[8] Mingyue Cheng, Fajie Yuan, Qi Liu, Xin Xin, and Enhong Chen. 2021. Learning transferable user representations with sequential behaviors via contrastive pre-training. In *2021 IEEE International Conference on Data Mining (ICDM)*. IEEE, 51–60.

[9] Xiangxiang Chu, Zhi Tian, Bo Zhang, Xinlong Wang, Xiaolin Wei, Huaxia Xia, and Chunhua Shen. 2021. Conditional positional encodings for vision transformers. *arXiv preprint arXiv:2102.10882* (2021).

[10] Jonathan Crabbé and Mihaela Van Der Schaar. 2021. Explaining time series predictions with dynamic masks. In *ICML*. PMLR, 2166–2177.

[11] Zhicheng Cui, Wenlin Chen, and Yixin Chen. 2016. Multi-scale convolutional neural networks for time series classification. *arXiv preprint arXiv:1603.06995* (2016).

[12] Angus Dempster, François Petitjean, and Geoffrey I Webb. 2020. ROCKET: exceptionally fast and accurate time series classification using random convolutional kernels. *Data Mining and Knowledge Discovery* 34, 5 (2020), 1454–1495.

[13] Angus Dempster, Daniel F Schmidt, and Geoffrey I Webb. 2021. Minirocket: A very fast (almost) deterministic transform for time series classification. In *ACM SIGKDD*. 248–257.

[14] Janez Demšar. 2006. Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine learning research* 7 (2006), 1–30.

[15] Ziheng Duan, Haoyan Xu, Yueyang Wang, Yida Huang, Anni Ren, Zhongbin Xu, Yizhou Sun, and Wei Wang. 2022. Multivariate time-series classification with hierarchical variational graph pooling. *Neural Networks* 154 (2022), 481–490.

[16] Ge Gao, Qitong Gao, Xi Yang, Miroslav Pajic, and Min Chi. [n. d.]. A Reinforcement Learning-Informed Pattern Mining Framework for Multivariate Time Series Classification. ([n. d.]).

[17] Hardik Goel, Igor Melnyk, and Arindam Banerjee. 2017. R2n2: Residual recurrent neural networks for multivariate time series forecasting. *arXiv preprint arXiv:1709.03159* (2017).

[18] Josif Grabocka, Nicolas Schilling, Martin Wistuba, and Lars Schmidt-Thieme. 2014. Learning time-series shapelets. In *ACM SIGKDD*. 392–401.

[19] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *IEEE CVPR*. 770–778.

[20] Min Hou, Chang Xu, Zhi Li, Yang Liu, Weiqing Liu, Enhong Chen, and Jiang Bian. 2022. Multi-Granularity Residual Learning with Confidence Estimation for Time Series Prediction. In *Proceedings of the ACM Web Conference 2022*. 112–121.

[21] Md Amirul Islam, Sen Jia, and Neil DB Bruce. 2020. How much position information do convolutional neural networks encode? *arXiv preprint arXiv:2001.08248* (2020).

[22] Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar, and Pierre-Alain Muller. 2019. Deep learning for time series classification: a review. *Data mining and knowledge discovery* 33, 4 (2019), 917–963.

[23] 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. *DMKD* 34, 6 (2020), 1936–1962.

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

[25] Jason Lines, Luke M Davis, Jon Hills, and Anthony Bagnall. 2012. A shapelet transform for time series classification. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*. 289–297.

[26] Minghao Liu, Shengqi Ren, Siyuan Ma, Jiahui Jiao, Yizhou Chen, Zhiguang Wang, and Wei Song. 2021. Gated transformer networks for multivariate time series classification. *arXiv preprint arXiv:2103.14438* (2021).

[27] Zhiding Liu, Mingyue Cheng, Zhi Li, Qi Liu, and Enhong Chen. 2022. One Person, One Model—Learning Compound Router for Sequential Recommendation. *2022 IEEE International Conference on Data Mining (ICDM)* (2022), 289–298.

[28] Matthew Middlehurst, James Large, Michael Flynn, Jason Lines, Aaron Bostrom, and Anthony Bagnall. 2021. HIVE-COTE 2.0: a new meta ensemble for time series classification. *Machine Learning* 110, 11 (2021), 3211–3243.

[29] Jeeheh Oh, Jiaxuan Wang, and Jenna Wiens. 2018. Learning to exploit invariances in clinical time-series data using sequence transformer networks. In *Machine Learning for Healthcare Conference*. PMLR, 332–347.

[30] Beanbonyka Rim, Nak-Jun Sung, Sedong Min, and Min Hong. 2020. Deep learning in physiological signal data: A survey. *Sensors* 20, 4 (2020), 969.

[31] Alejandro Pasos Ruiz, Michael Flynn, James Large, Matthew Middlehurst, and Anthony Bagnall. 2021. The great multivariate time series classification bake off: a review and experimental evaluation of recent algorithmic advances. *DMKD* 35, 2 (2021), 401–449.

[32] Marc Rußwurm and Marco Körner. 2020. Self-attention for raw optical satellite time series classification. *ISPRS journal of photogrammetry and remote sensing* 169 (2020), 421–435.

[33] Pritam Sarkar and Ali Etemad. 2020. Self-supervised ECG representation learning for emotion recognition. *IEEE Transactions on Affective Computing* (2020).

[34] Patrick Schäfer and Ulf Leser. 2017. Multivariate time series classification with WEASEL+ MUSE. *arXiv preprint arXiv:1711.11343* (2017).

[35] Omer Berat Sezer, Mehmet Ugur Gudelek, and Ahmet Murat Ozbayoglu. 2020. Financial time series forecasting with deep learning: A systematic literature review: 2005–2019. *Applied soft computing* 90 (2020), 106181.

[36] Chang Wei Tan, Angus Dempster, Christoph Bergmeir, and Geoffrey I Webb. 2022. MultiRocket: multiple pooling operators and transformations for fast and effective time series classification. *DMKD* (2022), 1–24.

[37] Wensi Tang, Guodong Long, Lu Liu, Tianyi Zhou, Michael Blumenstein, and Jing Jiang. 2021. Omni-Scale CNNs: a simple and effective kernel size configuration for time series classification. In *International Conference on Learning Representations*.

[38] Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. 2020. Efficient transformers: A survey. *ACM Computing Surveys (CSUR)* (2020).

[39] Alasdair Tran, Alexander Mathews, Cheng Soon Ong, and Lexing Xie. 2021. Radflow: A recurrent, aggregated, and decomposable model for networks of time series. In *Proceedings of the Web Conference 2021*. 730–742.

[40] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).

[41] Sinong Wang, Belinda Z Li, Madian Khabsa, Han Fang, and Hao Ma. 2020. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768* (2020).

[42] Wenhai Wang and Enze at al Xie. 2021. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. In *IEEE/CVF ICCV*. 568–578.

[43] Qingsong Wen, Tian Zhou, Chaoli Zhang, Weiqi Chen, Ziqing Ma, Junchi Yan, and Liang Sun. 2022. Transformers in time series: A survey. *arXiv preprint arXiv:2202.07125* (2022).

[44] Haixu Wu, Jialong Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. 2022. Flowformer: Linearizing Transformers with Conservation Flows. In *International Conference on Machine Learning*.

[45] Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, Xiaojun Chang, and Chengqi Zhang. 2020. Connecting the dots: Multivariate time series forecasting with graph neural networks. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*. 753–763.

[46] Lexiang Ye and Eamonn Keogh. 2009. Time series shapelets: a new primitive for data mining. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. 947–956.

[47] Yi Yin and Pengjian Shang. 2016. Forecasting traffic time series with multivariate predicting method. *Appl. Math. Comput.* 291 (2016), 266–278.

[48] 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*. 2114–2124.

[49] Xuchao Zhang, Yifeng Gao, Jessica Lin, and Chang-Tien Lu. 2020. Tapnet: Multivariate time series classification with attentional prototypical network. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 6845–6852.

[50] Yi Zheng, Qi Liu, Enhong Chen, Yong Ge, and J Leon Zhao. 2014. Time series classification using multi-channels deep convolutional neural networks. In *International conference on web-age information management*. Springer, 298–310.

[51] 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. In *Proceedings of the AAAI*, Vol. 35. 11106–11115.