



Minerva Access is the Institutional Repository of The University of Melbourne

Author/s:

Le, X-M;Luo, L;Aickelin, U;Tran, M-T

Title:

ShapeFormer: Shapelet Transformer for Multivariate Time Series Classification

Date:

2024-08-25

Citation:

Le, X. -M., Luo, L., Aickelin, U. & Tran, M. -T. (2024). ShapeFormer: Shapelet Transformer for Multivariate Time Series Classification. Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, pp.1484-1494. ACM. <https://doi.org/10.1145/3637528.3671862>.

Persistent Link:

<https://hdl.handle.net/11343/350477>

License:

CC BY



ShapeFormer: Shapelet Transformer for Multivariate Time Series Classification

Xuan-May Le
xuanmay.le@student.unimelb.edu.au
The University of Melbourne
Melbourne, Victoria, Australia

Uwe Aickelin
uwe.aickelin@unimelb.edu.au
The University of Melbourne
Melbourne, Victoria, Australia

Ling Luo
ling.luo@unimelb.edu.au
The University of Melbourne
Melbourne, Victoria, Australia

Minh-Tuan Tran
tuan.tran7@monash.edu
Monash University
Melbourne, Victoria, Australia

ABSTRACT

Multivariate time series classification (MTSC) has attracted significant research attention due to its diverse real-world applications. Recently, exploiting transformers for MTSC has achieved state-of-the-art performance. However, existing methods focus on generic features, providing a comprehensive understanding of data, but they ignore class-specific features crucial for learning the representative characteristics of each class. This leads to poor performance in the case of imbalanced datasets or datasets with similar overall patterns but differing in minor class-specific details. In this paper, we propose a novel Shapelet Transformer (ShapeFormer), which comprises class-specific and generic transformer modules to capture both of these features. In the class-specific module, we introduce the discovery method to extract the discriminative subsequences of each class (i.e. shapelets) from the training set. We then propose a Shapelet Filter to learn the difference features between these shapelets and the input time series. We found that the difference feature for each shapelet contains important class-specific features, as it shows a significant distinction between its class and others. In the generic module, convolution filters are used to extract generic features that contain information to distinguish among all classes. For each module, we employ the transformer encoder to capture the correlation between their features. As a result, the combination of two transformer modules allows our model to exploit the power of both types of features, thereby enhancing the classification performance. Our experiments on 30 UEA MTSC datasets demonstrate that ShapeFormer has achieved the highest accuracy ranking compared to state-of-the-art methods. The code is available at <https://github.com/xuanmay2701/shapeformer>.

CCS CONCEPTS

• Information systems → Data mining; • Computing methodologies → Machine learning.



This work is licensed under a Creative Commons Attribution International 4.0 License.

KDD 2024, Aug 25 - 29, 2024, Barcelona, Spain
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0490-1/24/08
<https://doi.org/10.1145/3637528.3671862>

KEYWORDS

time series; shapelet; transformer; attention; classification

ACM Reference Format:

Xuan-May Le, Ling Luo, Uwe Aickelin, and Minh-Tuan Tran. 2024. ShapeFormer: Shapelet Transformer for Multivariate Time Series Classification. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '24)*, August 25–29, 2024, Barcelona, Spain. ACM, New York, NY, USA, 11 pages. <https://doi.org/10.1145/3637528.3671862>

1 INTRODUCTION

A multivariate time series (MTS) is a collection of data points where each point is composed of multiple variables that have been observed or measured over time. This data structure is prevalent in various fields, such as economics [29], weather prediction [27], education [7], and healthcare [34]. Time series classification stands out as a fundamental and crucial aspect within the domain of time series analysis [30]. However, there are still many challenges in the research on MTS classification (MTSC) [30], especially in capturing the correlations among variables.

Over the past few decades, various approaches have been introduced to enhance the performance of MTSC [12, 23, 33, 41, 48, 50]. Among these, shapelets, which are class-specific time series subsequences, have demonstrated their effectiveness in [14, 21, 23, 44]. This success comes from the fact that each shapelet contains class-specific information representative of its class. It is evident that the distance between the shapelet and the time series of its class is far smaller than the time series of other classes (see Figure 1). Hence, there has been an increased focus on harnessing the capabilities of shapelets in the field of MTSC.

In 2017, Vaswani et al. [40] introduced the breakthrough Transformer architecture, initially designed for Natural Language Processing but later demonstrating success in Computer Vision tasks [8]. Following these successes, Transformer-based models have been effectively applied to MTSC. GTN [26] employs a two-tower multi-headed attention approach to extract distinctive information from input series, SVP-T [50] captures short- and long-term dependencies among subseries using clustering and employing them as inputs for the Transformer, and ConvTran [10] integrates absolute and relative position encoding for improved position embedding in the Transformer model.

Obviously, Transformers utilised in MTSC have demonstrated state-of-the-art (SOTA) performances [10, 47, 50]. Existing methods

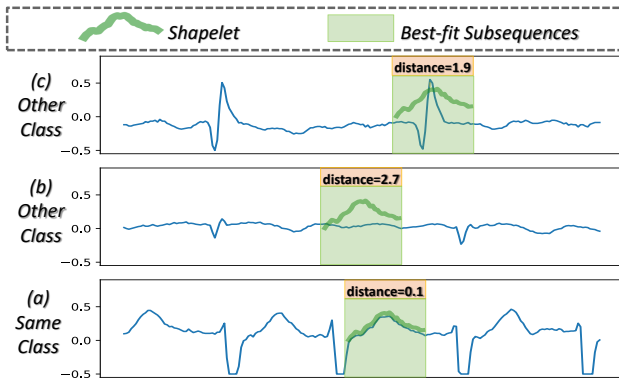


Figure 1: The illustration depicts the shapelet in the Atrial Fibrillation dataset. The best-fit subsequence is the subsequence with the shortest distance to the shapelet in the time series. It is clear that the shapelet can discriminate between classes by utilising their distance to the best-fit subsequences.

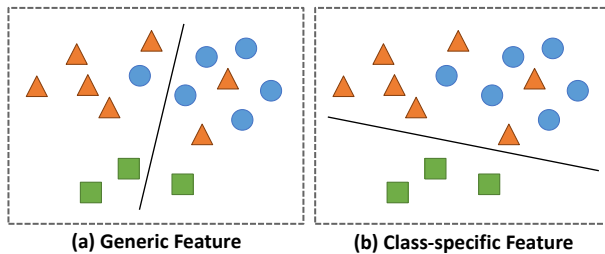


Figure 2: The separating hyperplane using (a) the generic feature has a higher overall accuracy, while the hyperplane using (b) the class-specific feature is better in classifying a single class.

only discover the generic features from timestamps [10, 26, 47] or common subsequences [50] in time series as inputs for the Transformer model to capture the correlation among them. These features merely contain generic characteristics of time series, offering a broad understanding of the data. Nevertheless, they overlook the essential class-specific features necessary to allow the model to capture the representative characteristics of each class. As a result, the model exhibits poor performance in two cases: 1) the dataset has instances that are very similar in overall patterns, differing only in minor class-specific patterns, effective classification cannot be achieved using solely generic features; 2) the imbalanced dataset, where generic features only focus on classifying the majority classes and ignore those of minority. As can be seen in Figure 2, the hyperplane created using the generic feature (Figure 2a) attempts to classify the majority classes (orange triangles and blue circles) and ignores the minority (green squares), while the class-specific feature (Figure 2b) tries to separate each class from the others.

To address the aforementioned problem, we propose a novel method called Shapelet Transformer (ShapeFormer), which comprises class-specific and generic transformer modules to capture both of these features. In the class-specific module, we initially introduce Offline Shapelet Discovery, inspired by [21], to MTS. Based on this, we extract a small number of high-quality shapelets from the training set. Subsequently, we propose a Shapelet Filter that leverages the precomputed shapelets to discover the best-fit subsequences in the input time series. Following this, the Shapelet Filter

learns the difference between the embedding of these shapelets and their most fitting subsequences derived from the input time series. As shown in Figure 1, the distance of shapelets to the time series in the same class is far smaller than the time series of other classes. Similar to the distance, our difference feature also highlights the substantial distinctions among classes. Additionally, rather than using the original shapelets extracted from the dataset, we propose considering these shapelets as the initialisation and then dynamically optimising shapelets during training to effectively represent the distinguishing information. In the generic module, we utilise convolution filters for the extraction of features over all classes. For each module, we employ the transformer encoder to capture the dependencies between their features. Through the integration of these two modules, our ShapeFormer excels in capturing not only class-specific features but also generic characteristics from time series data. This dual capability contributes to an enhancement in the overall performance of classification tasks.

Our contributions can be summarised as follows:

- We introduce ShapeFormer, which effectively captures both class-specific and generic discriminative features in time series.
- We propose the Offline Shapelet Discovery for MTS to effectively and efficiently extract shapelets from training set.
- We propose the Shapelet Filter, which learns the difference between shapelets and input time series, which contain important class-specific features. The shapelets are also dynamically optimised during training to effectively represent the class distinguishing information.
- We conduct experiments on all 30 UEA MTS datasets and demonstrate that ShapeFormer has achieved the highest accuracy ranking compared to SOTA methods.

To the best of our knowledge, our ShapeFormer is a pioneering transformer-based approach that leverages the power of shapelets for MTSC.

2 RELATIVE WORKS

2.1 Multivariate Time Series Classification

We categorise the MTSC methods into two main categories: non-deep learning, and deep learning.

Non-deep learning methods. They primarily utilise distance measures [38, 39], such as Euclidean Distance [20], Dynamic Time Warping, and its diverse variants [3, 19], to calculate the similarity between time series. Otherwise, they leverage special features, such as bag of patterns [24], Symbolic Aggregate approximation [22], bag of SFA symbols [32], and convolution kernel features [5, 35] for classification. [31] gives a comprehensive survey of the conventional methods mentioned.

Deep learning methods. Various neural network methods were proposed for MTSC [16]. Specifically, the LSTM-FCN [17] model features an LSTM layer and stacked CNN layers which directly extract features from time series. These features are subsequently fed into a softmax layer to produce class probabilities. However, it has a limitation in capturing long dependencies among different variables. To address this, Hao et al [15], proposed to use of two cross-attention modules to enhance their CNN-based model. TapNet [48] constructs an attentional prototype network that incorporates LSTM, and CNN to learn multi-dimensional interaction

features. RLPAM [12] adopts a unique approach by transforming MTS into a univariate cluster sequence and subsequently employs reinforcement learning for pattern selection. WHEN [41] was proposed to learn heterogeneity by utilising a hybrid attention network, incorporating both DTW attention and wavelet attention.

2.2 Transformer-Based Time Series Classifiers

In 2017, Vaswani et al. introduced the Transformer architecture, achieving a breakthrough in Natural Language Processing [40] and demonstrating notable success in Computer Vision tasks [8]. Recently, it has proven effective in time series classification tasks. Specifically, GTN [26] utilises a two-tower multi-headed attention approach for extracting distinctive information from the input series. The integration of the output from the two towers is achieved through gating, implemented by a learnable matrix. ConvTran [10] was proposed to enhance the position embedding by leveraging both absolute and relative position encoding. SVP-T [50] uses clustering to identify time series subsequences and employs them as inputs for the Transformer, enabling the capture of long- and short-term dependencies among subseries. Recently, the application of pretrained transformer-based self-supervised learning models like BERT [6] has achieved significant success not only in the field of NLP but also in other areas [36, 37, 43, 45]. Inspired by these successes, many models attempt to adopt a similar structure for time series classification [46, 47]. It is noteworthy that most previous transformer-based methods effectively exploit the generic information of time series.

2.3 Shapelet Discovery for Time Series

Shapelets refer to short subsequences within time series that contain class-specific information by exhibiting a small distance to the time series of the target class and a larger distance to other classes (see Figure 1). Additionally, each shapelet can encompass crucial subsequences located at different positions and variables within a time series. This coverage enables them to effectively represent the time series. In the last decade, the effectiveness of shapelets for time series has been proven by many related studies [13, 21, 23, 25, 44]. The original shapelet discovery method [44] extracts all possible subsequences in the training set and considers the subsequences as shapelets when they have the highest information gain ratio. It requires excessive computing time and is hard to apply to MTSC. Other methods use random shapelets that lack position and variable information [14], or employ the common subsequences as shapelets, which unfortunately have limited discriminative features [23]. Recently, [21] proposed the hyperfast Offline Shapelet Discovery (OSD), which utilises important points to extract a small number of high-quality shapelets from the original time series data. It has been demonstrated to be a SOTA method for univariate time series classification.

3 PRELIMINARIES

Multivariate Time Series Classification. We represent MTS as $\mathbf{X} \in \mathbb{R}^{V \times T}$, where V denotes the number of variables and T represents the length of the time series. Here, $\mathbf{X} = X^1, \dots, X^V$, and each X^v corresponds to a time series for variable v . Specifically,

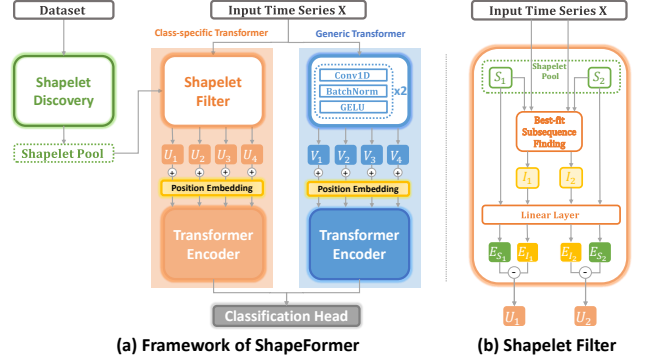


Figure 3: The general architecture of ShapeFormer.

$X^v = x_1^v, \dots, x_T^v$, where x_t^v signifies a value for variable v at timestamp t within \mathbf{X} . Consider a training dataset $\mathcal{D} = \{(X_i, Y_i)\}_{i=1}^M$, where M is the number of time series instances and the pair (X_i, Y_i) represents a training sample and its corresponding label, respectively. The objective of MTSC is to train a classifier $f(\mathbf{X})$ to predict a class label for a multivariate time series with an unknown label.

Time Series Subsequence. Given a time series X of length T , a time series subsequence $X[p_s : p_e] = x_{p_s}, \dots, x_{p_e}$ is a consecutive subsequence of time series X , where p_s is a start index and p_e is an end index.

Perceptual Subsequence Distance (PSD). Given a time series X of length T , and a subsequence $S = s_1, \dots, s_l$ of length l , with $l \leq T$, the PSD [21] of X and S is determined as:

$$PSD(X, S) = \min_{j=1}^{T-l+1} (\text{CID}(T[j : j+l-1], S)), \quad (1)$$

where CID is the complexity-invariant distance, which is commonly used in time series mining, in general, [2] and shapelet discovery in particular [21].

4 SHAPELET TRANSFORMER MODEL

We propose ShapeFormer, a transformer-based method that leverages the strength of both class-specific and generic features in time series. In contrast to existing transformer-based MTSC methods [26, 47, 50], our approach first extracts shapelets from the training datasets (Section 4.1). Subsequently, these extracted shapelets are used to discover discriminative features in time series through the use of a class-specific transformer module (Section 4.2). Additionally, we introduce the use of convolution layers with a generic transformer module to extract generic features in time series (Section 4.3). Finally, the overall architecture of ShapeFormer is summarised in Section 4.4 and Figure 3.

4.1 Shapelet Discovery

This section introduces the Offline Shapelet Discovery (OSD) method, inspired by [21], to multivariate time series. In contrast with other methods, our OSD employs Perceptually Important Points (PIPs) [4], condensing time series data by choosing points that closely resemble the original, to efficiently select high-quality shapelets. The selection process is based on the reconstruction distance, with the highest index continuously chosen. We define the reconstruction distance as the perpendicular distance between a target point

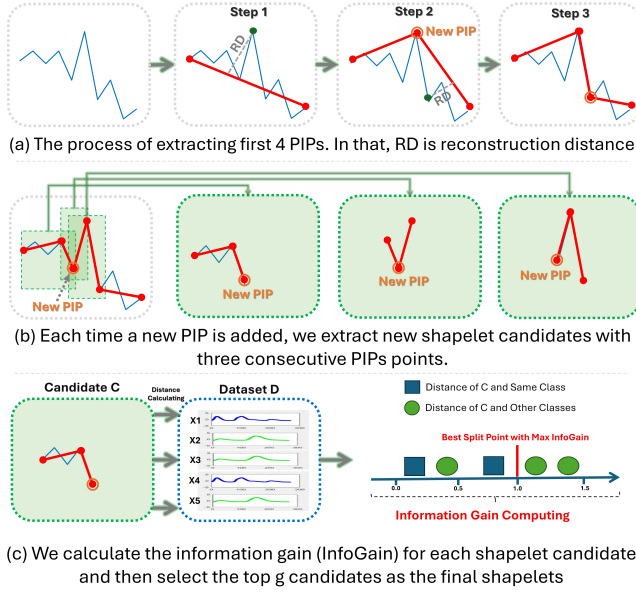


Figure 4: The process of Offline Shapelet Discovery.

and a line reconstructed by the two nearest selected important points [4, 21]. The process of our OSD is illustrated in Figure 4 and the pseudo-code is presented in Algorithm 1. Given the dataset $\mathcal{D} = \{(X_i, Y_i)\}_{i=1}^M$, our method contains two main phases, including shapelet extraction and shapelet selection.

In the first phase, our OSD initially extracts shapelet candidates by identifying PIPs. Specifically, the first and last indices are added to the PIPs set. Subsequently, the index with the highest reconstruction distance is continuously added to the PIPs set. Each time a new PIP is added, we extract new shapelet candidates with three consecutive PIPs points. This means that, with each new PIP, a maximum of three shapelet candidates can be added to the set. In this paper, we set the number of PIPs as $npip = 0.2 \times T$, where T represents the time series length. Our method aims to select a maximum of $3 \times npip$ candidates, therefore, we only extract an average of 5900 candidates for each dataset. This count is significantly smaller than the 45 million candidates typically extracted through classic shapelet discovery methods [25, 44], thereby significantly speeding up the process. We then store four types of information for each shapelet, including the value vector of shapelets, its start index, end index, and variables.

In the second phase, our method selects an equal number of shapelets for each class. Given the shapelet candidate S_i of class Y_i , we first compute its PSD with all instances in the training datasets (Eq. 1). After that, their distance will be used to find optimal information gain. This implies that the optimal information gain is the highest ratio achievable by the shapelet S_i [21]. Finally, the top g candidates with the highest information gain are chosen as the shapelets and stored in the shapelet pool \mathcal{S} .

4.2 Class-Specific Transformer

To utilise the class-specific characteristics of shapelets, we first propose the Shapelet Filter which is used to effectively discover input tokens for the transformer model.

Algorithm 1: Offline Shapelet Discovery

```

Input:  $\mathcal{D} = \{(X_i, Y_i)\}_{i=1}^M$ : dataset; time series length:  $T$ ; number
of variables:  $V$ ; number of PIPs:  $k$ ; number of shapelets:  $g$ ;
set of classes:  $\mathcal{Y}$ ;  $|\mathcal{Y}|$  is the number of classes.

/* Shapelet Extracting */
1  $C = []$ ;
2 foreach  $X \sim \mathcal{D}$  do
3   for  $v = 1$  to  $V$  do
4      $P = [1, T]$  # Add the first and last index into PIPs set:  $P$ ;
5     for  $j = 1$  to  $k-2$  do
6       Find index  $p$  from 1 to  $T$  with maximum
reconstruction distance;
7        $P.append(p).sorted()$  # Add a new index  $p$  into  $P$ ;
8        $idx = P.index(p)$  for  $z = 0$  to 2 do
9         # Validating newly generated candidates;
10        if  $idx + 2 \leq |I|$  and  $idx - z \geq 1$  then
11           $idx_s = P[idx - z]$ ;
12           $idx_e = P[idx + 2 - z]$ ;
13           $C = X[idx_s : idx_e]$ ;
14          Add new candidates  $C$ , its start index  $idx_s$ ,
end index  $idx_e$ , and its variables  $v$  into  $C$ .

/* Shapelet Selecting */
15 foreach  $S_i \sim C$  do
16    $D = []$ ;
17   foreach  $X \sim \mathcal{D}$  do
18      $d = PSD(X, S_i)$  (Eq. 1);
19      $D.append(d)$ ;
20   Compute the optimal information gain of  $S_i$  using  $D$ ;
21  $\mathcal{S} = []$ ;
22 foreach  $Y^i \sim \mathcal{Y}$  do
23   Select the top  $g/|\mathcal{Y}|$  shapelets ranked by information gain in
class  $Y^i$  from  $C$ ; Add them to set  $\mathcal{S}$ ;
24 return  $\mathcal{S}$ 

```

Shapelet Filter. Given a shapelet pool \mathcal{S} (as discussed in Section 4.1), an input time series X and its label Y , we first select the best-fit subsequence for each shapelet in \mathcal{S} (refer to Figure 5a). Specifically, with each shapelet $S_i \in \mathcal{S}$, its length l , start index p_s^i , end index p_e^i and variables v^i , we calculate the distance CID of them with all subsequences in time series X [21]. After that, the subsequence with the shortest distance will be selected as an important subsequence I_i of S_i .

$$\text{index} = \underset{j=0}{\text{argmin}}^{T-l+1} CID(X[j : j+l], S_i), \quad (2)$$

$$I_i = X[\text{index} : \text{index} + l]. \quad (3)$$

To reduce computing time and effectively utilise the position information of the shapelet, we propose limiting the search for the best-fit subsequence to a neighbouring area within the hyperparameter window size w on both the left and right sides of the actual position of the shapelet. This means that one shapelet only calculates the distance with maximum $2w + 1$ subsequences in X .

$$\text{index} = \underset{j=p_s-w}{\text{argmin}}^{p_s+w-1} CID(X[j : j+l], S_i), \quad (4)$$

$$I_i = X[\text{index} : \text{index} + l]. \quad (5)$$

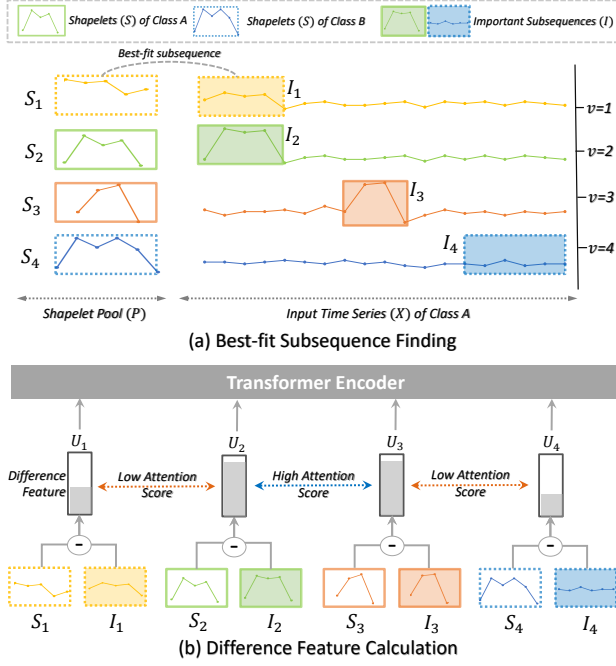


Figure 5: The illustrations for: (a) best-fit subsequence finding method; (b) difference feature calculation method.

Subsequently, we compute the difference features $U_i \in \mathbb{R}^{1 \times d_{spe}}$ between the embedding of each shapelet and their most fitting subsequences derived from the input time series (see Figure 5b).

$$U_i = \mathcal{P}_I(I_i) - \mathcal{P}_S(S_i), \quad (6)$$

where \mathcal{P} is the linear projector of $\mathbb{R}^{l \times d_{spe}}$ with l is length of shapelet and d_{spe} is the embedding size of difference features.

Similar to the distance between shapelet and time series, our difference feature also highlights the substantial distinctions among classes. Furthermore, by directly incorporating the shapelets in computing the difference features (Eq. 6), the shapelets are now considered as the learnable parameters of the Shapelet Filter component. Therefore, rather than using fixed shapelets, we can use them as the initial parameters of the Shapelet Filter, which will be optimised during training.

Position Embedding. The difference features U_i are then integrated with position embeddings to capture their order. To better indicate the position information of shapelets, the embeddings of three types of positions are considered, including the start index, end index, and variables. Specifically, we propose to use a one-hot vector representation for these indices and then employ a linear projector to learn their embedding.

$$\text{PE}(p) = \text{Linear}(\text{one-hot}(p)), \quad (7)$$

$$U_i = U_i + \text{PE}(p_s^i) + \text{PE}(p_e^i) + \text{PE}(v^i). \quad (8)$$

We also observed that the performance is enhanced when we only use the position of shapelets instead of the position of best-fit subsequences. This improvement can be attributed to the fact that the fixed position is easier to learn than the unstable position of best-fit subsequences.

Transformer Encoder. The class-specific difference features, along with their corresponding position embeddings, are then input into a transformer encoder to learn their correlation. Specifically, we employ the multi-head attention mechanism (MHA) [40] for this purpose. Given an input series, $U = U_1, \dots, U_g$ and the projections $W_q, W_k, W_v \in \mathbb{R}^{d_{spe} \times d_{spe}}$ represent query, key, and value matrices, respectively. These matrices, W_q, W_k, W_v , undergo reshaping into $\mathbb{R}^{h \times d_{spe} \times (d_{spe}/h)}$ to signify the h attention heads and are subsequently concatenated into standard dimensions after computation. Each attention head within this set is capable of capturing distinct relationships of the features. Finally, these matrices are used to compute an output $Z^{\text{spe}} = Z_1^{\text{spe}}, \dots, Z_g^{\text{spe}}$ where $Z_i^{\text{spe}} \in \mathbb{R}^{d_{spe}}$:

$$Z_i^{\text{spe}} = \sum_{j=1}^g a_{i,j} (U_j * W_v), \quad (9)$$

where $a_{i,j}$ is an attention score which is calculated as:

$$a_{i,j} = \text{softmax}\left(\frac{(U_i * W_q)(U_j * W_k)}{\sqrt{d_{spe}}}\right), \quad (10)$$

Thanks to the class-represented characteristics of these features, the attention score for features within the same class is boosted compared to features in different classes. This enhancement helps the model better distinguish between different classes. Additionally, owing to the nature of shapelets, the difference features possess the ability to identify significant subsequences across different temporal locations and variables within the time series. This capability enables the module to effectively capture temporal and variable dependencies in time series data.

Class Token. Existing transformer-based methods apply averaging pooling to Z^{spe} , to obtain the final token for classification [10, 26]. However, our class-specific transformer module utilises difference features that capture the distinctive characteristics of each shapelet. Applying average pooling may diminish these properties, potentially limiting performance. To address this, we propose using only the first difference feature of the highest information gain shapelet Z_1^{spe} as the class token Z_*^{spe} for final classification. The reason for this is that when averaging all tokens, there is a loss of information regarding distinct features U_i . Moreover, the first token Z_1^{spe} , which carries the highest information gain, harbors the most crucial features for effectively classifying time series.

4.3 Generic Transformer

Besides leveraging the power of class-specific features, in this section, we introduce the generic transformer module, utilising convolution filters to extract generic features in the time series. Specifically, we employ two CNN components [10, 11], each comprising Conv1D, BatchNorm, and GELU, to effectively discover generic features. The first block is designed to capture the temporal patterns in time series by using the Conv1D filter $\in \mathbb{R}^{1 \times d_c}$. On the other hand, the second block uses the Conv1D filter $\in \mathbb{R}^{V \times 1}$ to capture the correlation between variables in time series. In this context, V represents the number of variables, and d_c is the kernel size of the convolution filter, which is fixed at 8 in all experiments. From that,

the output generic feature $V_i \in \mathbb{R}^{1 \times d_{gen}}$ is calculated as follows:

$$\text{ConvBlock}(X) = \text{GELU}(\text{BatchNorm}(\text{Conv1D}(X))), \quad (11)$$

$$V = \text{ConvBlock}(\text{ConvBlock}(X)). \quad (12)$$

Afterward, these features will be fed to the h multi-attention heads to learn the correlation. Each attention head has the capacity to capture distinct patterns within time series data.

$$Z^{\text{gen}} = \text{MHA}(V + P), \quad (13)$$

as the position of each element $V_i \in V$ lacks inherent meaning, we utilise the learnable position embedding P for representing them. Furthermore, since the module takes classic features as input tokens, we employ averaging pooling to derive the final class token.

$$Z_*^{\text{gen}} = \text{AvgPooling}(Z^{\text{gen}}). \quad (14)$$

4.4 Overall Architecture of ShapeFormer

To enhance clarity, we present the overall architecture of ShapeFormer in Figure 3. Our method initiates by extracting shapelets from the training datasets. Subsequently, for a given input time series X , it is processed through dual transformer modules, comprising the class-specific shapelet transformer and the generic convolution transformer. The outputs from these two modules are then concatenated and fed into the final classification head.

$$Z = \text{concat}(Z_*^{\text{spe}}, Z_*^{\text{gen}}), \quad (15)$$

$$\hat{y} = \text{argmax}(\text{softmax}(\text{Linear}(Z))). \quad (16)$$

The predictions \hat{y} are used to optimise the model parameters based on the following objective function:

$$\mathcal{L} = \mathcal{L}_{\text{CE}}(\hat{y}, Y). \quad (17)$$

where, CE is the Cross-Entropy Loss, which can be calculated as $\mathcal{L}_{\text{CE}}(\hat{y}, Y) = \sum_i^{|Y|} y_i \log(\hat{y}_i)$.

5 EXPERIMENTS

5.1 Experimental Setting

Dataset. We assess our approach using the UEA archive, a well-known benchmark made up of 30 distinct datasets for MTSC [1]. It covers various domains, including Human Activity Recognition, Motion classification, ECG classification, EEG/MEG classification, Audio Spectra classification, and more. The sample sizes of datasets in the UEA archive range from 27 to 50,000, the time series lengths spanning 8 to 17,984, and dimensions varying from 2 to 1,345.

Metrics. We use classification accuracy to evaluate model performance and compare methods based on their average ranks and win/draw/loss counts on all datasets. Finally, we evaluate the statistical significance of performance differences using the p-value of Friedman and Wilcoxon signed-rank test [1].

Implementation Details. Our model was trained using the RAdam optimiser with an initial learning rate set as 0.01, a momentum of 0.9, and a weight decay of $5e-4$. The training process involved a batch size of 16 for a total of 200 epochs. We configured the number of attention heads to be 16 and followed the protocol outlined in [47, 50]. This protocol involves splitting the training set into 80% for training and 20% for validation, allowing us to fine-tune

hyperparameters. Once the hyperparameters were finalised, we conducted model training on the entire training set and subsequently evaluated its performance on the designated official test set.

Environment. All the experiments are conducted on a machine with one Intel(R) Xeon(R) Silver 4214 CPU @ 2.20GHz and one NVIDIA Tesla V100 SXM2.

5.2 Baselines

We have selected 12 baseline methods for the comparative experiments, comprising two distance-based methods: *EDI*, *DTW_D* [1]; a pattern-based algorithm: *WEASEL+MUSE* [33]; a feature-based algorithm: *MiniRocket* [5]; an ensemble method: *LCEM* [9]; three deep learning models: *MLSTM-FCNs* [18], *Tapnet* [48], *Shapenet* [23]; an attention-based model: *WHEN* [41]; and three transformer-based models: *TST* [47], *ConvTran* [10], *SVP-T* [50]. They all attained the SOTA performance described in the most recent research. The details of 12 baseline methods are shown in Appendix A.

5.3 Performance Evaluation

Table 1 illustrates the experimental results of our method with 12 other competitors on the UEA multivariate time series classification archive [1]. The accuracy of 12 baseline methods are taken from [50], except the results of *WHEN*, and *ConvTran* are taken from their original papers [10, 41]. The best result on each dataset is indicated in bold, and the summarised information is provided in the last six lines of the table.

The results show that among all methods, ShapeFormer achieves the best performance in both the highest average rank (2.5) and the largest number of top-1 (the best in 15 out of 30 datasets). This indicates that ShapeFormer can be taken as a SOTA for MTSC. The rank index signifies that, even on some datasets where our model does not exhibit the highest performance, its results remain highly competitive. Specifically, the average rank of our method is slightly higher compared to that of the runner-up, *WHEN*, a difference of 0.617. Meanwhile, the gap in average rank between ShapeFormer and three Transformer-based methods (*TST*, *ConvTran*, *SVP-T*) is large, with 5.617, 3.15, and 2.783 respectively. The p-value is ≤ 0.05 , which confirms these ranks have statistically significant differences. Specifically, the p-values for ShapeFormer in comparison to all methods are below 0.05, which indicates the results are statistically significant except for *WHEN*. However, regarding the number of top-1, our ShapeFormer attained SOTA results in 15 datasets compared to *WHEN*, only 4 datasets.

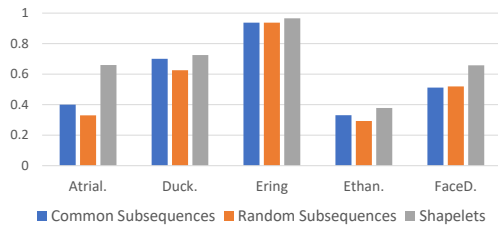
5.4 Ablation Study and Model Design

Effectiveness of Using Shapelets. In Figure 6, we compare the performance when using random subsequences, common subsequences as mentioned in [50], and shapelets in our methods. The results demonstrate that the shapelets outperform the other two methods in terms of accuracy across all five datasets. This highlights the benefit of highly discriminative shapelet features in increasing the performance of the transformer-based model, thereby indicating the contribution of our work.

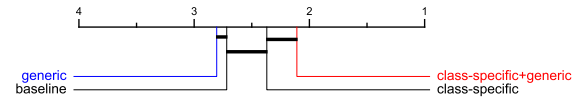
Component Evaluation. We begin by evaluating the impact of two key modules in our ShapeFormer: the Class-specific Transformer (Section 4.2) and the Generic Transformer (Section 4.3),

Table 1: Accuracies of our proposed method ShapeFormer and 12 compared methods on all datasets of the UEA archive [1].

	EDI	DTWD	WEASEL +MUSE	MiniRocket	LCEM	MLSTM -FCNs	Tapnet	Shapenet	WHEN	TST	ConvTran	SVPT	Our
ArticulatoryWordRecognition	0.970	0.987	0.990	0.992	0.993	0.973	0.987	0.987	0.993	0.983	0.983	0.993	0.993
AtrialFibrillation	0.267	0.220	0.333	0.133	0.467	0.267	0.333	0.400	0.467	0.200	0.400	0.400	0.660
BasicMotions	0.676	0.975	1.000	1.000	1.000	0.950	1.000	1.000	1.000	0.975	1.000	1.000	1.000
CharacterTrajectories	0.964	0.989	0.990	0.993	0.979	0.985	0.997	0.980	0.996	0.000	0.992	0.990	0.996
Cricket	0.944	1.000	1.000	0.986	0.986	0.917	0.958	0.986	1.000	0.958	1.000	1.000	1.000
DuckDuckGeese	0.275	0.600	0.575	0.650	0.375	0.675	0.575	0.725	0.700	0.480	0.620	0.700	0.725
ERing	0.133	0.929	0.133	0.981	0.200	0.133	0.133	0.133	0.959	0.933	0.963	0.937	0.966
EigenWorms	0.549	0.618	0.890	0.962	0.527	0.504	0.489	0.878	0.893	N/A	0.593	0.925	0.925
Epilepsy	0.666	0.964	1.000	1.000	0.986	0.761	0.971	0.987	0.993	0.920	0.986	0.986	0.993
EthanolConcentration	0.293	0.323	0.430	0.468	0.372	0.373	0.323	0.312	0.422	0.337	0.361	0.331	0.378
FaceDetection	0.519	0.529	0.545	0.620	0.614	0.545	0.556	0.602	0.658	0.681	0.672	0.512	0.658
FingerMovements	0.550	0.530	0.490	0.550	0.590	0.580	0.530	0.580	0.660	0.776	0.560	0.600	0.700
HandMovementDirection	0.278	0.231	0.365	0.392	0.649	0.365	0.378	0.338	0.554	0.608	0.405	0.392	0.486
Handwriting	0.200	0.286	0.605	0.507	0.287	0.286	0.357	0.452	0.561	0.305	0.375	0.433	0.507
Heartbeat	0.619	0.717	0.727	0.771	0.761	0.663	0.751	0.756	0.780	0.712	0.785	0.790	0.800
InsectWingbeat	0.128	N/A	N/A	0.595	0.228	0.167	0.208	0.250	0.657	0.684	0.713	0.184	0.314
JapaneseVowels	0.924	0.949	0.973	0.989	0.978	0.976	0.965	0.984	0.995	0.994	0.989	0.978	0.997
LSST	0.456	0.551	0.590	0.643	0.652	0.373	0.568	0.590	0.663	0.381	0.616	0.666	0.700
Libras	0.833	0.870	0.878	0.922	0.772	0.856	0.850	0.856	0.933	0.844	0.928	0.883	0.961
MotorImagery	0.510	0.500	0.500	0.550	0.600	0.510	0.590	0.610	0.630	N/A	0.560	0.650	0.670
NATOPS	0.850	0.883	0.870	0.928	0.916	0.889	0.939	0.883	0.978	0.900	0.944	0.906	0.989
PEMS-SF	0.973	0.711	N/A	0.522	0.942	0.699	0.751	0.751	0.925	0.919	0.828	0.867	0.925
PenDigits	0.705	0.977	0.948	N/A	0.977	0.978	0.980	0.977	0.987	0.974	0.987	0.983	0.990
PhonemeSpectra	0.104	0.151	0.190	0.292	0.288	0.110	0.175	0.298	0.293	0.088	0.306	0.176	0.293
RacketSports	0.868	0.803	0.934	0.868	0.941	0.803	0.868	0.882	0.934	0.829	0.862	0.842	0.895
SelfRegulationSCP1	0.771	0.775	0.710	0.925	0.839	0.874	0.652	0.782	0.908	0.925	0.918	0.884	0.911
SelfRegulationSCP2	0.483	0.539	0.460	0.522	0.550	0.472	0.550	0.578	0.589	0.589	0.583	0.600	0.633
SpokenArabicDigits	0.967	0.963	0.982	0.620	0.973	0.990	0.983	0.975	0.997	0.993	N/A	0.986	0.997
StandWalkJump	0.200	0.200	0.333	0.333	0.400	0.067	0.400	0.533	0.533	0.267	0.333	0.467	0.600
UWaveGestureLibrary	0.881	0.903	0.916	0.938	0.897	0.891	0.894	0.906	0.919	0.903	0.891	0.941	0.922
Average rank	11.200	9.783	7.933	5.900	6.600	9.833	8.233	6.850	3.117	8.117	5.650	5.283	2.500
Number of top-1	1	1	4	6	4	0	2	2	4	3	4	5	15
Wins	29	29	24	21	25	30	28	27	16	25	24	24	-
Draws	0	1	2	2	2	0	1	2	9	0	2	5	-
Loses	1	0	4	7	3	0	1	1	5	5	4	1	-
P-value	0.000	0.000	0.001	0.014	0.003	0.000	0.000	0.002	0.475	0.005	0.024	0.000	-

**Figure 6: Accuracies of using shapelets and two other types of subsequences.**

in comparison with the baseline method, SVP-T [50] on the first 10 datasets of UEA archive [1]. In this process, individual components are incrementally incorporated to assess their impact on the ultimate accuracy. As depicted in Figure 7, applying the generic transformer alone exhibits a lower accuracy compared to the baseline. In contrast, utilising only the class-specific module results in significantly improved performance over the baselines, emphasising the effectiveness of class-specific features in the transformer-based time series model. Furthermore, the combination of class-specific

**Figure 7: Average ranks for 3 variations of ShapeFormer and the baseline (SVP-T [50] - the current SOTA transformer-based method).**

and generic components shows a positive impact on the enhancement of classification accuracy. This combination harnesses the power of both features, significantly boosting overall performance. **Choosing between the Position of Shapelets and Best-fit Subsequences.** Our ShapeFormer leverages shapelets to find the best-fit subsequences and employ the difference features U_i calculated by them as the inputs for the Transformer encoder. Then there is a question "Should we choose the positions of the shapelets or the best-fit subsequences for position embedding?". Figure 8 illustrates a comparison between the accuracies achieved by employing the position of the best-fit subsequences and shapelets, as indicated in Eq. 8, for position embedding in the Transformer encoder. The outcomes indicate that our approach exhibits superior performance when utilising the position of shapelets across all five datasets under consideration. This enhancement can be ascribed to the fact

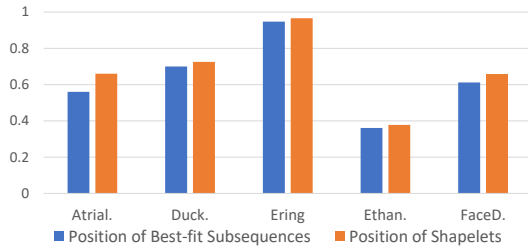


Figure 8: Accuracies of using the position of best-fit subsequences and shapelets.

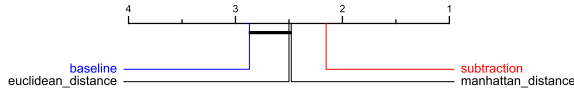


Figure 9: Average accuracy ranks of various calculation methods for difference features.

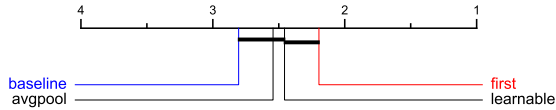


Figure 10: Average accuracy ranks of different class token designs.

that learning from the fixed position of shapelets is easier compared to the unstable position of the best-fit subsequences.

Comparison with Various Methods for Calculating Difference Features. The critical difference diagram in Figure 9 displays the performance of using different methods for calculating difference features in Eq. 6, including Manhattan Distance, Euclidean Distance, and the subtraction between $\mathcal{P}_I(I_i)$ and $\mathcal{P}_S(S_i)$. The results demonstrate that: 1) All calculation methods for difference features yield better results compared to the SVP-T baseline; 2) Using subtraction exhibits the highest performance. Although the subtraction is simple, its superiority lies in effectively capturing relative changes by considering both the magnitude and direction of changes between embedding vectors $\mathcal{P}_I(I_i)$ and $\mathcal{P}_S(S_i)$.

Different Class Token Designs. The output of the class-specific transformer consists of a series of tokens $Z_1^{\text{spe}}, \dots, Z_g^{\text{spe}}$. The question at hand is, "Are there any effective ways to design class tokens before feeding them to the classification head?". In Figure 10, we analyse the impact of different class token designs on the performance of ShapeFormer. The results indicate that: 1) Our ShapeFormer outperforms the baseline with all types of class token designs, demonstrating the advantage of our method; 2) Utilising the first token Z_1^{spe} as the final class token Z_*^{spe} for ShapeFormer yields the best performance. This is due to the fact that learning or averaging all tokens results in a loss of information on difference features U_i . Furthermore, the first token, containing the highest information gain, possesses the most discriminative features for classifying time series.

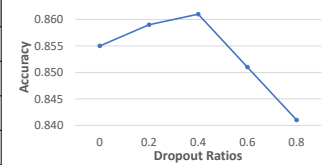
5.5 Hyperparameter Sensitivity

Tuning Window Size and Number of Shapelets. In our method, there are two main parameters related to shapelet discovery that need tuning, including the window size when calculating the distances between shapelets and subsequences and the number of

Table 2: The average accuracy for various numbers of PIPs.

Npips ($\times T$)	0.05	0.1	0.15	0.2	0.25	0.3	0.4	0.5
Accuracy	0.832	0.848	0.856	0.864	0.864	0.864	0.864	0.864

$d_{\text{spe}} \backslash d_{\text{gen}}$	32	64	128	256
32	0.845	0.858	0.864	0.86
64	0.838	0.855	0.861	0.852
128	0.834	0.842	0.859	0.858
256	0.829	0.836	0.844	0.855



(a) Different scale factors d_{global} and d_{local}

(b) Different dropout ratios

Figure 11: Effectiveness of (a) class-specific and generic scale factors and (b) different dropout ratios.

shapelets. Regarding the window size, we propose to tune it exclusively during the shapelet discovery phase. For each dataset, we will select a window size from the set [10, 20, 50, 100, 200], aiming to provide the top 100 shapelets with the highest information gain. This tuning technique can significantly reduce training time since it only operates during the shapelet discovery phase. As for the number of shapelets, considering the diverse characteristics of different datasets, we choose this number from [1, 3, 10, 30, 100]. The details of our tuned parameters are shown in Appendix B.

Number of PIPs. As shown in the following Table 2, the model accuracy increases as we increase the number of PIPs (npips) from 0.05T to 0.2T. Afterward, accuracy remains stable even with further increases in npips. Therefore, we set npips at 0.2 for all of our experiments.

The Scale Factors d_{spe} and d_{gen} . In Figure 11a, we compare the impact of different scale factors of the class-specific and generic embedding sizes on the classification accuracy of ShapeFormer. The results show that: 1) The pair of $d_{\text{spe}} = 128$ and $d_{\text{gen}} = 32$ has achieved the highest accuracy; 2) In general, a larger class-specific embedding size has achieved better performance, indicating the benefit of using shapelets in a transformer-based time series model.

Dropout Ratios. In Figure 11b, we analyse the impact of different dropout ratios of ShapeFormer. It is evident that our methods work well and achieve high performance with small dropout ratios, with the ratio of 0.4 yielding the highest performance.

5.6 Improving Performance in Specific Datasets by Optimizing Scale Factor

In MTSC, it is crucial to develop models that generalise well across a majority of datasets rather than models tailored to specific datasets. For example, in terms of *InsectWingbeat* dataset, we observed that setting d_{gen} (embedding size of generic feature) to 256 leads to significantly better performance (0.704) compared to d_{gen} at 32 (0.314) (our chosen parameter). However, this improvement comes at the cost of decreased performance on other datasets (from 0.864 to 0.831). Therefore, we recommend tuning this hyperparameter to achieve better performance on specific datasets if needed.

Table 3: Accuracy for *InsectWingbeat* dataset and the first 10 UEA datasets with various d_{gen} factors.

d_{gen}	32 (our choice)	64	128	256
InsectWingbeat	0.314	0.500	0.634	0.704
First 10 UEA datasets	0.864	0.852	0.844	0.831

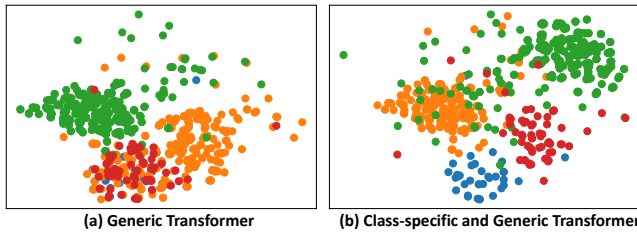


Figure 12: The t-SNE visualisation in 4 classes of LSST dataset using (a) our generic transformer and (b) both class-specific and generic transformers. Each point indicates an instance and the colors of the points signify the true labels.

5.7 A Case Study of LSST (Imbalanced Datasets)

To illustrate the effectiveness of combining both class-specific and generic features transformer modules to classify imbalanced data, we conducted experiments on the *LSST* dataset. The *LSST* dataset comprises 16 classes, and we randomly selected 4 classes to be represented by the colors blue, orange, green, and red, with 35, 270, 382, and 63 instances, respectively. It is clear that the sizes of blue and red classes are significantly smaller compared to the sizes of the green and orange classes. Figure 12a shows that the generic transformer prioritises majority classes (green and orange), but neglects minority ones (blue and red). However, in Figure 12b, the combination of class-specific and generic transformers effectively distinguishes all four classes.

5.8 A Case Study of BasicMotions

To interpret ShapeFormer results, we use the *BasicMotions* dataset from the UEA archive [1], focusing on human activity recognition with 4 classes (playing badminton, standing, walking, and running). Each class is associated with 6 variables, and 10 shapelets are set for analysis. Randomly selecting a ‘walking’ instance from the training set. Figure 13a showcases the top three shapelets for this class and three from others, highlighting ShapeFormer’s ability to identify crucial subsequences across diverse locations and variables in the time series. Moreover, shapelets within the same ‘walking’ class tend to share greater similarity with best-fit subsequences than those from other classes.

In Figure 13b, the attention heat map for all 40 shapelets across 4 classes reveals that shapelets within the same class generally attain higher attention scores. For instance, S_{20} and S_{23} belonging to the ‘walking’ class show a small difference feature (Eq. 6), resulting in higher attention scores. This enhanced attention allows the model to focus more on the correlation between shapelets within the same classes, thereby improving overall performance.

6 CONCLUSION

In this paper, we propose a novel Shapelet Transformer (ShapeFormer) for multivariate time series classification. It consists of dual transformer modules aimed at identifying class-specific and generic features within time series data. In particular, the first module discovers class-specific features by utilising discriminative subsequences (shapelets) extracted from the entire dataset. Meanwhile, the second transformer module employs convolution filters

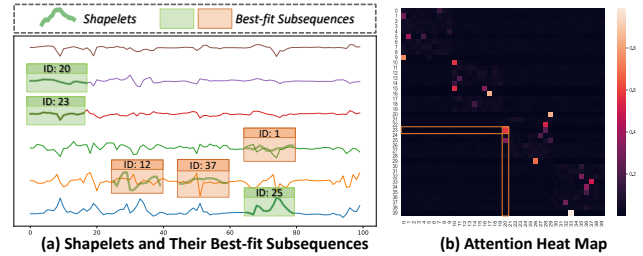


Figure 13: (a) The green box depicts the top three shapelets, and the orange box displays three random shapelets from other classes, extracted in one random input time series of the ‘walking’ class in the *BasicMotions* dataset. (b) The attention heat map for all shapelets.

to extract generic features across all classes. The experimental results show that by combining both modules, our ShapeFormer has achieved the highest rank in terms of classification accuracy when compared to the SOTA methods. In future work, we intend to utilise the power of shapelets in many different time series analysis tasks such as forecasting or anomaly detection.

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 preprint arXiv:1811.00075* (2018).
- [2] Gustavo EAPA Batista, Xiaoyue Wang, and Eamonn J Keogh. 2011. A complexity-invariant distance measure for time series. In *Proceedings of the 2011 SIAM international conference on data mining*. SIAM, 699–710.
- [3] Donald J Berndt and James Clifford. 1994. Using dynamic time warping to find patterns in time series. In *Proceedings of the 3rd international conference on knowledge discovery and data mining*. 359–370.
- [4] Fu Lai Korris Chung, Tak-Chung Fu, Wing Pong Robert Luk, and Vincent To Yee Ng. 2001. Flexible time series pattern matching based on perceptually important points. In *Workshop on Learning from Temporal and Spatial Data in International Joint Conference on Artificial Intelligence*.
- [5] 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*. 248–257.
- [6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).
- [7] Thuc-Doan Do, Tuan Minh Tran, Xuan-May Thi Le, and Thuy-Van T Duong. 2017. Detecting special lecturers using information theory-based outlier detection method. In *Proceedings of the International Conference on Compute and Data Analysis*. 240–244.
- [8] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xi-aohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929* (2020).
- [9] Kevin Fauvel, Élisabeth Fromont, Véronique Masson, Philippe Favardin, and Alexandre Termier. 2020. Local cascade ensemble for multivariate data classification. *arXiv preprint arXiv:2005.03645* (2020).
- [10] Navid Mohammadi Foumani, Chang Wei Tan, Geoffrey I Webb, and Mahsa Salehi. 2023. Improving Position Encoding of Transformers for Multivariate Time Series Classification. *arXiv preprint arXiv:2305.16642* (2023).
- [11] Seyed Navid Mohammadi Foumani, Chang Wei Tan, and Mahsa Salehi. 2021. Disjoint-cnn for multivariate time series classification. In *2021 International Conference on Data Mining Workshops (ICDMW)*. IEEE, 760–769.
- [12] Ge Gao, Qitong Gao, Xi Yang, Miroslav Pajic, and Min Chi. 2022. A reinforcement learning-informed pattern mining framework for multivariate time series classification. In *In the Proceeding of 31th International Joint Conference on Artificial Intelligence (IJCAI-22)*.
- [13] Josif Grabocka, Nicolas Schilling, Martin Wistuba, and Lars Schmidt-Thieme. 2014. Learning time-series shapelets. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. 392–401.
- [14] Josif Grabocka, Martin Wistuba, and Lars Schmidt-Thieme. 2016. Fast classification of univariate and multivariate time series through shapelet discovery. *Knowledge and information systems* 49 (2016), 429–454.

- [15] Yifan Hao and Huiping Cao. 2020. A new attention mechanism to classify multivariate time series. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*.
- [16] 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.
- [17] Fazle Karim, Somshubra Majumdar, Houshang Darabi, and Samuel Harford. 2019. Multivariate LSTM-FCNs for time series classification. *Neural networks* 116 (2019), 237–245.
- [18] Fazle Karim, Somshubra Majumdar, Houshang Darabi, and Samuel Harford. 2019. Multivariate LSTM-FCNs for time series classification. *Neural networks* 116 (2019), 237–245.
- [19] Eamonn Keogh and Chotirat Ann Ratanamahatana. 2005. Exact indexing of dynamic time warping. *Knowledge and information systems* 7 (2005), 358–386.
- [20] Sang-Wook Kim, Dae-Hyun Park, and Heon-Gil Lee. 2004. Efficient processing of subsequence matching with the Euclidean metric in time-series databases. *Information processing letters* 90, 5 (2004), 253–260.
- [21] Xuan-May Le, Minh-Tuan Tran, and Van-Nam Huynh. 2022. Learning Perceptual Position-Aware Shapelets for Time Series Classification. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 53–69.
- [22] Xuan-May Thi Le, Tuan Minh Tran, and Hien T Nguyen. 2020. An improvement of SAX representation for time series by using complexity invariance. *Intelligent Data Analysis* 24, 3 (2020), 625–641.
- [23] Guozhong Li, Byron Choi, Jianliang Xu, Sourav S Bhowmick, Kwok-Pan Chun, and Grace Lai-Hung Wong. 2021. Shapenet: A shapelet-neural network approach for multivariate time series classification. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 35. 8375–8383.
- [24] Jessica Lin, Rohan Khade, and Yuan Li. 2012. Rotation-invariant similarity in time series using bag-of-patterns representation. *Journal of Intelligent Information Systems* 39 (2012), 287–315.
- [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] Amy McGovern, Derek H Rosendahl, Rodger A Brown, and Kelvin K Droegemeier. 2011. Identifying predictive multi-dimensional time series motifs: an application to severe weather prediction. *Data Mining and Knowledge Discovery* 22 (2011), 232–258.
- [28] Yuqi Nie, Nam H Nguyen, Phanwadee Sinthong, and Jayant Kalnaganam. 2022. A time series is worth 64 words: Long-term forecasting with transformers. *arXiv preprint arXiv:2211.14730* (2022).
- [29] Andrew J Patton. 2012. A review of copula models for economic time series. *Journal of Multivariate Analysis* 110 (2012), 4–18.
- [30] 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. *Data Mining and Knowledge Discovery* 35, 2 (2021), 401–449.
- [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. *Data Mining and Knowledge Discovery* 35, 2 (2021), 401–449.
- [32] Patrick Schäfer. 2015. The BOSS is concerned with time series classification in the presence of noise. *Data Mining and Knowledge Discovery* 29 (2015), 1505–1530.
- [33] Patrick Schäfer and Ulf Leser. 2017. Multivariate time series classification with WEASEL+ MUSE. *arXiv preprint arXiv:1711.11343* (2017).
- [34] ABA Stevner, Diego Vidaurre, Joana Cabral, K Rapuano, Søren Føns Vind Nielsen, Enzo Tagliazucchi, Helmut Laufs, Peter Vuust, Gustavo Deco, Mark W Woolrich, et al. 2019. Discovery of key whole-brain transitions and dynamics during human wakefulness and non-REM sleep. *Nature communications* 10, 1 (2019), 1035.
- [35] 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. *Data Mining and Knowledge Discovery* 36, 5 (2022), 1623–1646.
- [36] Minh-Tuan Tran, Trung Le, Xuan-May Le, Mehrtash Harandi, and Dinh Phung. 2024. Text-Enhanced Data-free Approach for Federated Class-Incremental Learning. *arXiv preprint arXiv:2403.14101* (2024).
- [37] Minh-Tuan Tran, Trung Le, Xuan-May Le, Mehrtash Harandi, Quan Hung Tran, and Dinh Phung. 2023. NAYER: Noisy Layer Data Generation for Efficient and Effective Data-free Knowledge Distillation. *arXiv preprint arXiv:2310.00258* (2023).
- [38] Tuan Minh Tran, Xuan-May Thi Le, Hien T Nguyen, and Van-Nam Huynh. 2019. A novel non-parametric method for time series classification based on k-Nearest Neighbors and Dynamic Time Warping Barycenter Averaging. *Engineering Applications of Artificial Intelligence* 78 (2019), 173–185.
- [39] Tuan Minh Tran, Xuan-May Thi Le, Vo Thanh Vinh, Hien T Nguyen, and Tuan M Nguyen. 2017. A weighted local mean-based k-nearest neighbors classifier for time series. In *Proceedings of the 9th International Conference on Machine Learning and Computing*. 157–161.
- [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] Jingyuan Wang, Chen Yang, Xiaohan Jiang, and Junjie Wu. 2023. WHEN: A Wavelet-DTW Hybrid Attention Network for Heterogeneous Time Series Analysis. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 2361–2373.
- [42] Haixu Wu, Tengge Hu, Yong Liu, Hang Zhou, Jianmin Wang, and Mingsheng Long. 2022. Timesnet: Temporal 2d-variation modeling for general time series analysis. In *The eleventh international conference on learning representations*.
- [43] Carl Yang and Jiawei Han. 2023. Revisiting citation prediction with cluster-aware text-enhanced heterogeneous graph neural networks. In *2023 IEEE 39th International Conference on Data Engineering (ICDE)*. IEEE, 682–695.
- [44] 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.
- [45] Wenhui Yu, Xiao Lin, Junfeng Ge, Wenwu Ou, and Zheng Qin. 2020. Semi-supervised collaborative filtering by text-enhanced domain adaptation. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2136–2144.
- [46] Zhihan Yue, Yujing Wang, Juanyong Duan, Tianmeng Yang, Congrui Huang, Yunhai Tong, and Bixiong Xu. 2022. Ts2vec: Towards universal representation of time series. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 36. 8980–8987.
- [47] 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.
- [48] 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.
- [49] Tian Zhou, Peisong Niu, Liang Sun, Rong Jin, et al. 2023. One fits all: Power general time series analysis by pretrained lm. *Advances in neural information processing systems* 36 (2023), 43322–43355.
- [50] Rundong Zuo, Guozhong Li, Byron Choi, Sourav S Bhowmick, Daphne Ngar Yin Mah, and Grace LH Wong. 2023. SVP-T: a shape-level variable-position transformer for multivariate time series classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 37. 11497–11505.

A BASELINES

12 baseline methods are utilised in the comparative experiments, comprising two distance-based methods, a pattern-based algorithm, a feature-based algorithm, an ensemble method, three deep learning models, an attention-based model, and three transformer-based models. They all attained the SOTA performance described in the most recent research.

- *EDI* and *DTW_D* [1]: The two benchmark classifiers based on Euclidean Distance and dimension-dependent dynamic time warping.
- *WEASEL+MUSE* [33]: A classifier based on a bag-of-pattern approach demonstrated SOTA performance when compared with similar competitors for MTSC. We choose this algorithm as the representative baseline among pattern-based methods.
- *MiniRocket* [5]: A feature-based method utilizes random convolutional kernels to discover features. It performs well in both univariate and multivariate time series classification.
- *LCEM* [9]: A hybrid ensemble method that integrates boosting, bagging, divide-and-conquer, and decision tree components. LCEM demonstrated superior performance when compared to other random forest methods. We choose it as a representative illustration of ensemble methods.
- *MLSTM-FCNs* [18]: A deep learning method for MTSC which utilizes LSTM layer and stacked CNN layers to discover features.

Table 4: The selected window size and number of shapelets for each UEA MTSC dataset.

Dataset	Window size	Number of shapelets (per class)
ArticularyWordRecognition	50	10
AtrialFibrillation	100	3
BasicMotions	100	10
CharacterTrajectories	50	3
Cricket	200	30
DuckDuckGeese	10	100
ERing	50	100
EigenWorms	10	10
Epilepsy	20	30
EthanolConcentration	200	100
FaceDetection	10	10
FingerMovements	20	30
HandMovementDirection	200	100
Handwriting	20	30
Heartbeat	200	100
InsectWingbeat	10	30
JapaneseVowels	10	1
LSST	20	10
Libras	10	30
MotorImagery	100	30
NATOPS	20	1
PEMS-SF	50	10
PenDigits	4	10
PhonemeSpectra	20	30
RacketSports	10	10
SelfRegulationSCP1	100	100
SelfRegulationSCP2	100	100
SpokenArabicDigits	10	100
StandWalkJump	10	100
UWaveGestureLibrary	10	10

- *Tapnet* [48]: A classifier constructs an attentional prototype network. Tapnet incorporates LSTM, and CNN to learn multi dimensional interaction features. We opt for it as another representative of the deep learning method.
- *Shapenet* [23]: Shapenet aims to learn representations of different shapelet candidates in a unified space and selects final shapelets by training a dilated causal CNN module followed by standard classification. This model can capture dependencies among variables. We choose it as a representative of the shapelet-based method.

- *WHEN* [41]: An attention-based method that learns heterogeneity by utilising a hybrid attention network, incorporating both DTW attention and wavelet attention. It achieved SOTA performance for MTSC on the UEA datasets.
- *TST* [47]: A transformer-based framework for MTS representation learning. TST is considered as baseline method that takes the values at each timestamp as the input for the Transformer model. It gains great performance for many sequential tasks, such as regression, and classification.
- *ConvTran* [10]: A transformer-based method for MTSC that proposed to improve the position embedding in the Transformer model by leveraging both absolute and relative position encoding.
- *SVP-T* [50]: A method uses clustering to identify time series subsequences and employs them as inputs for the Transformer, enabling the capture of long- and short-term dependencies among subseries. It achieved SOTA performance for MTSC. We choose it as another representative of the transformer-based method.

B SELECTED HYPERPARAMETERS

In this section, we follow the setting mentioned in Section 5.5 to tune the hyperparameter of window size and number of shapelets per class. In Table 4, we provide the selected window size and number of shapelets for each class on 30 UEA MTSC datasets.

C COMPARISON WITH TIME SERIES REPRESENTATION METHODS

We conducted an experiment to compare the average accuracy of our ShapeFormer against SOTA representation methods, specifically TST [47], PatchTST [28], TimesNet [42], and GPT2 [49] (as shown in Table 5). By adhering to the GPT2 protocol across 10 datasets, our method outperforms the others on all datasets, achieving an average accuracy of 0.773.

Table 5: Comparison between our proposed ShapeFormer and SOTA time series representation methods.

	TST	GPT2	TimesNet	PatchTST	Ours
Averaging Accuracy	0.736	0.740	0.736	0.679	0.773