SVP-T: A Shape-Level Variable-Position Transformer for Multivariate Time Series Classification

Rundong Zuo[1], Guozhong Li[1], Byron Choi[1], Sourav S Bhowmick[2], Daphne Ngar-yin Mah[3], Grace L.H. Wong[4]

[1] Department of Computer Science, Hong Kong Baptist University, Hong Kong SAR
[2] School of Computing Engineering, Nanyang Technological University, Singapore
[3] Department of Geography, Hong Kong Baptist University, Hong Kong SAR
[4] Department of Medicine and Therapeutics, The Chinese University of Hong Kong, Hong Kong SAR
{csrdzuo, csgzli, bchoi}@comp.hkbu.edu.hk, assourav@ntu.edu.sg, daphnemah@hkbu.edu.hk, wonglaihung@cuhk.edu.hk

Abstract

Multivariate time series classification (MTSC), one of the most fundamental time series applications, has not only gained substantial research attentions but has also emerged in many real-life applications. Recently, using transformers to solve MTSC has been reported. However, current transformer-based methods take data points of individual timestamps as inputs (timestamp-level), which only capture the temporal dependencies, not the dependencies among variables. In this paper, we propose a novel method, called SVP-T. Specifically, we first propose to take time series subsequences, which can be from different variables and positions (time interval), as the inputs (*shape-level*). The temporal and variable dependencies are both handled by capturing the long- and short-term dependencies among *shapes*. Second, we propose a variable-position encoding layer (VP-layer) to utilize both the *variable* and *position* information of each shape. Third, we introduce a novel VP-based (Variable-Position) self-attention mechanism to allow the enhancing of the attention weights of overlapping shapes. We evaluate our method on *all* UEA MTS datasets. SVP-T achieves the best accuracy rank compared with several competitive state-of-the-art methods. Furthermore, we demonstrate the effectiveness of the VP-layer and the VP-based self-attention mechanism. Finally, we present one case study to interpret the result of SVP-T.

1 Introduction

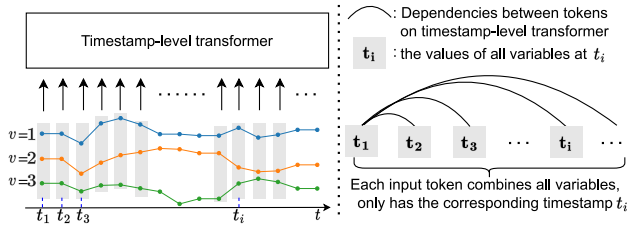
Multivariate time series (MTS), recording a group of variables at each timestamp, is ubiquitous in diverse domains, such as economics, smart city, medicine, and astronautics (Palpanas 2015). Time series classification is one of the most fundamental time series applications (Bagnall et al. 2017). However, compared to univariate time series classification (UTSC) (Bagnall et al. 2017; Shifaz et al. 2020), multivariate time series classification (MTSC) has received less research attention (Li et al. 2021). One of the key issues for MTSC is to capture the interactions among different variables (Ruiz et al. 2020), *e.g.*, one activity (playing badminton) determined by several shapes from different variables around the same time in human activity recognition (Bagnall et al. 2018).

Several methods have been proposed to solve the issue of such interactions of variables. (Hao and Cao 2020) proposed two novel cross-attention modules on their network. However, the model starts with a convolution operation with local neighborhood information layer by layer, which is known to be prone to the position-dependent prior bias on variables. ShapeNet captures the potential interactions between variables by learning subsequence representations (Li et al. 2021). Nevertheless, the representations do not consider the information about the variables and the positions of subsequences.

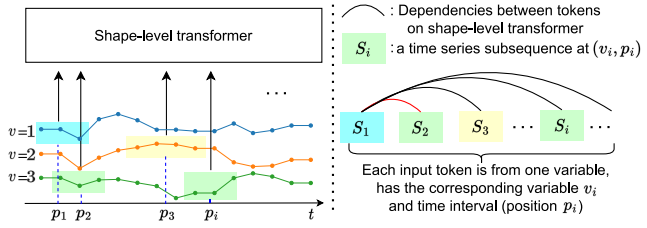
Recently, transformer models have exhibited superior performance in capturing long- and short-term dependencies in the applications of natural language processing (NLP) (Vaswani et al. 2017; Devlin et al. 2019). Moreover, some transformer-based methods have been proposed for time series applications. For instance, Informer employs transformer for time series forecasting (Zhou et al. 2021) for efficiently handling extremely long input sequences. (Zerveas et al. 2021) have proposed a transformer-based framework for time series representation learning for downstream tasks (*e.g.*, regression and classification). Applying a transformer on MTS avoids using a sequence-aligned recurrent architecture, *i.e.*, the models learn the temporal dependencies without position-dependent prior bias.

Challenges. We note that the input token of previous transformer-based methods (called *timestamp-level* transformer in this paper) for MTS is taken from one timestamp of all variables (Zhou et al. 2021; Zerveas et al. 2021), which means they only consider the temporal dependencies, and do not yet consider the dependencies between variables, since the transformer is for learning the dependencies among input tokens, not the dependencies within each individual token (shown in Figure 1(a)). Meanwhile, the input length of the timestamp-level transformer depends on the length of time series T, and thus the time complexity of the self-attention mechanism in the transformer is $O(T^2)$. Therefore, a time series with a large T makes a transformer time-consuming.

We are the first to take shapes as the input for the transformer (also called *shape-level* transformer in this paper, shown in Figure 1(b)) to handle both the temporal and variable dependencies. When using these shapes as the input, there are two additional challenges. First, the variable



(a) LHS: the timestamp-level transformer; RHS: the dependencies (illustrated with curved lines) of all tokens (timestamps) to the first token learned by the transformer. The input token fed into the timestamp-level transformer combines all variables at one timestamp, and thus the transformer only captures the temporal dependencies.



(b) LHS: our shape-level transformer; RHS: the dependencies (illustrated with curved lines) of all tokens (shapes) to the first token learned by the transformer. The input token of the shape-level transformer is from one variable v_i together with the corresponding time interval (position p_i). By capturing the dependencies between shapes, the dependencies of variables and positions are learned. The red line denotes the dependency between the 1-st and 3-rd variables.

Figure 1: (a) Timestamp-level input vs (b) our shape-level input for transformer.

and position (VP) information of the shapes have not been considered by the encoder of the transformer. Second, the shapes that appear almost simultaneously (*overlap in time*) are important to many applications, such as human activity recognition (Zhou, De la Torre, and Hodgins 2012; Wang et al. 2019), which can be specifically emphasized.

Contributions. In this paper, we propose a *shape-level variable-position* transformer model, named SVP-T, to address the above-mentioned challenges. To the best of our knowledge, this is the first work to take time series subsequences as the input (shape-level) and consider their variable and position information in a transformer architecture. An overview of SVP-T is presented in Figure 2.

SVP-T takes the shapes from different variables and positions as input. The novelty is that we ① utilize shapes nearby the cluster centers, obtained from a clustering algorithm, rather than each timestamp, as the input for SVP-T. Figure 1 shows the difference between a previous timestamp-level transformer (Zerveas et al. 2021) (Figure 1(a)) and our shape-level transformer, SVP-T (Figure 1(b)). Instead of using a fixed position encoding in (Vaswani et al. 2017) or a fully learnable position encoding in (Devlin et al. 2019; Zerveas et al. 2021), we ② propose a variable-position encoding layer (VP-layer) to specifically utilize the variable and position of our shapes, which makes the model sense the input time series instance. Finally, we ③ introduce a novel VP-based (Variable-Position) self-attention mechanism to

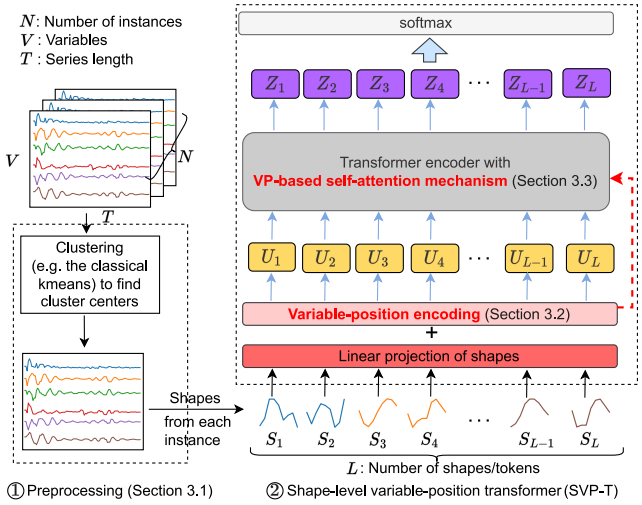


Figure 2: The overview of our method. ① Preprocessing. A variable-wise clustering method discovers L shapes of each instance as the input tokens (Section 3.1). ② SVP-T. A VP-layer encodes variable-position information of shapes (Section 3.2) and a VP-based self-attention mechanism in the transformer encoder which captures long- and short-dependencies between shapes (Section 3.3). $\{S_1, S_2, \cdots, S_L\}$ are the shapes shown in Figure 1(b).

enable enhancement of the attention weights of overlapping shapes for MTSC. As a proof of concept, we propose to increase the attention weight of two shapes that are close in time but from different variables.

We conduct experiments on all datasets of the UEA MTS archive. The evaluation shows that our SVP-T outperforms 12 baseline methods in terms of accuracy rank and the additional experiments show the effectiveness of our proposed modules. Furthermore, we perform a case study to show how our SVP-T handles input shapes by analyzing the internal representation.

Organization. The rest of this paper is organized as follows. Section 2 presents the related work on MTS. The definition of MTS and details of SVP-T are introduced in Section 3. Section 4 reports the experimental results and one case study. Section 5 concludes the paper and presents avenues for future work.

2 Related Work

In this section, we present the existing methods for MTSC and the transformer-based methods for time series.

2.1 Multivariate Time Series Classification

We categorize the MTSC methods into two types, namely traditional (non-deep learning) methods and deep learning ones. For details of traditional methods, interested readers can refer to an excellent survey paper (Ruiz et al. 2020). Hence, we focus more on deep learning methods.

LSTM-FCN (Karim et al. 2019) was proposed with an LSTM layer and stacked CNN layers to generate the features from time series. The features are then transmitted

to a softmax layer to output the probability of each class. (Hao and Cao 2020) found an issue that CNN-based models cannot capture the long dependencies well among different variables. To handle the issue, they added two cross attention modules on their CNN-based model. TapNet applies LSTM, CNN, and attentional prototype learning techniques for MTSC (Zhang et al. 2020). Furthermore, it utilizes unlabeled data for semi-supervised learning and obtains better performances. (Li et al. 2021) proposed ShapeNet to learn the time series subsequences representation into a unified space for the final shapelet selection. However, they did not consider the information of positions in time series subsequences. RLPAM (Gao et al. 2022) first converts MTS into a univariate cluster sequence and then uses reinforcement learning for selecting patterns.

2.2 Transformer-Based Methods for Time Series

The transformer architecture has successfully been applied to important applications including natural language processing (Vaswani et al. 2017; Devlin et al. 2019) and computer vision (Dosovitskiy et al. 2021; Liu et al. 2021).

Recently, there have been some successes in time series applications, such as forecasting (Wu et al. 2020; Lim et al. 2021) and representation learning (Eldele et al. 2021). Informer employs transformer for efficiently forecasting long sequence time series under long-tail distribution (Zhou et al. 2021). TS-TCC (Eldele et al. 2021) utilizes transformer in temporal contrasting to extract the temporal features for unsupervised time series representation learning. (Zerveas et al. 2021) proposed a novel framework based on the transformer encoder to learn the representation of MTS for downstream tasks (*e.g.*, classification and regression). The efficiency of this method is limited by the length of the time series and its performance is relatively weak in low dimension datasets (Zerveas et al. 2021). We notice that all the previous transformer-based methods take the data at timestamps of the time series as the input (as shown in Figure 1(a)).

3 Methods

In this section, we first give some notations of multivariate time series. Then we propose a novel shape-level variable-position transformer, called SVP-T, for MTSC. Specifically, we propose shape inputs for the transformer, a variable-position encoding layer, and a VP-based self-attention mechanism.

Multivariate Time Series (MTS). Prior to the technical discussions, we provide some notations here. We denote MTS as $\mathbf{X} \in \mathcal{R}^{V \times T}$, where V is the number of variables and T is the series length of each variable. Specifically, $\mathbf{X} = \left\{X^1, X^2, \cdots, X^V\right\}$, where X^v is the time series of the v-th variable. $X^v = (x_1^v, x_2^v, \cdots, x_T^v)$, where x_t^v is the value of the t-th timestamp in the v-th variable. An MTS dataset shown in the top left of Figure 2 consists of N instances and each instance has one corresponding class label.

3.1 Shapes Input for Transformer

An excellent review paper (Ruiz et al. 2020) demonstrates that the discriminative time series subsequences (*e.g.*, interval-based, dictionary-based, and shapelet-based methods) significantly improve the accuracy performance of MTSC. Inspired by the aforementioned results, we transform original time series into shapes as the input for our SVP-T. We follow the existing works (Zakaria et al. 2016; Grabocka, Wistuba, and Schmidt-Thieme 2016; Li et al. 2021) to apply clustering for discovering shapes.

For self-containedness, we describe how shapes are generated as follows. First, we generate numerous time series subsequences for all instances. Second, we employ a classic clustering algorithm (*e.g.*, kmeans) based on Euclidean Distance to each variable. The K cluster centers for variable v are $\mathbf{C}^v = \{C_1^v, C_2^v, \cdots, C_K^v\}$, which are representative of a number of time series subsequences of v. The nearest time series subsequences $\mathbf{S}^v = \{S_1^v, S_2^v, \cdots, S_K^v\}$ to $\mathbf{C}^v$ are the tokens as input shapes of SVP-T. After that, one MTS instance $\mathbf{X}$ is transformed into a set of shapes $\mathbf{S} = \{S_1, S_2, \cdots, S_L\}$, where $L = K \times V$.

From Figure 1(a), we can observe that the input length of previous transformer-based methods are determined by the length of MTS (namely, T). As the time complexity of the self-attention mechanism is $O(T^2)$ (Vaswani et al. 2017), this would be inefficient in handling long time series in a timestamp-level transformer. In contrast, L in our shape-level transformer is tunable. From our experiments shown in Figure 7, we achieve good accuracy when L is much smaller than T (*e.g.*, EigenWorms, and StandWalkJump).

3.2 Variable-Position Encoding Layer

Since the transformer model contains no recurrence and no convolution, a fixed sinusoidal position encoding, is proposed to capture the input sequence in the seminal work of transformer (Vaswani et al. 2017). BERT (Devlin et al. 2019) proves that a fully learnable position encoding performs better. Both the fixed position encoding and fully learnable position encoding utilize the order of sequence for input tokens, *i.e.*, word embeddings in NLP (Vaswani et al. 2017). However, such order of sequence may not carry the semantics for the need of other applications (Shiv and Quirk 2019).

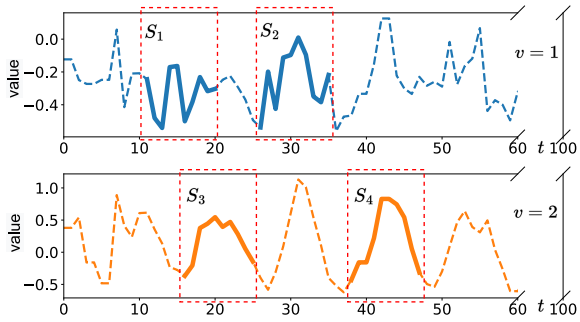
As we mentioned in Section 3.1, the input tokens of SVP-T are the shapes. In the "shapes-as-input-tokens" situation, the order of sequence for shapes is less important when compared to the relative positions of the shapes in the original time series instance, which are the corresponding variables and time intervals (called VP information). An example of VP information for input shapes is illustrated in Figure 3.

To include such VP information in learning, we propose a *variable-position encoding layer* (VP-layer) in SVP-T, which is shown in the middle right-hand side of Figure 2.

The input of the VP-layer is the VP information. Specifically, for the i-th shape S_i, the corresponding VP information P_i is defined as follows:

$$P_i = (\frac{v_i}{V}, \frac{t_{i,start}}{T}, \frac{t_{i,end}}{T}) \tag{1}$$

where v_i, $t_{i,start}$ and $t_{i,end}$ are the variable, the first timestamp, and the last timestamp of S_i, respectively. We normalize them to the range of $[0, 1]$ by dividing them with the number of variables V or their original time series length T.



(a) The shapes S_1, S_2, S_3, and S_4 in their original time series instance (Blue and orange are the two variables $(1, 2)$ of the partial MTS instance from the BasicMotions dataset)

Shape	S_1	S_2	S_3	S_4
VP information	$(\frac{1}{6}, 0.11, 0.2)$	$(\frac{1}{6}, 0.26, 0.35)$	$(\frac{2}{6}, 0.16, 0.25)$	$(\frac{2}{6}, 0.38, 0.47)$

(b) The VP information (P_i in Formula 1) of the shapes. The first, second, and third components are the first timestamp, the last timestamp, and the corresponding variable, respectively (The length $T = 100$ and the number of variables $V = 6$).

Figure 3: An example to illustrate VP information

P_i is transmitted into a linear layer with a trainable weight W_p:

$$P'_i = P_i W_p, W_p \in \mathcal{R}^{3 \times d_{model}} \tag{2}$$

where d_{model} is the dimension of the input in the transformer (Vaswani et al. 2017). The final i-th input U_i of our transformer encoder is:

$$U_i = P'_i + W_s S_i \tag{3}$$

where W_s is the shared weight of a linear layer (shown in Figure 2: a linear projection of shapes).

3.3 VP-Based Self-Attention Mechanism

We propose a VP-based self-attention mechanism for the transformer encoder to capture the long- and short-term dependencies between different shapes. The canonical self-attention transforms the input embedding U into three feature spaces Q, K, and V, where all of them are vectors (Vaswani et al. 2017). They are defined as follows:

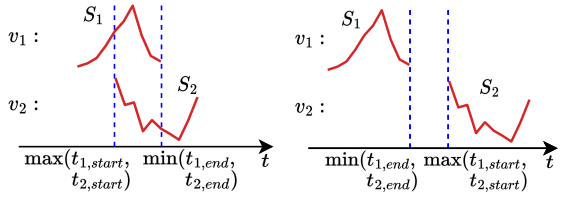
$$Q = UW_q, K = UW_k, V = UW_v \tag{4}$$

where $W_q, W_k \in \mathcal{R}^{d_{model} \times d_k}$, $W_v \in \mathcal{R}^{d_{model} \times d_v}$. For clarity, we let $d = d_k = d_v$ in this paper. The projection space Q, K, and V are called query, key, and value space, respectively. They are used to compute the attention by mapping a query and a set of key-value pairs to an output. The attention is computed as follows:

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d}})V \tag{5}$$

The attention weights for the values are calculated as:

$$Score = \text{softmax}(\frac{QK^T}{\sqrt{d}}) \tag{6}$$



(a) (S_1, S_2) overlap (b) (S_1, S_2) do not overlap

Figure 4: An illustration of overlapping calculation (shown in Formula 8). (S_1, S_2) are two shapes of different variables (v_1, v_2) from the BasicMotions dataset. (a) $\min(t_{1,end}, t_{2,end}) > \max(t_{1,start}, t_{2,start})$. (b) $\min(t_{1,end}, t_{2,end}) < \max(t_{1,start}, t_{2,start})$.

There is, however, a special characteristic of MTS datasets that needs further consideration. When two shapes ① overlap with each other and ② are from different variables, they together can be important shapes for MTSC and their attention weight should be higher. It should be remarked that the overlapping behavior has been successfully applied to the 2D-image data in object detection (Girshick et al. 2014; Rezatofighi et al. 2019). We are the first to utilize the corresponding characteristics for MTSC. Figure 4(a) shows an example of overlapping shapes S_1 and S_2.

Incorporating the above-mentioned characteristics into MTSC, we propose a VP-based self-attention mechanism, which allows the attention weights of these overlapping shapes to be explicitly enhanced. Our goal is to modify the attention weights matrix $Score$ to $Score'$ using a matrix M called the Overlapping-Enhancement matrix:

$$Score' = \text{softmax}(Score \odot M) \tag{7}$$

The overlapping of two shapes S_i, S_j is defined as:

$$O_{lap}(S_i, S_j) = \begin{cases} \max(\min(t_{i,end}, t_{j,end}) - \\ \quad \max(t_{i,start}, t_{j,start}), 0) & v_i \neq v_j \\ 0 & v_i = v_j \end{cases} \tag{8}$$

For presentation simplicity, we use v, $t_{i,start}$, and $t_{i,end}$ to present $\frac{v_i}{V}$, $\frac{t_{i,start}}{T}$, and $\frac{t_{i,end}}{T}$, when T, V are not relevant to the discussion.

We remark a special case where two shapes are from the same variable, their overlapping part is repetitive and thus, $O_{lap} = 0$. For shapes from different variables, we give an example of overlapping shapes in Figure 4.

Example 1. *(S_1, S_2) are two time series subsequences from two variables. There are two cases. ① When there is an overlapping between (S_1, S_2), namely, $\min(t_{1,end}, t_{2,end})$ is larger than $\max(t_{1,start}, t_{2,start})$ as shown in Figure 4(a). Their O_{lap} is then larger than 0. ② When there is no overlapping between (S_1, S_2), namely, $\min(t_{1,end}, t_{2,end})$ is smaller than $\max(t_{1,start}, t_{2,start})$, as shown in Figure 4(b). Their O_{lap} is 0.*

Finally, one element $M(i, j)$ of the Overlapping-Enhancement matrix is calculated by:

$$M(i, j) = \alpha e^{\text{relu}(O_{lap}(S_i, S_j) - \beta)} \tag{9}$$

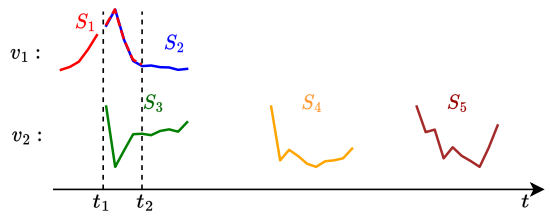


Figure 5: An example of where the attention weight should be increased. There are three circumstances. ① (S_1, S_2), they overlap in time, but they come from the same variable. ② both (S_1, S_4) and (S_1, S_5), they do not overlap in time. ③ (S_1, S_3), they overlap in time and come from different variables. The dash line is the overlapping part of (S_1, S_2). Only the attention weight of ③ should be increased.

where $\alpha \geq 1$, and $\beta \in [0, 1]$. If the overlapping of two shapes (S_i, S_j) is larger than β, the relu function leads it to stay the same and the result of $M(i, j)$ would be larger than one, which amplifies the corresponding attention weight. Otherwise, the relu makes it to be 0 and the attention weight remains unchanged. That is, no matter how far two shapes are in position, if their overlapping is less than β, the self-attention mechanism would learn the long- and short-term dependencies equally, as (S_1, S_4) and (S_1, S_5) shown in Figure 5. Meanwhile, if the above characteristic does not exist, we can set $\beta \geq 1$ and the self-attention mechanism could be as before.

4 Experiments

In this section, we evaluate the performance of SVP-T. We start by introducing the experiment setting, including the environment, datasets, metrics, and the implementation details. Then, we present the baseline methods compared in Section 4.2. In Section 4.3, we report the accuracies of all methods. We investigate the effectiveness of our VP-layer and VP-based self-attention mechanism in Section 4.4. In addition, we conduct an experiment on how L influences the classification results in Section 4.5. Finally, we present a case study of a dataset named BasicMotions.

4.1 Experiment Setting

Environment. All the experiments are implemented on a machine with one Xeon Gold 6226 CPU @ 2.70GHz and one NVIDIA Tesla V100S. Python 3.8, and Pytorch 1.10.0 are used to build and train our model. **Datasets.** We evaluate our method on a well-known benchmark of MTS, the UEA archive, which is consists of 30 different datasets. Details of these datasets can be found in (Bagnall et al. 2018). **Metrics.** We choose classification accuracy as our evaluation metric. Meanwhile, we compute the average rank, the number of Top-1, Top-3, and Top-5 accuracy, the number of wins/draws/losses, to show the robustness of different methods. Finally, we conduct Friedman and Wilcoxon signed-rank test following the process in (Demšar 2006) to evaluate whether the result is statistically significant. **Implementation details.** We set $\alpha = 1.5$ and $\beta = 0$ in Formula 9. Since the benchmark datasets are highly heterogeneous, as well as the MTS data in nature, we follow the previous work (Zerveas et al. 2021), that splits the training set into two parts, $80\% - 20\%$. Then, we take the 20% part as the validation set to tune the hyperparameters. When the hyperparameters are fixed, we train our model on the whole training set and finally, evaluate it on the official test set. Our tuned hyperparameters of all datasets are shown in Appendix A.1. We adopt batch normalization, instead of layer normalization, for better performance on time series applications (Zerveas et al. 2021).

4.2 Baselines

We compare SVP-T with 12 methods and briefly introduce each of them. Readers who are interested in them may refer to the original papers. **Three benchmarks** (Bagnall et al. 2018) (EDI, $DTWI$, and $DTWD$) are based on Euclidean Distance, dimension-independent dynamic time warping, and dimension-dependent dynamic time warping. **MLSTM-FCNs** (Karim et al. 2019) is a deep learning method for MTS, which applies an LSTM layer and stacked CNN layers to generate features. **WEASEL-MUSE** (Schäfer and Leser 2017) is a bag-of-pattern based approach which extracts and represents features to words. **Scalable Representation Learning (SRL)** (Franceschi, Dieuleveut, and Jaggi 2019) employs negative sampling techniques with an encoder-based architecture to learn the representation. **TapNet** (Zhang et al. 2020) is a recent model with an attentional prototype learning in its deep learning-based network for MTSC. **ShapeNet** (Li et al. 2021) projects the MTS subsequences into a unified space and applies clustering to find the shapelets. **Rocket, MiniRocket** (Dempster, Petitjean, and Webb 2020; Dempster, Schmidt, and Webb 2021) use random convolutional kernels to extract features from univariate time series, and they extended their codes to MTSC. **RLPAM** (Gao et al. 2022) introduces reinforcement learning to the pattern mining of MTS. **TStamp Transformer** (Zerveas et al. 2021) takes the values at each timestamp as the input for a transformer encoder as Figure 1(a) shown, which is the baseline for evaluating our idea of taking shapes as input.

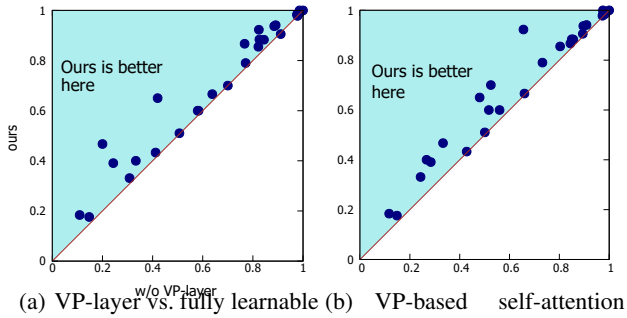
4.3 Performance Evaluation

The accuracies of all the baseline experimental results are taken from the original papers or the survey (Ruiz et al. 2020) except for MiniRocket and TStamp Transformer (Zerveas et al. 2021). The details of them are shown in Appendix A.2. We set a fixed random seed for reproducibility. For consistency of presentation, we follow the results of (Gao et al. 2022) to keep three decimal places.

As Table 1 shown, the overall accuracy of SVP-T outperforms all the related methods. Specifically, the average rank of SVP-T is 4.017, which is the best among 13 methods. Meanwhile, the gap in terms of average rank between SVP-T and the runner-up, MiniRocket, is about 1, which shows a clear lead considering that the average ranks of MiniRocket and RLPAM are nearly the same (less than 0.1 difference). For the number of top-1 accuracy, we find that SVP-T is slightly lower than RLPAM and MiniRocket. However, the number of top-3 accuracies and the number of top-5 accuracies of SVP-T are both higher than all of the other methods,

	EDI	DTWI	DTWD	MLSTM -FCNs	WEASEL +MUSE	SRL	TapNet	ShapeNet	Rocket	Mini Rocket	RLPAM	TStamp Transformer	Ours
AWR	0.970	0.980	0.987	0.973	0.990	0.987	0.987	0.987	**0.996**	0.992	0.923	0.983	0.993
AF	0.267	0.267	0.220	0.267	0.333	0.133	0.333	0.400	0.249	0.133	**0.733**	0.200	0.400
BM	0.676	1.000	0.975	0.950	**1.000**	**1.000**	**1.000**	**1.000**	0.990	**1.000**	**1.000**	0.975	**1.000**
CT	0.964	0.969	0.989	0.985	0.990	0.994	**0.997**	0.980	N/A	0.993	0.978	N/A	0.990
CK	0.944	0.986	**1.000**	0.917	**1.000**	0.986	0.958	0.986	**1.000**	0.986	**1.000**	0.958	**1.000**
DDG	0.275	0.550	0.600	0.675	0.575	0.675	0.575	**0.725**	0.461	0.650	0.700	0.480	0.700
EW	0.549	N/A	0.618	0.504	0.890	0.878	0.489	0.878	0.863	**0.962**	0.908	N/A	0.923
EP	0.666	0.978	0.964	0.761	**1.000**	0.957	0.971	0.987	0.991	**1.000**	0.978	0.920	0.986
ER	0.133	0.914	0.929	0.133	0.133	0.133	0.133	0.133	**0.981**	**0.981**	0.819	0.933	0.937
EC	0.293	0.304	0.323	0.373	0.430	0.236	0.323	0.312	0.447	**0.468**	0.369	0.337	0.331
FD	0.519	0.000	0.529	0.545	0.545	0.528	0.556	0.602	**0.694**	0.620	0.621	0.681	0.512
FM	0.550	0.520	0.530	0.580	0.490	0.540	0.530	0.580	0.553	0.550	0.640	**0.776**	0.600
HMD	0.278	0.306	0.231	0.365	0.365	0.270	0.378	0.338	0.446	0.392	**0.635**	0.608	0.392
HW	0.200	0.316	0.286	0.286	**0.605**	0.533	0.357	0.452	0.567	0.507	0.522	0.305	0.433
HB	0.619	0.658	0.717	0.663	0.727	0.737	0.751	0.756	0.718	0.771	0.779	0.712	**0.790**
IW	0.128	N/A	N/A	0.167	N/A	0.160	0.208	0.250	N/A	0.595	0.352	**0.684**	0.184
JV	0.924	0.959	0.949	0.976	0.973	0.989	0.965	0.984	0.965	0.989	0.935	**0.994**	0.978
LB	0.833	0.894	0.870	0.856	0.878	0.867	0.850	0.856	0.906	**0.922**	0.794	0.844	0.883
LSST	0.456	0.575	0.551	0.373	0.590	0.558	0.568	0.590	0.632	0.643	0.643	0.381	**0.666**
MI	0.510	N/A	0.500	0.510	0.500	0.540	0.590	0.610	0.531	0.550	0.610	N/A	**0.650**
NT	0.850	0.850	0.883	0.889	0.870	0.944	0.939	0.883	0.885	0.928	**0.950**	0.900	0.906
PD	0.705	0.939	0.977	0.978	0.948	0.983	0.980	0.977	**0.996**	N/A	0.982	0.974	0.983
PM	**0.973**	0.734	0.711	0.699	0.000	0.688	0.751	0.751	0.856	0.522	0.632	0.919	0.867
PH	0.104	0.151	0.151	0.110	0.190	0.246	0.175	**0.298**	0.284	0.292	0.175	0.088	0.176
RS	0.868	0.842	0.803	0.803	**0.934**	0.862	0.868	0.882	0.928	0.868	0.868	0.829	0.842
SCP1	0.771	0.765	0.775	0.874	0.710	0.846	0.652	0.782	0.866	**0.925**	0.802	**0.925**	0.884
SCP2	0.483	0.533	0.539	0.472	0.460	0.556	0.550	0.578	0.514	0.522	**0.632**	0.589	0.600
SAD	0.967	0.959	0.963	0.990	0.982	0.956	0.983	0.975	0.630	0.620	0.621	**0.993**	0.986
SWJ	0.200	0.333	0.200	0.067	0.333	0.400	0.400	0.533	0.456	0.333	**0.667**	0.267	0.467
UGL	0.881	0.868	0.903	0.891	0.916	0.884	0.894	0.906	0.944	0.938	**0.944**	0.903	0.941
Avg.Rank	10.533	9.450	8.850	8.750	6.883	7.100	6.950	5.517	5.417	5.000	5.050	7.483	**4.017**
Num.Top-1	1	1	1	0	5	1	2	3	5	7	**8**	5	5
Num.Top-3	1	2	1	1	6	6	3	7	13	14	16	9	**18**
Num.Top-5	2	2	3	5	15	12	13	18	17	21	20	10	**24**
Wins	27	27	28	27	21	21	24	19	17	17	15	21	-
Draws	0	2	1	0	3	2	1	2	1	1	3	0	-
Losses	3	1	1	3	6	7	5	9	12	12	12	9	-
P-value	0.000	0.000	0.000	0.000	0.006	0.003	0.000	0.118	0.217	0.765	0.967	0.047	-

Table 1: Accuracies of our method and 12 related methods on all datasets of the UEA archive



(a) VP-layer vs. fully learnable positional encoding in transformer (b) VP-based self-attention mechanism vs. original self-attention mechanism

Figure 6: Effectiveness analysis

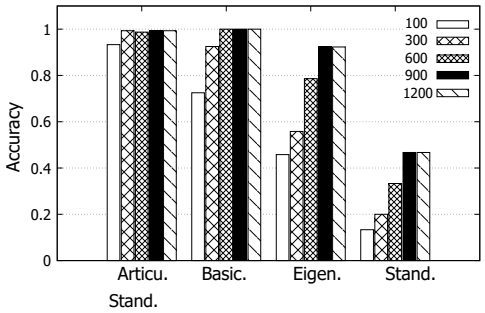


Figure 7: Accuracy by varying the number of shapes L. The series length $T = 144, 100, 17984, 2500$ for four datasets.

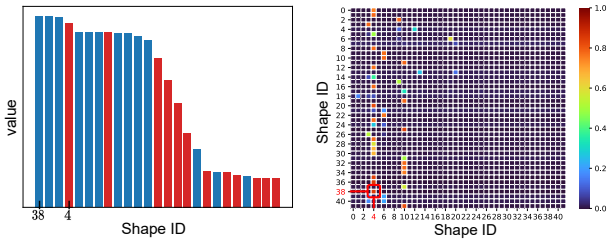
which shows the results of SVP-T are more robust. Also, the performance of RLPAM relies on the quality of its univariate cluster sequence and MiniRocket needs random kernels for transformation. In terms of 1-to-1 comparison with other methods, SVP-T wins/draws on at least 18 out of 30 datasets. For the Friedman and Wilcoxon test, we set the significant level to $\alpha = 0.05$ as (Bagnall et al. 2018; Li et al. 2021). The statistical significance is $p \leq 0.05$, which confirms there is a significant difference among the 13 methods. The p-values of SVP-T to all methods are less than 0.05, which indicates the results are statistically significant except for ShapeNet, Rocket, MiniRocket, and RLPAM.

Comparison with Timestamp-level transformer. To evaluate our contribution of taking shapes as the input of a transformer, we compare our SVP-T with TStamp Transformer (Zerveas et al. 2021) shown in the last two columns of Table 1. In terms of 1-to-1 comparison, SVP-T wins on 21 out of the 30 datasets. The p-value of SVP-T to TStamp Transformer is smaller than 0.05, which shows a statistically significant improvement of taking shapes as input.

4.4 Effectiveness Analysis

We conduct experiments to demonstrate the effectiveness of the VP-layer and the VP-based self-attention mechanism.

VP-layer vs. fully learnable positional encoding. To study the performance of the VP-layer, we change the VP-layer to fully learnable positional encoding and conduct an experiment on all the datasets. Figure 6(a) shows that the accuracies of using the VP-layer are clearly better than fully learnable positional encoding, which supports the effectiveness the VP-layer proposed in Section 3.2.



(a) Attention (b) Attention weights

Figure 8: Visualization of one instance from the class playing badminton: (a) The X-axis denotes the ID of input shapes, and the Y-axis denotes the attention. The shapes are ranked by the values of Y-axis. (b) The X-axis and Y-axis denote the ID of input shapes. For (i, j), the color indicates the attention weight between two shapes (S_i, S_j). *e.g.*, the attention weight between shapes S_4 and S_{38} is shown in red.

VP-self-attention mechanism vs. original self-attention. To study the performance of our VP-based self-attention mechanism, we change the self-attention to the original self-attention mechanism (Vaswani et al. 2017). The result in Figure 6(b) shows that the accuracies achieved using the VP-based self-attention mechanism proposed in Section 3.3 are above directly using the original self-attention mechanism.

4.5 Evaluation the Effect of L

We conduct experiments to investigate how the number of input shapes L would influence the classification result.

Generally, the more shapes, the more accurate SVP-T. On the other hand, the time complexity of the transformer-based model is $O(L^2)$. Hence, there is a trade-off between accuracy and efficiency. We report four datasets because they have a wide range of lengths and variables.

Figure 7 shows the accuracy by varying L. There is no significant improvement in the accuracy of ArticularyWordRecognition when L is larger than 100, which means a small number $L = 100$ shapes could have a great performance. The same phenomenon can be observed in other datasets when L varies. Therefore, the efficiency of our model could be further improved by finding the minimum L for each dataset. We also find that when L is larger than 900, the accuracies of all datasets remain unchanged. Therefore, we set the default L for all datasets to 900.

4.6 A Case Study of BasicMotions

To interpret the result of SVP-T, a human activity recognition dataset from UEA archive, BasicMotions with four classes (namely, playing badminton, standing, walking, and running) and six variables (Bagnall et al. 2018), is employed to illustrate without domain knowledge. We follow the steps of visualization attention and attention weights in (Abnar and Zuidema 2020).

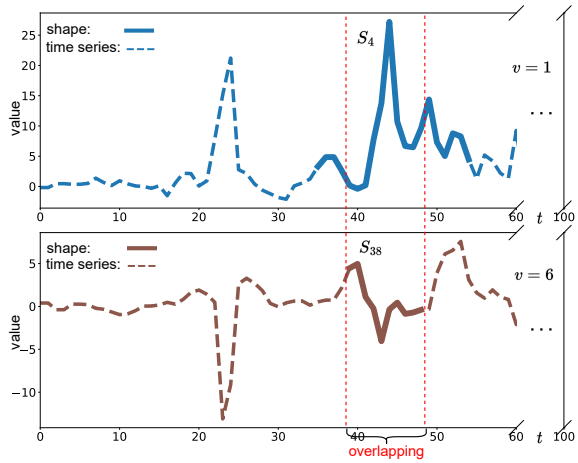


Figure 9: The overlapping shapes S_4 and S_{38} of one instance from the class "playing badminton". The shapes S_4 and S_{38} are from different variables and overlap in time.

We randomly select one instance from the class "playing badminton" in the test set for analysis. Figure 8(a) shows the attention of different shapes. The red bars and the blue bars are from the first three variables (3D accelerometer), and the last three variables (3D gyroscope), respectively. We can observe that the shapes with higher attention are mostly from the last three variables (blue). The possible reason is that the discriminative shapes more likely exist in the three gyroscope variables (blue) for the class "playing badminton" than for the other three classes.

We also find the shape S_4 has high attention while it belongs to the first variable (3D accelerometer). To seek the reason, we further visualize the attention weights matrix ($Score'$ in Formula 7) shown in Figure 8(b). We discover the attention weights given to S_4 are relatively higher than others. Specifically, the high attention weight between S_{38} and S_4 indicates a higher dependency. By tracking them back to the original time series instance, a clear overlapping behavior (shown in Figure 9) can be observed, which is described in Section 3.3, and further illustrates the effectiveness of our VP-based self-attention mechanism. Meanwhile, Figure 8(b) shows the high data sparsity (the colored points vs. the dark points) in our SVP-T, which has the potential of utilizing sparsity to improve efficiency (Zhou et al. 2021).

5 Conclusion

In this paper, we propose a novel shape-level variable-position transformer method, named SVP-T, for MTSC. We use time series subsequences, rather than timestamps, as the input tokens of a transformer-based model, which captures both variable and position dependencies in MTS. In particular, a variable-position encoding layer is proposed to utilize the VP information of each shape. In addition, we propose a variable-position self-attention mechanism in SVP-T to enhance the attention weights of the overlapping shapes. The experiment shows the accuracy of SVP-T has the highest rank when compared to the state-of-the-art methods. As for future work, we plan to utilize the sparsity we observed from Section 4.6 to improve the efficiency of SVP-T.

Acknowledgments

Thanks for the helpful feedbacks from the anonymous reviewers. This work was supported by the Hong Kong Research Grant Council (HKRGC), RIF R2002-20F, and Hong Kong Baptist University's Interdisciplinary Research Clusters Matching Scheme, HKBU IRCMS/19-20/H01.

References

Abnar, S.; and Zuidema, W. 2020. Quantifying Attention Flow in Transformers. In *ACL*.

Bagnall, A.; Dau, H. A.; Lines, J.; Flynn, M.; Large, J.; Bostrom, A.; Southam, P.; and Keogh, E. 2018. The UEA multivariate time series classification archive, 2018. arXiv:1811.00075.

Bagnall, A.; Lines, J.; Bostrom, A.; Large, J.; and Keogh, E. 2017. The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances. *DMKD*.

Dempster, A.; Petitjean, F.; and Webb, G. I. 2020. ROCKET: Exceptionally Fast and Accurate Time Series Classification Using Random Convolutional Kernels. *DMKD*.

Dempster, A.; Schmidt, D. F.; and Webb, G. I. 2021. MiniRocket: A Very Fast (Almost) Deterministic Transform for Time Series Classification. In *SIGKDD*.

Demšar, J. 2006. Statistical comparisons of classifiers over multiple data sets. *JMLR*.

Devlin, J.; Chang, M.; Lee, K.; and Toutanova, K. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL*.

Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; Uszkoreit, J.; and Houlsby, N. 2021. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *ICLR*.

Eldele, E.; Ragab, M.; Chen, Z.; Wu, M.; Kwoh, C. K.; Li, X.; and Guan, C. 2021. Time-Series Representation Learning via Temporal and Contextual Contrasting. In *IJCAI*.

Franceschi, J.-Y.; Dieuleveut, A.; and Jaggi, M. 2019. Unsupervised Scalable Representation Learning for Multivariate Time Series. In *NeurIPS*, 4652–4663.

Gao, G.; Gao, Q.; Yang, X.; Pajic, M.; and Chi, M. 2022. A Reinforcement Learning-Informed Pattern Mining Framework for Multivariate Time Series Classification. In *IJCAI*.

Girshick, R.; Donahue, J.; Darrell, T.; and Malik, J. 2014. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. In *ICCV*.

Grabocka, J.; Wistuba, M.; and Schmidt-Thieme, L. 2016. Fast classification of univariate and multivariate time series through shapelet discovery. *KAIS*, 49(2): 429–454.

Hao, Y.; and Cao, H. 2020. A New Attention Mechanism to Classify Multivariate Time Series. In *IJCAI*.

Karim, F.; Majumdar, S.; Darabi, H.; and Harford, S. 2019. Multivariate LSTM-FCNs for time series classification. *Neural Networks*, 116: 237–245.

Li, G.; Choi, B.; Xu, J.; Bhowmick, S. S.; Chun, K.-P.; and Wong, G. L. 2021. Shapenet: A shapelet-neural network approach for multivariate time series classification. In *AAAI*.

Lim, B.; Arık, S. Ö.; Loeff, N.; and Pfister, T. 2021. Temporal fusion transformers for interpretable multi-horizon time series forecasting. *International Journal of Forecasting*.

Liu, Z.; Lin, Y.; Cao, Y.; Hu, H.; Wei, Y.; Zhang, Z.; Lin, S.; and Guo, B. 2021. Swin Transformer: Hierarchical Vision Transformer using Shifted Windows. In *ICCV*.

Palpanas, T. 2015. Data series management: The road to big sequence analytics. *ACM SIGMOD Record*, 44(2): 47–52.

Rezatofighi, H.; Tsoi, N.; Gwak, J.; Sadeghian, A.; Reid, I.; and Savarese, S. 2019. Generalized intersection over union: A metric and a loss for bounding box regression. *CVPR*.

Ruiz, A. P.; Flynn, M.; Large, J.; Middlehurst, M.; and Bagnall, A. 2020. The great multivariate time series classification bake off: a review and experimental evaluation of recent algorithmic advances. *DMKD*.

Schäfer, P.; and Leser, U. 2017. Multivariate time series classification with WEASEL+ MUSE. *arXiv preprint arXiv:1711.11343*.

Shifaz, A.; Pelletier, C.; Petitjean, F.; and Webb, G. I. 2020. TS-CHIEF: a scalable and accurate forest algorithm for time series classification. *DMKD*.

Shiv, V.; and Quirk, C. 2019. Novel positional encodings to enable tree-based transformers. In *NeurIPS*.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. In *NeurIPS*.

Wang, H.; Ho, E. S.; Shum, H. P.; and Zhu, Z. 2019. Spatio-temporal manifold learning for human motions via long-horizon modeling. *TVCG*, 27(1): 216–227.

Wu, N.; Green, B.; Ben, X.; and O'Banion, S. 2020. Deep transformer models for time series forecasting: The influenza prevalence case. *arXiv preprint arXiv:2001.08317*.

Zakaria, J.; Mueen, A.; Keogh, E.; and Young, N. 2016. Accelerating the discovery of unsupervised-shapelets. *DMKD*.

Zerveas, G.; Jayaraman, S.; Patel, D.; Bhamidipaty, A.; and Eickhoff, C. 2021. A Transformer-Based Framework for Multivariate Time Series Representation Learning. In *SIGKDD*.

Zhang, X.; Gao, Y.; Lin, J.; and Lu, C.-T. 2020. Tapnet: Multivariate time series classification with attentional prototypical network. In *AAAI*.

Zhou, F.; De la Torre, F.; and Hodgins, J. K. 2012. Hierarchical aligned cluster analysis for temporal clustering of human motion. *TPAMI*, 35(3): 582–596.

Zhou, H.; Zhang, S.; Peng, J.; Zhang, S.; Li, J.; Xiong, H.; and Zhang, W. 2021. Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting. In *AAAI*.