CNN Kernels Can Be the Best Shapelets

Eric Qu*
University of California, Berkeley
Berkeley, CA 94720
ericqu@berkeley.edu

Yansen Wang
Microsoft Research Asia
Shanghai 200232, China
yansenwang@microsoft.com

Xufang Luo
Microsoft Research Asia
Shanghai 200232, China
xufluo@microsoft.com

Wenqiang He
University of Science and Technology of China
Hefei 230026, China
wenqianghe@mail.ustc.edu.cn

Dongsheng Li
Microsoft Research Asia
Shanghai 200232, China
dongsheng.li@microsoft.com

Abstract

Shapelets and CNN are two typical approaches to model time series. Shapelets aim at finding a set of sub-sequences that extract feature-based interpretable shapes, but may suffer from accuracy and efficiency issues. CNN performs well by encoding sequences with a series of hidden representations, but lacks interpretability. In this paper, we demonstrate that shapelets are essentially equivalent to a specific type of CNN kernel with a squared norm and pooling. Based on this finding, we propose ShapeConv, an interpretable CNN layer with its kernel serving as shapelets to conduct time-series modeling tasks in both supervised and unsupervised settings. We also find human knowledge can be easily injected to ShapeConv by adjusting its initialization and model performance is boosted with it. Experiments show that ShapeConv can achieve state-of-the-art performance on time-series benchmarks without sacrificing interpretability and controllability.

1 Introduction

Interpretable time-series modeling, a key task in machine learning, aims to encode sequences and make predictions in a human-understandable way. On the one hand, this fundamental task is central to numerous applications such as forecasting [25], classification [43], clustering [28] and multimodal learning [2]. On the other hand, people are keen to understand the reasoning behind the model's decisions. For example, in the healthcare domain, when the model reviews a patient's electroencephalogram (EEG) and diagnoses them with epilepsy, explaining for this judgment at the same time is crucial for both the patient and the doctor.

Early works to model time series proposed several methods to extract interpretable features from the sequence, among which *shapelets* draw much attention and are applied to several downstream tasks. The original Shapelets [38] are a set of discriminative sub-sequences selected from the input time series, and the minimum distance between a shapelet and all possible sub-sequences of the raw input is calculated in the shapelet transform step [23], providing features indicating a shapelet's

*Work done during an internship in Microsoft Research Asia.

Preprint. Under review.

presence in a sequence. Shapelet is recognized as an interpretable approach, because it can capture local discriminative patterns from the original data. However, traditional shapelets can be inefficient due to their brute-force search requirement and high time complexity, and also ineffective to handle various downstream tasks since the possible shapelets are limited. Some follow-up research [29, 21] which tries to learn the shapelets rather than extracting from the raw sequences, however, often fails to balance between the performance and the interpretability.

As the new era of deep learning comes, more and more works seek to fit the sequence with a high dimensional non-convex function using deep neural networks such as RNN [16], CNN [13], Transformer [37], etc. to model the time series. These *deep-learning-based methods* have attracted much more attention than shapelets, thanks to their great performance when the number of data is sufficient, but they are more likely to overfit when the signal-to-noise ratio is relatively large and the data are scarce. Also, the representations (often called hidden representations) are almost impossible to interpret and control due to the black-box nature of neural networks.

In this paper, we aim to seamlessly inject the interpretability of shapelets into the convolutional layer while retaining the advantages and characteristics of both. Despite the apparent disparity between shapelets and deep models in time-series modeling, we theoretically prove that **extracting features with shapelets can be equivalently conducted by passing the input time series to a specific convolutional layer (1-layer CNN) with a squared norm and pooling**. This equivalence bridges the gap between shapelets and the learnable CNN kernel.

Building on our finding, we devise ***ShapeConv***, an interpretable CNN layer with its kernel serving as shapelets, to address the time-series modeling task. First, following the proved equivalence, we add a squared norm term to the original convolutional operator, making ShapeConv layer behaviors exactly the same as shapelet transform, and convolutional kernels play the same role as shapelet. Next, to make ShapeConv's kernel resemble original time-series sub-sequences, we initialize the kernels with raw sequences and apply a Euclidean distance regularization to enforce similarity between learned kernels and sub-sequences. Furthermore, ShapeConv equips with some special designs to realize high accuracy and applicability, including the diversity regularizer to encourage different shapelets and an extension for multivariate tasks.

We also discover that using kernels as shapelets offers another advantage: the controllable injection of human prior knowledge. For instance, in the above example of EEG, we may encounter some common situations in healthcare problems, such as a lack of annotated data, or the data noise being too large to learn from. However, at this time, doctors may have some prior knowledge about what kind of waveforms are indicative of epilepsy. With ShapeConv, these knowledge can be easily injected by adjusting the initialization of the kernel. Starting with human knowledge provides a strong foundation, helps ShapeConv avoid fitting to noise, and enhances performance.

Experiments demonstrate that ShapeConv excels in both supervised classification and unsupervised clustering tasks, outperforming other learning-based shapelet methods and recent deep models for time-series classification and clustering, while maintaining interpretability. Moreover, model performance can be further boosted with human knowledge, and time complexity is significantly reduced compared to prior shapelet methods.

2 Background

Shapelets Shapelet [38], defined as a maximally discriminative sub-sequence in time-series data, is originally designed to capture inter-class features in terms of small sub-sequences rather than the full sequence. Shapelet-based methods will usually find a subset of shapelets $S^* \subset \hat{S}$ from data to maximize information gain by splitting data with the shapelets as nodes of a decision tree. $\hat{S}$ is the candidate shapelet set containing all possible sub-sequences of the original data.

Afterwards, a shapelet transform [23] is introduced to decouple the shapelet discovering step and downstream classifier. This shapelet transform step will extract features for an input signal $\mathbf{X}$ based on the distances between shapelets and the input. Specifically, for each selected shapelet $\mathbf{s} \in S^*$, we calculate the minimal distance between $\mathbf{s}$ and all sub-sequences of $\mathbf{X}$, i.e.,

$$d_{\mathbf{s},\mathbf{X}} = \min_{\mathbf{x} \in \hat{\mathbf{X}}} dist(\mathbf{s}, \mathbf{x}), \tag{1}$$

where $\hat{\mathbf{X}}$ is the set containing all sub-sequences with the same length as $\mathbf{s}$, and $dist(\mathbf{s}, \mathbf{x})$ is the distance function (usually the Euclidean distance) measuring the similarity between a shapelet and a raw sub-sequence. Using all shapelets in S^*, the feature vector with length $|S^*|$ can be obtained for each $\mathbf{X}$, and these features are used for building different kinds of classifiers besides tree-based models.

These traditional shapelet discovering methods enjoy good interpretability. However, they generally require brute-force candidate search because the candidate space is not continuous, and thus suffer from high time complexity. Following works extend the candidate set to $\mathbb{R}^{l_s}$ where l_s is the length of shapelet, and try to solve the problem with optimization-based methods, e.g. adversarial training[6], auto-encoder[21], etc. However, without more justifications for the distance between learned shapelets and the original data, these methods often suffer from the trade-off between performance and interpretability. We provide more discussion on related work in Appendix A.

Convolutional Neural Networks CNN is one of the most commonly used network structures and its several variants have been used in time-series modeling for years [20]. A CNN layer is essentially some sliding filters known as convolutional kernels followed by the activation function and pooling layer. Here, we consider a simple case to illustrate the CNN layer, where the input is 2-dimensional signal $\mathbf{X} \in \mathbb{R}^{n_{in} \times l_x}$ with n_{in} input channels and l_x time steps. Applying a 1-D convolution over it can be formulated as:

$$\mathbf{Y} = \sum_{k=1}^{n_{in}} \mathbf{W}_k * \mathbf{X}_k. \tag{2}$$

$W \in \mathbb{R}^{n_{in} \times n_{out} \times l_s}$ denotes the weights of convolutional kernels. $*$ denotes the cross-correlation operator on the k-th input channel between each raw vector of $\mathbf{W}_k$ and the input $\mathbf{X}_k$. Note that for most CNN kernels, $\mathbf{W}_k$ is learnable so this cross-correlation is equivalent to the convolution in terms of optimization. After that, a non-linear activation function and a pooling layer are often applied to extract the aggregated value among its neighbors $N(j)$ for each channel i at location j, which can be written as:

$$\mathbf{Y}_{ij}^{pool} = \operatorname*{pool}_{j' \in N(j)} \sigma(\mathbf{Y}_{ij'} + \mathbf{b}_i). \tag{3}$$

Here, $\mathbf{b}$ is the bias term and sometimes set to zero for simplicity. σ is the activation function such as ReLU, Tanh. pool denotes a pooling function such as $\max$ or mean.

3 How can CNN Kernels be the Best Shapelets?

In this section, we give a comprehensive answer to the question in the title. First, we provide a formal proof to show the equivalence between CNN and shapelets in Sec. 3.1. Then, we introduce ShapeConv, a novel convolutional layer well utilizing the equivalence in Sec. 3.2. Finally, we show how ShapeConv can be used for supervised learning and unsupervised learning in Sec. 3.3 and Sec. 3.4, respectively.

3.1 Equivalence Between CNN Kernels and Shapelets

The core idea is that when the calculation of Euclidean distance in the shapelet transform step is expanded, one of the terms is exactly the same as the forward passing in a CNN layer. Therefore, using these shapelets to extract features with Euclidean distance for $\mathbf{X}$ can be equivalently done by convolving X with n_{out} kernels from a 1-D CNN layer added by squared L_2 norm, followed by a maximum pooling. The difference between these two can be easily handled and omitted in practice. We summarize the finding in the following theorem:

Theorem 3.1 *Assume the input $\mathbf{X} \in \mathbb{R}^{l_x}$ is a 1-dimensional single-variate signal of length l_x, and n_{out} shapelets $S^* = \{\mathbf{s_1}, \mathbf{s_2}, ...\mathbf{s_{n_{out}}}\}$ with length l_s are discovered. The feature extracted from $\mathbf{X}$ with $\mathbf{s_i}$ and Euclidean distance is $d_{\mathbf{s_i},\mathbf{X}}$. Then we have*

$$d_{\mathbf{s_i},\mathbf{X}} = -2 \max_{j \in \{1,2,...,l_x - l_s + 1\}} [\mathbf{Y}_{ij} - \mathcal{N}(\mathbf{s_i}, \mathbf{X}_{j:j+l_s-1})], \tag{4}$$

where $\mathbf{Y} = \mathbf{s_i} * \mathbf{X}$ is the cross-correlation defined in Eq. (2) and $\mathcal{N}(\mathbf{s_i}, \mathbf{X}') = (\|\mathbf{s_i}\|_2^2 + \|\mathbf{X}'\|_2^2)/2$ is squared L_2 norm term.

Detailed proof can be found in the Appendix C. Eq. (4) bridges shapelets and the learnable CNN kernel. The left-hand side is the feature extracted by a specific shapelet, and the right-hand side contains the maximum pooling over the convolution between the kernel and the input time series and a squared norm term. The difference between these two is a constant factor -2 which can be absorbed in the learnable parameters.

There are two more gaps between the practical use of CNN in Eq. (3) and Theorem 3.1. One is the non-linear activation function. When the activation function σ is monotonically increasing, the order to apply maximum pooling and activation function can be swapped, i.e., $\max(\sigma(\cdot)) = \sigma(\max(\cdot))$. This fits for most cases in practice with ReLU, Tanh, Sigmoid and their variants as activation functions. Another is the bias term. Since $\mathbf{b}$ is often designed to be independent of the position, we can have $\max_j(\mathbf{Y}_{ij} + \mathbf{b}_i) = \max_j(\mathbf{Y}_{ij}) + \mathbf{b}_i$. Now we can rewrite Eq. (3) as:

$$\mathbf{Y}_{ij}^{max} = \max_{j' \in N(j)} \sigma(\mathbf{Y}_{ij'} + \mathbf{b}_i) = \sigma(\max_{j' \in N(j)} (\mathbf{Y}_{ij'}) + \mathbf{b}_i). \tag{5}$$

This suggests that we can optionally add the bias term and the non-linear activation after the features are extracted with the shapelets to obtain a complete equivalence.

3.2 ShapeConv: An Interpretable CNN Layer with Its Kernels Serving as Shapelets

Motivated by the above established equivalence, we introduce ***ShapeConv***, a novel interpretable CNN layer for time-series data, with its kernels serving as shapelets. Specifically, in addition to the cross-correlation operator and max pooling in the original CNN layer, we add a squared norm term $\mathcal{N}(\mathbf{s_i}, \mathbf{X}')$ in Eq. (4) before the pooling function to the CNN layer. Consequently, ShapeConv's forward pass mirrors the shapelet transform, calculating the minimum distance between a shapelet and all possible raw input sub-sequences, and convolutional kernels in ShapeConv play the same role as shapelets. Although initially univariate, ShapeConv can directly adapt to multivariate data tasks by adjusting the number of input channels while maintaining its other designs, enabling the model to learn multiple kernels/shapelets for each variate.

Shaping Kernels To serve as a shapelet, ShapeConv's kernel should have the similar shape as the maximally discriminative sub-sequence in original time-series data. Discriminability is achieved by optimizing kernel weights via task-specific loss. To shape kernels like original data, we first design the initialization method. Depending on whether class labels are available, we suggest different methods for initial kernel-data proximity, elaborated in the following subsections. Then, we introduce a shape regularizer to keep the kernel similar to data during training. Specifically, we calculate distances between the kernel weight, i.e., shapelet $\mathbf{s_i}$, and sub-sequences in the input, and the shape regularizer is defined as the minimal distances, i.e.,

$$\mathcal{R}_{shape} = \frac{1}{n_{out}} \sum_{i=1}^{n_{out}} \min_{\mathbf{x} \in \hat{\mathbf{X}}_i} dist(\mathbf{s_i}, \mathbf{x}), \tag{6}$$

where $\hat{\mathbf{X}}_i$ is the set containing all sub-sequences with same length as $\mathbf{s_i}$. We opt for Euclidean distance here to allow for the reuse of the distance calculated in the shapelet transform step. This approach ensures kernel weights are initialized with original data shapes and remain close during training, yielding interpretable kernels with discriminative shapes.

Increasing Diversity Since shapelets in our model are learnable weights with large flexibility, they tend to fall into local optimality where all kernels are similar but not catching diverse and discriminative features. We provide experimental results on this phenomenon in Sec. 4.1, Figure 2. Therefore, we further introduce a diversity regularizer to relieve this issue. We first use the pairwise ℓ_2 distance between different kernels (or shapelets) to construct a distance matrix $\mathbf{D}_s$, where $\mathbf{D}_s(i,j) = \exp(-\ell_2(\mathbf{s_i}, \mathbf{s_j}))$, and the diversity regularizer is defined as F-norm of $\mathbf{D}_s$, i.e.,

$$\mathcal{R}_{div} = \|\mathbf{D}_s\|_F. \tag{7}$$

To summarize, the above basic designs make ShapeConv a differentiable CNN layer with its kernels serving as shapelets. ShapeConv can keep the advantages of both CNN and shapelets. On the one hand, ShapeConv is a fully differentiable CNN layer, so it can be optimized effectively with lower time complexity than traditional methods. On the other hand, kernels of ShapeConv have good interpretability since they are similar to discriminative sub-sequences in original data.

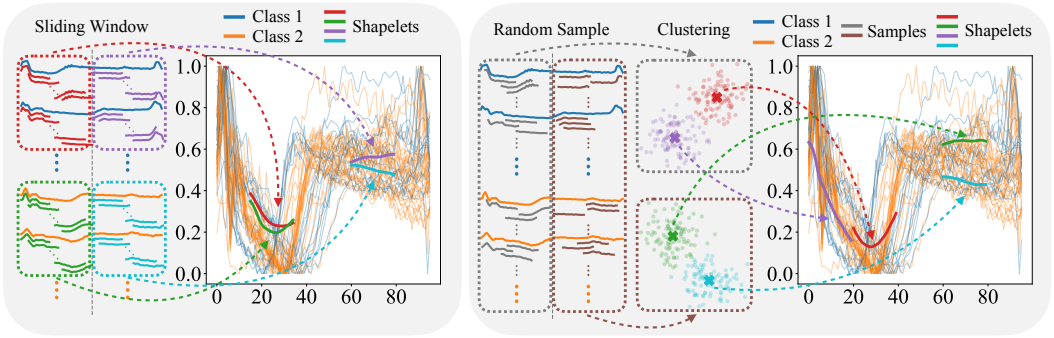

Figure 1: Illustration of initialization methods for ShapeConv kernels. $n_{out} = 4$, $n_{cls} = 2$ for both figures. Left part shows initialization in supervised learning. $k = 2$ here. Right part shows initialization in unsupervised learning. $n_{cut} = 2$ and $k = 2$ here.

3.3 Supervised Learning

In this section, we show the method to apply ShapeConv in a supervised learning task. Here, we focus on the time-series classification task, which is a typical application for shapelets.

Initialization The target of initialization is to provide convolutional kernels with a good starting point capturing class-specific sub-sequences in the data. As illustrated in the left part of Figure 1, the main idea is that for every class of data, we split them along the time axis, and calculate the mean of sub-sequences in each split part. Suppose we have n_{out} output channels, corresponding to n_{out} shapelets, with each shapelet of length l_s. The input length is l_X, and there are n_{cla} classes. First, we assign $k = n_{out}/n_{cla}$ shapelets to each class. Within each class, we divide the time series into k equal parts of length $l_k = l_X/k$ (along the time axis). For each part, we use a sliding window to find all possible candidates of length l_s. The final initialization of the kernel is obtained by calculating the mean of all candidates. This method not only incorporates class information into the kernel but also associates each kernel with a specific region of the time-series data in the dataset. As a result, the model can more effectively capture local information within the initialization region.

Classifier and Loss Function The output of the ShapeConv layer is the shapelet transformed distances, representing features extracted by learned shapelets, and these features are further utilized by the downstream classifier. Here, we append a multi-layer perceptron (MLP) after the ShapeConv layer, to map the shapelet transformed distance to the class labels. This method allows for end-to-end training of the entire model, optimizing both the ShapeConv layer and the classifier simultaneously. The loss function is designed as follows,

$$\mathcal{L} = \mathcal{L}_{cls} + \lambda_{shape}\mathcal{R}_{shape} + \lambda_{div}\mathcal{R}_{div}. \tag{8}$$

Here, $\mathcal{L}_{cls}$ is the task-specific classification loss, such as cross-entropy. $\mathcal{R}_{shape}$ is the shape regularizer defined in Eq. (6). $\mathcal{R}_{div}$ is the diversity regularizer defined in Eq. (7). λ_{shape} and λ_{div} controls the balance between each terms. Note that ShapeConv is compatible with other classifiers. We discuss this in Appendix D.1 and include this variant in our experiment in Sec. 4.1.

3.4 Unsupervised Learning

We now apply ShapeConv to the unsupervised learning task, where ShapeConv needs to capture the most representative sub-sequences in data to cluster unlabelled time series.

Initialization Initialization in the unsupervised learning setting is more challenging, since no class information is given now. Therefore, we design a pre-clustering method to find out some discriminative sub-sequences. The main idea is dividing the time-series data into portions along the time axis and performing separate clustering within each portion to find initial shapelets, as illustrated in the right part of Figure 1. Suppose we have n_{out} shapelets in the ShapeConv layer, each with a length of l_s. The input length is l_X. First, all input time series are divided into n_{cut} equal parts along the time axis, each with a length of $l_{cut} = l_X/n_{cut}$. Each part is assigned with $k = n_{out}/n_{cut}$

Table 1: Testing accuracy of supervised time-series classification tasks on UCR and UEA datasets.

	125 UCR datasets			30 UEA datasets		
Method	Avg. Acc.	Avg. Rank	Training Time (hours)	Avg. Acc.	Avg. Rank	Training Time (hours)
DTW	0.727	5.22	–	0.650	4.64	–
TNC	0.761	4.39	228.4	0.670	4.58	91.2
TST	0.641	6.14	17.1	0.617	5.30	28.6
TS-TCC	0.757	4.31	1.1	0.668	4.32	3.6
T-Loss	0.806	3.67	38.0	0.658	3.95	15.1
TS2Vec	0.836	2.34	0.9	0.704	3.12	**0.6**
ShapeConv	**0.860**	**1.81**	**0.5**	**0.750**	**2.02**	0.9

shapelets. We then sample a large number of subsequences (e.g., 10,000) of length l_s from each part and perform KMeans clustering with k centers on them. Finally, the cluster centers are utilized as the initialization of the shapelets. In this approach, the class information is implicitly introduced during the clustering process, and the division into cuts allows the shapelets to focus on different regions of the time series. This enables the model to better capture the local patterns of different classes.

Loss Function As for the task-specific loss, we employ Davies-Bouldin Index (DBI) [9] to optimize ShapeConv for better clustering results. Overall, DBI loss aims to minimize the intra-cluster distances while maximizing the inter-cluster distances, ensuring that the extracted shapelet transformed distances yield well-separated clusters [21]. The detailed formulation of DBI loss can be found in Appendix D.2. The overall loss function in unsupervised learning is designed as,

$$\mathcal{L} = \mathcal{L}_{\text{DBI}} + \lambda_{shape}\mathcal{R}_{shape} + \lambda_{div}\mathcal{R}_{div}. \tag{9}$$

Here, $\mathcal{L}_{DBI}$ is DBI loss. $\mathcal{R}_{shape}$ is the shape regularizer defined in Eq. (6). $\mathcal{R}_{div}$ is the diversity regularizer defined in Eq. (7). λ_{shape} and λ_{div} controls the balance between each terms.

Incorporating Human Knowledge The characteristic of ShapeConv makes it easy to incorporate human knowledge, which means that human experts can "tell" the model what some key subsequences look like, and the model can use these knowledge for improving its performance. On the other hand, in unsupervised learning tasks, the model will first learn to minimize the shapelet transform distance, which may lead it to converge to local minima. If the shapelet is initialized in a non-discriminative region, the model's performance may be negatively affected. Therefore, we propose using human knowledge for shapelet initialization.

Specifically, we first visualize the dataset and ask the human labeler to identify the most discriminative regions. Once these regions are labeled, we calculate the mean of each region and use it as the initialization for the shapelets. Then, the shapelet will tend to converge in the targeted region. As shown in experiments in Section 4.2, this approach makes the model learn high-quality shapelets.

4 Experiments and Analysis

4.1 Supervised Time-Series Classification

Settings We evaluate our ShapeConv model on time-series classification tasks using the UCR univariate time-series dataset [8] and UAE multivariate times series dataset [1]. Hyperparameters are tuned via grid search based on the validation set performance, and they are reported in Appendix E.2.

Compared Methods We compare ShapeConv with three kinds of baselines: (1) shapelet-based methods (IGSVM [18], FLAG [19], LTS [14], ADSN [29]), (2) common deep learning methods (MLP, CNN, ResNet [36]), and (3) state-of-the-art time-series classification models (DTW [5], TNC [35], TST [41], TS-TCC [11], T-Loss [13], TS2Vec [39]). In addition, we also design some variants of ShapeConv for ablation studies, including training without diversity loss (w/o div), training with random initialization (w/o init), and using an SVM classifier (w/ SVM).

Results The performance of ShapeConv, along with various baselines and variants, is evaluated on the 25 UCR datasets and presented in Table 2. Additionally, the summary of results for state-of-the-art time-series classification models on 125 UCR and 29 UEA datasets are shown in Table 1. The full

Table 2: Testing accuracy of supervised time-series classification tasks on 25 UCR datasets.

Dataset	MLP	CNN	ResNet	IGSVM	FLAG	LTS	ADSN	ShapeC. w/o init	ShapeC. w/ SVM	ShapeC. w/o div	ShapeC.
Adiac	75.2	85.7	82.6	23.5	75.2	51.9	79.8	70.6	82.8	87.5	**88.2**
Beef	83.3	75.0	76.7	90.0	83.3	76.7	93.3	85.2	91.3	91.2	**94.1**
Chlorine.	87.2	84.3	82.8	57.1	76.0	73.0	88.0	82.9	90.6	90.4	**92.4**
Coffee	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**
Diatom.	96.4	93.0	93.1	93.1	96.4	94.2	98.7	**100.0**	**100.0**	**100.0**	**100.0**
DPLittle	70.1	70.3	70.1	66.6	68.3	73.4	72.7	74.1	70.4	73.1	**75.9**
DPMiddle	72.1	73.6	72.3	69.5	71.3	74.1	78.4	77.9	80.1	80.3	**81.3**
DPThumb	70.5	70.1	70.5	69.6	70.5	75.2	73.6	73.3	73.8	74.9	**75.1**
ECGFiveDays	97.0	98.5	95.5	99.0	92.0	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**
FaceFour	83.0	93.2	93.2	97.7	90.9	94.3	**97.7**	94.3	92.7	94.1	96.5
GunPoint	93.3	**100.0**	99.3	**100.0**	96.7	99.6	98.7	**100.0**	**100.0**	**100.0**	**100.0**
Herring	64.1	68.1	64.1	64.1	64.1	64.1	70.3	70.8	72.6	74.1	**75.0**
ItalyPower.	96.6	97.0	96.0	93.7	94.6	95.8	97.2	**100.0**	**100.0**	**100.0**	**100.0**
Lightning7	64.4	**86.3**	83.6	63.0	76.7	79.0	80.8	80.3	77.1	78.9	81.9
MedicalImages	72.9	**79.2**	77.2	55.2	71.4	71.3	72.0	66.4	70.3	70.1	70.9
MoteStrain	86.9	95.0	89.5	88.7	88.8	90.0	90.6	90.0	90.8	91.2	**93.0**
MPLittle	70.3	75.8	72.6	70.7	69.3	74.3	75.8	74.1	73.2	76.8	**77.2**
MPMiddle	75.0	80.0	77.5	76.9	75.0	77.5	79.1	75.0	80.4	76.9	**81.3**
PPLittle	71.0	75.3	76.1	72.1	67.1	71.0	71.5	66.4	75.6	76.7	**78.2**
PPMiddle	70.7	78.4	75.3	75.9	73.8	74.9	78.6	73.5	78.1	79.6	**80.7**
PPThumb	72.6	74.5	70.8	75.5	67.4	70.5	69.5	70.5	74.1	73.1	**74.8**
Sony.	72.7	96.8	**98.5**	92.7	92.9	91.0	91.5	92.8	94.9	95.2	97.1
Symbols	85.3	96.2	87.2	84.6	87.5	94.5	96.3	95.3	96.4	97.6	**98.2**
SyntheticC.	95.0	99.0	**100.0**	87.3	99.7	97.3	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**
Trace	82.0	**100.0**	**100.0**	98.0	99.0	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**
TwoLeadECG	85.3	**100.0**	**100.0**	**100.0**	99.0	**100.0**	98.6	**100.0**	**100.0**	**100.0**	**100.0**
Avg. Acc	80.5	86.4	84.8	79.4	82.6	83.2	86.6	85.1	87.1	87.8	**88.9**
Avg. Rank	8.8	5.3	6.5	8.2	8.8	6.8	5.1	5.9	4.4	3.8	**2.4**
Best	1	6	5	3	1	4	5	8	8	8	**24**

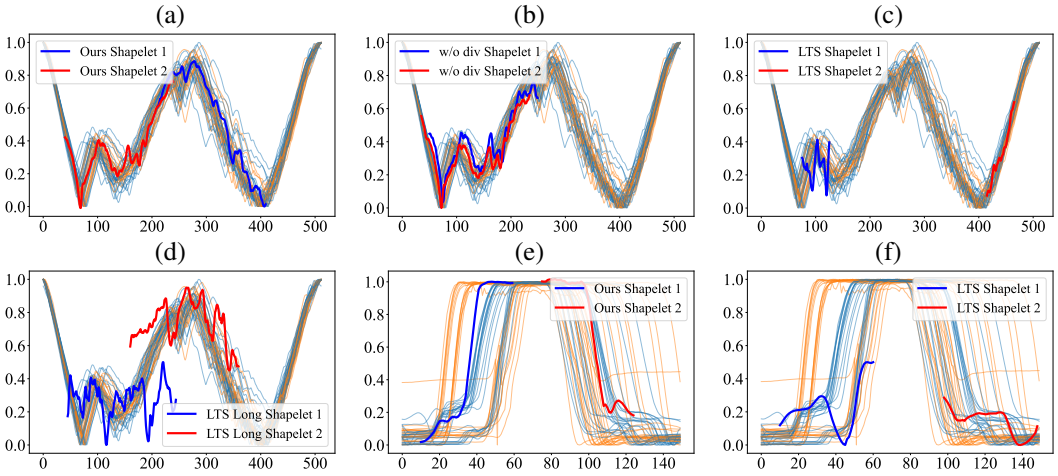

Figure 2: Learned shapelets by different methods on the Herring and GunPoint dataset. Dataset: (a-d) Herring dataset; (e-f) GunPoint dataset. Methods: (a and e) ShapeConv; (b) ShapeConv without diversity loss; (c and f) LTS; (d) LTS with shapelet length set to 200.

results are presented in the Appendix E.3 In general, ShapeConv consistently outperforms all other baselines and variants, ranking first on average. These results demonstrate that ShapeConv not only provides interpretability but also excels in performance, making it a competitive choice for time-series classification compared to state-of-the-art methods. Besides, our learning formulation for shapelets enables ShapeConv to learn better shapelets than other shapelet-based methods, leading to superior performance in time-series classification tasks. Moreover, better performance than standard CNN models shows that modifications in ShapeConv are proved to be useful. Results of ablation studies tell that the effectiveness of our initialization method and diversity loss contributes to improved performance compared to the variants. Lastly, the choice of downstream classifier, either SVM or MLP, does not significantly impact the performance of the ShapeConv model, indicating its flexibility and robustness in different classification settings.

Table 3: Testing accuracy and training time of ShapeConv and LTS with different shapelet lengths.

Length		50	100	150	200	250	300	350	400
Accuracy	ShapeConv	61.3	67.2	67.2	68.8	75.0	70.3	71.9	71.9
	LTS	64.1	60.9	59.4	59.4	59.4	59.4	59.4	59.4
Time	ShapeConv	9.20s	9.09s	9.15s	9.30s	9.27s	9.19s	9.02s	9.09s
	LTS	3.37h	3.20h	3.23h	3.17h	2.69h	2.25h	1.87h	1.21h

Table 4: Unsupervised time-series clustering results (NMI on test data) from 20 UCR datasets.

Dataset	KMeans	k-Shape	U-ShapeL	DTC	USSL	DTCR	STCN	AutoS.	ShapeC. w/o Init	ShapeC. w/o DBI	ShapeC.	ShapeC. w/ Human
Arrow	0.4816	0.5240	0.3522	0.5000	**0.6322**	0.5513	0.5240	0.5624	0.5195	0.6181	0.6231	**0.6531**
Beef	0.2925	0.3338	0.3413	0.2751	0.3338	**0.5473**	0.5432	0.3799	0.3095	0.3951	0.4132	**0.6312**
Chlorine.	0.0129	0.0000	0.0135	0.0013	0.0133	0.0195	**0.0760**	0.0133	0.0173	0.0496	0.0561	**0.0847**
Dist.ageG	0.1880	0.2911	0.2577	0.3406	0.3846	0.4553	**0.5037**	0.4400	0.4421	0.4730	0.4812	**0.5381**
ECG200	0.1403	0.3682	0.1323	0.0918	0.3776	0.3691	0.4316	0.3928	0.3015	0.5419	**0.5623**	**0.6359**
ECGFiveDays	0.0002	0.0002	0.1498	0.0022	0.6502	0.8056	0.3582	0.7835	0.6421	0.8163	**0.8312**	0.7964
GunPoint	0.0126	0.3653	0.3653	0.0194	0.4878	0.4200	**0.5537**	0.4027	0.3969	0.4426	0.4629	**0.5839**
Ham	0.0093	0.0517	0.0619	0.1016	0.3411	0.0989	0.2382	0.3211	0.1972	0.3891	**0.4185**	**0.4892**
Herring	0.0013	0.0027	0.1324	0.0143	0.1718	0.2248	0.2002	0.2019	0.1462	0.2315	**0.2642**	**0.2475**
MoteStrain	0.0551	0.2215	0.0082	0.0094	0.5310	0.4094	0.4063	0.4257	0.3153	0.5612	**0.5823**	**0.6139**
OSULeaf	0.0208	0.0126	0.0203	0.2201	0.3353	0.2599	0.3544	0.4432	0.3014	**0.5194**	**0.5194**	0.4852
Plane	0.8598	0.9642	**1.0000**	0.8678	**1.0000**	0.9296	0.9615	0.9982	**1.0000**	**1.0000**	**1.0000**	**1.0000**
Sony.	0.6112	**0.7107**	0.5803	0.2559	0.5597	0.6634	0.6112	0.6096	0.5493	0.6356	**0.6428**	0.6149
SwedishLeaf	0.0168	0.1041	0.3456	0.6187	0.9186	0.6663	0.6106	**0.9340**	0.5870	0.8643	0.8742	0.8651
Symbols	0.7780	0.6366	0.8691	0.7995	0.8821	0.8989	0.8940	0.9147	0.8173	**0.9428**	**0.9428**	0.9275
ToeSeg.1	0.0022	0.3073	0.3073	0.0188	0.3351	0.3115	0.3671	0.4610	0.3073	0.4760	**0.4893**	**0.5893**
ToeSeg.2	0.0863	0.0863	0.1519	0.0096	0.4308	0.3249	**0.5498**	0.4664	0.1305	0.5062	0.5273	**0.6734**
TwoPatterns	0.4696	0.3949	0.2979	0.0119	0.4911	0.4713	0.4110	0.5150	0.3153	0.5196	**0.5423**	**0.6194**
TwoLeadECG	0.0000	0.0000	0.0529	0.0036	0.5471	0.4614	**0.6911**	0.5654	0.4471	0.6143	0.6427	**0.7313**
WordsS.	0.5435	0.4154	0.3933	0.3498	0.4984	0.5448	0.3947	0.5112	0.4061	0.5942	**0.6143**	**0.6734**
Avg Acc	0.2291	0.2895	0.2917	0.2256	0.4961	0.4717	0.4840	0.5171	0.4074	0.5595	0.5745	0.6227
Avg Rank	9.48	8.57	8.36	9.57	4.81	4.90	4.48	4.31	7.14	2.62	**1.76**	NA
Best acc	0	1	1	0	2	1	5	1	1	3	**12**	15

Analysis In this section, we investigate two main research questions (RQs): (1) why does ShapeConv outperform other shapelet-based methods? (2) how are ShapeConv's interpretability results compared to other shapelet-based methods when they yield similar results?

We examine the Herring and GunPoint datasets from UCR [8]. Learned shapelets with minimum distance from the original data by the model with best validation accuracy are plotted in Figure 2. In the Herring dataset, ShapeConv (testing accuracy 75.0) significantly outperforms the LTS method (testing accuracy 64.1). In response to the first RQ, we observe that ShapeConv's shapelets (Figure 2 (a)) cover all turning points in the time series, where the two classes differ the most, while LTS's shapelets (Figure 2 (c)) do not cover the targeted regions. This indicates that ShapeConv's learned shapelets are better at distinguishing classes, leading to improved performance. Besides, the ablation on the diversity regularizer (Figure 2 (b)) verify its effect on learning different shapelets.

We find that the learned shapelet by ShapeConv is much longer than that by LTS. The result of forcing the shapelet learned by LTS to be longer (Figure 2 (d)) reveals that LTS fails to learn a high-quality long shapelet. We also provide an ablation on the shapelet length in Table 3. It shows ShapeConv's accuracy increases with shapelet length up to a certain point, while LTS's accuracy does not benefit from the increased length. This is likely due to optimization issues in the LTS method, which cannot handle long-length shapelets. ShapeConv, on the other hand, can be efficiently computed in parallel, leading to better optimization results and significantly faster training time (about $1000\times$ faster).

In the GunPoint dataset, both ShapeConv and LTS methods achieve saturated accuracy (100). In response to the second RQ, we utilize this dataset to compare the interpretability of the learned shapelets by ShapeConv (Figure 2 (e)) and LTS (Figure 2 (f)). It is evident that the shapelet learned by ShapeConv captures the distinguishing features of the class effectively. In contrast, the shapelets learned by LTS do not align well with either of the classes, especially for shapelet 1 in blue. Based on this observation, we conclude that ShapeConv is capable of learning more interpretable shapelets compared to the LTS method.

4.2 Unsupervised Time-Series Clustering

Settings We evaluate our ShapeConv model on time-series clustering task using 36 UCR univariate time-series datasets [8]. We first learn shapelets using a ShapeConv layer, then apply KMeans on the shapelet-transformed distance. We use the Normalized Mutual Information (NMI) metric to evaluate

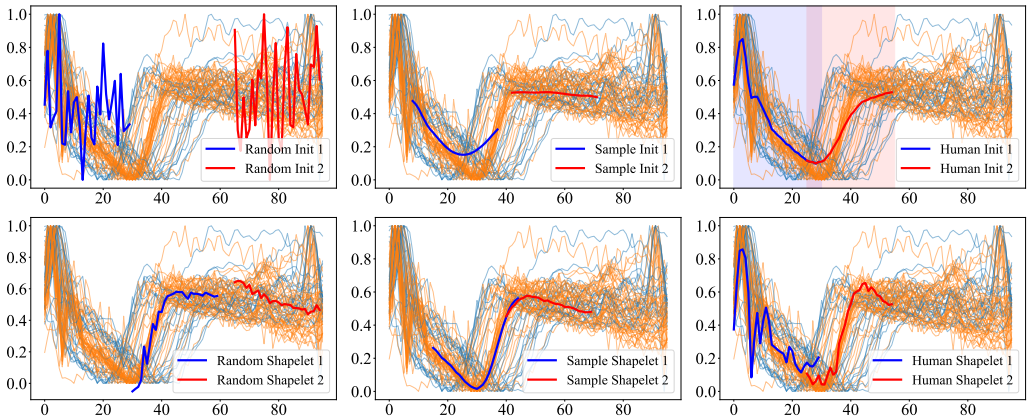

Figure 3: Visualization of cases from ECG200 dataset. First row: illustration of the shapelet initializations; second row: illustration of the learned shapelets. Left: random initialization; middle: cut and sample initialization (Sec. 3.4); right: human knowledge initialization (Sec. 3.4)

the models. Hyperparameters are tuned via grid search based on validation set performance, and they are reported in Appendix E.2.

Compared Methods We compare ShapeConv with three kinds of baselines: (1) pure clustering methods (KMeans [17] applied to the entire time series), (2) shapelet-based methods (U-Shapelet [40], AutoShape [21]), and (3) state-of-the-art time-series clustering models (k-Shape [31], DTC [30], USSL [42], DTCR [28], STCN [27]). In addition, we also design some variants of ShapeConv for ablation studies, including training with random initialization (w/o Init), training without DBI Loss (w/o DBI), and using human knowledge to initialize shapelets (w/ Human).

Results The results of all models on the 20 UCR datasets are presented in Table 4. We include a more comprehensive test on 36 UCR datasets in the Appendix E.4. In general, our ShapeConv outperforms all other baselines and variants, achieving the highest average rank among the compared methods. From these results, we observe that ShapeConv not only outperforms state-of-the-art time-series clustering methods, verifying the effectiveness of learning interpretable shapelets, but also surpasses other shapelet-based methods, suggesting that our formulation can better capture shapelets in the data.

Furthermore, ShapeConv's superior performance over the variant with random initialization method highlights the importance of proper initialization in achieving good clustering performance. Notably, ShapeConv with human knowledge initialization attains the best performance among all methods, emphasizing the crucial role of initialization and the model's ability to effectively incorporate human knowledge to guide the learning process and improve clustering results. Overall, the results demonstrate the potential of ShapeConv as a powerful and interpretable method for unsupervised time-series clustering, achieving superior performance compared to existing methods while maintaining the ability to learn interpretable shapelets and incorporate human knowledge.

Analysis of the Initialization We now provide a case study to analyze the effect of initialization for ShapeConv in unsupervised learning tasks. We select the ECG200 dataset from UCR [8] for this analysis and results are plotted in Figure 3.

First, we observe that in the time-series clustering task, *the learned shapelets are close to their initializations*. This is because, during the first step of learning shapelets, we solely minimize the shapelet-transformed distance, which tends to optimize within the local region. Therefore, determining the initialization of the shapelets is critical for unsupervised learning.

In both the random and sample initialization, one of the shapelets matches the right part of the time series, where the two classes are indistinguishable. In contrast, when using human initialization, we choose the two regions with the most significant differences between the classes (the shaded regions in Figure 3) and use the average of those regions as initialization. Consequently, the shapelets are converged in the these regions, effectively capturing the differences between the classes.

5 Conclusion

In this paper, we bridge the gap between CNNs and shapelets in time-series modeling by finding the equivalance between them. Based on the finding, we further proposed ShapeConv, an interpretable convolutional kernel with its kernels serving as shapelets, and we apply ShapeConv to both supervised and unsupervised tasks. ShapeConv is designed to maintain the advantages of both CNNs and shapelets, providing excellent performance without sacrificing interpretability and controllability. Our experiments on various benchmark datasets showed that ShapeConv outperforms other shapelet-based methods and state-of-the-art time-series classification and clustering models. Moreover, the incorporation of human knowledge can further enhance the performance of ShapeConv, highlighting its potential in real-world applications where expert knowledge is available.

References

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

[2] Tadas Baltrušaitis, Chaitanya Ahuja, and Louis-Philippe Morency. Multimodal machine learning: A survey and taxonomy. *IEEE transactions on pattern analysis and machine intelligence*, 41(2):423–443, 2018.

[3] Mustafa Gokce Baydogan, George Runger, and Eugene Tuv. A bag-of-features framework to classify time series. *IEEE transactions on pattern analysis and machine intelligence*, 35(11):2796–2802, 2013.

[4] Kai-Wei Chang, Biplab Deka, Wen-Mei W Hwu, and Dan Roth. Efficient pattern-based time series classification on gpu. In *2012 IEEE 12th International Conference on Data Mining*, pages 131–140. IEEE, 2012.

[5] Yanping Chen, Bing Hu, Eamonn Keogh, and Gustavo EAPA Batista. Dtw-d: time series semi-supervised learning from a single example. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 383–391, 2013.

[6] Ziqiang Cheng, Yang Yang, Wei Wang, Wenjie Hu, Yueting Zhuang, and Guojie Song. Time2graph: Revisiting time series modeling with dynamic shapelets. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 3617–3624, 2020.

[7] Edward Choi, Mohammad Taha Bahadori, Jimeng Sun, Joshua Kulas, Andy Schuetz, and Walter Stewart. Retain: An interpretable predictive model for healthcare using reverse time attention mechanism. *Advances in neural information processing systems*, 29, 2016.

[8] Hoang Anh Dau, Anthony Bagnall, Kaveh Kamgar, Chin-Chia Michael Yeh, Yan Zhu, Shaghayegh Gharghabi, Chotirat Ann Ratanamahatana, and Eamonn Keogh. The ucr time series archive. *IEEE/CAA Journal of Automatica Sinica*, 6(6):1293–1305, 2019.

[9] David L Davies and Donald W Bouldin. A cluster separation measure. *IEEE transactions on pattern analysis and machine intelligence*, (2):224–227, 1979.

[10] Hui Ding, Goce Trajcevski, Peter Scheuermann, Xiaoyue Wang, and Eamonn Keogh. Querying and mining of time series data: experimental comparison of representations and distance measures. *Proceedings of the VLDB Endowment*, 1(2):1542–1552, 2008.

[11] Emadeldeen Eldele, Mohamed Ragab, Zhenghua Chen, Min Wu, Chee Keong Kwoh, Xiaoli Li, and Cuntai Guan. Time-series representation learning via temporal and contextual contrasting. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pages 2352–2359, 2021.

[12] Vincent Fortuin, Matthias Hüser, Francesco Locatello, Heiko Strathmann, and Gunnar Rätsch. Som-vae: Interpretable discrete representation learning on time series. *arXiv preprint arXiv:1806.02199*, 2018.

[13] Jean-Yves Franceschi, Aymeric Dieuleveut, and Martin Jaggi. Unsupervised scalable representation learning for multivariate time series. *Advances in neural information processing systems*, 32, 2019.

[14] Josif Grabocka, Nicolas Schilling, Martin Wistuba, and Lars Schmidt-Thieme. Learning time-series shapelets. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 392–401, 2014.

[15] Antoine Guillaume, Christel Vrain, and Wael Elloumi. Random dilated shapelet transform: A new approach for time series shapelets. In *Pattern Recognition and Artificial Intelligence: Third International Conference, ICPRAI 2022, Paris, France, June 1–3, 2022, Proceedings, Part I*, page 653–664, Berlin, Heidelberg, 2022. Springer-Verlag.

[16] Tian Guo, Tao Lin, and Nino Antulov-Fantulin. Exploring interpretable lstm neural networks over multi-variable data. In *International conference on machine learning*, pages 2494–2504. PMLR, 2019.

[17] John A Hartigan and Manchek A Wong. Algorithm as 136: A k-means clustering algorithm. *Journal of the royal statistical society. series c (applied statistics)*, 28(1):100–108, 1979.

[18] Jon Hills, Jason Lines, Edgaras Baranauskas, James Mapp, and Anthony Bagnall. Classification of time series by shapelet transformation. *Data mining and knowledge discovery*, 28:851–881, 2014.

[19] Lu Hou, James Kwok, and Jacek Zurada. Efficient learning of timeseries shapelets. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30, 2016.

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

[21] Guozhong Li, Byron Choi, Jianliang Xu, Sourav S Bhowmick, Daphne Ngar-yin Mah, and Grace Lai-Hung Wong. Autoshape: An autoencoder-shapelet approach for time series clustering. *arXiv preprint arXiv:2208.04313*, 2022.

[22] Tianfu Li, Zhibin Zhao, Chuang Sun, Li Cheng, Xuefeng Chen, Ruqiang Yan, and Robert X Gao. Waveletkernelnet: An interpretable deep neural network for industrial intelligent diagnosis. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 52(4):2302–2312, 2021.

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

[24] Jason Lines, Sarah Taylor, and Anthony Bagnall. Time series classification with hive-cote: The hierarchical vote collective of transformation-based ensembles. *ACM Transactions on Knowledge Discovery from Data*, 12(5), 2018.

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

[26] Yingtao Luo, Chang Xu, Yang Liu, Weiqing Liu, Shun Zheng, and Jiang Bian. Learning differential operators for interpretable time series modeling. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 1192–1201, 2022.

[27] Qianli Ma, Sen Li, Wanqing Zhuang, Jiabing Wang, and Delu Zeng. Self-supervised time series clustering with model-based dynamics. *IEEE Transactions on Neural Networks and Learning Systems*, 32(9):3942–3955, 2020.

[28] Qianli Ma, Jiawei Zheng, Sen Li, and Gary W Cottrell. Learning representations for time series clustering. *Advances in neural information processing systems*, 32, 2019.

[29] Qianli Ma, Wanqing Zhuang, Sen Li, Desen Huang, and Garrison Cottrell. Adversarial dynamic shapelet networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 5069–5076, 2020.

[30] Naveen Sai Madiraju, Seid M Sadat, Dimitry Fisher, and Homa Karimabadi. Deep temporal clustering: Fully unsupervised learning of time-domain features. *arXiv preprint arXiv:1802.01059*, 2018.

[31] John Paparrizos and Luis Gravano. k-shape: Efficient and accurate clustering of time series. In *Proceedings of the 2015 ACM SIGMOD international conference on management of data*, pages 1855–1870, 2015.

[32] Thanawin Rakthanmanon and Eamonn Keogh. Fast shapelets: A scalable algorithm for discovering time series shapelets. In *proceedings of the 2013 SIAM International Conference on Data Mining*, pages 668–676. SIAM, 2013.

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

[34] Mit Shah, Josif Grabocka, Nicolas Schilling, Martin Wistuba, and Lars Schmidt-Thieme. Learning dtw-shapelets for time-series classification. In *Proceedings of the 3rd IKDD Conference on Data Science, 2016*, pages 1–8, 2016.

[35] Sana Tonekaboni, Danny Eytan, and Anna Goldenberg. Unsupervised representation learning for time series with temporal neighborhood coding. *arXiv preprint arXiv:2106.00750*, 2021.

[36] Zhiguang Wang, Weizhong Yan, and Tim Oates. Time series classification from scratch with deep neural networks: A strong baseline. In *2017 International joint conference on neural networks (IJCNN)*, pages 1578–1585. IEEE, 2017.

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

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

[39] Zhihan Yue, Yujing Wang, Juanyong Duan, Tianmeng Yang, Congrui Huang, Yunhai Tong, and Bixiong Xu. Ts2vec: Towards universal representation of time series. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 8980–8987, 2022.

[40] Jesin Zakaria, Abdullah Mueen, and Eamonn Keogh. Clustering time series using unsupervised-shapelets. In *2012 IEEE 12th International Conference on Data Mining*, pages 785–794. IEEE, 2012.

[41] George Zerveas, Srideepika Jayaraman, Dhaval Patel, Anuradha Bhamidipaty, and Carsten Eickhoff. A transformer-based framework for multivariate time series representation learning. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 2114–2124, 2021.

[42] Qin Zhang, Jia Wu, Peng Zhang, Guodong Long, and Chengqi Zhang. Salient subsequence learning for time series clustering. *IEEE transactions on pattern analysis and machine intelligence*, 41(9):2193–2207, 2018.

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

Appendix

A Related Work

Interpretable Time Series Modelling Research on interpretable time series modelling can be categorized into two groups: classical feature engineering methods [33] and deep learning-based models [20]. The former aims at representing time series with explainable features handcrafted from multitude perspectives, which range from designing distance metrics [10], such as Dynamic Time Warping (DTW) and Edit Distance-based measures, to ensembling statistical and spectral patterns, e.g. TSBF [3] and HIVE-COTE [24]. Despite interpretable, these methods are computationally inefficient and lack of scalability.

Recent years have also witnessed the advancement of interpretable deep learning methods for time series modeling. Edward Choi et al. [7] proposed a RNN-based two-level attention neural network named RETAIN for interpretable Electronic Health Records (EHR) prediction. Guo et al. [16] further explored the structure of LSTM and tried to learn variable-wise hidden states in an explainable way. In addition to RNN models, Fortuin et al. [12] developed a CNN-based SOM-VAE method to learn the topologically interpretable discrete representations of time series in a probabilistic fashion. Moreover, Luo et al. [26] employed convolutional kernels to approximate the partial differential equations of time series so as to explain the nonlinear dynamics of their sequential patterns. Li et al. [22] designed a wavelet convolution layer and put forward the WaveletKernelNet (WKN) to help CNNs discover filters with certain physical meaning. Different from these methods, we innovatively view convolution kernels from the shapelet perspective and provide shapelet-based interpretability on model parameters.

Shapelet Discovering Current shapelet discovery methods can also be roughly divided into two categories: the search-based approaches and the learning-based ones.

Principle behind the searching-based methods is to search the raw dataset for effective shapelets. Ye et al. [38] initially proposed the shapelet-based method for time series classification, where they utilized cross-entropy as quality measure and integrated the searching process into decision tree construction. To speed up searching, Chang et al. [4] implemented a parallel shapelet searching algorithm via GPU and Rakthanmanon et al. [32] took advantage of the Symbolic Aggregation Approximation (SAX) technique to convert continuous time series into discrete low-dimensional representations. Unlike those methods where shapelet searching is integrated with classification, Lines et al. [23] proposed the Shapelet Transform to separate the two processes, where the F-statistic is employed as quality measure to search for top-k shapelets and all time series are transformed into Euclidean distance vectors for downstream classification. Besides, Guillaume et al. [15] proposed Random Dilated Shapelet Transform and Zakaria et al. [40] intended to search unsupervised shapelets for time series clustering. Compared with the search-based methods, our method is much more efficient and able to extend to the multivariate scenarios.

In terms of learning-based methods, the core idea is to capture shapelets automatically through model optimization. Grabocka et al. [14] were the first to propose learning time series shapelets with gradient descent. Shah et al. [34] extended this approach by replacing the Euclidean distance with DTW measure to learn DTW-shapelets. However, these two methods are hard to optimize and unstable during training when the shapelet size is large. In contrast, Ma et al. [29] explored the use of adversarial training to learn sample-specific shapelets. But this approach is susceptible to instability and mode collapse due to adversarial learning. More recently, Cheng et al. [6] attempted to obtain time series representations by learning time-aware shapelets with graph embedding techniques, and Li et al. [21] introduced AUTOSHAPE, which utilizes an autoencoder to learn high-quality shapelets in an unsupervised manner. Nevertheless, both of these methods involve a shapelet candidate generation process before shapelet learning, thus being indirect and less efficient than our method, which can learn shapelets directly via optimization.

B Limitations

Despite its promising performance and interpretability, ShapeConv has some limitations that should be acknowledged. One of the main limitations is its sensitivity to initialization. The performance of

ShapeConv is highly dependent on the initialization of the convolutional kernels. As the model tends to optimize within the local region, selecting a poor initialization may lead to suboptimal shapelets that fail to capture the important patterns in the data. Although we have proposed initialization methods that mitigate this issue, further research is needed to develop more robust initialization strategies that can consistently yield high-quality shapelets.

Another limitation of ShapeConv is that although it can function as a single convolutional layer, we have not yet explored the use of it incorporated into advanced deep models, such as transformers, multi-layer CNNs, RNNs, etc. These complex designs are necessary for tackling problems with intricate temporal patterns and dependencies. Future work could explore how to integrate ShapeConv into other advanced deep models, while maintaining its key advantage of providing interpretable insights.

C Proof of Theorem 3.1

Theorem 3.1 *Assume the input $\mathbf{X} \in \mathbb{R}^{l_x}$ is a 1-dimensional single-variate signal of length l_x, and n_{out} shapelets $S^* = \{\mathbf{s_1}, \mathbf{s_2}, ...\mathbf{s_{n_{out}}}\}$ with length l_s are discovered. The feature extracted from $\mathbf{X}$ with $\mathbf{s_i}$ and Euclidean distance is $d_{\mathbf{s_i},\mathbf{X}}$. Then we have*

$$d_{\mathbf{s_i},\mathbf{X}} = -2 \max_{j \in \{1,2,\ldots,l_x - l_s + 1\}} [\mathbf{Y}_{ij} - \mathcal{N}(\mathbf{s_i}, \mathbf{X}_{j:j+l_s-1})], \tag{10}$$

where $\mathbf{Y} = \mathbf{s_i} * \mathbf{X}$ is the cross-correlation defined in Eq. (2) and $\mathcal{N}(\mathbf{s_i}, \mathbf{X}') = (\|\mathbf{s_i}\|_2^2 + \|\mathbf{X}'\|_2^2)/2$ is squared L_2 norm term.

Proof. According to Eq.(1), shapelet transform step extract features using minimal distance and can be expanded as:

$$\begin{aligned}
d_{\mathbf{s_i},\mathbf{X}} &= \min_{\mathbf{x} \in \hat{\mathbf{X}}} dist(\mathbf{s_i}, \mathbf{x}) = \min_{j \in \{1,2,\cdots,l_x - l_s + 1\}} \|\mathbf{s_i} - \mathbf{X}_{j:j+l_s-1}\|_2^2 \\
&= \min_{j \in \{1,2,\cdots,l_x - l_s + 1\}} (\|\mathbf{s_i}\|_2^2 + \|\mathbf{X}_{j:j+l_s-1}\|_2^2 - 2 \sum_{k=1}^{l_s} \mathbf{s_i}_k \mathbf{X}_{j+k}) \\
&= \min_{j \in \{1,2,\cdots,l_x - l_s + 1\}} [\|\mathbf{s_i}\|_2^2 + \|\mathbf{X}_{j:j+l_s-1}\|_2^2 - 2(\mathbf{s_i} * \mathbf{X})_k], \\
&= -2 \max_{j \in \{1,2,\cdots,l_x - l_s + 1\}} [\mathbf{Y}_{ij} - \mathcal{N}(\mathbf{s_i}, \mathbf{X}_{j:j+l_s-1})]
\end{aligned}$$

with $\mathbf{Y}$ and $\mathcal{N}(\mathbf{s_i}, \mathbf{X}_{j:j+l_s-1})$ defined as in the theorem. □

D Details on Model Designs

D.1 Compatible with Other Classifiers

ShapeConv can also be used together with traditional classifiers, such as support vector machines (SVMs), decision trees, or random forests. In this case, the whole model cannot be optimized via an end-to-end fashion, so we decompose the shapelet learning step and classification. Specifically, no additional module is appended to the ShapeConv layer, and the output of the layer is directly optimized using the above loss function, but with $\mathcal{L}_{cls}$ term in Eq. (8) removed. Then, the learned features are fed into the chosen classifier for training and prediction. We also include this variant in our experiment in Sec. 4.1.

D.2 Davies-Bouldin Index (DBI) Loss

When the number of cluster is set to k, DBI can be denoted as

$$\mathcal{I}_{\text{DBI}} = \frac{1}{k} \sum_{i=1}^{k} \max_{j=1\ldots k, j \neq i} \frac{r_i + r_j}{d_{ij}}. \tag{11}$$

Here, r_i is the diameter of cluster i, which is defined as the average distance between each element in cluster i and the center of cluster i. $d_{i,j}$ is the distance between the center of cluster i and cluster j.

However, this formulation is not tractable for optimization due to the $\max$ operator. Thus, following [21], the $\max$ operator is replaced and approximated by the following calculation,

$$\mathcal{L}_{\mathrm{DBI}} = \frac{1}{k}\sum_{i=1}^{k}\frac{\sum_{j=1}^{k} m_{ij}\cdot e^{\alpha m_{ij}}}{\sum_{j=1}^{k} e^{\alpha m_{ij}}}, \tag{12}$$

where $m_{ij} = \frac{r_i+r_j}{d_{ij}}$. By numerical verification, $\alpha = 100$ is enough for Eq. (12) to approximate the true maximum.

E Details on Experiments and Analysis

E.1 Environment

All experiments are performed on the PyTorch platform using a 24-cores AMD Epyc 7V13 2.5GHz CPU, 220GB RAM, and an NVIDIA A100 80GB PCIe GPU. The server is provided by the Azure cloud computing platform.

E.2 Hyperparameters

Supervised Learning Hyperparameters are tuned via grid search based on validation set performance. The number of shapelets is chosen from $\{1, 2, 3, 4, 5\}$ times the number of classes, and the shapelet length is evaluated over $\{0.1, 0.2, \cdots, 0.8\}$ times the time series length. λ_{shape} is chosen from $\{0.01, 0.1, 1, 10\}$. λ_{div} is evaluated over $\{0.01, 0.1, 1, 10\}$. Learning rate is chosen from $\{0.001, 0.005, 0.01, 0.05, 0.1\}$.

Unsupervised Learning Hyperparameters are tuned via grid search based on validation set performance. The number of shapelets is chosen from $\{1, 2, 3, 4, 5\}$ times the number of classes, and the shapelet length is evaluated over $\{0.1, 0.15, 0.2, 0.25, \cdots, 0.8\}$ times the time series length. λ_{shape} is chosen from $\{0.01, 0.1, 1, 10\}$. λ_{div} is evaluated over $\{0.01, 0.1, 1, 10\}$. Learning rate is chosen from $\{0.001, 0.005, 0.01, 0.05, 0.1\}$.

E.3 Results of Supervised Learning Tasks

In this section, we present the full results of supervised time-series classification tasks on 125 UCR datasets (Table 5) and 30 UEA datasets (Table 6).

Table 5: Testing accuracy of supervised time-series classification tasks on 125 UCR datasets. Mean accuracy ± std over 3 independent experiments with different random seeds is reported.

	DTW	TNC	TST	TS-TCC	T-Loss	TS2Vec	ShapeConv
Adiac	0.604	0.726	0.550	0.767	0.675	0.775	**0.882**±0.014
ArrowHead	0.703	0.703	0.771	0.737	0.766	0.857	**0.915**±0.011
Beef	0.633	0.733	0.500	0.600	0.667	0.767	**0.941**±0.006
BeetleFly	0.700	0.850	**1.000**	0.800	0.800	0.900	**1.000**±0.000
BirdChicken	0.750	0.750	0.650	0.650	0.850	0.800	**1.000**±0.000
Car	0.733	0.683	0.550	0.583	0.833	0.883	**0.992**±0.016
CBF	0.997	0.983	0.898	0.998	0.983	**1.000**	**1.000**±0.000
ChlorineConcentration	0.648	0.760	0.562	0.753	0.749	0.832	**0.924**±0.013
CinCECGTorso	0.651	0.669	0.508	0.671	0.713	**0.827**	0.785±0.010
Coffee	**1.000**	**1.000**	0.821	**1.000**	**1.000**	**1.000**	**1.000**±0.000
Computers	0.700	0.684	0.696	**0.704**	0.664	0.660	0.656±0.017
CricketX	0.754	0.623	0.385	0.731	0.713	0.805	**0.914**±0.005
CricketY	0.744	0.597	0.467	0.718	0.728	**0.769**	0.729±0.011
CricketZ	0.754	0.682	0.403	0.713	0.708	**0.792**	0.764±0.027
DiatomSizeReduction	0.967	0.993	0.961	0.977	0.984	0.987	**1.000**±0.000
DistalPhalanxOutlineCorrect	0.717	0.754	0.728	0.754	0.775	0.775	**0.804**±0.027
DistalPhalanxOutlineAgeGroup	0.770	0.741	0.741	0.755	0.727	0.727	**0.798**±0.015
DistalPhalanxTW	0.590	0.669	0.568	0.676	0.676	0.698	**0.781**±0.012
Earthquakes	0.719	0.748	0.748	0.748	0.748	0.748	**0.784**±0.025
ECG200	0.770	0.830	0.830	0.880	0.940	0.920	**1.000**±0.000
ECG5000	0.924	0.937	0.928	0.941	0.933	0.935	**0.963**±0.008
ECGFiveDays	0.768	0.999	0.763	0.878	**1.000**	**1.000**	**1.000**±0.000
ElectricDevices	0.602	0.700	0.676	0.686	0.707	0.721	**0.743**±0.025
FaceAll	0.808	0.766	0.504	0.813	0.786	0.805	**0.853**±0.008

FaceFour	0.830	0.659	0.511	0.773	0.920	0.932	**0.965**±0.013
FacesUCR	0.905	0.789	0.543	0.863	0.884	0.930	**0.957**±0.002
FiftyWords	0.690	0.653	0.525	0.653	0.732	**0.774**	0.699±0.011
Fish	0.823	0.817	0.720	0.817	0.891	**0.937**	0.920±0.024
FordA	0.555	0.902	0.568	0.930	0.928	0.948	**0.983**±0.011
FordB	0.620	0.733	0.507	0.815	0.793	0.807	**0.841**±0.038
GunPoint	0.907	0.967	0.827	0.993	0.980	0.987	**1.000**±0.000
Ham	0.467	**0.752**	0.524	0.743	0.724	0.724	0.733±0.002
HandOutlines	0.881	0.930	0.735	0.724	0.922	0.930	**0.954**±0.022
Haptics	0.377	0.474	0.357	0.396	0.490	0.536	**0.580**±0.006
Herring	0.531	0.594	0.594	0.594	0.594	0.641	**0.750**±0.026
InlineSkate	0.384	0.378	0.287	0.347	0.371	0.415	**0.432**±0.009
InsectWingbeatSound	0.355	0.549	0.266	0.415	0.597	**0.630**	0.597±0.012
ItalyPowerDemand	0.950	0.928	0.845	0.955	0.954	0.961	**1.000**±0.000
LargeKitchenAppliances	0.795	0.776	0.595	0.848	0.789	0.875	**0.924**±0.004
Lightning2	**0.869**	**0.869**	0.705	0.836	**0.869**	**0.869**	0.819±0.016
Lightning7	0.726	0.767	0.411	0.685	0.795	**0.863**	0.808±0.016
Mallat	0.934	0.871	0.713	0.922	0.951	0.915	**0.984**±0.026
Meat	0.933	0.917	0.900	0.883	0.950	**0.967**	0.950±0.007
MedicalImages	0.737	0.754	0.632	0.747	0.750	**0.793**	0.709±0.004
MiddlePhalanxOutlineCorrect	0.698	0.818	0.753	0.818	0.825	0.838	**0.856**±0.029
MiddlePhalanxOutlineAgeGroup	0.500	0.643	0.617	0.630	0.656	0.636	**0.669**±0.003
MiddlePhalanxTW	0.506	0.571	0.506	0.610	0.591	0.591	**0.642**±0.024
MoteStrain	0.835	0.825	0.768	0.843	0.851	0.863	**0.930**±0.002
NonInvasiveFetalECGThorax1	0.790	0.898	0.471	0.898	0.878	0.930	**0.951**±0.012
NonInvasiveFetalECGThorax2	0.865	0.912	0.832	0.913	0.919	0.940	**0.942**±0.005
OliveOil	0.833	0.833	0.800	0.800	0.867	**0.900**	0.833±0.004
OSULeaf	0.591	0.723	0.545	0.723	0.760	0.876	**0.914**±0.007
PhalangesOutlinesCorrect	0.728	0.787	0.773	0.804	0.784	**0.823**	0.797±0.010
Phoneme	0.228	0.180	0.139	0.242	0.276	**0.312**	0.139±0.000
Plane	**1.000**	**1.000**	0.933	**1.000**	0.990	**1.000**	**1.000**±0.000
ProximalPhalanxOutlineCorrect	0.784	0.866	0.770	0.873	0.859	0.900	**0.913**±0.017
ProximalPhalanxOutlineAgeGroup	0.805	0.854	0.854	0.839	0.844	0.844	**0.883**±0.049
ProximalPhalanxTW	0.761	0.810	0.780	0.800	0.771	0.824	**0.847**±0.016
RefrigerationDevices	0.464	0.565	0.483	0.563	0.515	0.589	**0.613**±0.005
ScreenType	0.397	**0.509**	0.419	0.419	0.416	0.411	0.493±0.007
ShapeletSim	0.650	0.589	0.489	0.683	0.672	**1.000**	**1.000**±0.000
ShapesAll	0.768	0.788	0.733	0.773	0.848	**0.905**	0.848±0.002
SmallKitchenAppliances	0.643	0.725	0.592	0.691	0.677	0.733	**0.803**±0.001
SonyAIBORobotSurface1	0.725	0.804	0.724	0.899	0.902	0.903	**0.971**±0.005
SonyAIBORobotSurface2	0.831	0.834	0.745	0.907	0.889	0.890	**0.921**±0.004
StarLightCurves	0.907	0.968	0.949	0.967	0.964	0.971	**0.987**±0.017
Strawberry	0.941	0.951	0.916	**0.965**	0.954	**0.965**	0.919±0.014
SwedishLeaf	0.792	0.880	0.738	0.923	0.914	0.942	**0.961**±0.023
Symbols	0.950	0.885	0.786	0.916	0.963	0.976	**0.982**±0.040
SyntheticControl	0.993	**1.000**	0.490	0.990	0.987	0.997	**1.000**±0.000
ToeSegmentation1	0.772	0.864	0.807	0.930	0.939	0.947	**0.961**±0.016
ToeSegmentation2	0.838	0.831	0.615	0.877	0.900	0.915	**0.938**±0.008
Trace	**1.000**	**1.000**	**1.000**	**1.000**	0.990	**1.000**	**1.000**±0.000
TwoLeadECG	0.905	0.993	0.871	0.976	0.999	0.987	**1.000**±0.000
TwoPatterns	**1.000**	**1.000**	0.466	0.999	0.999	**1.000**	**1.000**±0.000
UWaveGestureLibraryX	0.728	0.781	0.569	0.733	0.785	0.810	**0.832**±0.035
UWaveGestureLibraryY	0.634	0.697	0.348	0.641	0.710	0.729	**0.754**±0.013
UWaveGestureLibraryZ	0.658	0.721	0.655	0.690	0.757	0.770	**0.805**±0.008
UWaveGestureLibraryAll	0.892	0.903	0.475	0.692	0.896	0.934	**0.953**±0.006
Wafer	0.980	0.994	0.991	0.994	0.992	**0.998**	0.984±0.012
Wine	0.574	0.759	0.500	0.778	0.815	0.889	**0.902**±0.001
WordSynonyms	0.649	0.630	0.422	0.531	0.691	0.704	**0.784**±0.018
Worms	0.584	0.623	0.455	0.753	0.727	0.701	**0.803**±0.003
WormsTwoClass	0.623	0.727	0.584	0.753	0.792	0.805	**0.815**±0.026
Yoga	0.837	0.812	0.830	0.791	0.837	**0.887**	0.772±0.004
ACSF1	0.640	0.730	0.760	0.730	0.900	0.910	**0.923**±0.027
AllGestureWiimoteX	0.716	0.703	0.259	0.697	0.763	0.777	**0.831**±0.009
AllGestureWiimoteY	0.729	0.699	0.423	0.741	0.726	0.793	**0.846**±0.016
AllGestureWiimoteZ	0.643	0.646	0.447	0.689	0.723	0.770	**0.848**±0.008
BME	0.900	0.973	0.760	0.933	0.993	0.993	**1.000**±0.000
Chinatown	0.957	0.977	0.936	**0.983**	0.951	0.968	**0.983**±0.010
Crop	0.665	0.738	0.710	0.742	0.722	**0.756**	0.722±0.034
EOGHorizontalSignal	0.503	0.442	0.373	0.401	0.605	0.544	**0.615**±0.008
EOGVerticalSignal	0.448	0.392	0.298	0.376	0.434	0.503	**0.537**±0.003
EthanolLevel	0.276	0.424	0.260	0.486	0.382	0.484	**0.708**±0.023
FreezerRegularTrain	0.899	0.991	0.922	0.989	0.956	0.986	**0.997**±0.006
FreezerSmallTrain	0.753	0.982	0.920	0.979	0.933	0.894	**0.999**±0.006
Fungi	0.839	0.527	0.366	0.753	**1.000**	0.962	0.983±0.000
GestureMidAirD1	0.569	0.431	0.208	0.369	0.608	**0.631**	0.573±0.001
GestureMidAirD2	**0.608**	0.362	0.138	0.254	0.546	0.515	0.585±0.003
GestureMidAirD3	0.323	0.292	0.154	0.177	0.285	0.346	**0.412**±0.001
GesturePebbleZ1	0.791	0.378	0.500	0.395	0.919	**0.930**	0.884±0.003
GesturePebbleZ2	0.671	0.316	0.380	0.430	**0.899**	0.873	0.874±0.014
GunPointAgeSpan	0.918	0.984	0.991	0.994	0.994	0.994	**1.000**±0.000
GunPointMaleVersusFemale	0.997	0.994	**1.000**	0.997	0.997	**1.000**	**1.000**±0.000

GunPointOldVersusYoung	0.838	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**±0.000
HouseTwenty	0.924	0.782	0.815	0.790	0.933	0.941	**0.984**±0.013
InsectEPGRegularTrain	0.872	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**±0.000
InsectEPGSmallTrain	0.735	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**±0.000
MelbournePedestrian	0.791	0.942	0.741	0.949	0.944	**0.959**	0.941±0.005
MixedShapesRegularTrain	0.842	0.911	0.879	0.855	0.905	0.922	**0.965**±0.004
MixedShapesSmallTrain	0.780	0.813	0.828	0.735	0.860	0.881	**0.938**±0.013
PickupGestureWiimoteZ	0.660	0.620	0.240	0.600	0.740	0.820	**0.871**±0.004
PigAirwayPressure	0.106	0.413	0.120	0.380	0.510	**0.683**	0.538±0.002
PigArtPressure	0.245	0.808	0.774	0.524	0.928	**0.966**	0.872±0.005
PigCVP	0.154	0.649	0.596	0.615	0.788	**0.870**	0.851±0.012
PLAID	0.840	0.495	0.419	0.445	0.555	0.561	**0.915**±0.004
PowerCons	0.878	0.933	0.911	0.961	0.900	**0.972**	0.911±0.011
Rock	0.600	0.580	0.680	0.600	0.580	**0.700**	**0.700**±0.010
SemgHandGenderCh2	0.802	0.882	0.725	0.837	0.890	0.963	**0.983**±0.007
SemgHandMovementCh2	0.584	0.593	0.420	0.613	0.789	0.893	**0.926**±0.006
SemgHandSubjectCh2	0.727	0.771	0.484	0.753	0.853	0.951	**0.981**±0.005
ShakeGestureWiimoteZ	0.860	0.820	0.760	0.860	0.920	**0.940**	0.860±0.018
SmoothSubspace	0.827	0.913	0.827	0.953	0.960	0.993	**1.000**±0.000
UMD	0.993	0.993	0.910	0.986	0.993	**1.000**	**1.000**±0.000
DodgerLoopDay	0.500		0.200			0.562	**0.631**±0.021
DodgerLoopGame	0.877		0.696			0.841	**0.915**±0.005
DodgerLoopWeekend	0.949		0.732			0.964	**0.971**±0.017
Average Accuracy	0.728	0.761	0.639	0.757	0.806	0.836	0.860
Average Rank	5.217	4.393	6.140	4.306	3.667	2.337	1.810

Table 6: Testing accuracy of supervised multivariate time-series classification tasks on 30 UEA datasets. Mean accuracy ± std over 3 independent experiments with different random seeds is reported.

	DTW	TNC	TST	TS-TCC	T-Loss	TS2Vec	ShapeConv
ArticularyWordRecognition	0.987	0.973	0.977	0.953	0.943	0.987	**1.000**±0.000
AtrialFibrillation	0.200	0.133	0.067	0.267	0.133	0.200	**0.530**±0.008
BasicMotions	0.975	0.975	0.975	**1.000**	**1.000**	0.975	**1.000**±0.000
CharacterTrajectories	0.989	0.967	0.975	0.985	0.993	**0.995**	0.983±0.002
Cricket	**1.000**	0.958	**1.000**	0.917	0.972	0.972	**1.000**±0.000
DuckDuckGeese	0.600	0.460	0.620	0.380	0.650	**0.680**	0.652±0.010
EigenWorms	0.618	0.840	0.748	0.779	0.840	**0.847**	0.796±0.001
Epilepsy	0.964	0.957	0.949	0.957	0.971	0.964	**0.980**±0.020
ERing	0.133	0.852	0.874	**0.904**	0.133	0.874	0.787±0.000
EthanolConcentration	**0.323**	0.297	0.262	0.285	0.205	0.308	0.251±0.009
FaceDetection	0.529	0.536	0.534	0.544	0.513	0.501	**0.648**±0.015
FingerMovements	0.530	0.470	0.560	0.460	0.580	0.480	**0.590**±0.009
HandMovementDirection	0.231	0.324	0.243	0.243	0.351	0.338	**0.419**±0.006
Handwriting	0.286	0.249	0.225	0.498	0.451	0.515	**0.530**±0.023
Heartbeat	0.717	0.746	0.746	0.751	0.741	0.683	**0.798**±0.001
JapaneseVowels	0.949	0.978	0.978	0.930	0.989	0.984	**1.000**±0.000
Libras	0.870	0.817	0.656	0.822	0.883	0.867	**0.893**±0.014
LSST	0.551	0.595	0.408	0.474	0.509	0.537	**0.614**±0.006
MotorImagery	0.500	0.500	0.500	0.610	0.580	0.510	**0.680**±0.008
NATOPS	0.883	0.911	0.850	0.822	0.917	0.928	**0.941**±0.009
PEMS-SF	0.711	0.699	0.740	0.734	0.676	0.682	**0.803**±0.009
PenDigits	0.977	0.979	0.560	0.974	0.981	**0.989**	0.973±0.025
PhonemeSpectra	0.151	0.207	0.085	**0.252**	0.222	0.233	0.198±0.001
RacketSports	0.803	0.776	0.809	0.816	0.855	0.855	**0.871**±0.019
SelfRegulationSCP1	0.775	0.799	0.754	0.823	0.843	0.812	**0.869**±0.013
SelfRegulationSCP2	0.539	0.550	0.550	0.533	0.539	0.578	**0.636**±0.001
SpokenArabicDigits	0.963	0.934	0.923	0.970	0.905	**0.988**	0.963±0.009
StandWalkJump	0.200	0.400	0.267	0.333	0.333	0.467	**0.639**±0.019
UWaveGestureLibrary	0.903	0.759	0.575	0.753	0.875	0.906	**0.948**±0.011
InsectWingbeat		0.469	0.105	0.264	0.156	0.466	**0.518**±0.026
Average Accuracy	0.650	0.670	0.617	0.668	0.658	0.704	0.750
Average Rank	4.638	4.583	5.300	4.317	3.950	3.117	2.017

E.4 Results of Unsupervised Learning Tasks

In this section, we report the full results of the unsupervised time-series clustering tasks on 36 UCR datasets in Table 5.

Table 7: Unsupervised time-series clustering results (NMI on test data) from 36 UCR datasets. Mean accuracy ± std over 3 independent experiments with different random seeds is reported.

Dataset	KMeans	k-Shape	U-ShapeL	DTC	USSL	DTCR	STCN	AutoS.	ShapeC. w/o Init	ShapeC. w/o DBI	ShapeC.	ShapeC. w/ Human
Arrow	0.4816	0.5240	0.3522	0.5000	**0.6322**	0.5513	0.5240	0.5624	0.5195	0.6181	0.6231±0.019	**0.6531**±0.003
Beef	0.2925	0.3338	0.3413	0.2751	0.3338	**0.5473**	0.5432	0.3799	0.3095	0.3951	0.4132±0.000	**0.6312**±0.015
BeetleFly	0.0073	0.3456	0.5105	0.3456	0.5310	0.7610	**1.0000**	0.5310	0.5039	**1.0000**	**1.0000**±0.000	**1.0000**±0.000
BirdChicken	0.0371	0.3456	0.2783	0.0073	0.6190	0.5310	**1.0000**	0.6352	0.5934	**1.0000**	**1.0000**±0.000	**1.0000**±0.000
Car	0.2540	0.3771	0.3655	0.1892	0.4650	0.5021	**0.5701**	0.4970	0.3752	0.4871	0.4915±0.003	0.5415±0.035
Chlorine.	0.0129	0.0000	0.0135	0.0013	0.0133	0.0195	**0.0760**	0.0133	0.0173	0.0496	0.0561±0.006	**0.0847**±0.002
Coffee	0.5246	**1.0000**	**1.0000**	0.5523	**1.0000**	0.6277	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**±0.000	**1.0000**±0.000
Diatom.	0.9300	**1.0000**	0.4849	0.6863	**1.0000**	0.9418	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**±0.000	**1.0000**±0.000
Dist.ageG	0.1880	0.2911	0.2577	0.3406	0.3846	0.4553	**0.5037**	0.4400	0.4421	0.4730	0.4812±0.032	**0.5381**±0.012
Dist.correct	0.0278	0.0527	0.0063	0.0115	0.1026	0.1180	**0.2327**	0.1333	0.0843	0.1052	0.1251±0.020	0.2174±0.038
ECG200	0.1403	0.3682	0.1323	0.0918	0.3776	0.3691	0.4316	0.3928	0.3015	0.5419	**0.5623**±0.013	**0.6359**±0.008
ECGFiveDays	0.0002	0.0002	0.1498	0.0022	0.6502	0.8056	0.3582	0.7835	0.6421	0.8163	**0.8312**±0.030	0.7964±0.003
GunPoint	0.0126	0.3653	0.3653	0.0194	0.4878	0.4200	**0.5537**	0.4027	0.3969	0.4426	0.4629±0.017	**0.5839**±0.027
Ham	0.0093	0.0517	0.0619	0.1016	0.3411	0.0989	0.2382	0.3211	0.1972	0.3891	**0.4185**±0.015	**0.4892**±0.019
Herring	0.0013	0.0027	0.1324	0.0143	0.1718	0.2248	0.2002	0.2019	0.1462	0.2315	**0.2642**±0.004	**0.2475**±0.019
Lighting2	0.0038	0.2670	0.0144	0.1435	0.3727	0.2289	0.3479	0.3530	0.3195	0.3951	**0.4291**±0.021	**0.4864**±0.010
Meat	0.2510	0.2254	0.2716	0.2250	0.9085	**0.9653**	0.9393	0.9437	0.3089	0.9537	0.9537±0.017	**1.0000**±0.000
Mid.ageG	0.0219	0.0722	0.1491	0.1390	0.2780	0.4661	**0.5109**	0.3940	0.1843	0.4452	0.4653±0.015	**0.6742**±0.016
Mid.correct	0.0024	0.0349	0.0253	0.0079	0.2503	0.1150	0.0921	**0.2873**	0.1968	0.2213	0.2316±0.007	**0.3521**±0.004
Mid.TW	0.4134	0.5229	0.4065	0.1156	0.9202	0.5503	0.6169	**0.9450**	0.8014	0.9267	0.9267±0.003	0.9381±0.004
MoteStrain	0.0551	0.2215	0.0082	0.0094	0.5310	0.4094	0.4063	0.4257	0.3153	0.5612	**0.5823**±0.001	**0.6139**±0.019
OSULeaf	0.0208	0.0126	0.0203	0.2201	0.3353	0.2599	0.3544	0.4432	0.3014	**0.5194**	**0.5194**±0.010	0.4852±0.008
Plane	0.8598	0.9642	**1.0000**	0.8678	**1.0000**	0.9296	0.9615	0.9982	**1.0000**	**1.0000**	**1.0000**±0.000	**1.0000**±0.000
Prox.ageG	0.0635	0.0110	0.0332	0.4153	0.6813	0.5581	0.6317	**0.6930**	0.5264	0.6398	0.6549±0.022	**0.7424**±0.011
Prox.TW	0.0082	0.1577	0.0107	0.6199	**1.0000**	0.6539	0.7330	0.8947	0.6064	0.8412	0.8523±0.009	0.8164±0.018
Sony.	0.6112	**0.7107**	0.5803	0.2559	0.5597	0.6634	0.6112	0.6096	0.5493	0.6356	**0.6428**±0.026	0.6149±0.012
Sony.II	0.5444	0.0110	0.5903	0.4257	0.6858	0.6121	0.5647	**0.7020**	0.5071	0.6874	0.6874±0.002	**0.7591**±0.014
SwedishLeaf	0.0168	0.1041	0.3456	0.6187	0.9186	0.6663	0.6106	**0.9340**	0.5870	0.8643	0.8742±0.009	0.8651±0.003
Symbols	0.7780	0.6366	0.8691	0.7995	0.8821	0.8989	0.8940	0.9147	0.8173	**0.9428**	**0.9428**±0.013	0.9275±0.001
ToeSeg.1	0.0022	0.3073	0.3073	0.0188	0.3351	0.3115	0.3671	0.4610	0.3073	0.4760	**0.4893**±0.030	**0.5893**±0.027
ToeSeg.2	0.0863	0.0863	0.1519	0.0096	0.4308	0.3249	**0.5498**	0.4664	0.1305	0.5062	0.5273±0.006	**0.6734**±0.002
TwoPatterns	0.4696	0.3949	0.2979	0.0119	0.4911	0.4713	0.4110	0.5150	0.3153	0.5196	**0.5423**±0.011	**0.6194**±0.005
TwoLeadECG	0.0000	0.0000	0.0529	0.0036	0.5471	0.4614	**0.6911**	0.5654	0.4471	0.6143	0.6427±0.005	**0.7313**±0.024
Wafer	0.0010	0.0010	0.0010	0.0008	0.0492	0.0228	**0.2089**	0.0520	0.0010	0.0824	0.0854±0.002	0.0623±0.009
Wine	0.0031	0.0119	0.0171	0.0000	**0.7511**	0.2580	0.5927	0.6045	0.2854	0.6198	0.6462±0.033	0.6834±0.041
WordsS.	0.5435	0.4154	0.3933	0.3498	0.4984	0.5448	0.3947	0.5112	0.4061	0.5942	**0.6143**±0.029	**0.6734**±0.002
Avg Acc	0.2132	0.2841	0.2777	0.2332	0.5427	0.4818	0.5478	0.5558	0.4290	0.5999	**0.6122**	**0.6591**
Avg Rank	9.6250	8.4306	8.5139	9.6806	4.5000	5.1389	4.1806	3.8194	6.9167	2.9722	**2.2222**	NA
Best acc	0	3	2	0	6	2	13	7	3	7	**14**	**25**