TapNet: Multivariate Time Series Classification with Attentional Prototypical Network

Xuchao Zhang[*,1] **Yifeng Gao**[*,2] **Jessica Lin,**[2] **Chang-Tien Lu**[1]
[1]Discovery Analytics Center, Virginia Tech, Falls Church, VA
[2]Department of Computer Science, George Mason University, Fairfax, VA
[1]{xuczhang, ctlu}@vt.edu, [2]{ygao12,jessica}@gmu.edu

Abstract

With the advance of sensors technologies, Multivariate Time Series classification (MTSC) problem, perhaps one of the most essential problem in time series data mining domain, has continuously received a significant amount of attention in recent decades. Traditional time series approaches based on Bag-of-Patterns or Time Series Shapelet have difficulty dealing with the huge amounts of feature candidates generated in high-dimensional multivariate data, but have promising performance in small training sets. By contrast, deep learning based methods can learn low-dimensional features efficiently but still suffer from the shortage of labeled data. In this paper, we propose a novel MTSC model with attentional prototype network to take the strengths of both traditional and deep learning based approaches. Specifically, we design a random group permutation method combined with multi-layer convolutional networks to learn the low-dimensional features from multivariate time series data. To handle the issue of limited training labels, we propose a novel attentional prototype network to train the feature representation based on their distance to class prototypes with inadequate data labels. Besides, we extend our model into its semi-supervised setting by utilizing the unlabeled data. Extensive experiments on 30 datasets in public UEA Multivariate time series archive with eight state-of-the-art baseline methods exhibit the effectiveness of the proposed model.

1 Introduction

Time series is a set of real value observations sequentially ordered by time. A multivariate time series is a set of co-evolving time series which is typically recorded by a set of sensors simultaneously over time. With the advance of sensors technologies, Multivariate Time Series Classification (MTSC) problem, identifying the labels for multivariate time series records, has received a great amount of attention in recent decades. Since time series data is a popular data type that exists in a wide range of research domain and applications, multivariate time series classification models have been used in many different real-world applications such as Human Activities Recognition (Minnen et al. 2006), EEG data analysis (Bagnall et al. 2018) and Motion Recognition (Rakthanmanon and Keogh 2013).

Most existing general time series classification approaches such as bag-of-patterns (Senin and Malinchik 2013) or time series shapelet (Ye and Keogh 2009) require a parsing step to convert time series into an extensive set of subsequences or patterns as feature candidates. The large feature space generated makes the feature selection step difficult, and may result in low accuracy in multivariate case (Schäfer and Leser 2017b). Recently, deep learning based methods (Karim et al. 2018b)(Zheng et al. 2014) achieve promising performance in time series classification task. These approaches can perfectly handle the issue concerning huge feature space by learning a low-dimensional feature representation via convolutional or recurrent networks directly from raw time series data. Moreover, neural network solutions require less domain knowledge in time series data than traditional methods. But these approaches require a large number of labeled data to train the massive model parameters. Different from computer vision and natural language processing domains, the availability of labeled data is limited in most of the time series datasets (Bagnall et al. 2018)(Bagnall et al. 2017). For instance, the "MotorImagery" dataset (Lal et al. 2005) provided by the University of Tubingen for brain activity detection contains only 378 labeled data samples, which is arduous to train a deep learning model with thousands of model parameters (Neyshabur, Tomioka, and Srebro 2014). However, the issue of limited training labels can be handled by traditional time series classification approach using distance-based methods such as DTW-1NN (Shokoohi-Yekta, Wang, and Keogh 2015). Table 1 presents the characteristics of existing MTS classification methods. We can see that traditional time series approaches can work with limited training samples, but they usually generate a large feature space and require domain knowledge in time series data. In contrast to traditional approaches, the deep learning methods can learn low-dimensional feature representations without domain knowledge, but these approaches suffer from the limitation of training labels.

To combine the advantages of traditional and deep learning MTSC approaches, this paper proposes a novel multivariate Time series classification model with Attentional Prototype Network, named TapNet. TapNet is capable of extracting low-dimensional features from the multivariate time series with little domain knowledge and handling the short-

[*]These two authors contributed equally.

Table 1: Characteristics of Existing Models

Method	Few Domain Knowledge	Small Feature Space	Short of Labels	Unlabeled Data
DTW			√	√
Shapelet			√	
Bag-of-Patterns			√	
Traditional DL	√	√		
TapNet (Ours)	√	√	√	√

age of labeled data. To learn the latent features from multivariate time series efficiently, we design a random group permutation method to reconstruct the dimensions of time series into groups, preceded by convolutional layers. To handle the issue of limited labeled data, we propose a novel attentional prototype network to train a low-dimensional feature representation for each time series based on their distances to the class prototype learned by small labeled samples. Since the training process is distance-based, it requires much fewer labeled samples than traditional deep neural network. To summarize, our work has the following main contributions:

- Propose an attentional prototype network to handle the limited training samples. A distance-based loss function between the class prototype and the time series samples with the same class label is utilized to train a deep neural network with limited labels.
- Learn a low-dimensional feature representation (embedding) for multivariate time series. The learned embeddings have an attractive characteristic that the distance between the samples with the same label is much smaller than the samples with different labels.
- Extend our model to semi-supervised settings to utilize the unlabeled data. The embeddings of unlabeled data are combined with labeled data to improve the estimation of class prototype embeddings when the training data is scarce.
- Conduct extensive experiments on the latest multivariate time series classification archive comprising of 30 datasets with a wide range of applications. Our method achieves the highest rank score compared with eight state-of-the-art methods.

The rest of the paper is organized as follows. Section 2 discusses related work in multivariate time series classification, and Section 3 introduces the problem definition and notations. Section 4 introduces the proposed model. The experimental results on 30 UAE Archive datasets are presented in Section 5, and the paper concludes with a summary of the research in Section 6.

2 Related Work

In this section, we briefly describe recent advances in time series classification research. We start our discussion with recent progress in univariate time series classification, and then we shift our focus to multivariate time series classification. Finally, we briefly describe recent work on semi-supervised time series classification.

2.1 Univariate Time Series Classification

Most state-of-the-art time series classification work extends upon one of the two major approaches: time series shapelet (Ye and Keogh 2009) and bag-of-patterns model (Senin and Malinchik 2013). For time series shapelet-based approaches (Rakthanmanon and Keogh 2013)(Wang et al. 2016)(Ye and Keogh 2009)(Fang, Wang, and Wang 2018), the models identify subsequences that are the most discriminating of the two classes. These subsequences can be used to transform the original, not linearly separable time series into a lower-dimensional space that is easier to classify. Bag-of-patterns models (Senin and Malinchik 2013)(Schäfer and Leser 2017a)(Li and Lin 2017) extract features from time series via a different mechanism — inspired by the bag-of-words models used for text data, they convert time series into a bag of discrete words, and use the histogram of words to represent the data. Recently, an increasing number of deep learning techniques have been developed for univariate time series classification. These techniques often use LSTM or CNN to extract the features (Karim et al. 2018a)(Zhao et al. 2017).

2.2 Multivariate Time Series Classification

Similar to univariate time series classification, most work on multivariate time series classification (Baydogan and Runger 2015)(Schäfer and Leser 2017b)(Wistuba, Grabocka, and Schmidt-Thieme 2015)](Baydogan and Runger 2016) follows one of the two directions using time series shapelet or bag-of-patterns models. Wistuba et al. introduced an approach named Ultra Fast Shapelets (UFS) (Wistuba, Grabocka, and Schmidt-Thieme 2015), which randomly selects shapelets in order to avoid the costly process of finding discriminative subsequences. A similar idea is also used in the work by Karlsson et al. (Karlsson, Papapetrou, and Boström 2016). They introduced an approach named Generalized Random Shapelet Forests (gRSF) which generates shapelet-based decision trees via randomly selected shapelets. It has been shown that the model is competitive compared with the interval feature based approaches such as LPS (Baydogan and Runger 2016) and ARKernel (Cuturi and Doucet 2011).

Following the idea of bag-of-patterns model, Baydogan et al. (Baydogan and Runger 2015) introduced an approach named Symbolic Representation for Multivariate Time series (SMTS). The algorithm attempts to capture the local relationships between different dimensions by generating a codebook directly from multivariate time series using a greedy approach. Recently, Schafer et al. (Schäfer and Leser 2017b) extends the Bag of SFA (Symbolic Fourier Approximation) symbol model into multivariate case by using the approach introduced in their previous work (Schäfer and Leser 2017a).

Recently, deep learning based methods (Karim et al. 2018b)(Zheng et al. 2014) achieve promising performance in multivariate time series classification task. These models often use LSTM layer and stacked CNN layer to extract features from time series, and a softmax layer is then applied to predict the label. Zheng et al. introduced

a model named Multi Channel Deep Convolutional Neural Network(MCDCNN)(Zheng et al. 2014), for which each univariate time series is passed through a separate CNN layer. The outputs of all univariate time series are concatenated and pass through a softmax layer to predict the label. In contrast, Karim et al. (Karim et al. 2018b) recently proposed a model consisting of an LSTM layer and stacked CNN layer along with Squeeze-and-Excitation block to generate latent features. Different from the conventional approaches, deep learning based methods learn the latent features by training convolutional or recurrent networks with large-scale labeled data. However, existing work does not address the problem with (labeled) data shortage.

2.3 Semi-supervised Time Series Classification

It has been shown that in many applications, collecting labeled data is often very difficult. As a result, for time series classification, semi-supervised approaches have gained much popularity in recent years. However, most existing work on semi-supervised time series classification focuses on univariate time series. The general approach is to use Dynamic Time Warping (DTW) (Wei and Keogh 2006)(Chen et al. 2013) to estimate labels for unlabeled time series. It has been shown that it can greatly improve the performance over the original DTW classifier. Marussy et al. (Marussy and Buza 2013) introduced an semi-supervised approach, SUCCESS, which consists of constrained hierarchical clustering and dynamic time warping. Begum et al. (Begum et al. 2014) introduced a minimum description length (MDL) based stopping criterion for semi-supervised learning. Most of these work are designed for univariate time series classification.

3 Problem Formulation

In this section, we begin by formulating the multivariate time series classification (MTSC) problem. Then a semi-supervised classification problem for multivariate time series (SMTSC) is formally defined. The notations used in this paper are summarized in Table 2.

3.1 Multivariate Time Series Classification (MTSC)

A multivariate time series (MTS) $X = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m\} \in \mathbb{R}^{m \times l}$ is an ordered sequence of $m \in \mathbb{N}$ streams with $\boldsymbol{x}_i = \big(x_{i,1}, \ldots, x_{i,l}\big)$, where l is the length of time series and m is the number of multivariate dimensions. For instance, when a dust sensor collects 100 sequential particle density records in three dimensions (PM1, PM2.5 and PM10), the multivariate time series X can be represented as a matrix with the dimension $m = 3$ and time series length $l = 100$. Each multivariate time series is associated with a class label $y \in \Omega$ from a predefined label set Ω. Given a group of multivariate time series $\mathcal{X} = \{X_1, \ldots, X_n\} \in \mathbb{R}^{n \times m \times l}$, where n is the number of time series, and the corresponding labels $\boldsymbol{y} = \{y_1, \ldots, y_n\} \in \mathbb{R}^n$ for each time series, the MTSC task is to train a classifier $f_X \mapsto y$ to predict a class label for a multivariate time series whose label is unknown.

Table 2: Math Notations

Notations	Explanations
$n, \tilde{n}$	size of labeled and unlabeled data samples
l, m	time series length and dimension size
$X \in \mathbb{R}^{m \times l}$	multivariate features for one time series
$\mathcal{X} \in \mathbb{R}^{n \times m \times l}$	collection of time series data samples
$\boldsymbol{y} \in \mathbb{R}^n$	collection of time series labels
$\tilde{\mathcal{X}} \in \mathbb{R}^{\tilde{n} \times m \times l}$	collection of time series data samples without labels
d	embedding dimension
$\boldsymbol{h}_i \in \mathbb{R}^d$	low-dimensional embedding of i^{th} data sample
$\boldsymbol{c}_k \in \mathbb{R}^d$	class prototype of the k^{th} class
S_k	indices of data samples with label k

3.2 Semi-supervised MTSC (SMTSC)

For semi-supervised MTSC problem, we assume the labeled samples are not sufficient to train a model for multivariate time series classification problem. So we intend to utilize unlabeled data during the training process to help improve the overall classification performance. The training set for semi-supervised setting is denoted as a tuple of labeled and unlabeled examples: $\big((\mathcal{X}, \boldsymbol{y}),\ \widetilde{\mathcal{X}}\big)$. The labeled portion is made up of the same input as MTSC problem, containing tuples of time series features $\mathcal{X}$ and labels $\boldsymbol{y}$. The unlabeled portion includes a set of time series $\widetilde{\mathcal{X}} = \big\{\widetilde{X}_1, \ldots, \widetilde{X}_{\tilde{n}}\big\}$ containing only time series features without labels, where $\widetilde{X}_i$ represents the input features of the i^{th} unlabeled time series and $\tilde{n}$ is the size of unlabeled samples. The SMTSC task is to train a classifier $\tilde{f}_X \mapsto y$ to predict a class label for a multivariate time series with not only the labeled training data samples $(\mathcal{X}, \boldsymbol{y})$ but unlabelled data samples $\widetilde{\mathcal{X}}$.

4 Model

In this section, we first introduce the overall architecture of new proposed TapNet model in Section 4.1. Then we explain the components of our model in Section 4.2-4.4. Lastly, we extend our model to semi-supervised setting (Semi-TapNet) in Section 4.5. We are releasing the source code of our implementation freely to the research community and it can be accessed at https://github.com/kdd2019-tapnet/tapnet.

4.1 Model Architecture Overview

Figure 1 shows the overall architecture of our proposed model, TapNet, comprising of three main components: random dimension permutation, multivariate time series encoding, and attentional prototype learning.

The input of our model is a set of multivariate time series with multiple dimensions. An example of 6-dimensional time series is shown in Figure 1. For each dimension, the time series share the same time series length. To model the interactive features between multivariate dimensions, we propose a random group permutation (RDP) method to randomly combine the dimensions into different groups with fixed group size. Take the multivariate time series in Figure 1 as an example, we divide the six dimensions of the time series into three groups with different dimension permutations. The detailed description of random dimension permutation method can be found in Section 4.2.

After the dimension permutation, a low-dimensional time series embedding is learned in the time series encoding component. Specifically, we apply both the LSTM and 1-Dimensional convolutional layers to model the sequential information of time series and the relationships between time series dimensions. After low-dimensional embeddings are learned, we use the embeddings of training samples as the input to learn the prototype for each class. Here, the class prototype is a feature representation (embedding) of each class, which contains the same embedding size as the time series. Specifically, the class prototype is a weighted combination of the training samples in the same class, where the weights of the training samples are trained by an attention layer. The intuition behind is to learn a class prototype for each class, which has smaller distances to the data samples in the same class, but larger distances to data samples in different classes. The details of time series encoding and attentional prototype learning can be found in Section 4.3 and 4.4, respectively.

Also, we extend the supervised TapNet model into its semi-supervised settings (Semi-TapNet). The Semi-TapNet model utilizes the feature information of unlabeled data (usually from the test set), which makes a notable improvement when the training labels are not sufficient. The details of semi-supervised attentional prototype learning can be found in Section 4.5.

4.2 Random Dimension Permutation

We propose a novel method, Random Dimension Permutation (RDP), to reorganize the time series dimensions into different groups based on random permutation orders, which helps to model the interactive features between multivariate dimensions.

Suppose the time series have m dimensions and they are divided into g groups, the size φ of each group can be represented as $\varphi = \lfloor \frac{m \cdot \alpha}{g} \rfloor$, where α is the scale factor parameter to control the ratio of the total dimensions in the new permutation over the original size and $\lfloor \cdot \rfloor$ represents the largest integer less than or equal to the given input. As the example shown in Figure 2, we have a time series with 6 dimensions ($m = 6$), and they are divided into $g = 3$ groups with scale factor $\alpha = 1.5$. Then the group size $\varphi = \lfloor \frac{6 \cdot 1.5}{3} \rfloor = 3$. We define σ_m as one random permutation of a set of numbers $\{1, \ldots, m\}$. To divide all the dimensions into g groups, we need to run the random permutation g times. For each permutation, we retrieve the first φ dimensions as the group candidates. To distinguish the different random permutations, we denote $\sigma_m^{(i)}$ as random permutation for the i^{th} group based on m dimensions. Then the candidates $\mathcal{G}_i$ of the i^{th} group can be defined as follows:

$$\mathcal{G}_i = \left\{ \sigma_m^{(i)}(1), \ldots, \sigma_m^{(i)}(\varphi) \right\}, \quad i = 1 \ldots g \tag{1}$$

The example in Figure 2 shows the time series with six dimensions are divided into three groups $\left\{(4, 5, 2), (1, 2, 6), (3, 6, 4)\right\}$. Three of six dimensions appears in two different groups, which is controlled by the scale factor parameter.

4.3 Multivariate Time Series Encoding

The time series encoding component is to learn a low-dimensional embedding $\boldsymbol{x} \in \mathbb{R}^d$ for each time series by a neural network based function $f_\Theta(X)$, where Θ is the set of function parameters and $X \in \mathbb{R}^{m \times l}$ is the time series data. To extract the sequential information as well as the multivariate features of the MTS data, we apply both Long short-term memory network (LSTM) (Sundermeyer, Schlüter, and Ney 2012) and a multi-layer convolutional network (Krizhevsky, Sutskever, and Hinton 2012). For the LSTM part, operating over the raw time series data $X \in \mathbb{R}^{m \times l}$ we obtain the contextual embedding $X_{\text{lstm}} \in \mathbb{R}^{m \times d_l}$, where d_l is the hidden dimension of LSTM with the default value 128. Then we apply a global average pooling operation on time series dimensions and get the output $X_{\text{lstm+pool}} \in \mathbb{R}^{1 \times d_l}$. For the convolutional network part, the input is the grouped permutations $\mathcal{G} = \{\mathcal{G}_1, \ldots, \mathcal{G}_g\}$. We apply three one-dimensional convolutional layers on each group $X_{\mathcal{G}_i} \in \mathbb{R}^{\varphi \times l}$. After each convolutional layer, we operate both Batch Normalization (Ioffe and Szegedy 2015) and Leaky Rectified Linear Units (Leaky ReLU) (Xu et al. 2015). Noted that we use separate parameters for the first convolutional layer but share the parameters for the 2nd and 3rd layers, which can help to learn separate features for each group but reduce the size of parameters in the last two layers. We set the default value of filters for the three convolutional layers as 256, 256 and 128 and set kernels as 8, 5 and 3. For time series with extremely long length, we also apply dilated convolution operation (Yu and Koltun 2015) to support exponential expansion of the receptive field without loss of long-length time series information. The output of the three convolutional layers for the i^{th} group is $X_{\text{conv}}^{(i)} \in \mathbb{R}^{d_f \times d_c}$, where d_f is the filter size of the last convolutional layer and d_c is the output dimension of convolutional operations. For each group, a global average pooling is applied on the convolutional dimensions d_c and generate the output $X_{\text{conv+pool}}^{(i)} \in \mathbb{R}^{d_f \times 1}$. Then we concatenate the results of all the groups of convolutional network and LSTM together and get the output $X_{\text{combo}} \in \mathbb{R}^{(d_f \times g + d_l) \times 1}$. Last we operate two fully connected layers on X_{combo} to generate the low-dimensional feature representation (embedding) of the multivariate time series $\boldsymbol{x} \in \mathbb{R}^d$, where d is the dimension of the embedding. The details of the parameter settings can be found in Appendix.

4.4 Attentional Prototype Learning

Since the training samples in time series data can be severely limited, traditional deep learning network may have an inductive bias (Neyshabur, Tomioka, and Srebro 2014). To address the issue, we propose a novel attentional prototype learning method by training the distance-based loss function. Specifically, in our approach, we learn a class prototype embedding (Snell, Swersky, and Zemel 2017) for each class and classify the input time series based on their distance to the prototype of each class. Figure 3 shows an example of the class prototypes in a 3-class classification scenario. The embeddings of class prototypes are marked by star shape (★) and the time series from the same class are represented

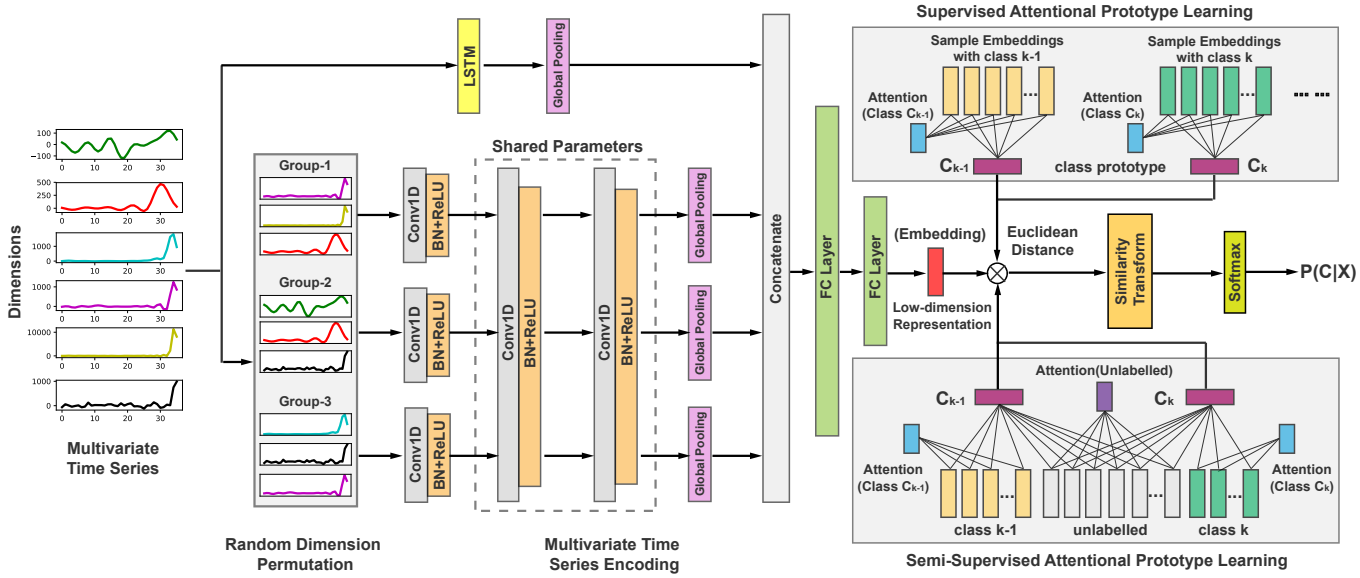

Figure 1: Overall Architecture of the TapNet Model

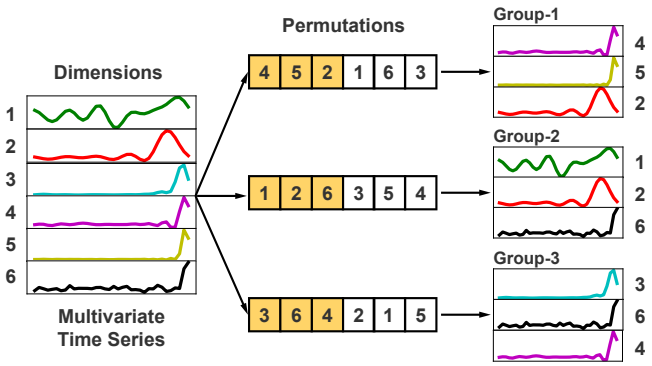

Figure 2: Example of random dimension permutation

by circle shape (◯) in the same color. The weights between class prototype and time series measures how much impact the time series have on the class prototype. The test samples will be labeled by the nearest class prototype. Let $H_k = [\boldsymbol{h}_1, \ldots, \boldsymbol{h}_{|S_k|}] \in \mathbb{R}^{|S_k| \times d}$ be a matrix of time series embeddings belonging to the class k, where S_k represents the set of indices for data samples with class label k. Then the prototype embedding of class k can be presented by a weighted sum of individual sample embeddings as follows:

$$\boldsymbol{c}_k = \sum_i A_{k,i} \cdot H_{k,i} \tag{2}$$

where $A_{k,i}$ is the weight of i^{th} data sample in class k and $H_{k,i}$ represents the embedding of the data sample.

Here the sample weight $A_{k,i}$ is not a predefined parameter but a trainable value according to the embeddings of time series. In particular, we regard the time series samples from the same class as a bag of independent instances. For each instance, an attention-based multi-instance pooling (Ilse, Tomczak, and Welling 2018) method is applied to learn its instance weight for the class prototype. The attention weights for the k^{th} class can be computed by the follows:

$$A_k = \text{softmax}\left(\boldsymbol{w}_k^T \tanh\left(V_k H_k^T\right)\right), \tag{3}$$

where $\boldsymbol{w}_k \in \mathbb{R}^{u \times 1}$ and $V_k \in \mathbb{R}^{u \times d}$ are trainable parameters for the attention model and u is the size of hidden dimension for both trainable parameters. Noted that we use separate parameters $\boldsymbol{w}_k$ and V_k for each class due to the assumption that the different classes may have distinct attentions on their feature spaces.

After we have the embedding vectors of class prototypes, the distribution over classes for a given time series $\boldsymbol{x} \in \mathbb{R}^d$ can be represented as a softmax over distances to the prototypes in the embedding space as follows:

$$p_\Theta(y = k|\boldsymbol{x}) = \frac{\exp\left(-D\left(f_\Theta(\boldsymbol{x}), \boldsymbol{c}_k\right)\right)}{\sum_i \exp\left(-D\left(f_\Theta(\boldsymbol{x}), \boldsymbol{c}_i\right)\right)}, \tag{4}$$

where the function $D : \mathbb{R}^d \times \mathbb{R}^d \mapsto [0, +\infty)$ is the distance function to measure the distances between two embedding vectors. The distance function can be chosen from *regular Bregman divergences* (Banerjee et al. 2005). Examples of Bregman divergences include squared euclidean distance and Mahalanobis distance (De Maesschalck, Jouan-Rimbaud, and Massart 2000). Here we applied the squared euclidean distance function $D(\boldsymbol{z}, \boldsymbol{z}') = \|\boldsymbol{z} - \boldsymbol{z}'\|^2$ to measure the distance between the time series embeddings. Noted that the probabilities over classes are based on the similarity between class prototype and time series, therefore, we multiply -1 in front of the distance function. Then the training of our model can be proceeded by minimizing the negative log probability $J(\Theta) = -\log p_\Theta(y = k|\boldsymbol{x})$ of the true class via Adam algorithm (Kingma and Ba 2014).

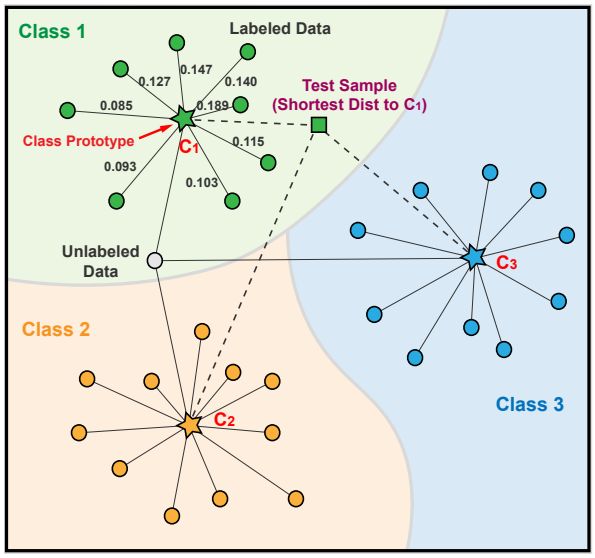

Figure 3: Example of Class Prototype

4.5 Semi-supervised TapNet

We now extend our supervised TapNet approach into its semi-supervised setting (Semi-TapNet) by utilizing the unlabeled data in the training phase. The unlabeled data can help to improve the estimation of the class prototype when the training data is scarce.

Let $\tilde{H} = \left[\tilde{\boldsymbol{h}}_1 \ldots, \tilde{\boldsymbol{h}}_{|\tilde{S}|}\right] \in \mathbb{R}^{|\tilde{S}| \times d}$ be a matrix of time series embeddings of unlabeled data, where $\tilde{S}$ is the set of indices for unlabeled data samples and $\tilde{\boldsymbol{h}}_i \in \mathbb{R}^{1 \times d}$ is the embedding vector for the i^{th} data sample in the unlabeled set. Then the prototype embedding of class k can be presented by a weighted sum of labeled and unlabeled data as follows.

$$\boldsymbol{c}_k = \frac{\sum_i A_{k,i} H_{k,i} + \sum_i \tilde{A}_{k,i} \tilde{H}_{k,i}}{\sum_i A_{k,i} + \sum_i \tilde{A}_{k,i}} \tag{5}$$

Similar to the supervised version, we also learn instance weight for the unlabeled data samples. Here $\tilde{A}_{k,i}$ is the weight of the i^{th} unlabeled data sample for class k and $\tilde{H}_{k,i}$ is the corresponding embedding of the data sample. The attention vector $\tilde{A}_k$ of class k for unlabeled data can be represented as follows:

$$\tilde{A}_k = \text{softmax}\left(\tilde{\boldsymbol{w}}_k^T \tanh\left(\tilde{V}_k \tilde{H}_k^T\right)\right) \tag{6}$$

where $\tilde{\boldsymbol{w}}_k \in \mathbb{R}^{u \times 1}$ and $\tilde{V}_k \in \mathbb{R}^{u \times d}$ are trainable parameters for the attention model. Then we apply the class prototype updated by unlabeled data into probability distribution over class in Equation (4) to train the semi-supervised model.

5 Experiments

In this section, the performance of the proposed model TapNet is evaluated using 30 datasets in the UEA multivariate time series classification archive. We begin by introducing the evaluation setting, with details on the datasets, metrics and baselines we use in our experiments. Then the performance of the proposed model in terms of supervised and semi-supervised classification accuracy is evaluated against several existing methods. Finally, the analyses on class prototype and random dimension permutation are elaborated. All the experiments are conducted on a single Tesla P100 GPU with 16GB memory.

5.1 Experimental Settings

Datasets In this experiment, we evaluate the proposed method on the latest multivariate time series classification archive (Bagnall et al. 2018). The archive is released in 2018 and consists of 30 datasets with a wide range of applications, dimensions and time series lengths. For each dataset, the time series in each dimension have equal length, and the train/test split is provided. According to the information in the released paper (Bagnall et al. 2018), the archive consists of 9 different Human Activity Recognition, 4 Motion classification, 3 ECG classification, 6 EEG/MEG classification, 5 Audio Spectra Classification, and three other type of data. The characteristics of each dataset can be found in the Appendix.

The dimension of multivariate time series ranges from 2 dimensions in trajectory classification data (Libras) to 1,131 dimensions in the audio spectra classification data (DuckDuckGeese). The length of the time series ranges from 8 (pen digit recognition dataset) to 17,901 (worm type classification data, EigenWorms). The datasets also have a large size range, from 27 (12 for training and 15 for testing in StandWalkJump) to 50,000 (30,000 for training and 20,000 for testing in InsectWingbeat). The number of classes ranges from 2 to 39.

We evaluate our proposed method using all 30 datasets. Since the datasets have diverse characteristics, we can evaluate different aspects of our proposed method, as well as showing statistical comparison between our proposed method and existing approaches.

Metrics For each dataset, we compute the classification accuracy as the evaluation metric. We also compute the average rank and the number of Win/Ties to compare different methods. In addition, we test the overall performance of the proposed approach via a statistical hypothesis test. The Critical Difference Diagram (Demšar 2006) is used to show the overall performance of TapNet.

Comparison Methods We compare our proposed approach with 8 different benchmark approaches, including the latest bag-of-patterns model based multivariate time series classification approach(Schäfer and Leser 2017b), deep learning framework and common distance-based classifiers(Karim et al. 2018b). Note that it has been shown in previous work(Schäfer and Leser 2017b)(Karim et al. 2018b) that these approaches are equivalent or better than many other feature-based approaches such as SMTS(Baydogan and Runger 2015), gRSF(Baydogan and Runger 2016), and LPS(Karlsson, Papapetrou, and Boström 2016). The details of the benchmarks we use are provided as follows:

- WEASEL-MUSE(Schäfer and Leser 2017b): This is the latest bag-of-patterns (BOP) based framework for multivariate time series classification. WEASEL-MUSE first generates multivariate symbols given the time series and uses a Chi-square test based feature selection pipeline to extract features. The generated features are then fed into a classifier to identify the label. We use the source code provided by the authors [1]. The approach is run under the recommended setting provided by the authors (Schäfer and Leser 2017b).
- MLSTM-FCN(Karim et al. 2018b): This is the latest general deep-learning framework for multivariate time series classification. The model consists of a LSTM layer and stacked CNN layer along with Squeeze-and-Excitation block to generate latent features. Then a MLP with softmax activation function is used to predict the label. We use the source code provided by the authors [2]. The approach is run using the default parameter setting (Karim et al. 2018b).
- 1NN-ED with (without) normalization: The one nerest neighbor classifier with Euclidean distance is the most popular baseline used for time series classification. We directly report the accuracy results provided in the archive (Bagnall et al. 2018).
- 1NN-DTW-i, dimension-independent dynamic time warping with (without) normalization: This one nearest neighbor classifier computes distances based on the sum of DTW distance for every dimension. Similar to 1NN-ED benchmark, we compare with the algorithm with (without) normalization. We directly report the performance results provided in the archive (Bagnall et al. 2018).
- 1NN-DTW-D, dimension-dependent dynamic time warping(Shokoohi-Yekta, Wang, and Keogh 2015) with (without) normalization. This is a variation of DTW-i approach which directly computes DTW distance based on multi-dimension points instead of treat each dimension separately. It has been shown in previous work that DTW-D can yield more accurate result compared with DTW-i. We directly report the performance results provided in the archive (Bagnall et al. 2018).

5.2 Classification Performance Evaluation

The classification accuracy, the average rank and number of Wins/Ties of each method is shown in Table 3. "N/A" in the table indicates the approach is out of memory, or the accuracy is not reported in (Bagnall et al. 2018). The best accuracy for each dataset is denoted with boldface. Overall, our approach outperforms all the other baseline approaches in terms of average rank. TapNet achieves the lowest overall average rank of 3.1. The approach with the second lowest average rank is WEASEL+MUSE, which is ranked at 3.56. DTW-1NN-D achieves the third lowest overall average rank of 4.5. MLSTM and DTW-1NN-D with normalization achieve similar average rank (4.85 and 4.86 respectively). DTW-1NN-i and ED-1NN have the worst performance among all approaches. In terms of the number of wins/ties, TapNet achieves 14 win (or ties), the best among all eight classifiers. WEASEL+MUSE achieves 12 wins/ties, but the deep learning model, MLSTM-FCN only wins (or ties) in 6 datasets. One nearest neighbor classifiers (ED or DTW) generally only win (or tie) in 1 to 3 datasets.

[1]https://github.com/patrickzib/SFA

[2]https://github.com/titu1994/MLSTM-FCN

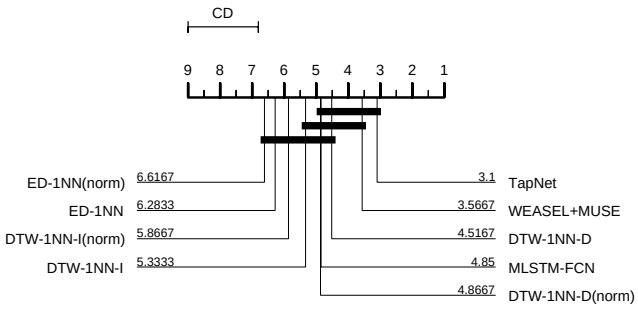

Figure 4: Critical Difference Diagram Generated via error rate of 30 tested dataset

Compared with MLSTM-FCN, TapNet can achieve better performance in most datasets containing small amount of data (e.g. the StandWalkJump dataset, which only contains 12 training samples). MLSTM-FCN often gets overall better performance in high dimensional datasets (e.g. DuckDuckGeese and SpokenArabicDigits). In contrast, TapNet can get better performance in both large and small data.

Compared with WEASEL+MUSE, TapNet can achieve better performance in most high dimensional or large time series, whereas WEASEL+MUSE may not be able to execute due to memory issue. This is because WEASEL+MUSE needs to generate symbols (words) for every subsequence per length per dimension. In the high dimensional time series, the dictionary size can increase dramatically, which makes it hard for the approach to handle large datasets.

Next we conduct statistical hypothesis test to evaluate performance of TapNet. Figure 4 shows the critical difference diagram (with alpha = 0.05) drawn based on Table 3 (Demšar 2006). The values in the figure are the average ranks; the approaches connected by a bold bar have no statistically significant difference from one another. TapNet is in the group of best classifiers. In addition, according the diagram, TapNet is the only method in the experiment that is significantly better than DTW-i, whereas all other latest methods do not show statistically better results compared with DTW-i.

5.3 Semi-Supervised Performance

We next demonstrate that semi-TapNet introduced in this paper can further improve the performance of classification. We select all datasets which have significantly imbalance training/test split. A total of 5 datasets are selected.

Table 3: Performance Comparison in UEA Multivariate Time Series Dataset

Dataset	TapNet	MLSTM -FCN	WEASEL +MUSE	ED-1NN	DTW-1NN-I	DTW-1NN-D	ED-1NN (norm)	DTW-1NN-I (norm)	DTW-1NN-D (norm)
ArticularyWordRecognition	0.987	0.973	**0.99**	0.97	0.98	0.987	0.97	0.98	0.987
AtrialFibrillation	**0.333**	0.267	**0.333**	0.267	0.267	0.2	0.267	0.267	0.22
BasicMotions	**1**	0.95	**1**	0.675	**1**	0.975	0.676	**1**	0.975
CharacterTrajectories	0.997	0.985	0.99	0.964	0.969	0.99	0.964	0.969	0.989
Cricket	0.958	0.917	**1**	0.944	0.986	**1**	0.944	0.986	**1**
DuckDuckGeese	0.575	**0.675**	0.575	0.275	0.55	0.6	0.275	0.55	0.6
EigenWorms	0.489	0.504	**0.89**	0.55	0.603	0.618	0.549	N/A	0.618
Epilepsy	0.971	0.761	**1**	0.667	0.978	0.964	0.666	0.978	0.964
ERing	**0.133**	**0.133**	**0.133**	**0.133**	**0.133**	**0.133**	**0.133**	**0.133**	**0.133**
EthanolConcentration	0.323	0.373	**0.43**	0.293	0.304	0.323	0.293	N/A	0.323
FaceDetection	**0.556**	0.545	0.545	0.519	0.513	0.529	0.519	0.5	0.529
FingerMovements	0.53	**0.58**	0.49	0.55	0.52	0.53	0.55	0.52	0.53
HandMovementDirection	**0.378**	0.365	0.365	0.279	0.306	0.231	0.278	0.306	0.231
Handwriting	0.357	0.286	0.605	0.371	0.509	**0.607**	0.2	0.316	0.286
Heartbeat	**0.751**	0.663	0.727	0.62	0.659	0.717	0.619	0.658	0.717
InsectWingbeat	**0.208**	0.167	N/A	0.128	N/A	0.115	0.128	N/A	N/A
JapaneseVowels	0.965	**0.976**	0.973	0.924	0.959	0.949	0.924	0.959	0.949
Libras	0.85	0.856	0.878	0.833	**0.894**	0.872	0.833	**0.894**	0.87
LSST	0.568	0.373	**0.59**	0.456	0.575	0.551	0.456	0.575	0.551
MotorImagery	**0.59**	0.51	0.5	0.51	0.39	0.5	0.51	N/A	0.5
N/ATOPS	**0.939**	0.889	0.87	0.86	0.85	0.883	0.85	0.85	0.883
PEMS-SF	**0.751**	0.699	N/A	0.705	0.734	0.711	0.705	0.734	0.711
PenDigits	**0.98**	0.978	0.948	0.973	0.939	0.977	0.973	0.939	0.977
Phoneme	0.175	0.11	**0.19**	0.104	0.151	0.151	0.104	0.151	0.151
RacketSports	0.868	0.803	**0.934**	0.868	0.842	0.803	0.868	0.842	0.803
SelfRegulationSCP1	0.652	**0.874**	0.71	0.771	0.765	0.775	0.771	0.765	0.775
SelfRegulationSCP2	**0.55**	0.472	0.46	0.483	0.533	0.539	0.483	0.533	0.539
SpokeN/ArabicDigits	0.983	**0.99**	0.982	0.967	0.96	0.963	0.967	0.959	0.963
StandWalkJump	**0.4**	0.067	0.333	0.2	0.333	0.2	0.2	0.333	0.2
UWaveGestureLibrary	0.894	0.891	**0.916**	0.881	0.869	0.903	0.881	0.868	0.903
Avg. Rank	**3.1**	4.85	3.56	6.28	5.33	4.51	6.61	5.86	4.86
Wins/Ties	**14**	6	12	1	3	3	1	3	2

Table 4: Performance of Semi-Supervised TapNet

Dataset (Training/Test)	**TapNet**	**Semi-TapNet**
Handwriting (150/850)	0.3565	**0.3882**
UWaveGestureLibrary (120/320)	0.894	**0.903**
ArticularyWordRecognition (275/300)	0.987	**0.993**
StandWalkJump (12/15)	**0.4**	**0.4**
JapaneseVowels (270/370)	0.965	**0.968**

The classification accuracy results of TapNet and Semi-TapNet in these 5 datasets are shown in Table 4. Semi-TapNet outperforms TapNet in 4 out of 5 selected datasets. Only one dataset, StandWalkJump, shows no improvement.

Semi-TapNet can achieve better performance in data that have significantly larger number of test samples compared with training samples. For example, in the Handwriting dataset, which has 850 test samples vs. 150 training samples, Semi-TapNet can improve the classification accuracy from 0.3565 to 0.3882 (approximately 9% improvement), whereas in the ArticularyWordRecognition dataset and JapaneseVowels, which have less imbalance training/test split and numbers of samples, the improvement is relatively small (from 0.987 to 0.993 and from 0.965 to 0.968 respectively). The results indicate that semi-TapNet can outperform TapNet when the dataset contains a limited number of labeled data but a larger number of unlabeled data.

5.4 Inspection of Class Prototype

In this section, we visualize the class prototypes and their corresponding time series embeddings to demonstrate the effectiveness of our trained low dimensional time series embedding. We used the t-SNE algorithm (Maaten and Hinton 2008) to visualize the 300-dimension time series embedding in the form of two-dimensional images. We use different colors to isolate different classes, and we use $\circ$ and $\times$ markers to represent the training and testing samples, respectively. The class prototype is shown by the $\bigstar$ mark. Figure 5 shows the embeddings learned for the *Character-trajectories* dataset, which contains 1422 training samples and 1436 testing samples in 20 different classes. From the results, we can conclude that: 1) the distances between data samples from different classes are much larger than the distances from the same class, which means we can easily use

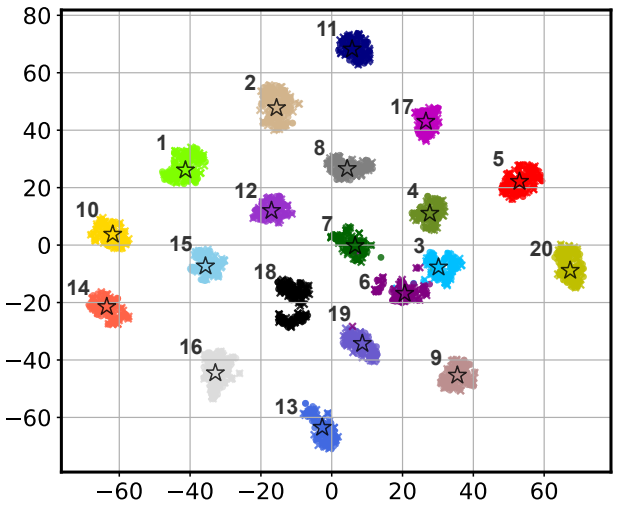

Figure 5: Class Prototype Inspection: visualize the 300-dimension multivariate time series embeddings in two dimensional image by t-SNE. The class prototype is marked by $\star$.

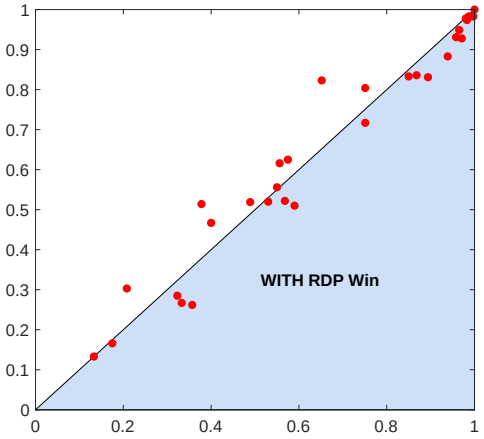

Figure 6: Performance Comparison on TapNet with (without) RDP

the learned multivariate time series embeddings to classify the time series; 2) the low dimensional time series embeddings give us a more interpretable perspective to understand the issues of the classifier. For instance, we can see the embeddings between class 3 and 6 are too close to separate the two classes. In fact, some testing examples in 6 are misclassified to class 3 and vice visa. It does help to identify the issue of a classifier and take further actions such as adding more training samples in the two classes.

5.5 Analysis on Random Dimension Permutation

We next evaluate the impact of accuracy by using Random Dimension Permutation. We compared with TapNet with (without) uses of random dimension permutation in all 30 datasets. Group number and scale factor used in the experiment can be found in appendix.

The pairwise accuracy comparison in all 30 datasets is shown in Figure 6. The x-axis is the accuracy of TapNet with Random Dimension Permutation and y-axis is the accuracy that TapNet achieved without Random Dimension Permutation. The point fallen into the shadow indicates that TapNet with Random Permutation improves the performance.

According to Figure 6, the performance increases in 22 out of 30 datasets, and only decrease in 8 datasets. These datasets include most of datasets contains 100 to 300 samples with dimension ranging from 2 to 100. The experiment shows that Random Dimension Permutation can overall increase the performance of TapNet.

6 Conclusion

In this paper, we present a novel distance-based deep learning framework, named TapNet, for multivariate time series classification problem. In particular, we propose a novel attentional prototype network to train the low-dimensional feature representations based on their distances to class prototype with limited training labels. Moreover, a random group permutation method is designed to learn the interactive features in multivariate dimensions combined with multi-layer convolutional networks. Additionally, we propose a semi-supervised model, Semi-TapNet, to utilize the unlabeled data in improving the classification performance when training samples are scarce. The experimental results demonstrate that our model can achieve the highest average rank on 30 datasets in the public UEA archive compared to eight state-of-the-art methods.

References

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. *Data Mining and Knowledge Discovery* 31(3):606–660.

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 preprint arXiv:1811.00075*.

Banerjee, A.; Merugu, S.; Dhillon, I. S.; and Ghosh, J. 2005. Clustering with bregman divergences. *Journal of machine learning research* 6(Oct):1705–1749.

Baydogan, M. G., and Runger, G. 2015. Learning a symbolic representation for multivariate time series classification. *Data Mining and Knowledge Discovery* 29(2):400–422.

Baydogan, M. G., and Runger, G. 2016. Time series representation and similarity based on local autopatterns. *Data Mining and Knowledge Discovery* 30(2):476–509.

Begum, N.; Hu, B.; Rakthanmanon, T.; and Keogh, E. 2014. A minimum description length technique for semi-supervised time series classification. In *Integration of reusable systems*. Springer. 171–192.

Chen, Y.; Hu, B.; Keogh, E.; and Batista, G. E. 2013. 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*, 383–391. ACM.

Cuturi, M., and Doucet, A. 2011. Autoregressive kernels for time series. *arXiv preprint arXiv:1101.0673*.

De Maesschalck, R.; Jouan-Rimbaud, D.; and Massart, D. L. 2000. The mahalanobis distance. *Chemometrics and intelligent laboratory systems* 50(1):1–18.

Demšar, J. 2006. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine learning research* 7(Jan):1–30.

Fang, Z.; Wang, P.; and Wang, W. 2018. Efficient learning interpretable shapelets for accurate time series classification. In *2018 IEEE 34th International Conference on Data Engineering (ICDE)*, 497–508. IEEE.

Ilse, M.; Tomczak, J. M.; and Welling, M. 2018. Attention-based deep multiple instance learning. *arXiv preprint arXiv:1802.04712*.

Ioffe, S., and Szegedy, C. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*.

Karim, F.; Majumdar, S.; Darabi, H.; and Chen, S. 2018a. Lstm fully convolutional networks for time series classification. *IEEE Access* 6:1662–1669.

Karim, F.; Majumdar, S.; Darabi, H.; and Harford, S. 2018b. Multivariate lstm-fcns for time series classification. *arXiv preprint arXiv:1801.04503*.

Karlsson, I.; Papapetrou, P.; and Boström, H. 2016. Generalized random shapelet forests. *Data Mining and Knowledge Discovery* 30(5):1053–1085.

Kingma, D. P., and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, 1097–1105.

Lal, T. N.; Hinterberger, T.; Widman, G.; Schröder, M.; Hill, N. J.; Rosenstiel, W.; Elger, C. E.; Birbaumer, N.; and Schölkopf, B. 2005. Methods towards invasive human brain computer interfaces. In *Advances in neural information processing systems*, 737–744.

Li, X., and Lin, J. 2017. Linear time complexity time series classification with bag-of-pattern-features. In *Data Mining (ICDM), 2017 IEEE International Conference on*, 277–286. IEEE.

Maaten, L. v. d., and Hinton, G. 2008. Visualizing data using t-sne. *Journal of machine learning research* 9(Nov):2579–2605.

Marussy, K., and Buza, K. 2013. Success: a new approach for semi-supervised classification of time-series. In *International Conference on Artificial Intelligence and Soft Computing*, 437–447. Springer.

Minnen, D.; Starner, T.; Essa, I.; and Isbell, C. 2006. Discovering characteristic actions from on-body sensor data. In *Wearable computers, 2006 10th IEEE international symposium on*, 11–18. IEEE.

Neyshabur, B.; Tomioka, R.; and Srebro, N. 2014. In search of the real inductive bias: On the role of implicit regularization in deep learning. *arXiv preprint arXiv:1412.6614*.

Rakthanmanon, T., and Keogh, E. 2013. Fast shapelets: A scalable algorithm for discovering time series shapelets. In *proceedings of the 2013 SIAM International Conference on Data Mining*, 668–676. SIAM.

Schäfer, P., and Leser, U. 2017a. Fast and accurate time series classification with weasel. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, 637–646. ACM.

Schäfer, P., and Leser, U. 2017b. Multivariate time series classification with weasel+ muse. *arXiv preprint arXiv:1711.11343*.

Senin, P., and Malinchik, S. 2013. Sax-vsm: Interpretable time series classification using sax and vector space model. In *Data Mining (ICDM), 2013 IEEE 13th International Conference on*, 1175–1180. IEEE.

Shokoohi-Yekta, M.; Wang, J.; and Keogh, E. 2015. On the non-trivial generalization of dynamic time warping to the multi-dimensional case. In *Proceedings of the 2015 SIAM international conference on data mining*, 289–297. SIAM.

Snell, J.; Swersky, K.; and Zemel, R. 2017. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems*, 4077–4087.

Sundermeyer, M.; Schlüter, R.; and Ney, H. 2012. Lstm neural networks for language modeling. In *Thirteenth annual conference of the international speech communication association*.

Wang, X.; Lin, J.; Senin, P.; Oates, T.; Gandhi, S.; Boedihardjo, A. P.; Chen, C.; and Frankenstein, S. 2016. Rpm: Representative pattern mining for efficient time series classification. In *19th International Conference on Extending Database Technology (EDBT)*, 185–196.

Wei, L., and Keogh, E. 2006. Semi-supervised time series classification. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, 748–753. ACM.

Wistuba, M.; Grabocka, J.; and Schmidt-Thieme, L. 2015. Ultra-fast shapelets for time series classification. *arXiv preprint arXiv:1503.05018*.

Xu, B.; Wang, N.; Chen, T.; and Li, M. 2015. Empirical evaluation of rectified activations in convolutional network. *arXiv preprint arXiv:1505.00853*.

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

Yu, F., and Koltun, V. 2015. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*.

Zhao, B.; Lu, H.; Chen, S.; Liu, J.; and Wu, D. 2017. Convolutional neural networks for time series classification. *Journal of Systems Engineering and Electronics* 28(1):162–169.

Zheng, Y.; Liu, Q.; Chen, E.; Ge, Y.; and Zhao, J. L. 2014. Time series classification using multi-channels deep convolutional neural networks. In *International Conference on Web-Age Information Management*, 298–310. Springer.

A Appendix

In the Appendix Section, the architecture of Multivariate time series encoding and experiment parameters setup are presented.

A.1 Architecture of Multivariate Time Series Encoding

Table 5 presents the architecture of the multivariate time series encoding component. According to the overall architecture of TapNet model in Figure 1, we divide the encoding component into sub-components: LSTM, Conv and FC. We denote a Long-short term memory network layer by "LSTM", in brackets we give the size of hidden dimensions of LSTM. The convolutional layer is denoted by "Conv", in brackets we provide the kernel size, stride and padding. The number of filters are provided after a dash. The batch normalization and leaky ReLU activation function are used after convolutional operation. We denote the fully connected layer by "FC" and the output dimension is provided after a dash.

Table 5: Implementation Details for MultivariateTime Series Encoding

	Layer	Type
LSTM	1	LSTM(128)
	2	Global Average Pooling (2)
Conv	1-1	Conv(8,1,0)-256 filters + Batch Norm + Leaky ReLU
	1-2	Conv(8,1,0)-256 filters + Batch Norm + Leaky ReLU
	1-3	Conv(8,1,0)-256 filters + Batch Norm + Leaky ReLU
	2	Conv(5,1,0)-256 filters + Batch Norm + Leaky ReLU
	3	Conv(3,1,0)-128 filters + Batch Norm + Leaky ReLU
	4	Global Average Pooling (2)
FC	1	FC-500 + Batch Norm + Leaky ReLU
	2	FC-300 + Batch Norm + Leaky ReLU

A.2 Experiment Parameter Settings

In Table 6, the details of the optimization procedure for TapNet are given, including the optimizer, learning rate, weight decay, epochs and stopping criteria. Since most of the datasets in our experiment contain limited training samples, it is not feasible to separate a validation set from the small size of training samples. For example, the dataset "AtrialFibrillation" contains only 15 training samples for 3 classes. It is impossible to split the incredibly small training samples into training and validation sets. Thus, we use the stop condition only based on the our training loss. During the training, we found our training loss can be converged to a close to zero number. Therefore, we set the stop condition as the difference of training loss between epochs is less than a small threshold, 1e-9.

Table 6: The optimization procedure details

Optimizer	Learning Rate	Weight decay	Epochs	Stopping criteria
Adam	0.00001	0.001	3000	Training loss difference between epochs less than 1e-9 or reach the maximum epoch

Due to the various types of our 30 experiment datasets, it is infeasible to use a fixed set of parameters to fit all the datasets. For instance, the "InsectWingbeat" dataset contains 30,000 training samples but the "AtrialFibrillation" dataset only contains 15 training samples. For some extremely large dataset, it is easy to get out-of-memory issue in GPU if we set a large filter number. Therefore, we use smaller values for filters for large size of dataset. Moreover, the time series length are extremely different in our datasets. For example, the "EigneWorms" dataset has time series with 17984 length but the time series length in "PenDigits" dataset is only 8. So it is infeasible to apply our default kernel size "8" into a 8-length time series in "PenDigits" dataset.

We demonstrate all the parameters for each dataset in Table 1 as well as the characteristics of the datasets such as number of samples, multivariate dimension, time series length and number of classes. The training and testing samples are presented in the format of "# of training samples$/$# of testing samples". The parameters varied by datasets include Random Dimension Permutation (RDP), Dilation, Filters, Kernels and Learning rate (LR). The RDP parameters are presented by the format "# of group$/$scale factor". The number of filters and kernels of all three convolutional layers are separated by comma in the format of "# of 1^{st} layer , # of 2^{nd} layer, # of 3^{rd} layer". All the other parameters and network structures are kept the same as the default settings for all the 30 datasets.

Table 7: Paramter Setting for All 30 UEA Multivariate Time Series Datasets

Dataset	Samples	Dim	Length	Class	RDP	Dilation	Filters	Kernels	LR
ArticularyWordRecognition	275/300	9	144	25	3/1.5	10	256,256,128	8,5,3	1e-5
AtrialFibrillation	15/15	2	640	3	3/1.5	1	256,256,128	8,5,3	1e-5
BasicMotions	40/40	6	100	4	3/1.5	1	256,256,128	8,5,3	1e-5
CharacterTrajectories	1422/1436	3	182	20	3/1.5	1	256,256,128	8,5,3	1e-5
Cricket	108/72	6	1197	12	3/1.5	1	256,256,128	8,5,3	1e-5
DuckDuckGeese	60/40	1345	270	5	3/1.5	1	256,256,128	8,5,3	1e-5
EigenWorms	128/131	6	17984	5	3/1.5	200	256,256,128	8,5,3	1e-5
Epilepsy	137/138	3	206	4	3/1.5	1	256,256,128	8,5,3	1e-5
ERing	30/30	4	65	6	3/1.5	1	256,256,128	8,5,3	1e-5
EthanolConcentration	261/263	3	1751	4	3/1.5	200	256,256,128	8,5,3	1e-6
FaceDetection	5890/3524	144	62	2	3/1.5	200	64,64,32	8,5,3	5e-5
FingerMovements	316/100	28	50	2	3/1.5	1	256,256,128	8,5,3	1e-5
HandMovementDirection	320/147	10	400	4	3/1.5	1	256,256,128	8,5,3	1e-5
Handwriting	150/850	3	152	26	3/1.5	1	256,256,128	8,5,3	1e-5
Heartbeat	204/205	61	405	2	3/1.5	200	64,64,32	8,5,3	1e-6
InsectWingbeat	30K/20K	200	78	10	2/4	10	4,8,4	8,5,3	1e-4
JapaneseVowels	270/370	12	29	9	3/1.5	1	256,256,128	8,5,3	1e-5
Libras	180/180	2	45	15	3/1.5	1	256,256,128	8,5,3	1e-5
LSST	2459/2466	6	36	14	3/1.5	1	256,256,128	8,5,3	1e-5
MotorImagery	278/100	64	3000	2	3/1.5	1	256,256,128	8,5,3	1e-5
NATOPS	180/180	24	51	6	3/1.5	1	256,256,128	8,5,3	1e-5
PEMS-SF	267/173	963	144	7	3/1.5	1	256,256,128	8,5,3	1e-5
PenDigits	7494/3498	2	8	10	3/1.5	1	256,256,128	4,1,1	1e-3
Phoneme	3315/3353	11	217	39	3/1.5	1	64,64,32	8,5,3	1e-3
RacketSports	151/152	6	30	4	3/1.5	1	256,256,128	8,5,3	1e-5
SelfRegulationSCP1	268/293	6	896	2	3/1.5	1	256,256,128	8,5,3	1e-6
SelfRegulationSCP2	200/180	7	1152	2	3/1.5	1	256,256,128	8,5,3	1e-9
SpokenArabicDigits	6599/2199	13	93	10	3/1.5	1	128,128,64	8,5,3	1e-4
StandWalkJump	12/15	4	2500	3	3/1	1	256,256,128	8,5,3	1e-5
UWaveGestureLibrary	120/320	3	315	8	3/1.5	1	256,256,128	8,5,3	1e-5