Time series classification using local distance-based features in multi-modal fusion networks

Brian Kenji Iwana[a,*], Seiichi Uchida[a]

[a]*Department of Advanced Information Technology, Kyushu University, Fukuoka, Japan*

Abstract

We propose the use of a novel feature, called local distance features, for time series classification. The local distance features are extracted using Dynamic Time Warping (DTW) and classified using Convolutional Neural Networks (CNN). DTW is classically as a robust distance measure for distance-based time series recognition methods. However, by using DTW strictly as a global distance measure, information about the matching is discarded. We show that this information can further be used as supplementary input information in temporal CNNs. This is done by using both the raw data and the features extracted from DTW in multi-modal fusion CNNs. Furthermore, we explore the effects of different prototype selection methods, prototype numbers, and data fusion schemes induce on the accuracy. We perform experiments on a wide range of time series datasets including three Unipen handwriting datasets, four UCI Machine Learning Repository datasets, and 85 UCR Time Series Classification Archive datasets.

Keywords: Convolutional Neural Network, Time Series Classification, Dynamic Time Warping, Distance Features

1. Introduction

Time series represent a large domain of data that cross numerous disciplines and time series recognition is a challenging but important task. A discrete time series is a sequence $\mathbf{s} = s_1, \ldots, s_t, \ldots, s_T$ of data elements $s_t \in \mathbb{R}^N$ across distinct time steps t. The difficulty of time series classification is the requirement to maintain the time dependence of the sequence.

Traditionally, time series classification has been tackled using distance-based methods [1], such as k-Nearest Neighbors (k-NN). However, due to the recent availability of data, artificial neural networks have become powerful tools for pattern recognition [2] and the use of neural networks does not exclude time series data. Most notably, Recurrent Neural Networks (RNN) [3] and Long Short-Term Memory RNNs (LSTM) [4] have had many successes on time series in natural language processing [5, 6] and time series prediction [7]. While these RNNs have traditionally been used to tackle time series problems, recent work has shown that variations of convolutional feedforward neural networks can effective and sometimes better for time series classification [8, 9]. However, despite the achievements that Convolutional Neural Networks (CNN) have had on image data [10, 11], there are still aspects of time series that have not been fully addressed, such as variable lengths and temporal distortions.

In contrast, tools like Dynamic Time Warping (DTW) [12] were specifically designed to address the difficulties with time series. DTW is a well-established distance measure for distance-based pattern recognition methods, especially in conjunction with k-NN [1, 13]. The standard use of DTW is to calculate the global distance between two time series patterns by taking the sum of the local distances between nonlinearly matched time series elements. The element matching in DTW is done using dynamic programming and allows for an elastic match that is robust to temporal distortions, variations in length, and variations in rate. However, when using DTW as a distance measure, only the summation, or global distance, is used and the information about the dynamic matching of sequence elements is discarded.

In this work, we show that these local distances within a DTW calculation can be useful for time series classification. We propose the creation of a new time series based on a sequence of the local distances between the elements matched by DTW, as shown in Fig. 1. In this way, classification is done based on the local relationships between elements rather than the raw data. Specifically, the local distances between the elements matched by DTW between an input time series and a prototype time series are used as features for a new local distance feature time series. The local distances are referred to as the *local distance features* and the original input time series are *coordinate features*.

Moreover, demonstrated in Fig. 1, multiple prototypes can be used as references for the input time series to obtain additional local distance features for each time step. By extracting the additional local distance features in relation to multiple prototypes, additional information is embedded into the new time series. Each additional prototype would increase the local distance feature-based time series by a dimension. However, using a large number of prototypes can be unwieldy and a small number of prototypes might not contain enough information for ac-

*Corresponding author

Email addresses: brian@human.ait.kyushu-u.ac.jp (Brian Kenji Iwana), uchida@ait.kyushu-u.ac.jp (Seiichi Uchida)

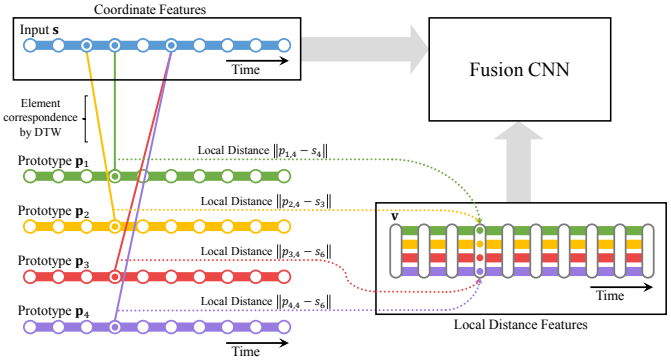

Figure 1: Illustration of the proposed method of creating a feature vector $\mathbf{v}$ from the local distances extracted from a DTW calculation. The lines represent the local distances between matched elements of the input $\mathbf{s}$ and the prototypes $\mathbf{p}_1$, $\mathbf{p}_2$, $\mathbf{p}_3$, and $\mathbf{p}_4$.

curate classification. Therefore, there is a question of how many and what methods should be used to select the prototypes. In this paper, we tackle these questions by evaluating the effects prototype number and prototype selection have on the proposed local distance features and their usefulness for time series classification.

We explore five different prototype selection methods, Borders, Closest, Spanning, K-Centers, and Random. Borders are the prototypes furthest away from the center of the training dataset. Closest is the ones near the center of the training dataset. Spanning prototypes are chosen evenly across the dataset and K-Centers prototype selection finds prototypes using K-Centers clustering [14]. Finally, Random selects prototypes at random. These methods are used both class-independent and classwise.

Using the prototypes selected from these methods, the local distance features can be extracted and used in conjunction with CNNs. One advantage of using the local distance features is that given an asymmetric slope constraint for DTW, the length of the local distance feature vectors can be fixed, even with time series of different lengths. This is ideal for the input of CNNs. Another advantage is that the dynamic matching warps the input which adds robustness to temporal distortions pre-built into the features.

Furthermore, since the local distance features portray the relationships between the input and prototypes, it is possible that they contain different information than the raw element data. Therefore, we combine the two features, local distance features and coordinate features, in multi-modal fusion networks. In other words, the two features are treated as two separate modalities and can be used with fusion CNNs to achieve more robust results.

The main contributions of this paper are:

- We propose the use of the local distances from element-wise matching through dynamic programming as features that can be used for time series classification. These local distance features can be combined in multi-modal fusion CNNs.
- We demonstrate that there is a diminishing return on the amount of information in adding prototypes and very few prototype patterns are actually required for state-of-the-art results.
- This paper explores the effects that the prototypes and the method of their selection have on the effectiveness of the model. We accomplish this by providing experimental results using five prototype selection methods, random selection, closest to the center pattern selection, border pattern selection, spanning, and selection using K-Centers clustering.
- The proposed method is evaluated on three Unipen online written character datasets [15], four UCI time series datasets [16], and the 85 UCR time series datasets [17], showing that the method can be generalized to many different types of time series.

2. Related work

2.1. Convolutional neural networks for time series and sequences

In recent times, CNNs have many successes in image recognition [10, 11]. CNNs have also been extended for time series recognition by considering time steps as a dimension for the convolutions. For example, the predecessor to CNNs, Time Delay Neural Networks (TDNN) [18] use a sliding window across multiple time steps, similar to 1D CNNs. As for CNNs, Zheng et al. [19] use them with 1D subsequences created from multivariate time series. There have also been attempts [20, 21, 22] to classify time series patterns by embedding them into 2D matrices for classification by CNN.

Another approach to using CNNs with time series is the use of dilated causal convolutional layers. Dilated convolutional layers [23] are convolutional layers that skip input values to increase the receptive field but overlap so that the size of the inputs does not decrease. WaveNet [24] is a groundbreaking work that employs the dilated causal convolutional layers to generate audio. Bai et al. [9] proposed Temporal Convolutional Networks (TCN) which also employ dilated convolutions but in a simpler architecture compared to WaveNets.

2.2. Multi-modal fusion networks

Fusion neural networks using multi-modal data is an expanding field in neural network development. Multi-modal recognition uses different types, or modalities, of data, often from different domains. Multi-modal fusion networks extend this by fusing the modalities within the architecture of one network. Zahavy et al. [25] used a decision-level fusion scheme combining title and image data for eCommerce classification. Wang et al. [26] tackle text aided image classification by combining image-based CNN features and word embedding-based CNN features using both early and late fusion. Fusion networks have also been used for combining title and image data [25] and for multi-source data types [27, 28], activities of daily life [27]. One work, in particular, Song et al. [29] tackle time series recognition by combining features from a CNN using correlation maps of the input and an encoding from LSTMs.

2.3. Dynamic time warping and feature extraction

Using DTW to derive distance-based features has been explored in the past. However, DTW-based feature methods normally use the global DTW distance as a feature. This can be seen in the work by Kate [30] where global DTW-based features are used for traditionally statistical methods such as a Support Vector Machine (SVM) and Symbolic Aggregate Approximation (SAX). DTW distances have also been used as features when used in conjunction with Dissimilarity Space Embedding (DSE) [31]. The difference between these methods and the proposed method is that the proposed method does not use the global DTW as a feature and instead uses the distances of the local matchings between elements. The method proposed in this paper is an extension of [32] where local distance-based features were first introduced. This work extends the previous work by proposing deliberate prototype selection methods, performing an ablation study that explores the effects of the prototypes, and demonstrates the proposed method's ability to achieve state-of-the-art results on many datasets across a diverse set of time series applications.

3. Local distance-based feature extraction

3.1. Feature extraction using DTW

DTW is a widely used algorithm for determining the distance between time series patterns. Unlike linear matching, DTW uses an optimized matching of elements in order to be robust to temporal distortions, such as differences in rate and translations in time. The matching determined by DTW is done by aligning similar elements using dynamic programming. Specifically, DTW elastically matches elements by estimating the minimal path on a cost matrix made of the local distances between elementwise pairings. This creates a matching between sequence elements that is warped in the time dimension.

Given two time series, the previously defined $\mathbf{s}$ and a prototype time series $\mathbf{p} = p_1, \ldots, p_u, \ldots, p_U$ with U number of time steps and $p_u \in \mathbb{R}^Q$, where Q is the dimensionality of each element, the global DTW distance is the summation of the local distances optimally matched. Namely,

$$\mathrm{DTW}(\mathbf{p}, \mathbf{s}) = \sum_{(u',t')\in\mathcal{M}} \|p_{u'} - s_{t'}\|, \tag{1}$$

where (u', t') are the indices of a match corresponding to the original indices u and t of $\mathbf{p}$ and $\mathbf{s}$, respectively. The summation of *local distances* between matches is then used as a distance measure between discrete time series.

However, as seen in Eq. (1), only the total global distance is used and the actual matching calculation is wasted. While this fact generally does not matter for traditional distance-based methods, it is possible that information about the structural relationship between the compared patterns is lost. Fig. 2 shows a comparison between DTW calculations where the local distances between elements can reveal additional information which would normally be lost when using DTW as a global distance measure. The figure shows four examples of different prototypes $\mathbf{p}_1$, $\mathbf{p}_2$, $\mathbf{p}_3$, and $\mathbf{p}_4$ which have similar DTW distances

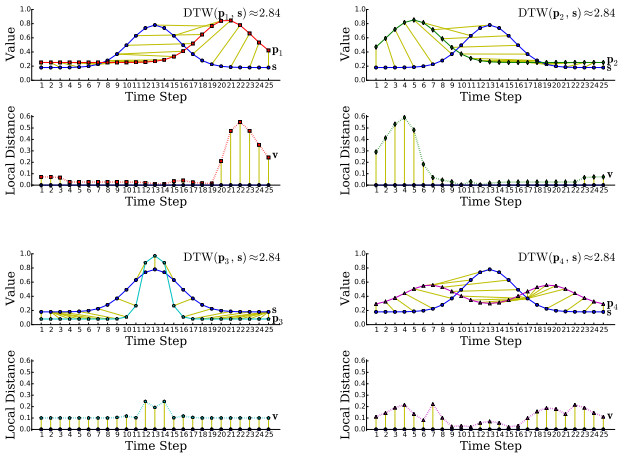

Figure 2: An example of a 1D time series $\mathbf{s}$ and four prototypes $\mathbf{p}_1$, $\mathbf{p}_2$, $\mathbf{p}_3$, and $\mathbf{p}_4$ that were crafted to have similar DTW-distances but very different structures and therefore different local distance features. Each prototype and sample pair have the two patterns illustrated above with the yellow connections representing matching by DTW and below is the local distance feature sequence.

to the common sample $\mathbf{s}$. Using a traditional distance-based classification method would not be able to distinguish the different prototypes even though the prototypes are significantly different from each other. On the other hand, the local distances between the matched elements of the time series maintain discriminating information that can be exploited. Thus, instead of using the total summation of matches, a sequence $\mathbf{v}$ is created using the local distances between matched elements, or:

$$\mathbf{v} = (\|p_1 - s_1\|), \ldots, (\|p_{u'} - s_{t'}\|), \ldots, (\|p_{U'} - s_{T'}\|) \tag{2}$$

for each $(u', t') \in \mathcal{M}$. We refer to the elements of $\mathbf{v}$ as *local distance features* and the original features of sequence $\mathbf{s}$ as *coordinate features*. A visualization of $\mathbf{v}$ is shown in Fig. 3 where each column represents the local distances from the sample to a prototype.

When used with multiple prototypes, as shown in Fig. 1, a multivariate sequence:

$$\mathbf{v} = \begin{pmatrix} \|p_{1,1} - s_{1,1}\| \\ \cdots \\ \|p_{K,1} - s_{K,1}\| \end{pmatrix}, \ldots, \begin{pmatrix} \|p_{1,u'} - s_{1,t'}\| \\ \cdots \\ \|p_{K,u'} - s_{K,t'}\| \end{pmatrix}, \ldots, \begin{pmatrix} \|p_{1,U'} - s_{1,T'}\| \\ \cdots \\ \|p_{K,U'} - s_{K,T'}\| \end{pmatrix} \tag{3}$$

with each element of $\mathbf{v}$ in $\mathbb{R}^K$, is created, where K is the total number of prototypes. When using a DTW slope constraint that ensures the time step always advances by one in relation to the prototype sequences, such as the asymmetric slope constraint defined by the recurrent function [33]:

$$D(u, v) = \|p_u - s_t\| + \min_{t'\in\{t,t-1,t-2\}} D(u-1, t'), \tag{4}$$

where $D(u, v)$ is the cumulative sum, input sequences of different lengths can be used to create the fixed length $\mathbf{v}$ as long as the prototypes are of the same length. This is because using this particular slope constraint guarantees that the number of matches in $\mathcal{M}$ will be always equal to the number of elements in the prototype sequence $\mathbf{p}$. Thus, the formulation of the local

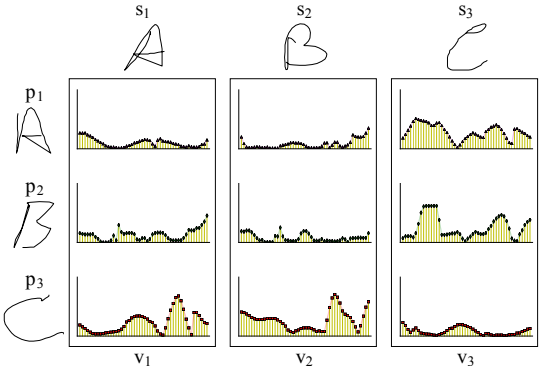

Figure 3: Examples of the local distance features and their corresponding online character time series. Each row relates to a prototype $\mathbf{p}_1$, $\mathbf{p}_2$, $\mathbf{p}_3$ and each column is the local distance features to a sample $\mathbf{s}_1$, $\mathbf{s}_2$, $\mathbf{s}_3$. The local distance feature-based time series $\mathbf{v}_1$, $\mathbf{v}_2$, $\mathbf{v}_3$ is the three dimensional combination of the prototypes for each sample.

distance feature sequences $\mathbf{v}$ provides us with a fixed length, temporally warped time series.

3.2. Classification using temporal CNNs

After the formulation of the local distance feature sequences, we propose the use of them as inputs of 1D convolution CNNs for time series classification. The coordinate features represent the raw time series pattern and the local distance features represent the relationships to prototype patterns. By combining both features, it is possible to train a CNN which combines the information from both types of data.

Combining the two different modalities of data is referred to as multi-modal classification. In the context of CNNs, there are different ways that the modalities can be combined. If CNNs with the two modalities are fused at the input-level, it is referred to as *data-level fusion* or *early fusion*. If the CNNs are fused at one of the intermediary layers, then it is called *feature-level fusion* or *middle fusion*. Finally, if the CNNs are fused at the end, right before the classification layer, then it is *decision-level fusion* or *late fusion*. The timing of the fusion depends on the author of a model and is chosen like a hyperparameter.

Figure 4 outlines the three fusion schemes in combination with the proposed feature extraction. Shown in Fig. 4 (a), the early fusion model takes the coordinate features and combines them with the local distance features to create a single time series input. This is the equivalent of considering the local distance features as additional dimensions a multivariate time series. Aside from the data fusion, the structure of the CNN is identical to a standard temporal CNN. For the middle fusion model in Fig. 4 (b), the two modalities are provided to separate sets of convolutional layers and are concatenated before the shared fully-connected layers. Each half would learn independent sets of convolutional weights respective to their individual modalities. The late fusion network in Fig. 4 (c) is structured similar to two distinct CNNs each with a data modality, but are combined as inputs for the output layer.

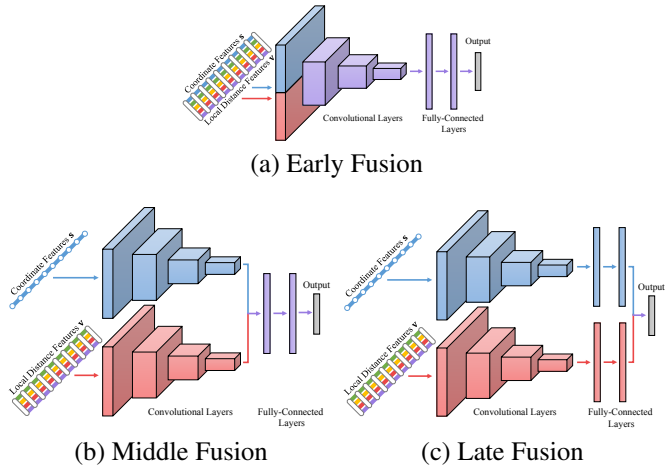

Figure 4: Comparison of the proposed method under the three fusion schemes.

3.3. Prototype selection

Classification in distance-based learning is done by observing the differences between prototypes and samples. However, with the expanding size of datasets, the number of prototypes can be large. Thus, selecting the best prototypes for the task can be an effective step toward reducing the computational time while retaining the required information for accurate classification. Prototype selection for distance-based methods, most notably k-NN, have been extensively studied in the past [34]. In addition, there have been many prototype selection and generation methods proposed for time series. For example, selection using AdaBoost [31] and generation using DTW Barycentric Averaging (DBA) [35].

In this paper, we borrow the idea of using of prototype selection from distance-based classifiers to choose prototypes for the creation of the local distance features $\mathbf{v}$ in order to selectively increase the amount of information embedded in $\mathbf{v}$. Specifically, a subset $\mathbf{P}'$ is determined from the entire training set $\mathbf{P}$ using *Random*, *Borders*, *Closest*, *Spanning*, and *K-Centers* prototype selection. Each of these methods can be performed on the entire dataset, *class-independent*, or within each class, *class-wise*. Except for random selection, each of the prototype selection methods was adapted to time series by using DTW as the distance measure for distance calculations. Examples of each prototype selection method done class-independent is shown in Fig. 5. The remainder of this section will detail the prototype selection methods used in the experiments.

Random prototype selection. Random selection is the simple method of selecting prototypes at random. This method is used as a baseline for the deliberate selection methods.

Borders prototype selection. The Borders method creates a prototype subset $\mathbf{P}'$ that represents the patterns on the borders of the training set $\mathbf{P}$. Namely,

$$\text{Borders}(\mathbf{P}) = \underset{\mathbf{p}_i \in \mathbf{P}}{\text{argmax}} \sum_{\mathbf{p}_j \in \mathbf{P}} \text{DTW}\left(\mathbf{p}_i, \mathbf{p}_j\right), \tag{5}$$

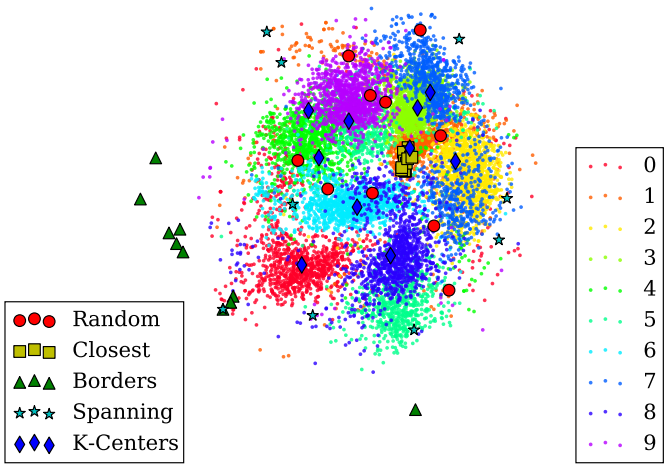

Figure 5: The Unipen 1a online handwritten digit dataset visualized using with PCA. The larger points are the results of the prototype selection methods where $K = 10$. The small points are colored in the digit classes.

where Borders($\mathbf{P}$) is the resulting index of a prototype in the training set $\mathbf{P}$ which has the total largest distance, determined by DTW, to all other patterns. This process is repeated K number of times with each round moving the selected prototype from $\mathbf{P}$ to $\mathbf{P}'$. When used in a classwise manner, the borders prototype selection method would generally pick patterns on the edge of each class, selecting the difficult to classify patterns or the patterns near the decision boundaries and when used in a class-independent way, the selected patterns lie at the edges of the entire training dataset.

Closest prototype selection. Closest takes the opposite approach of Borders. It constructs the prototype subset $\mathbf{P}'$ from the patterns closest to every other pattern in the training set $\mathbf{P}$. In other words, the center of $\mathbf{P}$, or:

$$\text{Closest}(\mathbf{P}) = \underset{\mathbf{p}_i \in \mathbf{P}}{\operatorname{argmin}} \sum_{\mathbf{p}_j \in \mathbf{P}} \text{DTW}\left(\mathbf{p}_i, \mathbf{p}_j\right), \tag{6}$$

repeating K times. Class-independent Closest selection is an intuitively poor selection method for the proposed method compared to the other methods due to the selected prototypes being similar to each other and providing little extra information. However, when used in a classwise manner, there could be a use in selecting the center prototypes of each class.

Spanning prototype selection. The objective of the Spanning prototype selection is to a uniform distribution across the dataset. The algorithm for Spanning prototype selection is outlined in the supplementary material. Unlike the previous methods, spanning considers the previously selected prototypes. It selects the prototype that is the furthest DTW-distance from all of the previous selections. The result is a prototype set $\mathbf{P}'$ that contains prototypes which are as far apart from each other as possible, spanning the entire original dataset $\mathbf{P}$.

K-Centers prototype selection. The K-Centers prototype selection method follows a K-Centers, or K-medoids, clustering [14] method to select the prototypes. The reason for selecting the medoid of the clusters in $\mathbf{P}$ is to acquire a distribution of prototypes that is similar to the distribution of the whole set.

The K-Centers prototype selection algorithm is also provided in the supplementary material. In order to calculate K-Centers, first, the prototype set $\mathbf{P}'$ is initialized using Spanning to create a deterministic initialization for K-Centers. Second, training samples $\mathbf{p} \in \mathbf{P}$ are assigned to clusters $\mathbf{C}_1, \ldots, \mathbf{C}_k, \ldots, \mathbf{C}_K$ based on their proximity to the nearest cluster center. Finally, new centers are determined for each cluster. This process is repeated until there are no changes in cluster centers.

4. Experimental results

4.1. Datasets

To evaluate the proposed method, we used the following datasets due to having many different properties.

The Unipen 1a, 1b, and 1c datasets [15] consist of pen-tip coordinates of isolated online handwritten digits, uppercase characters, and lowercase characters, respectively. The Unipen datasets are well-established as a benchmark for time series classification. Each dataset consists of about 13,000 patterns. For the experiments, the dataset was split into 10 folds for 10-fold cross-validation with the prototypes selected from each of the training sets.

The UCI Machine Learning Repository [16] is a large repository of datasets across many domains. From the repository, four time series datasets were used for the experiments. The datasets were selected based on the criteria of being time series classification tasks, having enough patterns in the training set, and having state-of-the-art baselines. The following datasets from the UCI Machine Learning Repository were used: The Activities of Daily Life (ADL) Recognition with Wrist-worn Accelerometer Data Set, the Spoken Arabic Digit Data Set (Arabic), the Australian Sign Language signs Data Set (Auslan), and the Hill-Valley Data Set. ADL is comprised of 705 3-axis accelerometer measurements separated in 7 ADL classes. Arabic contains 13-frequency Mel-Frequency Cepstrum Coefficients (MFCC) in 10 spoken digit classes. The dataset has 8,800 patterns, 6,600 of which make up the training set and 2,200 for a speaker independent test set. Next, Auslan has 6,650 sign language words from 95 classes. Finally, the Hill-Valley Data Set is a synthetic dataset made of 606 time series patterns, each 100 time steps, which are classified into a "hill" or a "valley." There are two versions, one without noise (HillValley) and one with noise (HillValley w/ noise). The training and test sets of datasets with predefined splits were used when available. 10-fold cross-validation was used when there were no predefined set splits.

The final category of datasets is the 85 1D time series datasets from the UCR Time Series Classification Archive [17]. These datasets span many different domains and have varying characteristics. The datasets have between 2 and 60 classes, 60 and 2709 time steps, and all contain predefined training and test sets.

4.2. Implementation details

To evaluate the usefulness of the proposed local distance features with CNNs, we used five implementations, a CNN with the time series features alone (Coord. Features), a CNN with the local distance features alone (Loc. Dist. Features), a data-level fusion CNN (Early Fusion), a feature-level fusion CNN (Middle Fusion), and a decision-level fusion CNN (Late Fusion). All of the implementations use 1D convolutional layers with a kernel size of 3 and stride 1. Unlike CNNs which are typically used with images, temporal CNNs can use 1D convolutions where the convolution is used across the time dimension. For this application, there is little difference in results when using a 1D or 2D convolution [32], thus a 1D convolution approach was chosen. After each convolutional layer, a 1D max pooling layer with a window size of 2 and a stride of 2 was used.

Due to the wide range of datasets outlined in Section 4.1, the number of convolutional layers and pooling layers was determined based on the length of the input. Specifically, the number of pooling layers L_{pool} and number of convolutional layers L_{conv} was defined by:

$$L_{pool} = L_{conv} = \text{round}\left(\log_2\left(T\right)\right) - 3, \tag{7}$$

where T is the maximum number of time steps in the input patterns. This formula was used to ensure that the length of the final feature maps before the fully-connected layers was reduced by max pooling to a range between 5 and 12 time steps. Using Eq. (7), the models in the experiments had between 2 and 8 convolutional and pooling layers. As for the number of nodes, the first convolutional layer was set to 64 nodes, the second was set to 128 nodes, and when applicable, the third or higher was set to 256 nodes. In addition, the fully-connected layers had 1,024 nodes with a dropout rate of 0.5. The fusion networks used parallel versions of these hyperparameters when needed. Rectified Linear Unit (ReLU) was used as the activation function for the hidden layers. It should be noted that other deep neural networks like VGG [36] were tested, however, the results were unsatisfactory and did not justify the extra parameters.

For the Unipen datasets, the models were trained for 100,000 iterations using batches of 100. The UCI and UCR datasets were trained for 50,000 iterations using batches of 32. The difference in the training regimens was due to the much larger size of the Unipen datasets when compared to the other datasets. Each of the models was trained using Adam optimizer [37] with an initial learning rate of 0.0001.

4.3. Comparison to state-of-the-art

4.3.1. Evaluation on Unipen online handwritten datasets

The experiments on the Unipen datasets are compared to nine state-of-the-art methods from literature, including both classical methods and neural network-based methods. For classical methods, the results from a Hidden Markov Model (HMM) based on statistical DTW (HMM CSDTW) [38], two SVMs, one that embeds DTW into the Gaussian kernel (SVM GDTW) [39] and one that uses piecewise polynomial functions (Inter. Kernel) [40], and the online scanning n-tuple classifier (OnSNT) [41] are reported. For neural network methods, we compare the results of the proposed method to a DTW neural network (DTW-NN) [42] which integrates DTW into a feed-forward neural network, a neuro-fuzzy system that uses stroke features in a neural network (FasArt) [43], a hybrid Kohonen-perceptron (KP) neural network (Fuzzy Rep. KP) [44], a CNN with dynamically aligned weights (CNN DWA) [45], and an LSTM [4, 45]. Finally, the results from Google's online handwriting recognition system (Google) [46] which uses a lattice encoding and beam search method is shown.

4.3.2. Evaluation on UCI Machine Learning Repository datasets

The UCI Machine Learning Repository and its datasets have many different past works tackling each dataset with specialized models. For the sake of brevity, only the state-of-the-art method found after a survey for each dataset is reported. For the ADL dataset, Iwana and Uchida [45] use CNN DWA. The state-of-the-art results for the Arabic dataset uses an HMM with the second-order derivatives of MFCC ($\Delta(\Delta\text{MFCC})$) [47]. There are many past works which use the Auslan dataset. However, most methods from literature only use subsets of the full dataset to limit the number of classes. De Rosa et al. [48], however, used six methods on the full set of 95 classes for the Auslan dataset. For the HillValley and HillValley w/ noise, we compare the proposed method to a neural network (NN) [49] and Random Bits Forest (RBF) [50], respectively.

4.3.3. Evaluation on UCR Time Series Classification Archive datasets

We surveyed many different methods including classic methods, ensemble methods, and deep learning methods. The methods include, the classic baseline methods of 1-NN Euclidean Distance (1-NN ED) [17], 1-NN DTW [17], and 1-NN DTW with the best warping window (1-NN DTW Best WW) [17]. The methods from literature include, Bag of SFA Symbols (BOSS) [51, 52], Complexity-Invariant Distance (CID) [53], Cross Recurrence Plots Compression Distance using MPEG 2 (CRPCD2) [54], DTW-distance Features (DTW_F) [30, 52], Learned Pattern Similarity (LPS) [55, 52], Learning Shapelets (LTS) [56], Move-Split-Merge (MSM) [57, 52], Proximity Forest (PF) [58], Time Series Bag of Features (TSBF) [59], and Time Series Forest (TSF) [60, 52]. Next, for ensemble methods, Elastic Ensemble (EE) [61, 52] is a combination of 11 1-NN classifiers using different distance measures, ST [62, 52] uses 8 shapelet transform methods, and flat Collective of Transformation-based Ensembles (COTE) [63, 52] uses 35 classifiers including multiple transforms in a single weighted ensemble. Finally, the results from four deep neural networks, Multi-scale CNN (MCNN) [64], Multi-Layer Perceptron (MLP) [8], Fully Convolutional Network (FCN) [8], and ResNet [8], are shown. In addition, an LSTM with two recurrent layers and two fully-connected layers was used as an RNN-based baseline. With exception to the number of recurrent layers, the LSTM had the

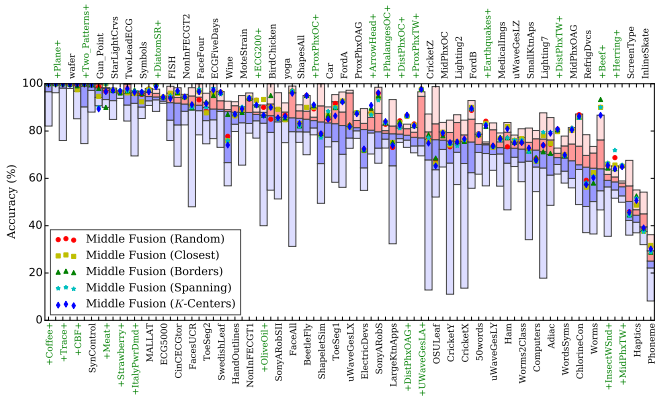

Figure 6: A comparison of baseline methods and the proposed method on 85 UCR Time Series Archive datasets sorted by median baseline accuracy and each dataset name is alternated on top and bottom. The bars display the range of methods from literature separated into quartiles. The markers are the accuracies from using the prototype selection methods with ($K = 5$) in a middle fusion CNN. The datasets highlighted in green with "+"s are the instances where at least one result had an equal or higher accuracy than the state-of-the-art for each dataset.

same hyperparameters and training regimen as the proposed method. It should be noted that there are many other methods in the literature that report results for the UCR Time Series Classification Archive datasets. We only list methods with the state-of-the-art result on at least one of the datasets.

4.4. Results

The results of using the five prototype selection methods described in Section 3.3 for the Unipen, UCI, and UCR datasets are shown in Tables 1, 2, and Fig. 6, respectively. A more detailed table of results for the UCR datasets is provided in the supplementary material. The results in the tables show the accuracies of using class-independent prototype selection with $K = 5$ in a middle fusion CNN which combines the coordinate features and the local distance features. The proposed methods are labeled Middle Fusion (Random), Middle Fusion (Closest), Middle Fusion (Borders), Middle Fusion (Spanning), and Middle Fusion (K-Centers) based on their respective prototype selection method.

Table 1 shows that the proposed method was able to achieve state-of-the-art results for the Unipen 1b and 1c datasets and was only surpassed by one classical method, OnSNT, and one modern method, Google. Furthermore, all of the prototype selection methods had higher accuracies compared to all of the other methods on the Unipen datasets aside from those two.

As for the results on the UCI datasets in Table 2, the proposed method performed exceptionally well. With exception to the Auslan dataset, the proposed method had perfect or near perfect accuracies which were large improvements over the state-of-the-art methods even though they were tailored to tackle their respective tasks. The Auslan dataset provides a difficult task for deep learning due to the dataset having 95 classes and only 70 patterns per class. This demonstrates a weakness of the proposed method as it does not perform as well with small datasets.

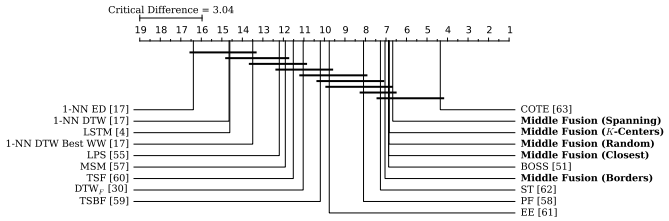

Figure 7: Average ranks on a critical difference diagram for methods using the 85 UCR time series datasets. The horizontal line segments indicate methods within one Critical Difference of each other (Eq. (8)). The bold methods are the proposed middle fusion CNNs with the prototype selection method in parenthesis and where $K = 5$.

The UCR datasets were also difficult for a similar reason. Most of the UCR datasets have very small training sets, which is generally not ideal for CNN and deep learning solutions. For that reason, when compared to the many comparative methods from literature, there were many individual successes but there were many times when one of the many comparison methods outperformed the proposed method.

Figure 7 is the critical difference diagram for 12 comparative methods and the proposed method. The 14 comparative methods used in Fig. 7 are the methods that report results on all 85 UCR time series datasets. In the diagram, the critical difference is defined by the Nemenyi post-hoc test [65] with $\alpha = 0.05$, or:

$$\text{Critical Difference} = q_\alpha \sqrt{\frac{R(R+1)}{6D}}, \tag{8}$$

where q_α is the Studentized range statistic for infinite degrees of freedom divided by $\sqrt{2}$, R is the number of classifiers, and D is the number of datasets. Critical difference diagrams is an established method for statistically comparing of classifiers over multiple datasets [66].

In addition, Fig. 6 and the supplementary material have results from 7 additional methods which only test on a selective subset of datasets. Notably, the figures show that the proposed method does well on average compared to most other methods and significantly outranks classic 1-NN DTW. Conversely, on average, COTE outperformed the proposed method, granted within one critical difference. However, while the results from COTE are good, it is "hugely computationally intensive" [52] and that "an algorithm that is faster than COTE but not significantly less accurate would be a genuine advance in the field," [52] as stated by the authors of COTE. The reason is that COTE gains its power from the collection of 35 different classifiers, including 11 1-NNs classifiers with different distance measures, 8 classifiers from Shapelet Transform (ST) [62], 8 using power spectrums, and 8 using auto-correlation functions.

5. Discussion

An ablation study was done to demonstrate the effects the prototypes and the fusion method had on the proposed model when classifying the Unipen datasets. Experiments were done by training the fusion methods with local distance feature sequences $\mathbf{v}$ with $K = 1$ to $K = 26$ prototypes for class-

Table 1: Comparisons on Unipen datasets

Method	Unipen Acc. (%) 1a	1b	1c
Middle Fusion ($K = 5$)			
Random	98.77±0.33	97.68±0.34	97.09±0.41
Closest	98.82±0.38	98.01±0.32	**97.37±0.42**
Borders	98.80±0.35	97.99±0.36	97.34±0.50
Spanning	98.87±0.41	**98.07±0.30**	97.20±0.38
K-Centers	98.76±0.36	98.01±0.48	97.21±0.35
Coord. features only	98.57±0.66	97.74±0.36	97.16±0.39
CNN DWA [45]	98.5	96.1	95.9
DTW-NN [42]	96.8	–	–
FasArt [43]	85.7	–	–
Fuzzy Rep. KP [44]	96.1	–	–
Google [46]	**99.2**	96.9	94.9
HMM CSDTW [38]	97.1	92.8	90.7
Inter. Kernel [40]	97.1	91.5	89.2
LSTM [45]	96.8	92.3	89.8
OnSNT [41]	98.9	95.7	92.1
SVM GDTW [39]	96.2	92.4	87.9

Table 2: Results on UCI Machine Learning Repository datasets

Dataset	S.o.t.A. Acc. (%)		Middle Fusion ($K = 5$) Acc. (%) Random	Closest	Borders	Spanning	K-Centers
ADL	90.0	CNN DWA [45]	97.4±1.7	97.6±1.3	**97.6±1.8**	**97.6±1.8**	97.3±1.3
Arabic	98.4	Δ(ΔMFCC) [47]	98.3	97.9	98.1	**98.4**	98.3
Auslan	**83.81**	DTW-d [48]	70.2±1.2	69.9±1.5	72.3±1.3	70.2±1.2	69.0±1.8
HillValley	**100**	NN [49]	**100**	**100**	**100**	**100**	**100**
HillValley w/ noise	97.5	RBF [50]	**100**	**100**	**100**	**100**	**100**
Average	92.12		93.18	93.08	**93.60**	93.24	92.92

independent selection and 1× to 10× the number of classes for both class-independent and classwise prototype selection. This was repeated for the three Unipen datasets and done for 10-fold cross-validation.

5.1. Effects of the number of prototypes

Intuitively, the more prototypes there are, the more information is embedded inside each local distance feature sequence, and therefore the more that can be learned. The intuition is confirmed by Fig. 8 where the accuracy drastically improves within the first few prototypes. However, there are severe diminishing returns on accuracy when using large numbers of prototypes. Despite using ten times the number of classes, i.e. 100 for 1a and 260 for 1b and 1c, there is very little improvement in the accuracy. Consequently, it is not worth increasing the computational time and complexity of the model beyond a minimal number of prototypes selected. Thus, the results from Tables 1, 2, and Fig. 6 were taken using only 5 prototypes, despite the ability to report higher accuracies using more prototypes.

5.2. Effects of the prototype selection method

One of the most important factors to be studied is the differences between the prototype selection methods. Similar to the effect of the number of prototypes, the accuracy of the prototype selection method is correlated to the variation in the prototypes selected. Methods that select an even distribution of the data tend to better than the methods which select prototypes from a narrow range of prototypes.

Classwise versus class-independent. Since Fig. 8 shows that there are severe diminishing returns on the accuracy when increasing the prototype set size and the minimum number of prototypes for classwise selection is fixed on the number of classes, the class-independent selection is more relevant for analysis. The results of classwise selection for the fusion network methods can be found in the supplementary material.

Closest prototype selection. Class-independent Closest as mentioned previously, is an intuitively poor prototype selection method and this is reflected in Fig. 8 (a). Across each dataset, with exception to $K = 1$, Closest performed the worst. The one exception of $K = 1$ is due to the center prototype being the average representation of the dataset. However, as the other prototype selection methods add additional prototypes, the advantage is lost because Closest continues to select the prototype from the central region of the dataset.

Borders prototype selection. As for the rest of the prototype selection methods, the difference is more subtle. Naively, Borders would have the largest variation due to Borders selecting the training patterns that are the furthest from all of the other points as described in Eq. (5). However, when visualizing one of the training sets for Unipen 1a using metric multidimensional

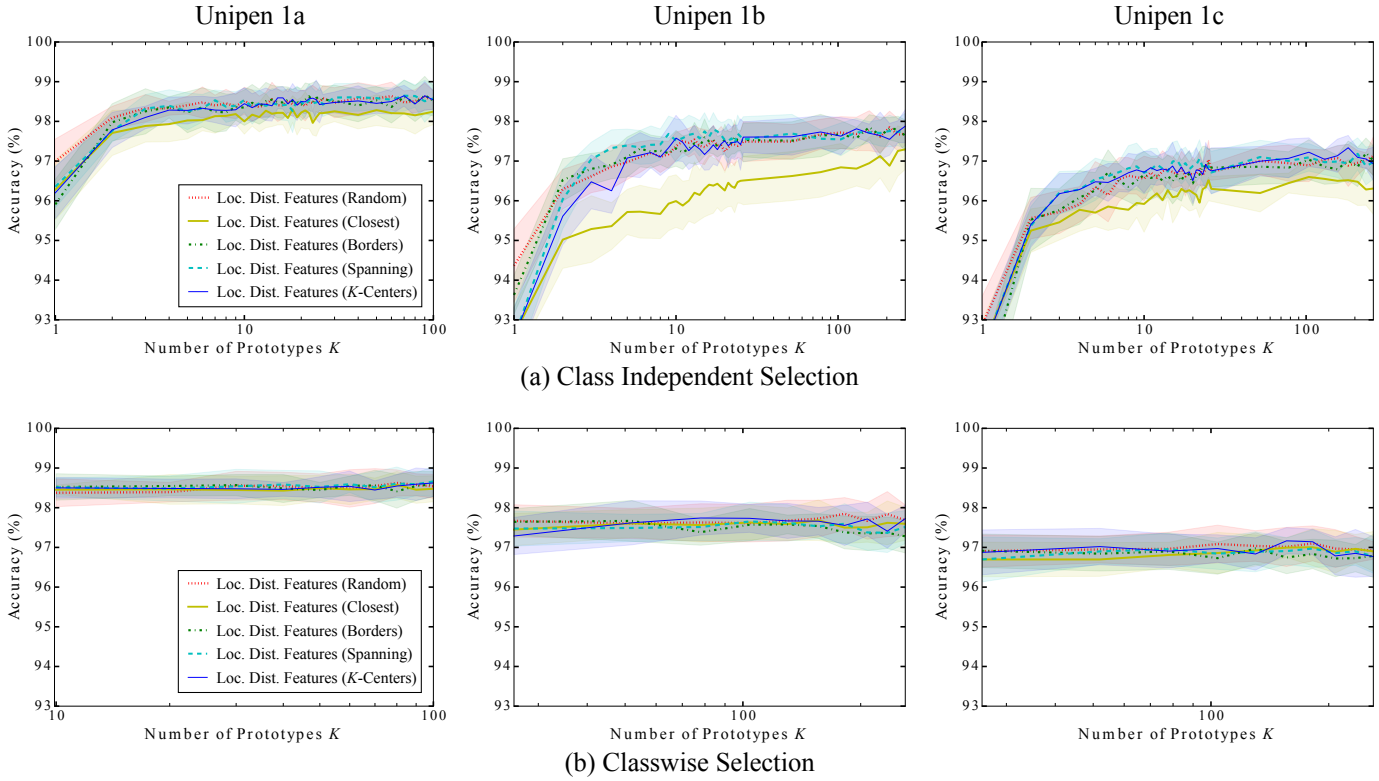

Figure 8: The test results on Unipen datasets using a CNN with only the local distance features. The lines represent the average of 10-fold cross-validation using the various prototype selection methods and the margin is the standard deviation above and below.

scaling (MDS) with Principal Component Analysis (PCA) using DTW as the distance measure, it can be seen in Fig. 5 that this is not always the case. The figure shows that while the selected border patterns are furthest from the other points in the dataset, they can be similar to each other.

Spanning prototype selection. Spanning takes an opposite approach to finding the furthest prototypes than Borders. Instead of determining the farthest from the dataset, Spanning finds the patterns that are the furthest from the already selected prototypes. This means that the Spanning prototypes are selected to be span across the entire training dataset leading to a representation of very different patterns. In general, the results in Fig. 8 demonstrate that Spanning is the best approach to selecting prototypes for local distance feature-based classification.

K-Centers prototype selection. Similar to Spanning, *K*-Centers prototype selection finds a good distribution of prototypes due to using the medoids of clusters. However, contrary to intuition, *K*-Centers generally did not outperform Spanning. This indicates that the center pattern of clusters is not as advantageous for local distance features as spanning the dataset evenly.

Random prototype selection. While Random selection was only included as a baseline for the other methods, the results show that selecting prototypes at random tended to do well. One reason for this phenomenon could be attributed to Random selecting patterns in a Gaussian distribution and therefore the selecting patterns with a good representation of the dataset. For the dataset in Fig. 5, the distribution of prototypes for Random and *K*-Centers appears about the same, and the results from all the datasets reflect that.

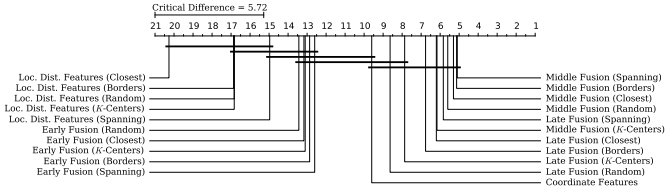

Figure 9: Critical difference diagram for the Unipen 1a, 1b, and 1c datasets comparing the fusion and selection methods (where $K = 5$). The average rank is calculated using 10-fold cross-validation with each training and test set the same for each comparison.

5.3. *Effects of the fusion method*

The model design choice with the most impact is the timing of the fusion method. From Fig. 9, fusing the CNNs at the feature-level generally has the best results. Also, the critical difference diagram shows that unlike when using the local distance features alone, there is not a significant difference between the prototype selection methods when using a fusion network.

Furthermore, in Fig. 10, it can be seen that increasing the number of prototypes, even up to ten times the number of classes, for the middle and late fusion networks had little effect on the accuracy. This shows that even with a very small number of prototypes, the local distance features contribute information

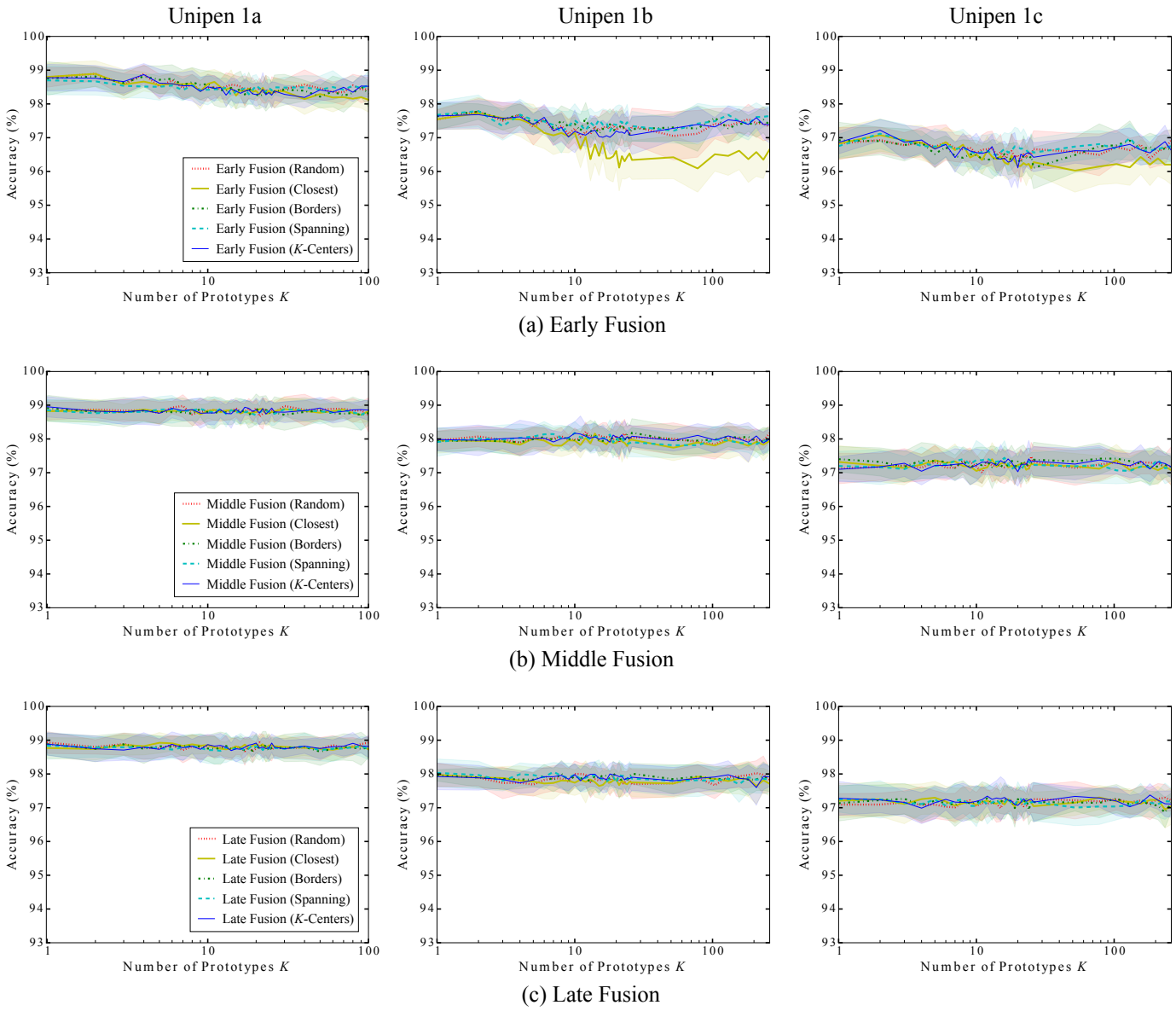

Figure 10: The test results for the three multi-modal fusion schemes. The lines represent the average of 10-fold cross-validation using the various prototype selection methods done class-independently and the margin is the standard deviation above and below.

for the multi-modal networks over the coordinate features alone (Table 1).

The early fusion accuracy graphs in Fig. 10 (a) is particularly interesting. The three datasets have a similar trend in that they have initially decreasing accuracies until they converge on the rising local distance feature only accuracies of Fig. 8 (a). The reason for this is due to the imbalance of data dimension when fused at the data level. For the Unipen datasets, the original coordinate features represent the two spatial dimensions of a plane. When K is small, the number of dimensions of the two features is balanced. However, as K grows large, the local distance features overshadow and even interfere with the coordinate features causing a poor accuracy. This is notable because by fusing the modalities later in the CNN overcomes this problem by having balanced feature representations at the fusion.

In addition, Figs. 10 (b), (c), and Fig. 9 shows that there is only a small difference between the prototype selection methods. Figure 11 shows the misclassifications of one of the cross-validation folds for the Unipen 1b dataset (uppercase characters). The figure further enforces that the difference between the prototype selection methods is subtle. More often than not, incorrectly classified characters were missed by multiple methods. Nonetheless, the figure also shows that the middle fusion CNNs using the coordinate features and the local distance features had fewer misclassifications than using the coordinate features alone.

6. Conclusion

This paper proposed a method of using multi-modal CNNs with the local distance-based features extracted from the DTW global distance calculation. In particular, using the dynamic

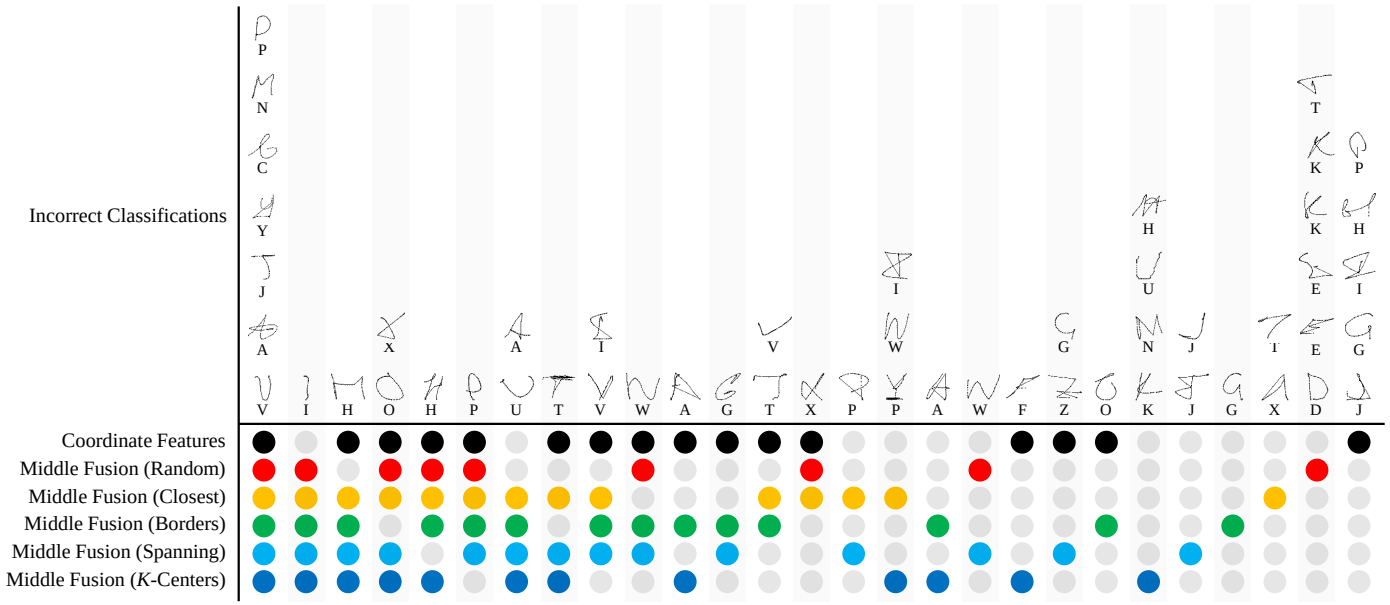

Figure 11: The misclassifications of using the coordinate features alone as well as the proposed method with different prototype selection methods for a common cross-validation fold on the Unipen 1b dataset. Each column shows the intersection of the misclassifications from the methods indicated with a colored circle.

element matching of DTW, the local distances between elements are used for time series classification. In this work, we demonstrated the effects that deliberate prototype selection has on those local distance features and their contribution toward classification. We used random selection, the closest prototypes to the center, the border prototypes, prototypes spanning the dataset, and the prototypes at the center of clusters found by *K*-Centers. These trials were also done with different numbers of prototypes and in a classwise and class-independent manner. We found that the prototype selection methods which used prototypes with a high variation tended to do better than the prototype methods which selected similar prototypes.

Many different datasets with different properties were used to evaluate the proposed method. This includes the Unipen 1a, 1b, and 1c online handwritten character datasets, four UCI Machine Learning Repository datasets with their variations, and 85 UCR Time Series Archive datasets. In the experiments, we show that it is possible to increase the accuracy of temporal CNNs by combining the local distance features with coordinate features in multi-modal CNNs. Furthermore, we found that using middle fusion, or feature-level fusion, generally performed better than early and late fusion. The experimental results on the other datasets show that the use of the proposed use of local distance features can generalize to many different types of time series.

Acknowledgment

This research was partially supported by MEXT-Japan (Grant No. J17H06100) and NTT Communication Science Laboratories.

References

[1] H. Ding, G. Trajcevski, P. Scheuermann, X. Wang, E. Keogh, Querying and mining of time series data, Proc. Very Larg. Data Base Endow. 1 (2) (2008) 1542–1552. doi:10.14778/1454159.1454226.

[2] J. Schmidhuber, Deep learning in neural networks: An overview, Neural Netw. 61 (2015) 85–117. doi:10.1016/j.neunet.2014.09.003.

[3] D. E. Rumelhart, G. E. Hinton, R. J. Williams, Learning representations by back-propagating errors, Nat. 323 (6088) (1986) 533–536. doi:10.1038/323533a0.

[4] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Comput. 9 (8) (1997) 1735–1780. doi:10.1162/neco.1997.9.8.1735.

[5] A. Graves, N. Jaitly, Towards end-to-end speech recognition with recurrent neural networks, in: Int. Conf. Mach. Learn., 2014, pp. 1764–1772.

[6] D. Bahdanau, K. Cho, Y. Bengio, Neural machine translation by jointly learning to align and translate, Comput. Res. Repos. abs/1409.0473. URL http://arxiv.org/abs/1409.0473

[7] Y. Qin, D. Song, H. Chen, W. Cheng, G. Jiang, G. W. Cottrell, A dual-stage attention-based recurrent neural network for time series prediction, in: Int. Joint Conf. Artif. Intell., 2017, pp. 2627–2633. doi:10.24963/ijcai.2017/366.

[8] Z. Wang, W. Yan, T. Oates, Time series classification from scratch with deep neural networks: A strong baseline, in: Int. Joint Conf. Neural Netw., 2017, pp. 1578–1585. doi:10.1109/ijcnn.2017.7966039.

[9] S. Bai, J. Z. Kolter, V. Koltun, An empirical evaluation of generic convolutional and recurrent networks for sequence modeling, Comput. Res. Repos. abs/1803.01271. URL http://arxiv.org/abs/1803.01271

[10] L. Wan, M. Zeiler, S. Zhang, Y. L. Cun, R. Fergus, Regularization of neural networks using dropconnect, in: Int. Conf. Mach. Learn., 2013, pp. 1058–1066.

[11] K. He, X. Zhang, S. Ren, J. Sun, Delving deep into rectifiers: Surpassing human-level performance on imagenet classification, in: IEEE Int. Conf. Comput. Vis., 2015, pp. 1026–1034. doi:10.1109/ICCV.2015.123.

[12] P. F. Felzenszwalb, R. Zabih, Dynamic programming and graph algorithms in computer vision, IEEE Trans. Pattern Anal. Mach. Intell. 33 (4) (2011) 721–740. doi:10.1109/tpami.2010.135.

[13] T. Rakthanmanon, B. Campana, A. Mueen, G. Batista, B. Westover, Q. Zhu, J. Zakaria, E. Keogh, Searching and mining trillions of time series subsequences under dynamic time warping, in: Int. Conf. Knowl. Discov. and Data Mining, 2012, pp. 262–270. doi:10.1145/2339530.2339576.

[14] L. Kaufman, P. J. Rousseeuw, Finding groups in data: An introduction to cluster analysis, Appl. Stat. 40 (3) (2009) 486. doi:10.2307/2347530.

[15] I. Guyon, L. Schomaker, R. Plamondon, M. Liberman, S. Janet, Unipen project of on-line data exchange and recognizer benchmarks, in: Int. Conf. Pattern Recogn., Vol. 2, 1994, pp. 29–33. doi:10.1109/icpr.1994.576870.

[16] D. Dheeru, E. Karra Taniskidou, UCI machine learning repository (2017). URL http://archive.ics.uci.edu/ml

[17] Y. Chen, E. Keogh, B. Hu, N. Begum, A. Bagnall, A. Mueen, G. Batista, The UCR time series classification archive, www.cs.ucr.edu/~eamonn/time_series_data/ (July 2015).

[18] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, K. J. Lang, Phoneme recognition using time-delay neural networks, in: Read. Speech Recog., Elsevier, 1990, pp. 393–404. doi:10.1016/b978-0-08-051584-7.50037-1.

[19] Y. Zheng, Q. Liu, E. Chen, Y. Ge, J. L. Zhao, Time series classification using multi-channels deep convolutional neural networks, in: Int. Conf. Web-Age Inform. Manag., 2014, pp. 298–310. doi:10.1007/978-3-319-08010-9_33.

[20] Y. Kim, Convolutional neural networks for sentence classification, in: Conf. Empir. Methods Nat. Lang. Process., 2014, pp. 1746–1751. doi:10.3115/v1/d14-1181.

[21] N. Razavian, D. Sontag, Temporal convolutional neural networks for diagnosis from lab tests, Comput. Res. Repos. abs/1511.07938. URL http://arxiv.org/abs/1511.07938

[22] Z. Wang, T. Oates, Encoding time series as images for visual inspection and classification using tiled convolutional neural networks, in: Assoc. Adv. Artif. Intell. Workshop, 2015, pp. 40–46.

[23] F. Yu, V. Koltun, Multi-scale context aggregation by dilated convolutions, Comput. Res. Repos. abs/1511.07122. URL http://arxiv.org/abs/1511.07122

[24] A. v. d. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, K. Kavukcuoglu, Wavenet: A generative model for raw audio, Comput. Res. Repos. abs/1609.03499. URL http://arxiv.org/abs/1609.03499

[25] T. Zahavy, A. Magnani, A. Krishnan, S. Mannor, Is a picture worth a thousand words? a deep multi-modal fusion architecture for product classification in e-commerce, Comput. Res. Repos. abs/1611.09534. URL http://arxiv.org/abs/1611.09534

[26] D. Wang, K. Mao, G.-W. Ng, Convolutional neural networks and multimodal fusion for text aided image classification, in: Int. Conf. Inform. Fusion, 2017, pp. 1–7. doi:10.23919/icif.2017.8009768.

[27] F. J. Ordóñez, D. Roggen, Deep convolutional and lstm recurrent neural networks for multimodal wearable activity recognition, Sens. 16 (1) (2016) 115. doi:10.3390/s16010115.

[28] X. Du, M. El-Khamy, J. Lee, L. Davis, Fused dnn: A deep neural network fusion approach to fast and robust pedestrian detection, in: IEEE Winter Conf. App. Comput. Vis., 2017, pp. 953–961. doi:10.1109/wacv.2017.111.

[29] D. Song, N. Xia, W. Cheng, H. Chen, D. Tao, Deep r-th root of rank supervised joint binary embedding for multivariate time series retrieval, in: ACM SIGKDD Int. Conf. Knowl. Discov. & Data Mining, 2018, pp. 2229–2238. doi:10.1145/3219819.3220108.

[30] R. J. Kate, Using dynamic time warping distances as features for improved time series classification, Data Mining and Knowl. Discov. 30 (2) (2016) 283–312. doi:10.1007/s10618-015-0418-x.

[31] B. K. Iwana, V. Frinken, K. Riesen, S. Uchida, Efficient temporal pattern recognition by means of dissimilarity space embedding with discriminative prototypes, Pattern Recogn. 64 (2017) 268–276. doi:10.1016/j.patcog.2016.11.013.

[32] B. K. Iwana, M. Mori, A. Kimura, S. Uchida, Introducing local distance-based features to temporal convolutional neural networks, in: Int. Conf. Front. Handw. Recogn., 2018, pp. 92–97. doi:10.1109/icfhr-2018.2018.00025.

[33] F. Itakura, Minimum prediction residual principle applied to speech recognition, IEEE Trans. Acoust., Speech, and Sig. Process. 23 (1) (1975) 67–72. doi:10.1109/tassp.1975.1162641.

[34] S. Garcia, J. Derrac, J. R. Cano, F. Herrera, Prototype selection for nearest neighbor classification: Taxonomy and empirical study, IEEE Trans. Pattern Anal. Mach. Intell. 34 (3) (2012) 417–435. doi:10.1109/tpami.2011.142.

[35] G. Forestier, F. Petitjean, H. A. Dau, G. I. Webb, E. Keogh, Generating synthetic time series to augment sparse datasets, in: IEEE Int. Conf. Data Mining, 2017, pp. 865–870. doi:10.1109/icdm.2017.106.

[36] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, Comput. Res. Repos. abs/1409.1556. URL http://arxiv.org/abs/1409.1556

[37] D. Kingma, J. Ba, Adam: A method for stochastic optimization, in: Int. Conf. Learn. Represent., 2015.

[38] C. Bahlmann, H. Burkhardt, The writer independent online handwriting recognition system frog on hand and cluster generative statistical dynamic time warping, IEEE Trans. Pattern Anal. Mach. Intell. 26 (3) (2004) 299–310. doi:10.1109/tpami.2004.1262308.

[39] C. Bahlmann, B. Haasdonk, H. Burkhardt, Online handwriting recognition with support vector machines-a kernel approach, in: Int. Workshop Front. Handwrit. Recogn., 2002, pp. 49–54. doi:10.1109/iwfhr.2002.1030883.

[40] K. Sivaramakrishnan, K. Karthik, C. Bhattacharyya, Kernels for large margin time-series classification, in: Int. Joint Conf. Neural Netw., 2007, pp. 2746–2751. doi:10.1109/ijcnn.2007.4371393.

[41] E. H. Ratzlaff, Methods, reports and survey for the comparison of diverse isolated character recognition results on the unipen database, in: Int. Conf. Doc. Anal. and Recogn., 2003, pp. 623–628. doi:10.1109/icdar.2003.1227737.

[42] B. K. Iwana, V. Frinken, S. Uchida, A robust dissimilarity-based neural network for temporal pattern recognition, in: Int. Conf. Front. Handwrit. Recogn., 2016, pp. 265–270. doi:10.1109/ICFHR.2016.0058.

[43] E. G. Sánchez, J. G. González, Y. A. Dimitriadis, J. C. Izquierdo, J. L. Coronado, Experimental study of a novel neuro-fuzzy system for on-line handwritten unipen digit recognition, Pattern Recogn. Lett. 19 (3) (1998) 357–364. doi:10.1016/s0167-8655(97)00178-5.

[44] J.-F. Hébert, M. Parizeau, N. Ghazzali, A new fuzzy geometric representation for online isolated character recognition, in: Int. Conf. Pattern Recogn., Vol. 2, 1998, pp. 1121–1123. doi:10.1109/icpr.1998.711891.

[45] B. K. Iwana, S. Uchida, Dynamic weight alignment for temporal convolutional neural networks, in: IEEE Int. Conf. Acoustics, Speech and Sig. Proc., 2019, pp. 3827–3831. doi:10.1109/icassp.2019.8682908.

[46] D. Keysers, T. Deselaers, H. A. Rowley, L.-L. Wang, V. Carbune, Multi-language online handwriting recognition, IEEE Trans. Pattern Anal. Mach. Intell. 39 (6) (2017) 1180–1194. doi:10.1109/tpami.2016.2572693.

[47] N. Hammami, M. Bedda, F. Nadir, The second-order derivatives of mfcc for improving spoken arabic digits recognition using tree distributions approximation model and HMMs, in: Int. Conf. Commun. and Inform. Tech., 2012, pp. 1–5. doi:10.1109/iccitechnol.2012.6285769.

[48] R. D. Rosa, N. Cesa-Bianchi, I. Gori, F. Cuzzolin, Online action recognition via nonparametric incremental learning, in: British Mach. Vis. Conf., 2014. doi:10.5244/c.28.113.

[49] Y. Wang, Y. Li, M. Xiong, Y. Y. Shugart, L. Jin, Random bits regression: a strong general predictor for big data, Big Data Anal. 1 (1) (2016) 12. doi:10.1186/s41044-016-0010-4.

[50] Y. Wang, Y. Li, W. Pu, K. Wen, Y. Y. Shugart, M. Xiong, L. Jin, Random bits forest: a strong classifier/regressor for big data, Scientific Rep. 6 (1). doi:10.1038/srep30086.

[51] P. Schäfer, The BOSS is concerned with time series classification in the presence of noise, Data Mining and Knowl. Discov. 29 (6) (2015) 1505–1530. doi:10.1007/s10618-014-0377-7.

[52] A. Bagnall, J. Lines, A. Bostrom, J. Large, E. Keogh, The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances, Data Mining and Knowl. Discov. 31 (3) (2017) 606–660. doi:10.1007/s10618-016-0483-9.

[53] G. E. Batista, X. Wang, E. J. Keogh, A complexity-invariant distance measure for time series, in: SIAM Int. Conf. Data Mining, 2011, pp. 699–710. doi:10.1137/1.9781611972818.60.

[54] D. F. Silva, V. M. A. De Souza, G. E. A. P. A. Batista, Time series classification using compression distance of recurrence plots, in: IEEE Int. Conf. Data Mining, 2013, pp. 687–696. doi:10.1109/icdm.2013.128.

[55] M. G. Baydogan, G. Runger, Time series representation and similarity based on local autopatterns, Data Mining and Knowl. Discov. 30 (2) (2016) 476–509. doi:10.1007/s10618-015-0425-y.

[56] J. Grabocka, N. Schilling, M. Wistuba, L. Schmidt-Thieme, Learning

time-series shapelets, in: ACM SIGKDD Int. Conf. Knowl. Discov. and Data Mining, 2014, pp. 392–401. doi:10.1145/2623330.2623613.

[57] A. Stefan, V. Athitsos, G. Das, The move-split-merge metric for time series, IEEE Trans. Knowl. and Data Eng. 25 (6) (2013) 1425–1438. doi:10.1109/tkde.2012.88.

[58] B. Lucas, A. Shifaz, C. Pelletier, L. O'Neill, N. Zaidi, B. Goethals, F. Petitjean, G. I. Webb, Proximity forest: an effective and scalable distance-based classifier for time series, Data Mining and Knowledge. Discov. 33 (3) (2019) 607–635. doi:10.1007/s10618-019-00617-3.

[59] M. G. Baydogan, G. Runger, E. Tuv, A bag-of-features framework to classify time series, IEEE Trans. Pattern Anal. and Mach. Intell. 35 (11) (2013) 2796–2802. doi:10.1109/tpami.2013.72.

[60] H. Deng, G. Runger, E. Tuv, M. Vladimir, A time series forest for classification and feature extraction, Inform. Sci. 239 (2013) 142–153. doi:10.1016/j.ins.2013.02.030.

[61] J. Lines, A. Bagnall, Time series classification with ensembles of elastic distance measures, Data Mining and Knowl. Discov. 29 (3) (2015) 565–592. doi:10.1007/s10618-014-0361-2.

[62] J. Lines, L. M. Davis, J. Hills, A. Bagnall, A shapelet transform for time series classification, in: ACM SIGKDD Int. Conf. Knowl. Discov. and Data Mining, 2012, pp. 289–297. doi:10.1145/2339530.2339579.

[63] A. Bagnall, J. Lines, J. Hills, A. Bostrom, Time-series classification with COTE: the collective of transformation-based ensembles, IEEE Trans. Knowl. and Data Eng. 27 (9) (2015) 2522–2535. doi:10.1109/tkde.2015.2416723.

[64] Z. Cui, W. Chen, Y. Chen, Multi-scale convolutional neural networks for time series classification, Comput. Res. Repos. abs/1603.06995.
URL http://arxiv.org/abs/1603.06995

[65] P. B. Nemenyi, Distribution-free multiple comparisons, Ph.D. thesis, Princeton University (1963).

[66] J. Demšar, Statistical comparisons of classifiers over multiple data sets, J. Mach. Learn. Res. 7 (2006) 1–30.