DTW-NN: A novel neural network for time series recognition using dynamic alignment between inputs and weights

Brian Kenji Iwana[a,*], Volkmar Frinken[b], Seiichi Uchida[a]

[a] *Department of Advanced Information Technology, Kyushu University, Fukuoka, Japan*
[b] *Onai Technology, Inc, San Jose, CA, USA*

Abstract

This paper describes a novel model for time series recognition called a Dynamic Time Warping Neural Network (DTW-NN). DTW-NN is a feedforward neural network that exploits the elastic matching ability of DTW to dynamically align the inputs of a layer to the weights. This weight alignment replaces the standard dot product within a neuron with DTW. In this way, the DTW-NN is able to tackle difficulties with time series recognition such as temporal distortions and variable pattern length within a feedforward architecture. We demonstrate the effectiveness of DTW-NNs on four distinct datasets: online handwritten characters, accelerometer-based active daily life activities, spoken Arabic numeral Mel-Frequency Cepstrum Coefficients (MFCC), and one-dimensional centroid-radii sequences from leaf shapes. We show that the proposed method is an effective general approach to temporal pattern learning by achieving state-of-the-art results on these datasets.

Keywords: Neural networks, dynamic time warping, temporal kernel, time series, dynamic programming

1. Introduction

A time series is a sequence of data ordered by time. The difficulty of time series recognition is that structural components and time dependencies need to be considered [1]. Thus, a robust time series recognition method would need to address these qualities as well as adjust for temporal distortions such as rate, time delay, and variable length.

For example, using distance-based models with Dynamic Time Warping (DTW) is a well-established method for time series classification [2]. In these methods, DTW is used as a global distance measure between time series which functions by using the sum of the distances between optimally matched elements. The elements are matched using the minimal path on an element-wise cost matrix estimated by dynamic programming. The combination of DTW and the nearest neighbor rule has shown to be successful for many time series recognition tasks [3, 4, 5]. However, with increasingly large datasets, exhaustive search methods begin to have increasingly large computational requirements. Consequently, these methods are not suitable for tasks with large amounts of data or tasks that need to be executed in reasonable amounts of time [5].

Recently, artificial neural networks (ANN) have been used successfully for applications in many domains [6], some of which include better than human performance in image classification [7], near-perfect accuracy in text [8], and having record-breaking results on established datasets [9, 10]. Besides their successes in pattern recognition, ANNs have been shown to be able to efficiently represent large amounts of data [11, 12].

However, when considering a time series, the successful feedforward neural networks such as Multi-Layer Perceptrons (MLP) and Convolutional Neural Networks (CNN) [13] do not specifically assess temporal factors. For example, time series with temporal distortions and non-uniform lengths cannot easily be processed by feedforward neural networks. Recurrent neural networks (RNN), on the other hand, are able to use temporal information by storing internal states within cyclic recurrent nodes. RNNs have been successful in many temporal pattern classification fields, such as speech [14, 15] and natural language processing [16]. While RNNs have been successful for time series recognition, they are complex, traditionally hard to train [17], and have very different structural properties compared to feedfoward networks.

This paper presents a feedforward model, called a Dynamic Time Warping Neural Network (DTW-NN) which is an adaptation of an MLP designed to tackle time series recognition. The core idea of a DTW-NN is to use *DTW nodes* to exploit DTW's ability to be flexible to temporal distortions in order to dynamically align the inputs to the weights. To accomplish this, we consider the weights of each DTW node as time series and replace the standard inner product of a node with DTW as a kernel-like method. Furthermore, the sequence-like weights are optimized through back propagation by using gradient descent on the DTW function. In this way, a complete feedforward neural network can be trained and used for the application of time series while still being invariant to temporal distortions.

The primary contributions of this paper are as follows:

- The proposal of DTW-NN, a novel feedforward neural net-

*Corresponding author
Email addresses: brian@human.ait.kyushu-u.ac.jp (Brian Kenji Iwana), volkmar@onai.com (Volkmar Frinken), uchida@ait.kyushu-u.ac.jp (Seiichi Uchida)

work that is robust to temporal distortions for time series recognition. This paper is the extension of a previous work [18].

- The description of a DTW node and the realization that the nodes of MLPs can be extended to other kernel-like methods in order to consider structural or other complex data types.

- The empirical evaluation of DTW-NNs on multiple datasets with varying characteristics including online handwritten characters, tri-axial accelerometer recordings of activities of daily life, spoken Arabic digit Mel-Frequency Cepstrum Coefficients (MFCC), and centroid-radii leaf shape outlines. We verify the proposed method's ability as an effective general solution for time series.

- Analysis of the characteristics of DTW-NN, the evaluated results, and comparative studies as well as an analysis of the trained DTW layer.

The remaining of this paper is organized as follows. A review of related work on time series recognition with neural networks is done in Section 2. Section 3 describes DTW and its relation to kernel functions. Section 4 details the proposed DTW-NN. In Section 5, we define the experiments, datasets, and discuss the results and performance of the proposed method. Finally, Section 6 draws the conclusion and proposes future work.

2. Related Work

The simplest application of neural networks to time series is to provide an MLP with subsequent elements in time. This technique has been heavily used in forecasting [19, 20, 21], where the network becomes a regression model with the output predicting future time steps given a sliding window of the input. However, a fully-connected approach like this does not maintain the temporal qualities of the time series and does not account for temporal distortions. For example, when considering an MLP, if patterns with temporal distortions are introduced, then the inputs of one pattern will not necessarily be aligned to the inputs of a different one.

One established method of handling time series data is the use of RNNs [22]. RNNs are a class of neural networks that store hidden states within recurrent nodes. In this way, each time step added to the network is dependent on the previous time steps through the recurrent connections. Long Short-Term Memory (LSTM) RNNs [23] extend RNNs by adding gates which control the update, output, and forgetting of the internal state in order to learn to circumvent the vanishing gradient problem and long-term dependencies [24]. They have been successful in many tasks such as handwriting [25, 26], speech [14], and can deal with natural language processing [16]. Gated Recurrent Units (GRU) [27] and Echo State Networks (ESN) [28] are more recent approaches to RNNs that have shown to have similar results to RNNs and LSTMs but with easier to train structures [29, 30, 31].

While recurrent models are normally used for time series, this paper addresses time series applications using feedforward neural networks. One approach to maintain the temporal information while using a feedforward neural network is a Time Delay Neural Network (TDNN) [32, 33]. Similar to the previous method, TDNNs input time steps into the network. To maintain temporal awareness, TDNNs invoke time-delay windows that tie multiple inputs to the subsequent hidden layer node. Currently, time-delays are handcrafted, yet Wöhler and Joachim [34] attempted to solve this using adaptive time-delays.

While CNNs have been successful with image data, there have also been approaches to adapt them for the use use of temporal pattern recognition. The convolutional kernels in one-dimensional (1D) CNNs are similar to the time-delays of TDNNs. Some examples of using CNNs with time series include splitting multivariate time series into channels of 1D subsequences [35] and embedding the data into 2D matrices [36, 37] with CNNs for classification. In another application, Zhang et al. [38] used an ESN with to create a 2D signal for convolutional layers in Spiking Echo State CNNs (SES CNN). For speech generation, WaveNet [39] uses 1D causal convolutional layers and uses dilation to increase the receptive field of the higher convolutions. Another use of using causal convolutional layers for is in Temporal Convolutional Networks (TCN) [40].

Other methods have been proposed to use neural networks based on features or characteristics of time series. Neural networks have been used in conjunction with Autoregressive Moving Average (ARMA) models [20, 41] and Autoregressive Integrated Moving Average (ARIMA) models [42]. Deep Belief Networks (DBN) have been used with Restricted Boltzman Machines (RBM) [43] and ARIMA models [42] for time series forecasting. Another example is using an MLP on wavelet transforms for forecasting and classification [44]. Extreme Learning Machines (ELM) have also been used for efficient time series forecasting [45]. Spiking neurons [46, 47] take advantage of brief spikes in data and time series to create sparse representations. While these are many uses of neural networks, they are commonly tied to specific domains of time series data, e.g. signals, speech, etc.

There have also been attempts to make use of dynamic aspects in neural networks. For example, Kalchbrenner et al. [48] use k-Max Pooling for sentence modeling. Deformable Convolutional Networks [49] use deformable convolutions to relax the constraints of a traditional convolution window. Recently, Yu et al. [50] introduced a model which combined GRUs with DTW. Furthermore, on a fundamental level, dynamic modeling is loosely related to the attention mechanism [51] in Transformers [52], the control in chaotic neural networks [53, 54, 55], and neural network pruning [56, 57].

In comparison to these methods, the proposed framework is a general solution for time series recognition using a feedforward network structure. DTW-NN was designed specifically to address difficulties of time series and is able to tackle these difficulties, such as temporal distortions, without manual parameterization.

3. Dynamic Time Warping as a Nonlinear Inner Product

The main idea behind the proposed method is to use DTW as a nonlinear inner product for time series and embed it into neural networks. DTW is an effective [4] method of estimating the optimal alignment between time series and sequence elements to measure a global distance between patterns. The proposed method exploits this matching for the alignment and the resulting DTW-distance for the output of DTW-based nodes.

3.1. Dynamic Time Warping

Given two discrete time series, a sequence $\mathbf{p} = p_1, \ldots, p_i, \ldots, p_I$ of length I and a sequence $\mathbf{s} = s_1, \ldots, s_j, \ldots, s_J$ of length J, where i and j are the index of each time step of sequence $\mathbf{p}$ and $\mathbf{s}$ respectively, the DTW-distance is the global summation of local distances between matched elements. Specifically, the DTW-distance can be denoted as:

$$\mathrm{DTW}(\mathbf{p}, \mathbf{s}) = \sum_{(i',j')\in\mathcal{M}} \left\| p_{i'} - s_{j'} \right\|, \tag{1}$$

where (i', j') is a pair of matched indices i' and j' corresponding to their original indices i of $\mathbf{p}$ and j of $\mathbf{s}$. $\mathcal{M}$ is the set of all matched pairs and $\|\cdot\|$ is a distance function between elements. The set $\mathcal{M}$ does not necessarily have a linear correspondence between each $1, \ldots, i, \ldots, I$ and $1, \ldots, j, \ldots, J$.

DTW uses dynamic programming to find estimated minimal warping path on an element-wise cost matrix given a cost function $\|p_i - s_j\|$. This warping path can be restricted by several constraints to prevent overfitting. For example, we enforce the asymmetric slope constraint [58] defined by the recurrence:

$$D(i, j) = \|p_i - s_j\| + \min_{j'\in\{j,j-1,j-2\}} D(i-1, j'), \tag{2}$$

where $D(i, j)$ is the cumulative minimum cost at the i-th and j-th element and the global DTW-distance is the value at the cumulative sum at $D(I, J)$. This particular slope constraint was selected to ensure that the number of local distances is always equal to the number of elements in $\mathbf{p}$ while still being flexible enough to overcome temporal distortions. It should be noted that the set of matched pairs $\mathcal{M}$ in Eq. (1) may contain repeated and skipped indices of i and j from the original sequences. This is possible due to the slope constraint defined in Eq. (2) and is not necessarily true of other slope constraints.

3.2. Relationship to Kernel Functions

Kernel functions are symmetric positive semi-definite functions that to map a feature space into vector space representations. With the application of a kernel function, data can be represented as a square matrix of pairwise comparisons. In this way, kernel functions can be seen as a similarity value between data points. For example, a linear kernel is the inner product of real vectors where parallel vectors would have a positive value and antiparallel vectors would have a negative value. While this example is useful for vector data, there are also kernels for other domains [59]. Some examples of popular kernel methods include SVMs [60, 61] with Radial Basis Function (RBF) kernels and kernel PCA [62].

In respect to kernel functions, DTW is also a positive semi-definite similarity function. However, unlike conventional kernels, DTW is asymmetric, where $\mathrm{DTW}(\mathbf{p}, \mathbf{s})$ does not necessarily equal $\mathrm{DTW}(\mathbf{s}, \mathbf{p})$. This is caused by the possibility that a different warping path is taken when using the slope constraint defined by Eq. (2). Despite this, DTW has shown to be effective as a similarity measure for Dissimilarity Space Embedding (DSE) [63, 64] as well as part of kernel functions for Support Vector Machines (SVM) [65, 66, 67].

4. Dynamic Time Warping Neural Network (DTW-NN)

MLPs, and feedforward neural networks in general, have layers of forward feeding nodes that are stacked in order to model nonlinear functions. The output of node n in a layer is:

$$a_n = \phi\left(\sum_i w_{n,i} x_i\right) = \phi\left(\mathbf{w}_n^{\mathrm{T}} \mathbf{x}\right), \tag{3}$$

where $\mathbf{x}$ is a vector of the input values $x_1, \ldots, x_i, \ldots, x_I$ and $\mathbf{w}_n$ is a vector of the respective weights $w_{n,1}, \ldots, w_{n,i}, \ldots, w_{n,I}$ to node n. The function $\phi(\cdot)$ is a nonlinear activation function applied to the result. The relationship between the weights and the inputs is much like a similarity function in that we can expect a larger activation a_n if the weight vector $\mathbf{w}_n$ and the input vector $\mathbf{x}$ are similar. In this way, the combination of weights and inputs is not unlike a linear kernel. However, this carries the assumption that each pairwise weight and input combination is linearly related. This may not be the case if the input is a time series with temporal distortions. The idea of the proposed method is to replace the inner product with a temporal similarity function, namely DTW.

4.1. DTW Layer Formulation

In order to tackle time series, we replace the inner product of a node with DTW as the kernel-like summation function. If we consider the weights as a sequence then during forward propagation, DTW can be used between the weights and the inputs of a node. Combining Eqs. (1) and (3), the activation of a node in becomes:

$$\begin{aligned} a_n &= \phi\left(\mathrm{DTW}\left(\mathbf{w}_n, \mathbf{x}\right)\right) \\ &= \phi\left(\sum_{(i',j')\in\mathcal{M}} \left\| w_{n,i'} - x_{j'} \right\|\right). \end{aligned} \tag{4}$$

This gives us a solution to optimize the alignment of each element in the weights $\mathbf{w}_n$ and inputs $\mathbf{x}$. For example, in Fig. 1, $w_{n,2}$, $w_{n,4}$, and $w_{n,4}$ are advanced one time step, causing two weights, $w_{n,1}$ and $w_{n,2}$, to be aligned with x_1. Also in this example, x_4 is skipped. This alignment is nonlinear is optimized in that the inputs $\mathbf{x}$ are warped to minimize the global distance between $\mathbf{w}_n$ and $\mathbf{x}$ given the constraints of DTW. Unlike the standard inner product of a fully-connected node, the DTW node

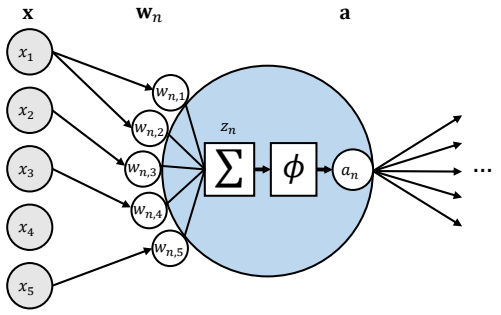


Figure 1: An example of a single DTW-NN node n. The input time series $\mathbf{x}$ and the weights $\mathbf{w}_n$ are aligned by DTW. The result of DTW is z_n and the output of the DTW node is a_n. Due to the dynamic alignment, x_1 corresponds with both $w_{n,1}$ and $w_{n,2}$, x_2 and x_3 are shifted one time step, and x_4 is skipped.

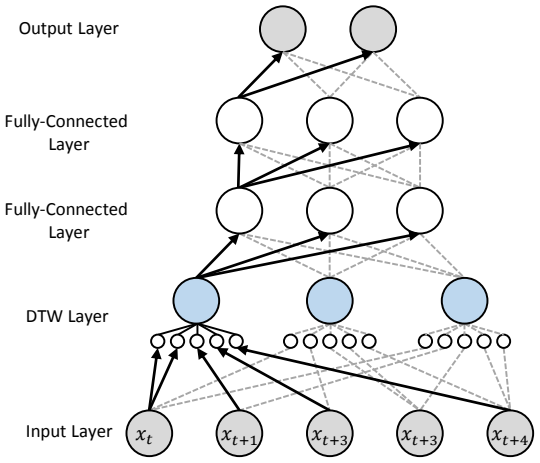


Figure 2: The proposed DTW-NN architecture with an input time series $\mathbf{x}$ with time step index t.

is able to accept inputs of different lengths. This is due to the dynamic alignment process and the slope constraint which allows for weights to skip inputs and connect to duplicate inputs. Thus, while the inputs can be different lengths, the output of each node is scalar.

This alignment process is performed on each forward propagation with the matches dynamically changing depending on each input sequence. In addition, similar to a standard MLP, the DTW layer can be *wide* with many parallel DTW nodes connected to the input. This way, DTW is calculated between the weights in each DTW node and each input sequence. The scalar activations of the DTW nodes are further used for subsequent fully-connected layers, as shown in Fig. 2.

4.2. Gradient of DTW

After each forward propagation, the gradient of a loss function with respect to the weights is back propagated through the network. This is commonly done using gradient descent in which the weights are moved negatively proportional to the derivative of the gradient. Namely, each weight is updated by:

$$w_{n,i} \leftarrow w_{n,i} - \eta \frac{\partial C}{\partial w_{n,i}}, \tag{5}$$

where C is the loss function and η is the learning rate.

To train DTW-NNs, we use the same back propagation principle to minimize the loss function. Similar to a traditional layer, the chain rule is used to find the gradient with respect to the weights in a DTW layer. For node n:

$$\frac{\partial C}{\partial w_{n,i}^{\mathrm{DTW}}} = \frac{\partial C}{\partial z_n^{\mathrm{DTW}}} \frac{\partial z_n^{\mathrm{DTW}}}{\partial w_{n,i}^{\mathrm{DTW}}}, \tag{6}$$

where the intermediate quantity z_n^{DTW} is the result of DTW. The partial derivative $\partial C / \partial z_n^{\mathrm{DTW}}$ is determined by back propagation and $\partial z_n^{\mathrm{DTW}} / \partial w_{n,i}^{\mathrm{DTW}}$ is the partial derivative of the output of DTW with respect to each weight $w_{n,i}^{\mathrm{DTW}}$. Under the DTW slope constraint defined in Eq. (2), each i' in $(i', j') \in \mathcal{M}$ directly corresponds to each i in $\mathbf{w}_n^{\mathrm{DTW}} = w_{n,1}^{\mathrm{DTW}}, \ldots, w_{n,i}^{\mathrm{DTW}}, \ldots, w_{n,I}^{\mathrm{DTW}}$. Thus, for each DTW node n, given an input sequence $\mathbf{x}$ and each dynamically matched input $x_{j'}$:

$$\begin{aligned} \frac{\partial z_n^{\mathrm{DTW}}}{\partial w_{n,i}^{\mathrm{DTW}}} &= \frac{\partial}{\partial w_{n,i}^{\mathrm{DTW}}} \mathrm{DTW}\left(\mathbf{w}_n^{\mathrm{DTW}}, \mathbf{x}\right) \\ &= \sum_{(i,j') \in \mathcal{M}} \frac{\partial}{\partial w_{n,i}^{\mathrm{DTW}}} \left\| \; w_{n,i}^{\mathrm{DTW}} - x_{j'} \; \right\|. \end{aligned} \tag{7}$$

Assuming a multivariate time series, each element of the input time series $x_{j'}$ consists of Q-dimensional vectors $(\chi_1, \ldots, \chi_Q)^{\mathrm{T}}$. The weight sequence $w_{n,i}^{\mathrm{DTW}}$ is denoted as $(\omega_1, \ldots, \omega_Q)^{\mathrm{T}}$ and is designed to be the same dimensionality as the input. To calculate the derivative of DTW, we find the partial derivative with respect to each dimension separately. Given the local distance function $\|\omega_q - \chi_q\|$ as the Euclidean distance, the partial derivative is:

$$\frac{\partial}{\partial (\omega_1, \ldots, \omega_Q)^{\mathrm{T}}} \left\| \begin{array}{c} \omega_1 - \chi_1 \\ \cdots \\ \omega_Q - \chi_Q \end{array} \right\| = \left(\begin{array}{c} \dfrac{\omega_1 - \chi_1}{\sqrt{\sum_{q'=1}^{Q} \left(\omega_{q'} - \chi_{q'}\right)^2}} \\ \cdots \\ \dfrac{\omega_Q - \chi_Q}{\sqrt{\sum_{q'=1}^{Q} \left(\omega_{q'} - \chi_{q'}\right)^2}} \end{array} \right). \tag{8}$$

The derivation of the Euclidean distance with respect to each coordinate dimension can be found in Appendix A.

Finally, by combining Eqs. (6), (7), and (8), we can formulate the gradient of the loss function in respect to the weights in a DTW node. The minimization of the loss with respect to the weights in a DTW node can be thought of as adjusting the coordinates of each element $w_{n,i}^{\mathrm{DTW}}$ in the weight sequence $\mathbf{w}_n^{\mathrm{DTW}}$ in order to obtain an optimal $\mathrm{DTW}\left(\mathbf{w}_n^{\mathrm{DTW}}, \mathbf{x}\right)$ result for the network.

4.3. Computational Complexity

The DTW algorithm itself is $O(IJ)$ where I and J are the number of elements in each sequence [68]. However, by using DTW with k-Nearest Neighbor (k-NN), the complexity of a test pattern becomes $O(XIJ + kX)$ where X is the number of patterns

Table 1: The Characteristics of the Datasets

Dataset	Dim.	Length	Classes	Features
Unipen 1a [69]	2	50	10	Pen Tip
Unipen 1b [69]	2	50	26	Pen Tip
Unipen 1c [69]	2	50	26	Pen Tip
ADL [70]	3	200	7	Acceleration
Spoken	13	40	10	MFCC
Arabic [71]	13	40	10	MFCC
Flavia [72]	1	100	32	Centroid-Radii

in the training set. This is due to k-NN requiring the calculation between every test pattern and every training pattern, followed by k nearest search.

In comparison, the test computational complexity of the proposed method is much lower. A forward pass of a fully-connected ANN with L number of layers is $O(LN_{\text{FC}}^2)$ assuming the number of nodes N_{FC} for each layer are of equal size. Adding a DTW layer requires a DTW calculation increasing the complexity to $O(LN_{\text{FC}}^2 + IJN_{\text{DTW}}N_{\text{FC}})$ where N_{DTW} is the number of nodes in the DTW layer. The number of nodes N is usually in the tens to hundreds and X could be many orders of magnitude larger. Thus, when X is large, $N \ll X$ and the complexity of a DTW-NN is much smaller than k-NN. However, it should be noted that k-NN requires no training time.

5. Experiments and Results

We demonstrate our method on four distinct datasets, the Unipen online isolated handwritten digits (Unipen 1a) and characters (Unipen 1b) and (Unipen 1c) [69], a dataset of activities of daily living (ADL) recognition with wrist-worn accelerometer data [70], a dataset containing MFCCs of spoken Arabic digits [71], and the Flavia leaf recognition dataset [72] using pseudo-time series based on centroid-radii. The datasets were chosen because of their availability and their varying time series characteristics, shown in Table 1.

5.1. Architecture Settings

5.1.1. Layer Hyperparameters

To evaluate the proposed method, we used a DTW-NN with the following hyperparameters. The first hidden layer is made of one DTW layer with 50 nodes. The next two are fully-connected layers containing 400 and 100 nodes respectively. The final output layer uses softmax with cross-entropy and with the appropriate number of nodes determined by the number of classes in the dataset. The batch size used in the experiments was dependent on the size of the training set; the Unipen datasets used a batch size of 100, the ADL dataset used a batch size of 5, and the Arabic digit and leaf shape datasets used a batch size of 50.

5.1.2. Weight Initialization

In the previous work, Iwana et al. [18] showed that there is little difference between initializing the DTW layer weights using a Gaussian distribution and initializing them using prototype patterns. Therefore, for simplicity, the weights between the input layer and the DTW were initialized using a Gaussian distribution with the same size, shape, and range as the input patterns of the respective datasets.

5.1.3. Learning Rate

We implemented a learning rate with a progressive $1/t$ decay defined by, $\eta_t = \frac{\eta_0}{1+\alpha t}$ where η_t is the learning rate at iteration t, η_0 is the initial learning rate, and α is the decay rate. For the training of the experiments, the proposed method used $\eta_0 = 0.001$ and $\alpha = 0.001$ for the DTW layer and a 0.0001 static learning rate for the fully-connected layers. The idea of using this progressive decay is to counter the slow convergence due to the previously defined weight initialization.

5.1.4. Activation Function

The result DTW is positive semi-definite with higher values equating to more dissimilar as opposed to similar in a linear kernel. So, to maintain similar qualities to a standard network for the output of a node, we adopt an inverted Rectified Linear Unit (ReLU) as $\phi(\cdot)$ in Eq. (3). We then employ batch normalization to scale the data and shift the mean to zero. Batch normalization has shown to increase performance by improving the network's generalization [73, 74]. As for the two fully-connected layers, the experiments use a hyperbolic tangent (tanh) activation function. This ensures that the output of the activation is bounded by -1 and 1.

5.1.5. DTW Slope Constraint

The slope constraint of DTW constrains the warping path of the dynamic programming calculation. For the experiments, we used the slope constraint as defined by Eq. (2). This slope constraint assures that the warping path is monotonic and guarantees that the matching of $\mathbf{p}$ always advances one step in time, therefore the number of matched elements is always equal to the number of elements I in $\mathbf{p}$. The advantage of a slope constraint with these characteristics is that a fixed-length $\mathbf{p}$ is useful for determining the similarity between sequences $\mathbf{s}$ of different lengths.

5.2. Comparison Methods

To illustrate the effectiveness of the proposed method, we compare the results of DTW-NN to established time series classification methods and results reported from literature for the respective datasets. For the six evaluated datasets, we use n-class classification, where n is equal to the number of classes available.

The datasets were split into three subsets, 90% for training, 10% for test, and 50 patterns set aside for validation. The one exception to this was the spoken Arabic dataset which contained a pre-defined speaker-independent training and test set split. Also, we use the same training, test, and validation sets for the 1-NN DTW and LSTM evaluations. The other reported results use their own defined divisions of the data. The results of the evaluations are shown in Table 2.

Table 2: Test Accuracy (%) Comparison for the Unipen 1a (Digits), 1b (Uppercase Letters), and 1c (Lowercase Letters) Datasets, the ADL Recognition with Wrist-worn Accelerometer Dataset, the Spoken Arabic Digit Dataset, and the Flavia Leaf Recognition Dataset

	Unipen			ADL	Arabic Digit	Flavia
Evaluation	1a	1b	1c	Accelerometer	MFCC	Leaf Shape
Proposed Method	98.2	94.9	94.5	**81.4**	96.1	79.5
1-NN w/ DTW	**98.5**	**97.1**	**95.8**	80.0	94.9	77.4
LSTM	96.5	92.1	90.5	80.0	96.0	81.6
DAG-SVM-GDTW [67]	96.2	92.4	87.9	-	-	-
HMM [75]	96.8	93.6	85.9	-	-	-
Fuzzy Rep. KP [76]	96.1	-	-	-	-	-
GMM + GMR [70]	-	-	-	63.1	-	-
Decision Tree [77]	-	-	-	80.9	-	-
Tree Distribution [78]	-	-	-	-	93.1	-
CHMM w/ $\Delta(\Delta$MFCC) [79]	-	-	-	-	**98.4**	-
WNN [80]	-	-	-	-	96.7	-
12-Feature PNN [72]	-	-	-	-	-	90.3*
SVM-BDT [81]	-	-	-	-	-	81.6
NFC [82]	-	-	-	-	-	50.2
ZM+HOG SVM [83]	-	-	-	-	-	**96.8***

* *Obtained using image features rather than time series features.*

5.2.1. Evaluated Methods

To evaluate the proposed method, we used the following two methods:

1-NN w/ DTW. A k-NN classifier where $k = 1$ and using a DTW as the distance measure is used as a comparison to our results. 1-NN w/ DTW is used because it is a standard baseline that outperforms many other state-of-the-art classifiers [3]. However, 1-NN w/ DTW is an exhaustive search and not a trained model, so testing was done by comparing each test pattern to the entire training set. Consequently, it is very computationally costly as outlined in Section 4.3.

LSTM. This evaluation uses the well-established neural network solution for variable length time series and sequence classification. For the experiments, we implemented an LSTM with one recursive hidden layers, a forget gate bias of 1.0, and two fully-connected layers. Each evaluation was given 50,000 iterations of the same batch size equal to the DTW-NN's evaluation. The experimental settings of the LSTM were set to be a fair comparison with similar complexity and depth as the proposed method.

5.2.2. Reported Methods

We compare our method with results reported from literature. The online handwritten character datasets are compared to an SVM using a Gaussian DTW kernel (DAG-SVM-GDTW) [67], a Hidden Markov Model (HMM) [75], and a Kohonen-Perceptron network using fuzzy vector representation (Fuzzy Rep. KP). For the ADL Accelerometer dataset, Bruno et al. [70] modeled the data using a Gaussian Mixture Modeling and Gaussian Mixture Regression (GMM + GMR) method and Kanna et al. [77] used a Decision Tree. The spoken Arabic digit comparisons from literature consist of a Tree Distribution model [78], a Continuous Hidden Markov Model of the second-order derivative MFCC (CHMM w/ $\Delta(\Delta$MFCC)) [79], and a Wavelet Neural Network (WNN) [80]. Finally, the Flavia leaf dataset is compared to the originally proposed Probabilistic Neural Network based on 12 digital morphological features (12-Feature PNN) [72], an SVM using a Binary Decision Tree (SVM-BDT) [81], Zernike Moment and Histogram of Oriented Gradients SVM [83], and the shape evaluation of a Neural Fuzzy Classifier (NFC) in [82]. Each comparative method was implemented using the same datasets as evaluated by the proposed method, however, the division in training and test sets may vary.

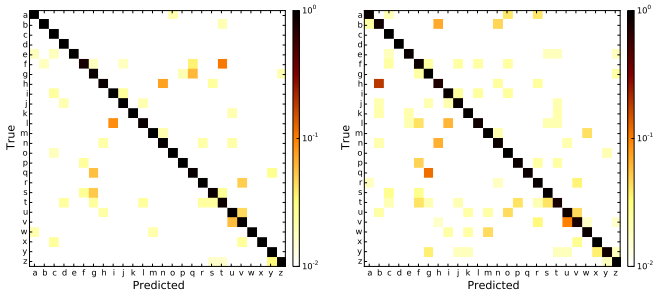
Many of the comparisons, such as CHMM w/ $\Delta(\Delta$MFCC) and WNN, are specific to a type of pattern. Also, the 12-Feature PNN and the ZM+HOG SVM evaluations are specific to leaf images because they use features specific to leaves such as vein features, physiological length and width, etc. In contrast to these methods, the proposed DTW-NN is a generic solution to many different types of time series patterns.

5.3. Isolated Online Handwritten Characters

Online handwriting is made of a sequence of coordinates extracted from the trajectory of a pen. In particular, isolated handwritten characters and digits are ideal time series to demonstrate temporal classification tasks due to having distinct classes despite having large intra-class variations.

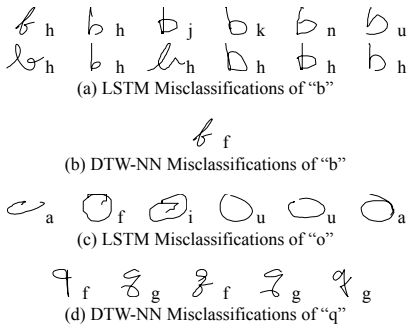
5.3.1. Dataset

The Unipen multi-writer 1a, 1b, and 1c datasets consist of trajectories of isolated digits, uppercase alphabet characters, and lowercase alphabet characters, respectively. The Unipen 1a dataset used was made of 13,000 digits evenly split into 10 classes representing the numerical digits "0" through "9." 1b and 1c datasets have 12,350 alphabetical characters in 26 classes. To ensure an equal distribution of each class, the datasets were reduced by randomly selecting patterns. Each



(a) DTW-NN (b) LSTM

Figure 3: Log scale confusion matrices for the test accuracy for Unipen 1c using the (a) proposed method and a (b) LSTM.



(a) LSTM Misclassifications of "b"

(b) DTW-NN Misclassifications of "b"

(c) LSTM Misclassifications of "o"

(d) DTW-NN Misclassifications of "q"

Figure 4: A comparison of the misclassified test results between DTW-NN and LSTM for the Unipen 1c (lowercase character) dataset. A rendering of each misclassification is shown with the class in which the pattern was classified as. (a) and (b) are all of the instances in the test set where "b" was predicted incorrectly by each model. (c) is LSTM's misclassifications of "o" and (d) is DTW-NN's misclassifications of "q." Conversely, DTW-NN did not incorrectly classify any "o"s and LSTM did not incorrectly classify any "q"s from the test set.

sequence trajectory was re-sampled to 50 elements and scaled to fit between $(0, 0)$ and $(1, 1)$.

5.3.2. Results

The results are reported in Table 2. Not unexpectedly, 1-NN w/ DTW performed extremely well on isolated handwritten characters. However, with over ten thousand training patterns, 1-NN w/ DTW is prohibitively slow. Using a desktop system with an Intel Xeon E5 2.6 GHz processor, 1-NN w/ DTW took an average of 77.6 seconds per classification. Comparatively, the proposed DTW-NN only requires DTW calculations with the 50 weight nodes. Using the same system with no GPU or neural network library only requires 0.2 seconds while still achieving 98.2%, 94.9%, and 94.5% accuracies for the Unipen 1a, 1b, and 1c datasets respectively. This indicates that DTW-NN can model the data almost as well as 1-NN w/ DTW but with a comparatively tiny amount of prototype patterns.

The confusion matrices in Fig. 3 compares the differences between DTW-NNs and LSTMs. The results of the DTW-NN in Fig. 3 (a) illustrate confusions between lowercase "t" and "f," "i" and "l," and "n" and "h". This indicates that the DTW-NN tends to confuse patterns with similar overall structures. However, LSTMs behave differently than DTW-NNs. For example, in Figs. 4 (a) and (b), DTW-NN correctly classified most of the "b" patterns which were difficult for the LSTM. This is because DTW computes the distance for the entire pattern, whereas the LSTM's recurrent nodes use discriminating features. The patterns in Fig. 4 (a) have the overall shape of a "b," so the DTW-NN correctly classified them, but the LSTM failed because the test patterns contain features shared by the other incorrect classes. Fig. 4 (c) and (d) further demonstrates the difference between the two network models. "o"s have features common to many classes such as the roundness of "a" or "u" which causes the LSTM to fail. On the other hand, the overall shape of an "o" is fairly distinct and DTW-NN succeeded in classifying all of the "o"s correctly. Conversely, lowercase "q"s often have the overall shape of a cursive "f" or "g" which causes DTW-NN to fail. This indicates that the proposed method was able to achieve a higher accuracy than the LSTM for the evaluated Unipen datasets because the spatial structure of the temporal patterns is more important for isolated characters and digits.

5.4. Accelerometer Based Activities of Daily Life

ADLs are activities that are basic tasks that are performed in normal life. The study of ADLs is useful for applications such as robotics, assisted ADL, and the monitoring of disabled or elderly patients [84].

5.4.1. Dataset

The dataset used for this experiment was collected by Bruno et al. [70] and it provides us with tri-axial acceleration-based data from ADLs. Bruno et al. collected the data using a wrist-mounted accelerometer recorded from 10 volunteers. The difficulty of using accelerometer data is that the patterns are very noisy and there is a high variation within each class. The dataset contains 705 recorded trials split into 7 classes: "Climbing stairs", "Drinking from a glass", "Getting up from bed", "Pouring water in a glass", "Sitting in a chair", "Standing from a chair", and "Walking." These activities represent various types of actions such as postural transitions, reiterated activities, and complex activities. They are recorded at 32 Hz with a range of $-1.5g$ to $1.5g$ where g is the acceleration due to gravity. For the experiment, they were sampled to 200 frames and scaled by $\frac{1}{1.5g}$. The original experiment in [70] uses a median filter to preprocess the data but this step is unnecessary for the proposed method.

5.4.2. Results

The proposed method had a test accuracy of 81.4%. This score is particularly good, considering that the training set only contained 585 patterns, which is small for most ANN models. Moreover, the proposed method outperformed all of the available benchmarks including 1-NN w/ DTW and LSTM, and far outperforming the method that was accompanied with the dataset, GMM + GMR with only a 63.1% accuracy. In addition, the results of the ADL experiment demonstrates the ability of the proposed method on noisy data.

Fig. 5 reveals that most of the proposed method's error comes from the confusion between two pairs of actions. One of the pairs, "Getting up from bed" and "Standing from a chair," are

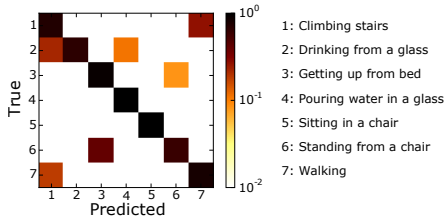
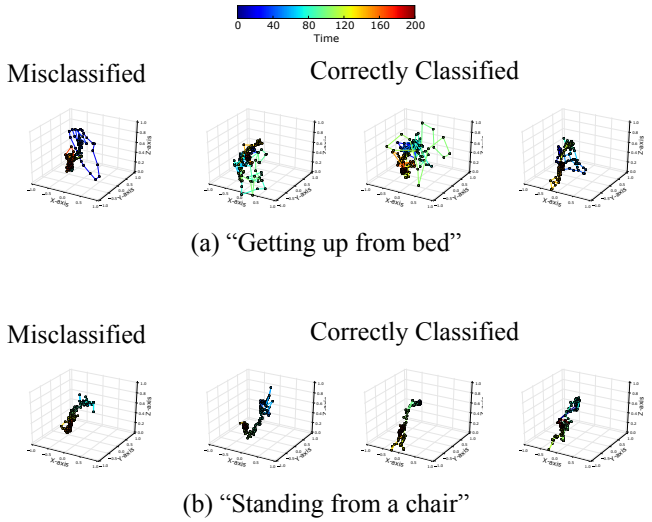


Figure 5: Log scale confusion matrix for ADL accelerometer test accuracy using DTW-NN.



(a) "Getting up from bed"

(b) "Standing from a chair"

Figure 6: (a) Patterns classified by DTW-NN as "Getting up from bed." The misclassified pattern is from the "Standing from a chair" class, but was classified as "Getting up from bed." (b) Patterns classified as "Standing from a chair" with the ground truth of the misclassified figure from "Getting up from bed."

postural transitions with similar movements. For example, the misclassified example in Fig. 6 (a) is very similar to the patterns labeled "Getting up from bed," but it belongs to the "Standing from a chair" class and the reverse is true about Fig. 6 (b). Another example is "Climbing stairs" and "Walking," which are both reiterated activities. Despite them being distinct activities, they tend to share the trait of having constant acceleration measurements over time. The confusion between these pairs is understandable and it further enforces that the DTW-NN is capable of classifying time series based on their overall structure.

5.5. Spoken Arabic Digits

Audio and speech are classic examples of time series patterns. MFCCs are calculated using Fourier transforms of the signal energy using filters at mel scale frequencies. Using MFCCs transforms signals into discrete time series features with the number of dimensions equaling the number of coefficients used.

5.5.1. Dataset

The spoken Arabic digit dataset is made up of 8,800 times series of 13-frequency MFCCs. The original samples were taken at a 11,025 Hz, 16-bit sampling rate, with a Hamming window, and with filters pre-emphasized $1 - 0.97Z^{-1}$. The samples were further resampled to 40 element sequences and scaled by $\frac{1}{15}$. Unlike the other evaluated datasets, the training set and test set division was provided by [71] and were made of 6,600 patterns and 2,200 speaker-independent patterns respectively.

5.5.2. Results

This evaluation shows that the proposed method is robust even with higher dimensional data. The accuracy of DTW-NN on the spoken Arabic digit MFCCs was 96.1% which is comparable to all of the evaluated methods. However, the wave specific feature representation methods, CHMM w/ Δ(ΔMFCC) and WNN, performed slightly better. On the other hand, this highlights the viability of DTW-NN as a general purpose solution to temporal pattern recognition.

5.6. Leaf Shapes

Plant leaf classification is a heavily explored field in pattern recognition. While many methods use texture features and image features, using shape features has the advantage of not being subject to problems with lighting and noise. In this way, leaf classification has been successfully used on various shape features, such as morphological shape features [85], ring projection [86], centroid-radii representations [87, 88], and hybrid shape-texture methods [89, 72].

5.6.1. Dataset

The Flavia leaf recognition dataset contains 1,907 isolated color leaf images. The dataset is composed of 50 to 72 leaves per class from 32 plant species. To classify leaf images using time series pattern recognition, we adopt a centroid-radii approach to convert the leaves into 1D pseudo-time series that describe the leaf's shape. The leaf images were preprocessed by using grayscale and then binarization. Next, the contour is found using a marching squares algorithm and the centroid is calculated. The time series is constructed taking the radius from the centroid to the outline starting from the lowest most point in 2D space and incrementing counterclockwise forming 100 evenly spaced sequence elements. The result is a 1D pseudo-time series representation of the leaf's shape. We normalized the 1D series so that the largest radius found in the dataset was scaled to 1. It should be noted that this method preprocessing into centroid-radii representations is rotation resistant but not entirely invariant depending on the location of the first time step on the leaf edge.

5.6.2. Results

The proposed method had a 79.5% accuracy which is comparable to the other shape only methods, 1-NN w/ DTW, LSTM, and SVM-DBT. However the best state-of-the-art method known to the authors, ZM+HOG SVM had a 96.8% accuracy using Zernike moments and histogram of oriented gradient features. Leaf classification using only the shape features is difficult because of the shape similarity of many classes. For example, Fig. 7 shows leaves from four classes and while the visual features of the leaves (e.g. texture, color, vein structure)

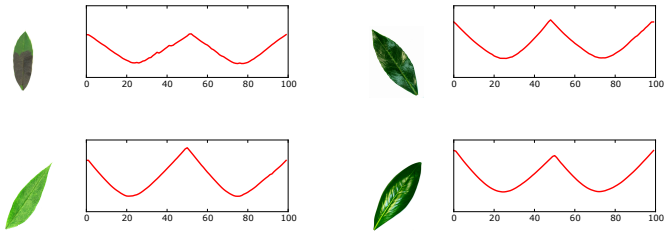


Figure 7: Leaves and their centroid-radii representations from four classes with similar shapes. (top right) *Berberis anhweienis*, (top left) *Cinnamomum camphora*, (bottom right) *Prunus persica*, and (bottom left) *Magnolia grandiflora*.

are distinct, the 1D shape representations are similar. For data with only small differences between classes, DTW traditionally performs poorly because noise often overshadows the differences between leaves [88]. Where DTW-NN excelled on patterns with large structural differences, such as handwriting, it suffered on data with subtle features such as leaf shapes.

5.7. Analysis of DTW Node Weights

The DTW node functions by aligning the inputs to the weights using DTW. This is possible because we consider the weights as a weight sequence rather than a weight vector. Furthermore, by treating the weights as if it were a time series, we can observe the training process. Fig. 8 follows the progression of a sample DTW node in each of the trained models. In each case, the weights are initialized by a Gaussian distribution, however, through many iterations, it can be observed that the weights converge to visible structures. For instance, the Unipen experiment's weights in Fig. 8 unraveled to form recognizable characters, such as a "6," an "R," and a "J", respectively.

The examples for the ADL, spoken Arabic digits, and leaf shape experiments also revealed clear structures despite being initialized at random. For example, Fig. 9 (c) has many weight sequences that are similar to the ones of Fig. 7, specifically, a line with two valleys. Another interesting observation is that the weights learned by the network start and end at a similar value which is consistent with a shape outline. Similarly, many patterns in Fig. 9 (a), (b), and (d) resemble patterns from the respective datasets.

However, Fig. 9 also shows that not all of the DTW nodes use weights that are recognizable shapes. The trained weight sequences converge into two forms, recognizable and unrecognizable structures. The recognizable structures are very similar to the input patterns, some of which resemble patterns from recognizable classes. Conversely, the unrecognizable structures appear to be noise. Although, it should be noted that even the DTW distance output of the nodes unrecognizable weight structures contribute to the proceeding fully-connected nodes. This means that the network not only uses the familiar generated patterns but also dissimilar patterns for discrimination.

5.8. Strengths and Limitations

Due to DTW-NNs being based on DTW, the accuracy of DTW-NN is dependent on the quality of DTW as a distance measure. Thus, the accuracy of the proposed method followed the accuracy of 1-NN w/ DTW closely. This is simultaneously a strength and a limitation. For example, 1-NN w/ DTW and DTW-NN had the most success in the Unipen experiments, but for the leaf shape experiment, the accuracy of both evaluations suffered due to similarly shaped classes. In addition, the close results of DTW-NN and 1-NN w/ DTW show that the proposed method is able to model the data despite using far fewer calculations. Had 1-NN w/ DTW been used with a prototype similar to the complexity of the 50 DTW-NN nodes, one would expect dramatically diminished results. This demonstrates that the proposed DTW-NN is able to generalize temporal patterns efficiently and is comparatively suitable for real-time applications.

On the other hand, the experiments showed that the LSTMs performed better when the differences between the classes were subtle. The trajectory-based datasets, such as the online handwriting and the ADL datasets are very noisy with a high variation within each class. The spoken Arabic and leaf shape datasets generally only have a few discriminating features that separate the classes. One reason for this observation could be because LSTMs use gating to emphasize learning from important time steps, but for many time series, no single point is more important than the overall structure. The advantage of a DTW-NN is the ability to have the power of an MLP while also having the robustness to overcome time distortions.

The evaluations also show that DTW-NN is competitive when compared to the domain specific methods. Another advantage of the proposed method over many of the other comparison methods is the ability to achieve a high accuracy with very little tuning and only rudimentary preprocessing. This proves that DTW-NN can be a general feedforward solution for time series, but there are some limitations to the proposed method.

5.8.1. Limitation 1: Time Series Shape

As mentioned previously, the advantage of implementing DTW as a weight alignment and the intermediate sum (Eq. (4)) is that the DTW-NN allows the network to overcome patterns of varying length and rate. However, because we consider the weight vector as a sequence for DTW, each element of the weight sequence should be fixed-dimensional equal to the input sequences. Also, given the slope constraint in Eq. (2), the weight sequence cannot be more than twice the size of the input sequence. This might lead to issues when the input time series length varies heavily. One way to overcome this is to use a slope constraint which is not subject to this limitation, such as the symmetric one proposed by Sakoe and Chiba [90]. However, using different constraints on the DTW warping path has advantages and disadvantages which would need to be chosen before training much like a hyperparameter.

5.8.2. Limitation 2: Weights as a Sequence

Another downside of considering the weights as a sequence is that the connections between inputs and weights are no longer simple multiplication functions and instead are distance functions. This means that there is no practical way to hand-craft time delays or zero out input sequence elements. In networks

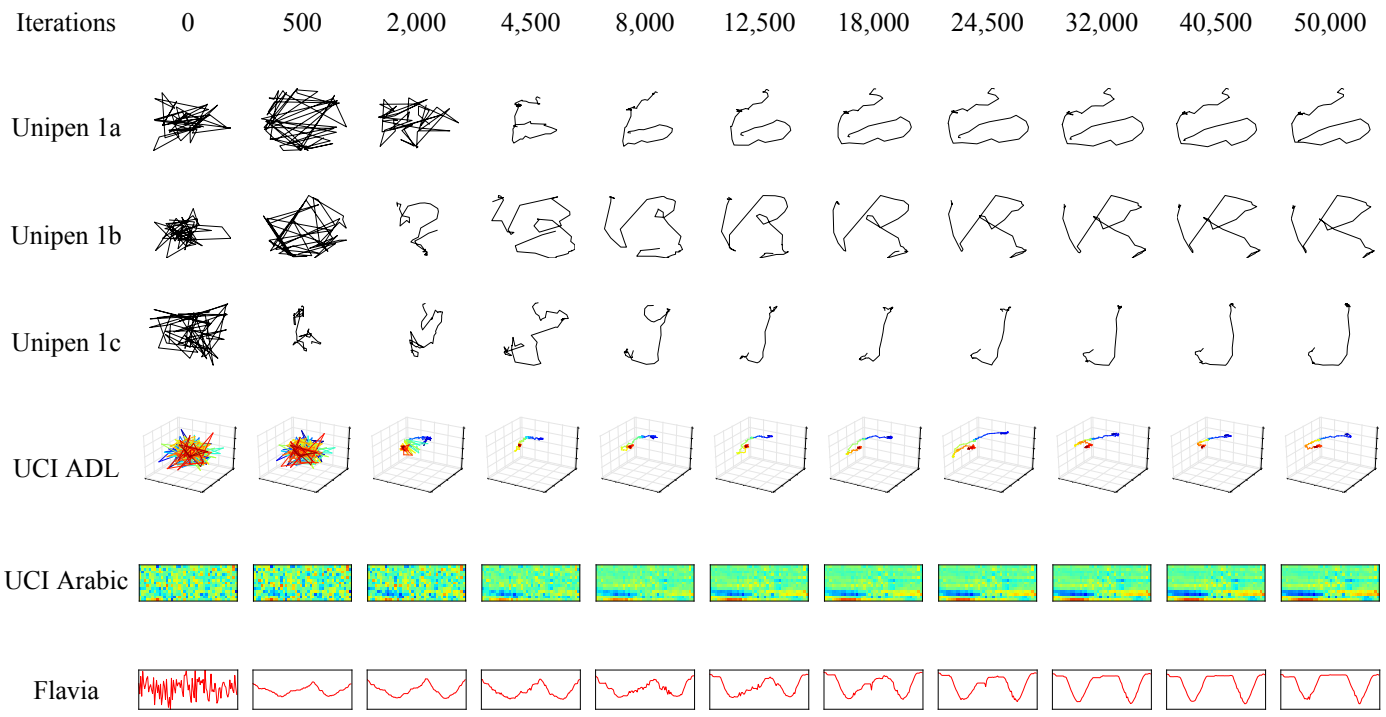


Figure 8: The progression of weights from a DTW node over the course of training.

such as fully-connected networks and RNNs, an element with zero in all dimensions would be ignored (but still advance time). Whereas in a DTW-NN, elements with zero in all dimensions is a valid sequence element with the possibility of a greater than zero distance to the weight sequence. However, a possible solution to overcome this limitation would be to ensure that the undesirable input element is skipped by the DTW slope constraint.

6. Conclusion

In this paper, we presented DTW-NN, a novel method of classifying discrete time series data. The proposed method is a temporally considerate feedforward ANN model which replaces the inner product of a neural node with DTW as a kernel-like function. Using DTW in this way allows the network to be robust to temporal distortions by aligning the weights to the optimal inputs. The result is a new feedforward network with automatic and dynamic time delays and advancements.

We evaluated the model on four different datasets: the Unipen online isolated handwritten digits (1a) and characters (1b and 1c), a dataset of ADL recognition with accelerometer data, MFCCs of spoken Arabic digits, and the Flavia leaf recognition dataset. Using these datasets, we were able to show that the proposed method works on a variety of time series types. First, with the handwritten digits dataset, we showed that DTW-NNs are a viable model and that it works well with structural data. Next, the ADL dataset demonstrated its ability with a small, noisy, and highly fluctuating dataset. The 13-dimensional spoken Arabic digits dataset proved that the DTW-NN can be used with multivariate data. And finally, the Flavia dataset showed that the proposed method can be used with shape outlines despite only having subtle feature differences between classes. The results of experiments demonstrate that the proposed method is a valuable general solution to time series recognition.

The concept of DTW layers can easily be extended to other feedforward neural networks. Furthermore, this paper lays the foundation for ANNs with DTW as a temporal kernel-like function and future work which can be done incorporating it into other applications. In addition, while the DTW-NN in this paper was achieved state-of-the-art results, there is room for improvement and future work can be put into further enhancing the network. Additional layers can be added, hyperparameters adjusted, and recent neural network optimization and generalization techniques can be applied. Another area of optimization would be the efficiency in the calculation of the DTW layer and the utilization of GPUs. Ongoing and future work will continue to enhance our method and approach other temporal domains.

Acknowledgment

This work was done while being supported by the Institute of Decision Science for a Sustainable Society and the Kyushu University Doctoral Scholarship.

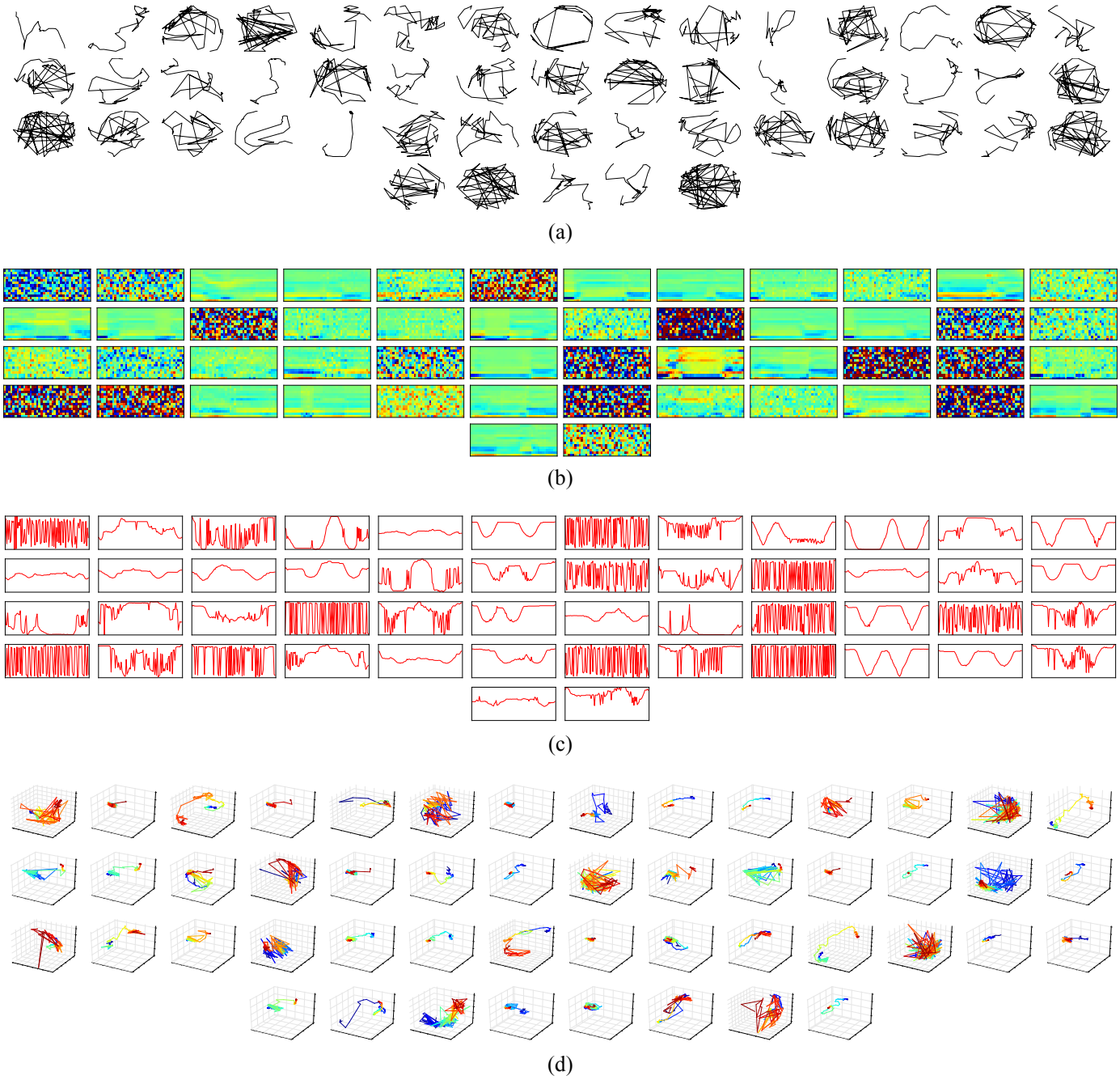


Figure 9: The state of the weights of each of the DTW nodes after 50,000 training iterations for the (a) Unipen 1c, the (b) spoken Arabic, the (c) Flavia leaf, and the (d) ADL accelerometer datasets.

Appendix A. The Derivative of Euclidean Distance

If $\|\cdot\|$ is denoted as the Euclidean distance, then the Euclidean distance between vectors $\mathbf{p}$ and $\mathbf{s}$ is:

$$\|\mathbf{p}-\mathbf{s}\| := \sqrt{\sum_{i=1}^{I}(p_i-s_i)^2}, \tag{A.1}$$

where I is the number of dimensions in the vector space. The derivative of $\|\mathbf{p}-\mathbf{s}\|$ with respect to one component of $\mathbf{p}$ is:

$$\begin{aligned}
\frac{\partial}{\partial p_i}\|\mathbf{p}-\mathbf{s}\| &= \frac{\partial}{\partial p_i}\sqrt{\sum_{i'=1}^{I}(p_{i'}-s_{i'})^2} \\
&= \frac{1}{2\sqrt{\sum_{i'=1}^{I}(p_{i'}-s_{i'})^2}}\frac{\partial}{\partial p_i}\sum_{i'=1}^{I}(p_{i'}-s_{i'})^2 \\
&= \frac{1}{2\sqrt{\sum_{i'=1}^{I}(p_{i'}-s_{i'})^2}} \\
&\qquad \left(\frac{\partial}{\partial p_i}(p_i-s_i)^2+\frac{\partial}{\partial p_i}\sum_{i'\neq i}(p_{i'}-s_{i'})^2\right) \\
&= \frac{1}{2\sqrt{\sum_{i'=1}^{I}(p_{i'}-s_{i'})^2}}2(p_i-s_i) \\
&= \frac{p_i-s_i}{\sqrt{\sum_{i'=1}^{I}(p_{i'}-s_{i'})^2}}.
\end{aligned} \tag{A.2}$$

This is repeated for all components of $\mathbf{p}$.

References

[1] Z. Xing, J. Pei, E. Keogh, A brief survey on sequence classification, ACM SIGKDD Explorations Newsletter 12 (1) (2010) 40. doi:10.1145/1882471.1882478.

[2] 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.

[3] X. Xi, E. Keogh, C. Shelton, L. Wei, C. A. Ratanamahatana, Fast time series classification using numerosity reduction, in: Int. Conf. Mach. Learning, 2006, pp. 1033–1040. doi:10.1145/1143844.1143974.

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

[5] 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. Discovery and Data Mining, 2012, pp. 262–270. doi:10.1145/2339530.2339576.

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

[7] H. Touvron, A. Vedaldi, M. Douze, H. Jégou, Fixing the train-test resolution discrepancy, CoRR abs/1906.06423.
URL http://arxiv.org/abs/1906.06423

[8] S. Uchida, S. Ide, B. K. Iwana, A. Zhu, A further step to perfect accuracy by training CNN with larger data, in: Int. Conf. Frontiers in Handwriting Recognition, 2016, pp. 405–410. doi:10.1109/ICFHR.2016.0082.

[9] K. Kowsari, M. Heidarysafa, D. E. Brown, K. J. Meimandi, L. E. Barnes, RMDL: Random multimodel deep learning for classification, in: Int. Conf. Inform. Sys. and Data Mining, 2018. doi:10.1145/3206098.3206111.

[10] Y. Huang, Y. Cheng, D. Chen, H. Lee, J. Ngiam, Q. V. Le, Z. Chen, GPipe: efficient training of giant neural networks using pipeline parallelism, CoRR abs/1811.06965.
URL http://arxiv.org/abs/1811.06965

[11] M. Banko, E. Brill, Scaling to very very large corpora for natural language disambiguation, in: Annu. Meeting Assoc. for Computational Linguistics, 2001, pp. 26–33. doi:10.3115/1073012.1073017.

[12] A. Krizhevsky, I. Sutskever, G. E. Hinton, Imagenet classification with deep convolutional neural networks, in: Advances in Neural Inform. Process. Syst., 2012, pp. 1097–1105.

[13] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proc. IEEE 86 (11) (1998) 2278–2324. doi:10.1109/5.726791.

[14] A. Graves, A.-R. Mohamed, G. Hinton, Speech recognition with deep recurrent neural networks, in: Int. Conf. Acoustics, Speech and Signal Process., 2013, pp. 6645–6649. doi:10.1109/icassp.2013.6638947.

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

[16] M. Sundermeyer, R. Schlüter, H. Ney, LSTM neural networks for language modeling, in: Interspeech, 2010, pp. 194–197.

[17] Y. Bengio, P. Simard, P. Frasconi, Learning long-term dependencies with gradient descent is difficult, IEEE Trans. Neural Networks 5 (2) (1994) 157–166. doi:10.1109/72.279181.

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

[19] G. Zhang, B. E. Patuwo, M. Y. Hu, Forecasting with artificial neural networks:: The state of the art, Int. J. Forecasting 14 (1) (1998) 35–62. doi:10.1016/S0169-2070(97)00044-7.

[20] H. B. Hwarng, H. T. Ang, A simple neural network for ARMA (p, q) time series, Omega 29 (4) (2001) 319–333. doi:10.1016/s0305-0483(01)00027-5.

[21] G. P. Zhang, Time series forecasting using a hybrid ARIMA and neural network model, Neurocomputing 50 (2003) 159–175. doi:10.1016/s0925-2312(01)00702-0.

[22] H. Jaeger, Tutorial on training recurrent neural networks, covering BPPT, RTRL, EKF and the "echo state network" approach, Tech. rep., German National Research Center for Inform. (2002).

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

[24] S. Hochreiter, Y. Bengio, P. Frasconi, J. Schmidhuber, Gradient flow in recurrent nets: the difficulty of learning long-term dependencies, Wiley-IEEE Press, 2001. doi:10.1109/9780470544037.ch14.

[25] A. Graves, M. Liwicki, S. Fernandez, R. Bertolami, H. Bunke, J. Schmidhuber, A novel connectionist system for unconstrained handwriting recognition, IEEE Trans. Pattern Anal. Mach. Intell. 31 (5) (2009) 855–868. doi:10.1109/tpami.2008.137.

[26] A. Graves, J. Schmidhuber, Offline handwriting recognition with multidimensional recurrent neural networks, in: Advances in Neural Inform. Process. Syst., 2009, pp. 545–552.

[27] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, Y. Bengio, Learning phrase representations using RNN encoder–decoder for statistical machine translation, in: Conf. Empirical Methods in Nat. Lang. Process., 2014, pp. 1724–1734. doi:10.3115/v1/d14-1179.

[28] H. Jaeger, H. Haas, Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication, Science 304 (5667) (2004) 78–80. doi:10.1126/science.1091277.

[29] J. Chung, C. Gulcehre, K. Cho, Y. Bengio, Empirical evaluation of gated recurrent neural networks on sequence modeling, CoRR abs/1412.3555.
URL http://arxiv.org/abs/1412.3555

[30] M. Čerňanský, P. Tiňo, Comparison of echo state networks with simple recurrent networks and variable-length markov models on symbolic sequences, in: Lecture Notes in Computer Science, Springer, 2007, pp. 618–627. doi:10.1007/978-3-540-74690-4_63.

[31] P. Tanisaro, G. Heidemann, Time series classification using time warping invariant echo state networks, in: IEEE Int. Conf. Mach. Learning and Appl., IEEE, 2016, pp. 831–836. doi:10.1109/icmla.2016.0149.

[32] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, K. J. Lang,

Phoneme recognition using time-delay neural networks, in: Readings in Speech Recog., Elsevier, 1990, pp. 393–404. doi:10.1016/b978-0-08-051584-7.50037-1.

[33] K. J. Lang, A. H. Waibel, G. E. Hinton, A time-delay neural network architecture for isolated word recognition, Neural Networks 3 (1) (1990) 23–43. doi:10.1016/0893-6080(90)90044-l.

[34] C. Wöhler, J. K. Anlauf, An adaptable time-delay neural-network algorithm for image sequence analysis, IEEE Trans. on Neural Networks 10 (6) (1999) 1531–1536. doi:10.1109/72.809100.

[35] 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. Management, 2014, pp. 298–310. doi:10.1007/978-3-319-08010-9_33.

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

[37] J. Yang, M. N. Nguyen, P. P. San, X. Li, S. Krishnaswamy, Deep convolutional neural networks on multichannel time series for human activity recognition, in: Int. Joint Conf. Artificial Intell., 2015, pp. 3995–4001.

[38] A. Zhang, W. Zhu, J. Li, Spiking echo state convolutional neural network for robust time series classification, IEEE Access (2018) 1–1doi:10.1109/access.2018.2887354.

[39] 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, CoRR abs/1609.03499.
URL http://arxiv.org/abs/1609.03499

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

[41] A. Ahadi, X. Liang, Wind speed time series predicted by neural network, in: IEEE Canadian Conf. Electrical & Computer Eng., 2018, pp. 264–269. doi:10.1109/ccece.2018.8447635.

[42] M. Qin, Z. Li, Z. Du, Red tide time series forecasting by combining ARIMA and deep belief network, Knowledge-Based Syst. 125 (2017) 39–52. doi:10.1016/j.knosys.2017.03.027.

[43] T. Kuremoto, S. Kimura, K. Kobayashi, M. Obayashi, Time series forecasting using a deep belief network with restricted boltzmann machines, Neurocomputing 137 (2014) 47–56. doi:10.1016/j.neucom.2013.03.047.

[44] A. K. Alexandridis, A. D. Zapranis, Wavelet neural networks: A practical guide, Neural Networks 42 (2013) 1–27. doi:10.1016/j.neunet.2013.01.008.

[45] G. Song, Q. Dai, A novel double deep ELMs ensemble system for time series forecasting, Knowledge-Based Syst. 134 (2017) 31–49. doi:10.1016/j.knosys.2017.07.014.

[46] W. Maass, Networks of spiking neurons: the third generation of neural network models, Neural Networks 10 (9) (1997) 1659–1671. doi:10.1016/s0893-6080(97)00011-7.

[47] R. Brette, M. Rudolph, T. Carnevale, M. Hines, D. Beeman, J. M. Bower, M. Diesmann, A. Morrison, P. H. Goodman, F. C. Harris, et al., Simulation of networks of spiking neurons: a review of tools and strategies, J. Computational Neuroscience 23 (3) (2007) 349–398. doi:10.1007/s10827-007-0038-6.

[48] N. Kalchbrenner, E. Grefenstette, P. Blunsom, A convolutional neural network for modelling sentences, in: Annu. Meeting Assoc. Computational Linguistics, 2014, pp. 655–665. doi:10.3115/v1/p14-1062.

[49] J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, Y. Wei, Deformable convolutional networks, in: IEEE Int. Conf. Computer Vision, 2017, pp. 764–773. doi:10.1109/iccv.2017.89.

[50] Z. Yu, Z. Niu, W. Tang, Q. Wu, Deep learning for daily peak load forecasting–a novel gated recurrent neural network combining dynamic time warping, IEEE Access 7 (2019) 17184–17194. doi:10.1109/access.2019.2895604.

[51] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, Y. Bengio, Show, attend and tell: Neural image caption generation with visual attention, in: Int. Conf. Mach. learning, 2015, pp. 2048–2057.

[52] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, I. Polosukhin, Attention is all you need, in: Advances in Neural Inform. Process. Sys., 2017, pp. 5998–6008.

[53] K. Shi, Y. Tang, S. Zhong, C. Yin, X. Huang, W. Wang, Nonfragile asynchronous control for uncertain chaotic lurie network systems with bernoulli stochastic process, Int. J. Robust and Nonlinear Control 28 (5) (2017) 1693–1714. doi:10.1002/rnc.3980.

[54] J. Wang, K. Shi, Q. Huang, S. Zhong, D. Zhang, Stochastic switched sampled-data control for synchronization of delayed chaotic neural networks with packet dropout, Applied Math. and Computation 335 (2018) 211–230. doi:10.1016/j.amc.2018.04.038.

[55] K. Shi, J. Wang, S. Zhong, X. Zhang, Y. Liu, J. Cheng, New reliable nonuniform sampling control for uncertain chaotic neural networks under markov switching topologies, Applied Math. and Computation 347 (2019) 169–193. doi:10.1016/j.amc.2018.11.011.

[56] J. Lin, Y. Rao, J. Lu, J. Zhou, Runtime neural pruning, in: Advances in Neural Inform. Process. Sys., 2017, pp. 2181–2191.

[57] A. Aghasi, A. Abdi, N. Nguyen, J. Romberg, Net-trim: Convex pruning of deep neural networks with performance guarantee, in: Advances in Neural Inform. Process. Sys., 2017, pp. 3177–3186.

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

[59] D. Haussler, Convolution kernels on discrete structures, Tech. rep., Depart. Comput. Sci., University of California Santa Cruz (1999).

[60] A. Aizerman, E. M. Braverman, L. Rozoner, Theoretical foundations of the potential function method in pattern recognition learning, Automation and Remote Control 25 (1964) 821–837.

[61] B. E. Boser, I. M. Guyon, V. N. Vapnik, A training algorithm for optimal margin classifiers, in: Annu. Workshop Computational Learning Theory, 1992, pp. 144–152. doi:10.1145/130385.130401.

[62] B. Schölkopf, A. Smola, K.-R. Müller, Nonlinear component analysis as a kernel eigenvalue problem, Neural Computation 10 (5) (1998) 1299–1319. doi:10.1162/089976698300017467.

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

[64] B. J. Jain, S. Spiegel, Time series classification in dissimilarity spaces, in: Int. Workshop Adv. Anal. and Learning Temporal Data, 2015, pp. 71–76.

[65] W. Chaovalitwongse, P. Pardalos, On the time series support vector machine using dynamic time warping kernel for brain activity classification, Cybern. and Syst. Anal. 44 (1) (2008) 125–138. doi:10.1007/s10559-008-0012-y.

[66] H. Shimodaira, K.-i. Noma, M. Nakai, S. Sagayama, Dynamic time-alignment kernel in support vector machine, in: Advances in Neural Inform. Process. Syst., 2002, pp. 921–928.

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

[68] C. A. Ratanamahatana, E. Keogh, Everything you know about dynamic time warping is wrong, in: Workshop Mining Temporal and Sequential Data, 2004, pp. 53–63.

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

[70] B. Bruno, F. Mastrogiovanni, A. Sgorbissa, T. Vernazza, R. Zaccaria, Analysis of human behavior recognition algorithms based on acceleration data, in: IEEE Int. Conf. Robotics and Automation, 2013, pp. 1602–1607. doi:10.1109/icra.2013.6630784.

[71] N. Hammami, M. Sellam, Tree distribution classifier for automatic spoken Arabic digit recognition, in: Int. Conf. Internet Technology and Secured Trans., 2009, pp. 1–4. doi:10.1109/icitst.2009.5402575.

[72] S. G. Wu, F. S. Bao, E. Y. Xu, Y.-X. Wang, Y.-F. Chang, Q.-L. Xiang, A leaf recognition algorithm for plant classification using probabilistic neural network, in: Int. Symp. Signal Process. and Inform. Technology, 2007, pp. 11–16. doi:10.1109/isspit.2007.4458016.

[73] Y. A. LeCun, L. Bottou, G. B. Orr, K.-R. Müller, Efficient backprop, in: Neural Networks: Tricks of the trade, Springer, 2012, pp. 9–48. doi:10.1007/978-3-642-35289-8_3.

[74] S. Ioffe, C. Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift, in: Int. Conf. Machine Learning, Vol. 37, 2015, pp. 448–456.

[75] J. Hu, S. G. Lim, M. K. Brown, Writer independent on-line handwriting recognition using an HMM approach, Pattern Recognition 33 (1) (2000) 133–147. doi:10.1016/s0031-3203(99)00043-6.

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

[77] K. R. Kanna, V. Sugumaran, T. Vijayaram, C. Karthikeyan, Activities of daily life (ADL) recognition using wrist-worn accelerometer, Int. J. Eng. and Technology 4 (3) (2016) 1406–1413.

[78] N. Hammami, M. Bedda, Improved tree model for arabic speech recognition, in: Int. Conf. Comp. Sci. and Inform. Technology, Vol. 5, 2010, pp. 521–526. doi:10.1109/iccsit.2010.5563892.

[79] 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. Technology, 2012, pp. 1–5. doi:10.1109/iccitechnol.2012.6285769.

[80] X. Hu, L. Zhan, Y. Xue, W. Zhou, L. Zhang, Spoken arabic digits recognition based on wavelet neural networks, in: IEEE Int. Conf. Syst., Man, and Cybern., 2011, pp. 1481–1485. doi:10.1109/icsmc.2011.6083880.

[81] K. Singh, I. Gupta, S. Gupta, SVM-BDT PNN and fourier moment technique for classification of leaf shape, Int. J. Signal Process., Image Process. and Pattern Recognition 3 (4) (2010) 67–78.

[82] J. Chaki, R. Parekh, S. Bhattacharya, Plant leaf recognition using texture and shape features with neural classifiers, Pattern Recognition Letters 58 (2015) 61–68. doi:10.1016/j.patrec.2015.02.010.

[83] D. G. Tsolakidis, D. I. Kosmopoulos, G. Papadourakis, Plant leaf recognition using Zernike moments and histogram of oriented gradients, in: Hellenic Conf. on Artificial Intell., 2014, pp. 406–417. doi:10.1007/978-3-319-07064-3_33.

[84] T. S. of the Benjamin Rose Hospital, Multidisciplinary studies of illness in aged persons, J. of Chronic Diseases 9 (1) (1959) 55 – 62. doi:10.1016/0021-9681(59)90137-7.

[85] J.-X. Du, X.-F. Wang, G.-J. Zhang, Leaf shape based plant species recognition, Appl. Math. and Computation 185 (2) (2007) 883–893. doi:10.1016/j.amc.2006.07.072.

[86] Q.-P. Wang, J.-X. Du, C.-M. Zhai, Recognition of leaf image based on ring projection wavelet fractal feature, in: Int. Conf. Intell. Computing, 2010, pp. 240–246. doi:10.1007/978-3-642-14932-0_30.

[87] J. Chaki, R. Parekh, Plant leaf recognition using shape based features and neural network classifiers, Int. J. Advanced Comput. Sci. and Applicat. 2 (10). doi:10.14569/ijacsa.2011.021007.

[88] L. Ye, E. Keogh, Time series shapelets: a new primitive for data mining, in: ACM Int. Conf. Knowledge Discovery and Data Mining, 2009, pp. 947–956. doi:10.1145/1557019.1557122.

[89] T. Beghin, J. S. Cope, P. Remagnino, S. Barman, Shape and texture based plant leaf classification, in: Int. Conf. Advanced Concepts for Intelligent Vision Syst., 2010, pp. 345–353. doi:10.1007/978-3-642-17691-3_32.

[90] H. Sakoe, S. Chiba, Dynamic programming algorithm optimization for spoken word recognition, IEEE Trans. Acoustics, Speech, and Sig. Process. 26 (1) (1978) 43–49. doi:10.1109/tassp.1978.1163055.