Dynamic Weight Alignment for Temporal Convolutional Neural Networks

Article · December 2017

CITATION
1

READS
195

2 authors:

Brian Kenji Iwana
Kyushu University
35 PUBLICATIONS **91** CITATIONS
SEE PROFILE

Seiichi Uchida
Kyushu University
319 PUBLICATIONS **2,906** CITATIONS
SEE PROFILE

Dynamic Weight Alignment for Convolutional Neural Networks

Brian Kenji Iwana Seiichi Uchida
Department of Advanced Information Technology
Kyushu University, Fukuoka, Japan
{brian,uchida}@human.ait.kyushu-u.ac.jp

Abstract

In this paper, we propose a method of improving Convolutional Neural Networks (CNN) by determining the optimal alignment of weights and inputs using dynamic programming. Conventional CNNs convolve learnable shared weights, or filters, across the input data. The filters use a linear matching of weights to inputs using an inner product between the filter and a window of the input. However, it is possible that there exists a more optimal alignment of weights. Thus, we propose the use of Dynamic Time Warping (DTW) to dynamically align the weights to optimized input elements. This dynamic alignment is useful for time series recognition due to the complexities of temporal relations and temporal distortions. We demonstrate the effectiveness of the proposed architecture on the Unipen online handwritten digit and character datasets, the UCI Spoken Arabic Digit dataset, and the UCI Activities of Daily Life dataset.

1. Introduction

Neural networks and perceptron learning models have become a powerful tool in machine learning and pattern recognition. Early models were introduced in the 1970s, but recently they have achieved state-of-the-art results due to improvements in data availability and computational power [46]. Convolutional Neural Networks (CNN) [34] in particular have achieved the state-of-the-art results in many areas of image recognition, such as offline handwritten digit recognition [52], text digit recognition [35, 50], and object recognition [6, 16, 17].

In addition to the image domain, CNN models have been used for time series patterns. A predecessor to CNNs, Time Delay Neural Networks (TDNN) [51, 33] used time-delay windows similar to the filters of a CNN. CNNs were also used to classify time series by embedding the sequences into vectors [59] and matrices [43, 55]. However, most recent successes in time series recognition have been through the use of Recurrent Neural Networks (RNN) [27] and in particular, Long Short-Term Memory (LSTM) networks [20].

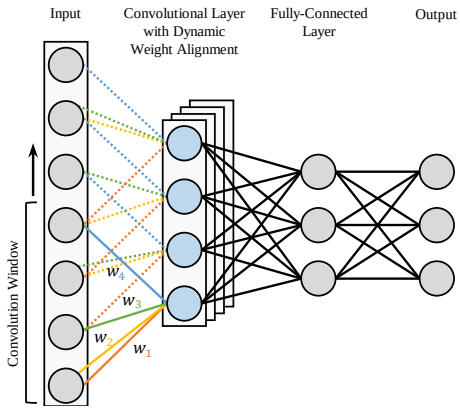


Figure 1. A 3-layer CNN with the first hidden layer as a convolutional layer with dynamically aligned weights. In each stride, the shared weights are individually aligned to the optimal input.

The key feature of CNNs are the sparsely connected shared weights used as a convolution. In this way, the shared weights, or filter, act as a feature extractor and maintain the structural aspects from the input. Furthermore, the shared weights are linearly aligned to each corresponding window value of the input. This interaction is much like a linear inner product or linear kernel in that the result is large when the filter is similar to the receptive field and small when it is different.

Difficulties arise with time series recognition due to issues with temporal distortions such as varying rates. The linear alignment assumes that each element of the receptive field correspond directly to each weight of the filter in a one-to-one fashion. Conversely, time series patterns do not have linear relationships between elements. RNNs address these issues with the use of recurrent connections to store persistent information across time without strict time step alignment. In addition, LSTMs are capable of overcoming local rate inconsistencies with the use of gates to control the learning and develop long-term dependencies. However, most feedforward network models, including CNNs,

do not have the same considerations for time series and do not protect against temporal distortions.

To overcome this, we propose a method of finding the optimal alignment between the weights and the inputs using dynamic programming, namely Dynamic Time Warping (DTW) [44]. DTW determines an optimal distance between two time series patterns by elastically matching elements using dynamic programming along a constrained path on a cost matrix. It then combines the local distances between matched elements to determine the optimized global distance. We exploit the elastic matching utility of DTW to align the weights of the filter to the elements of the corresponding receptive field to create more effective feature extractors for CNNs.

The contribution of this paper is twofold. First, we propose a novel method of aligning weights within the convolutional filters of CNNs. This alignment of weights allows CNNs to effectively tackle time series patterns and be used for time series recognition, shown in Fig. 1. Second, we demonstrate the effectiveness of the proposed method on multiple time series datasets including: Unipen online handwritten digit and character datasets, the UCI Spoken Arabic Digit dataset, and the UCI Activities of Daily Life dataset. We perform a comparative study using a traditional CNN to reveal the benefits of the proposed weight alignment. In addition, we compare the proposed method to other state-of-the-art methods for time series classification.

The rest of this paper is organized as follows. In Section 2, we review other neural methods of time series recognition. Section 3 details the proposed dynamic weight alignment method for CNNs. Section 4 reports the experimental results and Section 5 discusses the advantages of dynamic weight alignment. Finally, in Section 6, we provide a conclusion and lay out future work.

2. Related work

Neural networks and perceptron learning methods for time series recognition have been extensively explored in literature. Many classic models were used for time series forecasting [57, 23, 58] where artificial neural networks were provided a sliding window view of a discrete signal and were tasked with predicting future time steps. However, these methods learn rigid relationships to temporal features and are susceptible to temporal distortions and shift variance.

TDNNs are classical feed-forward neural networks designed to address shift variance [51] by introducing wider sliding window receptive fields to represent multiple points in time. These windows, or *time delays*, are passed through learned filters to create subsequent feature map time series. TDNNs have been successful in speech and phoneme recognition [48]. CNNs continue the idea of learned filters permeated across a structural input to the domain of images. They have been used for time series recognition by considering the time steps as a dimension for the convolutions. Zheng *et al*. [59] use CNNs with 1D subsequences created from multivariate time series. There also have been attempts [31, 43, 55, 53] to classify time series patterns by embedding them into 2D matrices for classification by CNN.

WaveNet [40] employs dilated causal convolutional layers to generate audio. The causal convolutions have relatively small receptive fields, however, the convolutions are dilated meaning that higher layers skip input values and increase the effective range of the receptive field. The WaveNet architecture stacks multiple dilated causal convolutions to create a very large receptive field which is considerate to order.

There are other notable feedforward network architectures that have been proposed to address the characteristics or features of time series patterns. For instance, Multi-Layer Perceptrons (MLP) have been applied to wavelet transforms [1] and $\mathrm{ARMA}(p, q)$ time series data [23] for forecasting and classification. Spiking neurons [36, 3] create sparse representations using brief spikes time series data.

Other than feedforward networks, recurrent models like RNNs have been successful in time series recognition. Similar to other models, RNNs are given a sliding window of the input time series. However, unlike feedforward networks, RNNs use recurrent connections and stored hidden states to maintain a dependence on time. LSTMs are an improvement on RNNs which use gates to control the amount of learned information to solve the vanishing gradient problem [19]. LSTMs have been successful in many time series and sequence domains such as handwriting [10, 12], speech [11], and natural language processing [49].

Dynamic neural networks is an emerging field in neural model learning There are generally two approaches, dynamically embedding trained parameters into a network and using dynamic connections. Dynamic Filter Networks (DFN) [7] use filter-generating networks to produce filters that are dynamically used depending on the input. Klein *et al*. [32] similarly uses filters that vary depending on the input. Dynamic Convolutional Neural Networks (DCNN) [29] use dynamic k-Max Pooling to simplify CNNs for sentence modeling. The distinction between these models and the proposed method is that we use dynamic programming to optimize the weight alignment inside nonlinear convolutions.

DTW-NNs [26] similarly use DTW as a nonlinear inner product for neural networks. The difference between a DTW-NN and the proposed model is that DTW-NN uses a sum of Euclidean distances between a learned prototype sequence and the input. This means the weights are only pseudo-weights whereas the CNN described in this paper

are used as true weights. In addition, we use a shared weight CNN model instead of sparsely aligned network. The advantage of a convolution is that the output of the node retains temporal qualities instead of a scalar output, which allows for the ability to stack multiple nonlinear inner product convolutional layers.

3. Dynamic weight alignment for CNNs

The goal of the proposed method is to exploit dynamic programming to determine the optimal alignment of weights for convolutional layers in CNNs. By using DTW, we can adapt the conventional linear inner product of a convolution to optimize the connections between convolutional filter weights and inputs. Figure 2 demonstrates the difference between a conventional convolutional layer with linear weight alignment and the proposed CNN with dynamic weight alignment.

3.1. Convolutional Neural Networks

A key feature of CNNs is that they maintain sparse connections that promote local connectivity. Each element of a convolutional node only has a receptive field of a small number of locally neighboring inputs. Besides enforcing local relations, the sparsity of a CNN reduces the number of parameters needed to be calculated and allows for the use of larger input sizes.

Parameter sharing further reduces the number of parameters required training a CNN. The idea is that the weights of a convolutional layer are shared for each corresponding output element's local receptive field. In this way, a forward calculation of a convolutional layer is identical to a convolution operation where the shared weights are the filter and the output is a feature map.

Formally, the feature map z_x^l of a convolutional layer is defined as:

$$z_x^l = \sum_{f=0}^{F} w_f^l a_{x+f}^{l-1} + b_x^l \tag{1}$$

for each element x, where l is the convolutional layer, $l-1$ is the previous layer, f is the index of the filter, and $F+1$ is the window size. We denote w_f^l, a_{x+f}^{l-1}, and b_x^l as the shared weights, the previous layer activations, and the bias respectively. In other words, z_x^l is the inner product of the shared weights $\mathbf{w}^l$ and each window of the previous layer $a_x^{l-1}, \ldots, a_{x+F}^{l-1}$.

This inner product linearly matches the weights to the inputs within the window. However, it is plausible that there exist instances where particular weights should be matched with more optimal inputs, for example skipping noisy elements or feature translation and scale variance within the filter.

3.2. Dynamic weight alignment

To optimize the alignment of weights w^l and the window of the previous layer $a_x^{l-1}, \ldots, a_{x+F}^{l-1}$, we adopt a dynamic programming solution, namely DTW. Given the nonlinear matching of weights and the values from the window, we consider the result of DTW as a nonlinear inner product.

3.2.1 Dynamic Time Warping

DTW is an asymmetric positive semi-definite similarity function that traditionally used as a distance measure between sequences. It is calculated using dynamic programming to determine the optimal match of elements between two sequences. By matching elements, the sequences are *warped* in the time dimension to align similar features of the time series. DTW distance has shown to be a successful distance measure for time series recognition [54, 8, 42].

To calculate DTW, the total cost over an optimal warping path of a local cost matrix is found using dynamic programming. Given two discrete time series, sequence $\mathbf{p} = p_1, \ldots, p_i, \ldots, p_I$ of length I and sequence $\mathbf{s} = s_1, \ldots, s_j, \ldots, s_J$, where i and j are the index of each time step and p_i and s_j are elements at each time step, the DTW-distance is the global summation of local distances between pairwise element matches. The DTW-distance is denoted as:

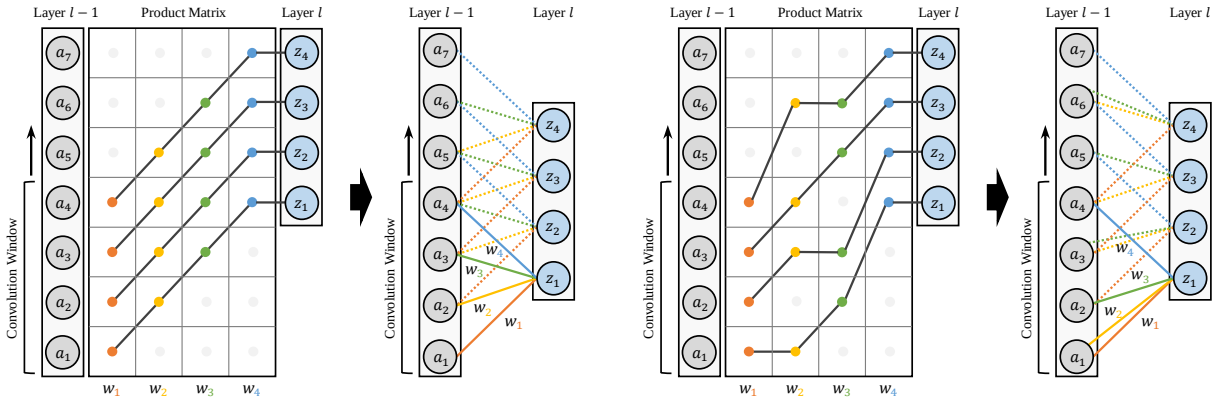
$$\mathrm{DTW}(\mathbf{p}, \mathbf{s}) = \sum_{(i',j') \in \mathcal{M}} ||p_{i'} - s_{j'}||\,, \tag{2}$$

where (i', j') is a pair of matched indices i' and j' corresponding to the original indices i of $\mathbf{p}$ and j of $\mathbf{s}$, respectively. The set $\mathcal{M}$ contains all matched pairs of i' and j'. It should be noted that the set of matched pairs $\mathcal{M}$ can contain repeated and skipped indices of i and j from the original sequences, therefore, $\mathcal{M}$ has a nonlinear correspondence to $1, \ldots, i, \ldots, I$ and $1, \ldots, j, \ldots, J$. $||\cdot||$ is a local distance function between elements.

The use of DTW as a distance measure or similarity function is not unlike a kernel function and has been integrated into the kernel functions of Support Vector Machines (SVM) [5, 47, 2] and as a replacement for the inner product of fully-connected neural network layers in Dynamic Time Warping Neural Networks (DTW-NN) [26].

3.2.2 Dynamic weight alignment with shared weights

If we define the distance function $||\cdot||$ of Eq. (2) as the product of $p_{i'}$ and $s_{j'}$, then we can consider DTW as a nonlinear inner product. Unlike the conventional convolution defined in Eq. (1), the result of DTW would become the nonlinear product of weights and the window of the previous layer.



(a) Convolution with linear weight alignment

(b) Convolution with dynamic weight alignment

Figure 2. The comparison between (a) a conventional linear convolution and (b) the proposed convolution with dynamic weight alignment. Both illustrate 1D convolutions with four weights $w_1, \ldots, w_4$ at stride 1. The layer $l-1$ is the previous layer with elements $a_1, \ldots, a_7$ and layer l is the resulting feature map from the convolution with elements $z_1, \ldots, z_4$. Each colored dot is the product of the corresponding weight and input, and $z_1, \ldots, z_4$ is the sum of the products.

Namely, we propose using DTW to determine z_x^l:

$$z_x^l = \text{DTW}(\mathbf{w}^l, \{a_x^{l-1}, \ldots, a_{x+F}^{l-1}\}) + b_x^l \tag{3}$$

$$= \sum_{(f',x')\in\mathcal{M}} w_{f'}^l a_{x'}^{l-1} + b_x^l, \tag{4}$$

where $\mathcal{M}$ is the set of optimally matched indices f' and x' corresponding to the index f of $\mathbf{w}^l$ and the index x in $\{x, \ldots, x+F\}$ of $\mathbf{a}^{l-1}$, respectively. When used in this manner, we create a nonlinear convolutional filter that acts as a feature extractor similar to using shapelets with DTW [56].

To create the nonlinearity of the weight alignment of the filters, DTW allows for a relaxed matching as determined by the warping path within constraints. For example, a slope constraint can be applied to the warping path to prevent overfitting. We propose using an asymmetric slope constraint which is a modification of the slope constraint proposed by Itakura [25]. In the traditional implementation of DTW, a the local distance metric is a dissimilarity function and the goal is to minimize the global distance. However, when using inner product space as the cost matrix, the local distance metric represents similarity, therefore we aim to find the maximal product. Thus, we use the slope constraint defined by the recurrent function:

$$D(f,x) = w_f^l a_x^{l-1} + \max_{x'\in\{x,x-1,x-2\}} D(f-1,x'), \tag{5}$$

where $D(f,x)$ is the cumulative sum at the f-th element of w_f^l and the x-th element of a_x^{l-1}. The global DTW distance, or the result of the nonlinear inner product, is defined as the value at $D(F,X)$.

In addition, there are many other possible slope and local constraints that have been researched [44, 38, 28]. However, the proposed method uses a slope constraint that ensures that the warping path is monotonic and that the number of matches is always equal to the number of elements in the filter.

3.3. Backpropagation of convolutions with dynamic weight alignment

In order to train the network, we use Stochastic Gradient Decent (SGD) to update the weights in order to minimize the loss. For a CNN, the gradient of the shared weights with respect to the error is the partial derivative:

$$\frac{\partial C}{\partial w_f^l} = \sum_x \frac{\partial C}{\partial z_x^l}\frac{\partial z_x^l}{\partial w_f^l}, \tag{6}$$

where C is the loss function. In a conventional CNN, w_f^l has a linear relationship to z_x^l, thus $\frac{\partial z_x^l}{\partial w_f^l}$ can be calculated simply. However, given the nonlinearity of the weight alignment, the calculation of the gradient is reliant on the matched elements determined by the forward pass in:

$$\frac{\partial C}{\partial w_f^l} = \sum_x \frac{\partial C}{\partial z_x^l}\frac{\partial\left(\sum_{(f',x')\in\mathcal{M}} w_{f'}^l a_{x'}^{l-1} + b_x^l\right)}{\partial w_f^l}. \tag{7}$$

Unlike a conventional CNN, the set of matched indicies $\mathcal{M}$ in Eq. (4) allows for duplicate and skipped values of $w_{f'}^l$ and $a_{x'}^{l-1}$. Thus, the relationship between $\sum_{(f',x')\in\mathcal{M}} w_{f'}^l a_{x'}^{l-1} + b_x^l$ and w_f^l is the sum of the windows of the matched activations $a_{x'}^{l-1}$ from the previous layer for each matched weight $w_{f'}^l$. Finally, the Eq. (7) can be solved, giving us:

$$\frac{\partial C}{\partial w_f^l} = \delta^{l+1} \sum_{(f',x')\in\mathcal{M}} a_{x'}^{l-1}, \tag{8}$$

where δ^{l+1} is the backpropagated error from the previous layer as determined by the chain rule.

4. Experiments

We demonstrate the effectiveness of the proposed method by quantitatively evaluating the architecture and comparing it to baseline methods.

4.1. Architecture settings

For the experiment, we implement a four layer dynamically aligned CNN. The first hidden layer is a convolutional layer with 50 nodes of the proposed dynamically aligned filters. The second and third layers are fully-connected layers with a hyperbolic tangent $\tanh$ activation and have 400 and 100 nodes respectively. The final output layer uses softmax with the number of outputs corresponding to the number of classes.

4.1.1 Batch normalization

Batch normalization helps in the generalization ability of a neural network [24]. It aims to reduce the effects of the internal covariate shift which forces higher layers of a network to drift which causes *whitening* of the inputs. To solve this problem, batch normalization shifts the values so that the mean is zero and the variance is one. The proposed method uses batch normalization as described by [24] after the convolutional layer.

4.1.2 Learning rate

By using a learning rate with a progressive decay, gradient decent of the network will quickly learn the weight parameters in the beginning and slowly refine them later in the training. The learning rate η_t at iteration t is defined as:

$$\eta_t = \frac{\eta_0}{1 + \alpha t}, \tag{9}$$

where η_0 is the initial learning rate and α is the decay parameter. For all of the experiments, we use the $1/t$ progressive learning rate with a $\eta_0 = 0.001$ and $\alpha = 0.001$.

4.2. Comparison methods

To evaluate the proposed method, we compare the accuracy to comparison methods. We report classification results from literature as well as evaluations using established state-of-the-art neural network methods.

4.2.1 Reported baselines

We compare the results of the proposed method to reported results from literature.

For the online handwritten digits and character evaluation, we compare results from two neural network models, a Regional-Fuzzy Representation (RFR) with an MLP [41] and a DTW-NN [26], and two other methods, an SVM using a Gaussian DTW kernel (DAG-SVM-GDTW) [2] and a Hidden Markov Model (HMM) [21]. We also compare our results to a Fuzzy Adaptive System ART-based (FasArt) [45] model and a Kohonen-Perceptron network using fuzzy vector representation (Fuzzy Rep. KP) [18].

For the spoken Arabic digits, there is one reported neural network solution using a Wavelet Neural Network (WNN) [22] as well as other models using a Tree Distribution model [14] and a Continuous Hidden Markov Model of the second-order derivative MFCC (CHMM w/ $\Delta(\Delta$MFCC)) [15].

For the activities of daily life (ADL) dataset, we compare our results to the original dataset proposal by Bruno *et al*. [4] who modeled the ADL accelerometer data using Gaussian Mixture Modeling and Gaussian Mixture Regression (GMM + GMR). We also report the best results of Kanna et al. [30] who used a Decision Tree.

4.2.2 Evaluated baselines

The evaluated baselines were designed to be direct comparisons for the proposed method. The LSTM is used as the established state-of-the-art neural network method for sequence and time series recognition and a traditional CNN is used as a direct comparison using standard convolutional layers. Both comparative models are provided with the same exact training, test, and validation sets as the proposed method. Furthermore, the evaluated methods use the same batch size and number of iterations as the proposed method for the respective trials.

For the LSTM evaluation, an LSTM with two recursive hidden layers and a softmax output layer was used. The LSTM was set to a forget gate bias of 1.0 and had a static learning rate of 0.001.

The second comparative evaluation was using a CNN with the same exact hyperparameters as the proposed method, but with standard convolutional nodes. It was constructed of one 50-node convolutional layer with a window size and stride equal each proposed method evaluation with batch normalization, two fully-connected $\tanh$ layers size 400 and 100 respectively, and a softmax output layer. The convolutional layer weights had the same progressive $1/t$ decay learning rate as evaluation of the proposed method with $\eta_0 = 0.001$ and $\alpha = 0.001$ and a static learning rate of 0.0001 between the fully-connected layers.

4.3. Classification of online handwritten digits and characters

The isolated online handwritten digits and characters offer an ideal source of data for time series recognition due to having distinct classes while having numerous within-class variations.

4.3.1 Dataset

The Unipen multi-writer 1a, 1b, and 1c datasets [13] are constructed from pen tip trajectories of isolated numerical digits, uppercase alphabet characters, and lowercase alphabet characters respectively. The datasets are provided by the International Unipen Foundation (iUF).[1]

We used a set of 13,000 randomly selected digits with 10 classes for the 1a experiment and a set of 12,350 randomly selected characters with 26 classes for the 1b and 1c experiments. The patterns were preprocessed by re-sampling them to 50 sequence elements and scaling them to fit in a square defined by the corner coordinates of $(-0.5, -0.5)$ and $(0.5, 0.5)$. The datasets were divided into three, a test of 10% of the data, a training set of 90% of the data, and 50 patterns set a side from the training set for a validation set.

4.3.2 Training

Given that the experimental dataset is made of sequences of two-dimensional coordinates, the filters should correspond accordingly. Thus, the convolutional filters were of size 7×2 and were applied to the input sequences at stride 2. A stride 2 was used to reduce redundant information and decrease computation time. The filters were initialized using a Gaussian distribution with a minimum value of -0.5 and a maximum of 0.5, which is within a similar range as the dataset. The experiment uses batch gradient decent with a batch size of 100 for 30,000 iterations.

4.3.3 Results

The results of the experiments on the Unipen 1a, 1b, and 1c datasets are shown in Table 1. The proposed method achieved a test accuracy of 98.08% for digits, 95.67% for uppercase characters, and 94.83% for lowercase characters after 30,000 iterations of training, surpassing the conventional CNN and the LSTM. Furthermore, the proposed method surpassed all of the comparative methods except for DTW-NN with 98.2% for the Unipen 1a trial.

Method	1a	1b	1c
DAG-SVM-GDTW [2]	96.2	92.4	87.9
HMM [21]	96.8	93.6	85.9
RFR [41]	96.9	-	85.6
DTW-NN [26]	**98.2**	94.9	94.5
FasArt [45]	85.66	-	-
Fuzzy Rep. KP [18]	96.1	-	-
LSTM	95.92	89.39	86.88
CNN	97.92	94.50	94.75
Proposed Method	98.08	**95.67**	**94.83**

Table 1. Accuracy (%) on the Unipen online handwritten digit (1a), uppercase character (1b), and lowercase character (1c) datasets. The highest accuracy for each dataset is in bold.

4.4. Classification of spoken arabic digits

Audio is a prime example of a time series and speech and phoneme classification is essential for speech recognition. One popular method of audio recognition is the use of Mel-Frequency Cepstrum Coefficients (MFCC) [9]. MFCCs transforms the audio signals into discrete time series by using mel-frequency cepstrum representations with the number of dimensions equaling the number of coefficients.

4.4.1 Dataset

We used the UCI Spoken Arabic Digit Data Set[2] for the experiments. The dataset consists of 8,800 13-frequency MFCCs of spoken Arabic numerals in 10 classes. The samples were taken at 11,025 Hz, 16 bit sampling rate with a Hamming window, and the filters pre-emphasized $1 - 0.97z^{-1}$.

The UCI Spoken Arabic Digit Data Set has a pre-defined division of the data with a speaker-independent training set and test set of 6,600 patterns and 2,200 patterns respectively. 50 patterns from the training set were separated as validation during training. For the experiments, we preprocessed the patterns with re-sampling to 40 sequence elements and scaling by $\frac{1}{15}$.

4.4.2 Training

The UCI Spoken Arabic Digit Data Set has 13-frequency MFCCs, therefore we use the data as 40 time steps of 13-dimensional data. The convolutional filters are set to 5×13 at stride 2 with a minimum value of -0.5 and a maximum of 0.5 to fit the data. We trained the proposed method with a batch size of 50 for 30,000 iterations.

4.4.3 Results

The proposed method had an accuracy of 96.95% for the predefined test set with 2,200 speaker independent MFCC

[1] URL: http://www.unipen.org/home.html

[2] URL: https://archive.ics.uci.edu/ml/datasets/Spoken+Arabic+Digit

Method	Accuracy (%)
Tree Distribution [14]	93.1
CHMM w/ Δ(ΔMFCC) [15]	**98.4**
WNN [22]	96.7
LSTM	96.0
CNN	94.77
Proposed Method	96.95

Table 2. Accuracy (%) on the UCI Spoken Arabic Digit Data Set. The highest accuracy for each dataset is in bold.

Method	Accuracy (%)
GMM + GMR [4]	63.1
Decision Tree [30]	80.9
LSTM	80.0
CNN	84.3
Proposed Method	**85.7**

Table 3. Accuracy (%) on the UCI ADL Recognition with Wrist-worn Accelerometer Data Set. The highest accuracy for each dataset is in bold.

spoken Arabic digits. When comparing the results in Table 2, it is noted that the proposed method was outperformed by CHMM w/ Δ(ΔMFCC). Nonetheless, the proposed CNN with dynamically aligned weights performed better than the conventional CNN, the LSTM, and the other reported results. In addition, CHMM w/ Δ(ΔMFCC) is a method tailored to MFCC speech recognition whereas the proposed method is a general method for time series.

4.5. Classification of activities of daily life

Study of ADL data is important for understanding the mechanics of daily life and helpful for for applications such as robotics, assisted ADL, and the monitoring of disabled or elderly patients [39].

4.5.1 Dataset

One common method of studying ADL is the use of wearable sensors. An example of this is the UCI ADL Recognition with Wrist-worn Accelerometer Data Set.[3] The dataset contains 705 recorded accelerometer data split into 7 classes of about 100 patterns each. The actions include "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." They were recorded at 32 Hz in the three spacial dimensions.

For the experiment, we scaled the data to $\frac{1}{1.5g}$ where g is the acceleration due to gravity. The time series were also sampled to 200 frames. The original experiment [4] uses a median filter, however we found it unnecessary and used the raw data for the experiment. For training, the dataset was divided at random into a training set of 600 patterns, a validation set of 5 patterns, and a test set of 70 patterns.

4.5.2 Training

The dataset is made of tri-axial accelerometer data. Due to the length of the samples, 200 frames, we used a larger convolutional filter of 9×3 at stride 4. They were initiated similarly with a Gaussian distribution with a minimum value of -0.5 and a maximum of 0.5 in all three dimensions. Finally, only a batch size of 5 was used due to the very small size of the dataset.

4.5.3 Results

Similar to the other datasets, the results in Table 3 show that the proposed method improved the accuracy over the conventional CNN with linearly aligned weights, with 85.7% and 84.3% respectively. The result of the proposed CNN with dynamically aligned weights far exceeds the accuracy of the original dataset proposal using GMM + GMR.

5. Discussion

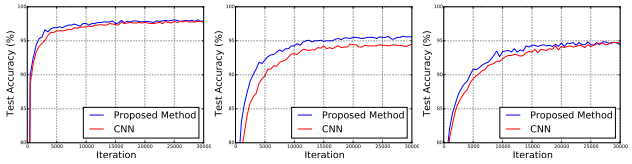
In this section, we discuss comparisons between the proposed method and the evaluated comparisons. We also offer explanations for the improved accuracy. Additional results such as the confusion matrices and accuracy comparisons are provided in the supplementary materials.

5.0.1 Proposed method versus LSTMs

LSTMs are the state-of-the-art sequence and time series neural network model. They are RNNs designed to maintain long-term dependencies through gating.

In the online handwriting and ADL experiments, the LSTM performed poorly compared to both CNNs. One reason for the limited performance of the LSTM is that each individual element of those datasets do not contain a significant amount of information and the model needs to know how all elements work together to form spatial structures. For example, for large tri-axial accelerometer data, individual long-term dependencies are not as important as the local and global structures whereas CNNs excels. Another reason for the poor performance of the ADL dataset could be the low amount of training data (600 training samples), high amounts of noise, and a high variation of patterns within each class. However, the LSTM did comparatively well on the spoken Arabic digits.

[3]URL: `https://archive.ics.uci.edu/ml/datasets/Dataset+for+ADL+Recognition+with+Wrist-worn+Accelerometer`



Unipen 1a Unipen 1b Unipen 1c

Figure 3. Comparison of test accuracies of the Unipen online handwritten datasets between a conventional CNN and the proposed CNN with dynamic weight alignment over the number of training iterations.

5.0.2 Proposed method versus conventional CNNs

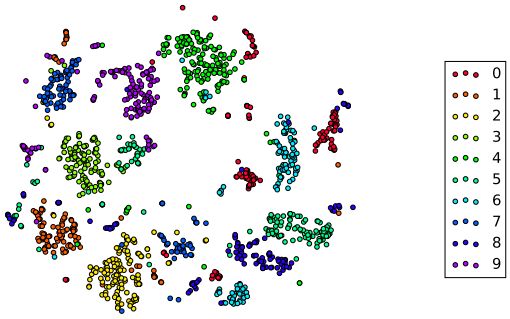
It is important to analyze the differences between a conventional CNN with linearly aligned weights and the proposed method with dynamically aligned weights. Table 1 reveals a 7.69% reduction in error for Unipen 1a, 21.21% reduction for 1b, and a 1.59% reduction for 1c. In addition to the increased accuracy, we can observe from Fig. 3 that compared to the conventional CNN, the proposed method achieves a higher accuracy during all parts of training but especially during the early stages. This indicates that the nonlinear alignment optimizes the use of the weights before the generalization of the CNN converges.

5.0.3 Efficient feature extraction

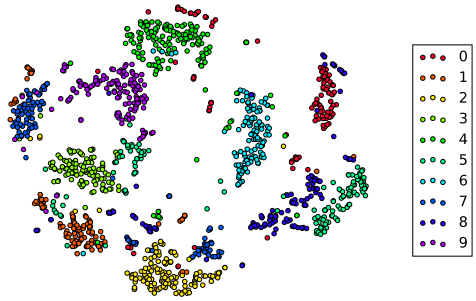
One explanation of the improved accuracy is that aligning the weights to their similar corresponding inputs is more efficient than conventional linear matching. The weights of a convolutional layer learned learned by a CNN act like filter for feature extraction [34]. And, the purpose of using dynamically aligned weights is to warp the assignment of weights to their most similar corresponding inputs. The DTW algorithm skips redundant and noisy inputs while ensuring that the target features are detected. This way, the features that are extracted by the filters are enhanced.

5.0.4 Improved low-level discriminability

Figure 4 is a visualization of the outputs of the convolutional layers from the test set using t-Distributed Stochastic Neighbor Embedding (t-SNE) [37]. t-SNE is an algorithm to visualize high-dimensional data which allows the visualization of the response from the convolutional node for each sample in low-dimensional space. The figure shows that the classes, represented by different colors, are more discriminable with the dynamic weight alignment. Figure 4 (a) is the test samples using a conventional CNN and it shows that many of the classes are made of multiple small clusters whereas Fig. 4 (b) generally contains larger unified clusters. For example, classes "0", "6", and "9" are made of multiple clusters in Fig 4 (a), but are unified in (b). The higher discriminability means that there would be less confusion for the higher layers. The t-SNE figures for the other evaluations can be found in the supplementary materials.



(a) CNN with linear weight alignment



(b) CNN with dynamic weight alignment

Figure 4. t-SNE visualization of the output of the convolutional node for the Unipen 1a (digits) experiment. Each point is a test sample colored to indicate the class membership. (a) is a conventional CNN and (b) is a CNN with the proposed weight alignment.

6. Conclusion

In this paper, we proposed a novel method of optimizing the weights within a convolutional filter of a CNN through the use of dynamic programming. We implemented DTW as a method of sequence element alignment between the weights of a filter and the inputs of the corresponding receptive field. By defining the local distance function of DTW as a product, we show that it is possible to use DTW as a less rigid nonlinear inner product. In this way, the weights of the convolutional layer are aligned to maximize their relationship to the data from the previous layer.

We show that the proposed model is an effective method to tackle time series recognition. We evaluated the proposed model on the Unipen online handwritten digit and character datasets. The results show n-class classification test accuracies of 98.08% for online handwritten digits, 95.67% for uppercase characters, and 94.83% for lowercase characters. In addition to handwriting, we demonstrate the effectiveness of the proposed model on other types of data including

13-coefficient MFCC speech of Arabic digits and tri-axial accelerometer ADL data. The spoken Arabic digit evaluate achieved a 96.95% accuracy and the accelerometer-based ADL evaluation achieved a 85.7% accuracy.

Dynamic weight alignment for convolutions of CNNs is a promising area of research. The proposed method can be applied to other CNN based models such as deep CNNs or fully convolutional networks. In the future, we plan on extending this method to other time series datasets as well as other domains, such as images.

References

[1] A. K. Alexandridis and A. D. Zapranis. Wavelet neural networks: A practical guide. *Neural Networks*, 42:1–27, 2013.

[2] C. Bahlmann, B. Haasdonk, and H. Burkhardt. Online handwriting recognition with support vector machines-a kernel approach. In *Int. Workshop Frontiers in Handwriting Recognition*, pages 49–54, 2002.

[3] 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):349–398, 2007.

[4] B. Bruno, F. Mastrogiovanni, A. Sgorbissa, T. Vernazza, and R. Zaccaria. Analysis of human behavior recognition algorithms based on acceleration data. In *IEEE Int. Conf. Robotics and Automation*, pages 1602–1607. IEEE, 2013.

[5] W. Chaovalitwongse and P. Pardalos. On the time series support vector machine using dynamic time warping kernel for brain activity classification. *Cybern. and Syst. Anal.*, 44(1):125–138, 2008.

[6] D.-A. Clevert, T. Unterthiner, and S. Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015.

[7] B. De Brabandere, X. Jia, T. Tuytelaars, and L. Van Gool. Dynamic filter networks. In *Advances in Neural Inform. Process. Systems*, 2016.

[8] H. Ding, G. Trajcevski, P. Scheuermann, X. Wang, and E. Keogh. Querying and mining of time series data: experimental comparison of representations and distance measures. *Proc. Very Large Data Base Endowment*, 1(2):1542–1552, 2008.

[9] T. Ganchev, N. Fakotakis, and G. Kokkinakis. Comparative evaluation of various mfcc implementations on the speaker verification task. In *Int. Conf. Speech and Comput.*, pages 191–194, 2005.

[10] A. Graves, M. Liwicki, S. Fernandez, R. Bertolami, H. Bunke, and J. Schmidhuber. A novel connectionist system for unconstrained handwriting recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 31(5):855–868, may 2009.

[11] A. Graves, A. rahman Mohamed, and G. Hinton. Speech recognition with deep recurrent neural networks. In *IEEE Int. Conf. Acoustics, Speech and Signal Process.* IEEE, May 2013.

[12] A. Graves and J. Schmidhuber. Offline handwriting recognition with multidimensional recurrent neural networks. In *Advances in Neural Inform. Process. Syst.*, pages 545–552, 2009.

[13] I. Guyon, L. Schomaker, R. Plamondon, M. Liberman, and S. Janet. Unipen project of on-line data exchange and recognizer benchmarks. In *Int. Conf. Pattern Recognition*, volume 2, pages 29–33. IAPR, 1994.

[14] N. Hammami and M. Bedda. Improved tree model for arabic speech recognition. In *IEEE Int. Conf. Comp. Sci. and Inform. Technology*, volume 5, pages 521–526. IEEE, 2010.

[15] N. Hammami, M. Bedda, and 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*, pages 1–5. IEEE, 2012.

[16] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *IEEE Int. Conf. on Comput. Vision*, pages 1026–1034, 2015.

[17] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *IEEE Conf. Comp. Vision and Pattern Recognition*, pages 770–778. IEEE, 2016.

[18] J.-F. Hébert, M. Parizeau, and N. Ghazzali. A new fuzzy geometric representation for online isolated character recognition. In *Int. Conf. Pattern Recognition*, volume 2, pages 1121–1123. IEEE, 1998.

[19] S. Hochreiter, Y. Bengio, P. Frasconi, and J. Schmidhuber. *Gradient flow in recurrent nets: the difficulty of learning long-term dependencies*. Wiley-IEEE Press, 2001.

[20] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, nov 1997.

[21] J. Hu, S. G. Lim, and M. K. Brown. Writer independent online handwriting recognition using an hmm approach. *Pattern Recognition*, 33(1):133–147, 2000.

[22] X. Hu, L. Zhan, Y. Xue, W. Zhou, and L. Zhang. Spoken arabic digits recognition based on wavelet neural networks. In *IEEE Int. Conf. Syst., Man, and Cybern.*, pages 1481–1485. IEEE, 2011.

[23] H. B. Hwarng and H. Ang. A simple neural network for arma (p, q) time series. *Omega*, 29(4):319–333, 2001.

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

[25] F. Itakura. Minimum prediction residual principle applied to speech recognition. *IEEE Trans. Acoustics, Speech, and Sig. Process.*, 23(1):67–72, 1975.

[26] B. K. Iwana, V. Frinken, and S. Uchida. A robust dissimilarity-based neural network for temporal pattern recognition. In *Int. Conf. Frontiers in Handwriting Recognition*, pages 265–270, 2016.

[27] H. Jaeger. *Tutorial on training recurrent neural networks, covering BPPT, RTRL, EKF and the" echo state network" approach*, volume 5. GMD-Forschungszentrum Informationstechnik, 2002.

[28] R. Kakisako, S. Uchida, and V. Frinken. Learning non-markovian constraints for handwriting recognition. In *Int. Conf. on Document Anal. and Recognition*, pages 446–450, 2015.

[29] N. Kalchbrenner, E. Grefenstette, and P. Blunsom. A convolutional neural network for modelling sentences. In *Annu. Meeting Assoc. for Comput. Linguistics*, pages 655–665, 2014.

[30] K. R. Kanna, V. Sugumaran, T. Vijayaram, and C. Karthikeyan. Activities of daily life (adl) recognition using wrist-worn accelerometer. *Int. J. of Eng. and Technology*, 4(3):1406–1413, 6 2016.

[31] Y. Kim. Convolutional neural networks for sentence classification. In *Conf. Empirical Methods in Natural Language Process.*, pages 1746–1751, 2014.

[32] B. Klein, L. Wolf, and Y. Afek. A dynamic convolutional layer for short range weather prediction. In *IEEE Conf. Comput. Vision and Pattern Recognition*, pages 4840–4848, 2015.

[33] K. J. Lang, A. H. Waibel, and G. E. Hinton. A time-delay neural network architecture for isolated word recognition. *Neural Networks*, 3(1):23–43, 1990.

[34] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proc. IEEE*, 86(11):2278–2324, 1998.

[35] C.-Y. Lee, P. W. Gallagher, and Z. Tu. Generalizing pooling functions in convolutional neural networks: Mixed, gated, and tree. In *Int. Conf. Artificial Intell. and Stat.*, pages 464–472, 2016.

[36] W. Maass. Networks of spiking neurons: the third generation of neural network models. *Neural Networks*, 10(9):1659–1671, 1997.

[37] L. v. d. Maaten and G. Hinton. Visualizing data using t-sne. *J. Mach. Learning Research*, 9(Nov):2579–2605, 2008.

[38] C. Myers, L. Rabiner, and A. Rosenberg. Performance tradeoffs in dynamic time warping algorithms for isolated word recognition. *IEEE Trans. Acoustics, Speech, and Sig. Process.*, 28(6):623–635, 1980.

[39] T. S. of the Benjamin Rose Hospital. Multidisciplinary studies of illness in aged persons. *J. of Chronic Diseases*, 9(1):55 – 62, 1959.

[40] A. v. d. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.

[41] M. Parizeau, A. Lemieux, and C. Gagné. Character recognition experiments using unipen data. In *Int. Conf. Document Anal. and Recognition*, pages 481–485, 2001.

[42] T. Rakthanmanon, B. Campana, A. Mueen, G. Batista, B. Westover, Q. Zhu, J. Zakaria, and E. Keogh. Searching and mining trillions of time series subsequences under dynamic time warping. In *Int. Conf. Knowledge Discovery and Data Mining*. ACM, 2012.

[43] N. Razavian and D. Sontag. Temporal convolutional neural networks for diagnosis from lab tests. *arXiv preprint arXiv:1511.07938*, 2015.

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

[45] E. G. Sánchez, J. G. González, Y. A. Dimitriadis, J. C. Izquierdo, and J. L. Coronado. Experimental study of a novel neuro-fuzzy system for on-line handwritten unipen digit recognition. *Pattern Recognition Letters*, 19(3):357–364, 1998.

[46] J. Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117, 2015.

[47] H. Shimodaira, K.-i. Noma, M. Nakai, and S. Sagayama. Dynamic time-alignment kernel in support vector machine. In *Advances in Neural Inform. Process. Syst.*, pages 921–928, 2002.

[48] M. Sugiyama, H. Sawai, and A. H. Waibel. Review of tdnn (time delay neural network) architectures for speech recognition. In *IEEE Int. Symp. on Circuits and Syst.*, pages 582–585. IEEE, 1991.

[49] M. Sundermeyer, R. Schlüter, and H. Ney. Lstm neural networks for language modeling. In *Interspeech*, 2010.

[50] S. Uchida, S. Ide, B. K. Iwana, and A. Zhu. A further step to perfect accuracy by training cnn with larger data. In *Int. Conf. Frontiers in Handwriting Recognition*, 2016.

[51] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K. J. Lang. Phoneme recognition using time-delay neural networks. *IEEE Trans. Acoust., Speech, Signal Process.*, 37(3):328–339, 1989.

[52] L. Wan, M. Zeiler, S. Zhang, Y. L. Cun, and R. Fergus. Regularization of neural networks using dropconnect. In *Int. Conf. Mach. Learning*, pages 1058–1066, 2013.

[53] Z. Wang and T. Oates. Encoding time series as images for visual inspection and classification using tiled convolutional neural networks. In *Assoc. Advancement Artificial Intell. Workshop*, pages 40–46, 2015.

[54] X. Xi, E. Keogh, C. Shelton, L. Wei, and C. A. Ratanamahatana. Fast time series classification using numerosity reduction. In *Int. Conf. Machine learning*, pages 1033–1040. ACM, 2006.

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

[56] L. Ye and E. Keogh. Time series shapelets: a new primitive for data mining. In *Int. Conf. Knowledge Discovery and Data Mining*, pages 947–956. ACM, 2009.

[57] G. Zhang, B. E. Patuwo, and M. Y. Hu. Forecasting with artificial neural networks:: The state of the art. *Int. J. Forecasting*, 14(1):35–62, 1998.

[58] G. P. Zhang. Time series forecasting using a hybrid arima and neural network model. *Neurocomputing*, 50:159–175, 2003.

[59] Y. Zheng, Q. Liu, E. Chen, Y. Ge, and J. L. Zhao. Time series classification using multi-channels deep convolutional neural networks. In *Web-Age Inform. Management*, pages 298–310. Springer Science + Business Media, 2014.

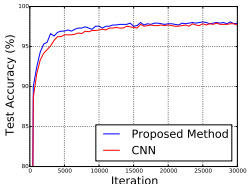


Figure 5. Comparison of test accuracies of a conventional CNN and the proposed method for Unipen 1a.

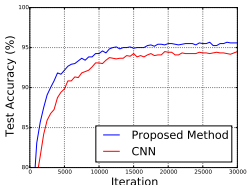


Figure 6. Comparison of test accuracies of a conventional CNN and the proposed method for Unipen 1b.

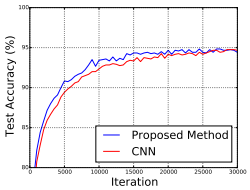


Figure 7. Comparison of test accuracies of a conventional CNN and the proposed method for Unipen 1c.

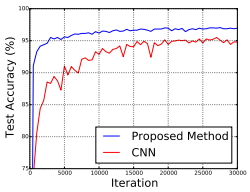


Figure 8. Comparison of test accuracies of a conventional CNN and the proposed method for UCI spoken Arabic digits.

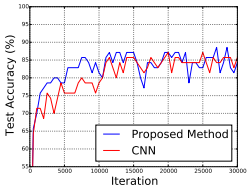


Figure 9. Comparison of test accuracies of a conventional CNN and the proposed method for UCI ADL accelerometer.

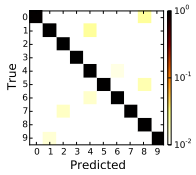


Figure 10. Confusion Matrix of test set of the proposed method for Unipen 1a.

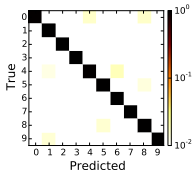


Figure 11. Confusion Matrix of test set of a conventional CNN for Unipen 1a.

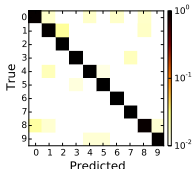


Figure 12. Confusion Matrix of test set of an LSTM for Unipen 1a.

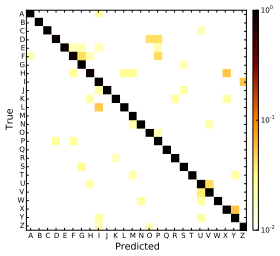


Figure 13. Confusion Matrix of test set of the proposed method for Unipen 1b.

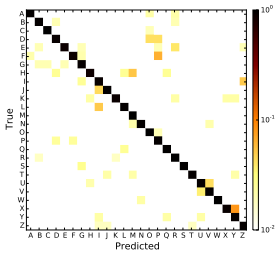


Figure 14. Confusion Matrix of test set of a conventional CNN for Unipen 1b.

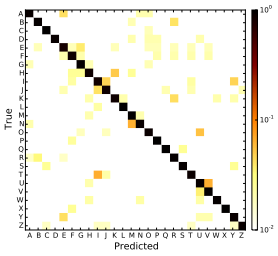


Figure 15. Confusion Matrix of test set of an LSTM for Unipen 1b.

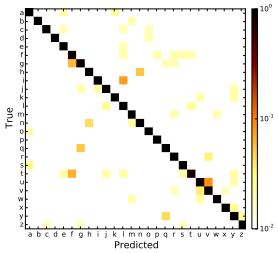


Figure 16. Confusion Matrix of test set of the proposed method for Unipen 1c.

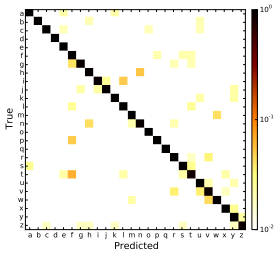


Figure 17. Confusion Matrix of test set of a conventional CNN for Unipen 1c.

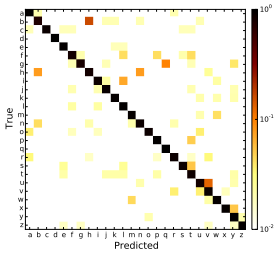


Figure 18. Confusion Matrix of test set of an LSTM for Unipen 1c.

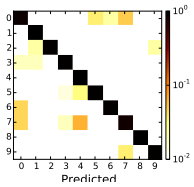


Figure 19. Confusion Matrix of test set of the proposed method for UCI spoken Arabic digits.

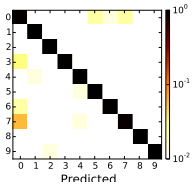


Figure 20. Confusion Matrix of test set of a conventional CNN for UCI spoken Arabic digits.

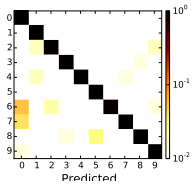


Figure 21. Confusion Matrix of test set of an LSTM for UCI spoken Arabic digits.

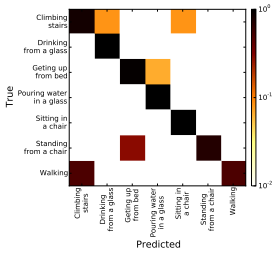


Figure 22. Confusion Matrix of test set of the proposed method for UCI ADL accelerometer.

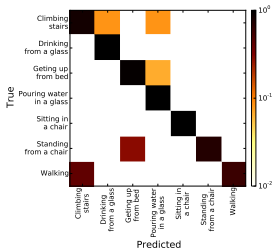


Figure 23. Confusion Matrix of test set of a conventional CNN for UCI ADL accelerometer.

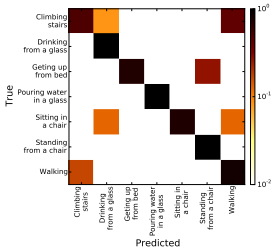


Figure 24. Confusion Matrix of test set of an LSTM for UCI ADL accelerometer.

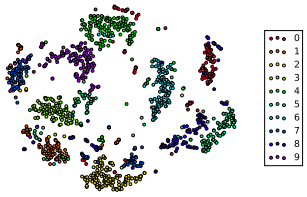


Figure 25. t-SNE of test set outputs of the convolutional layer of the proposed method for Unipen 1a.

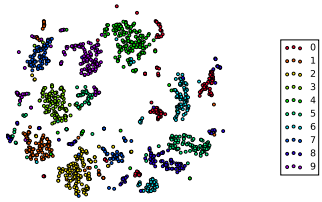


Figure 26. t-SNE of test set outputs of the convolutional layer of a conventional CNN for Unipen 1a.

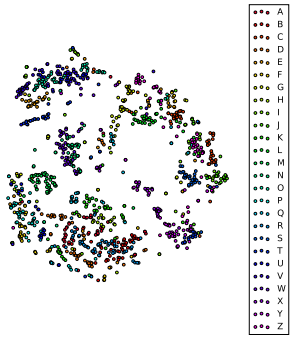


Figure 27. t-SNE of test set outputs of the convolutional layer of the proposed method for Unipen 1b.

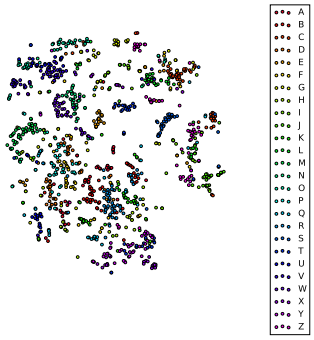


Figure 28. t-SNE of test set outputs of the convolutional layer of a conventional CNN for Unipen 1b.

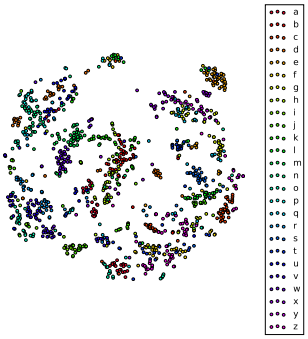


Figure 29. t-SNE of test set outputs of the convolutional layer of the proposed method for Unipen 1c.

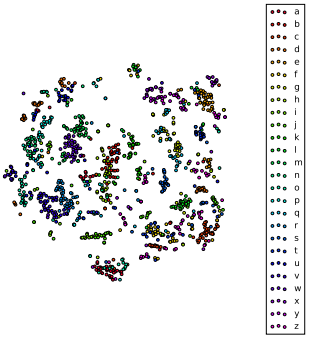


Figure 30. t-SNE of test set outputs of the convolutional layer of a conventional CNN for Unipen 1c.

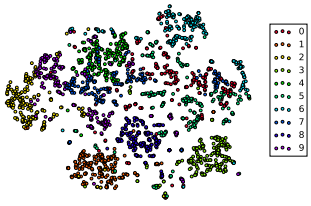


Figure 31. t-SNE of test set outputs of the convolutional layer of the proposed method for UCI spoken Arabic digits.

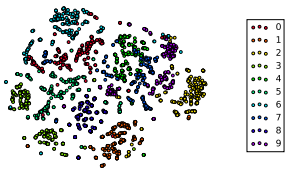


Figure 32. t-SNE of test set outputs of the convolutional layer of a conventional CNN for UCI spoken Arabic digits.

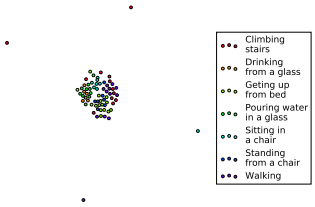


Figure 33. t-SNE of test set outputs of the convolutional layer of the proposed method for UCI ADL accelerometer.

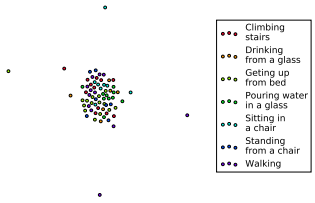


Figure 34. t-SNE of test set outputs of the convolutional layer of a conventional CNN for UCI ADL accelerometer.