DYNAMIC WEIGHT ALIGNMENT FOR TEMPORAL CONVOLUTIONAL NEURAL NETWORKS

Brian Kenji Iwana, Seiichi Uchida

Department of Advanced Information Technology, Kyushu University, Fukuoka, Japan

ABSTRACT

In this paper, we propose a method of improving temporal Convolutional Neural Networks (CNN) by determining the optimal alignment of weights and inputs using dynamic programming. Conventional CNN convolutions linearly match the shared weights to 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 the input of the convolutional layer. Specifically, the dynamic alignment overcomes issues such as temporal distortion by finding the minimal distance matching of the weights and the inputs under constraints. 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.

***Index Terms*—** Time series classification, convolutional neural network, dynamic programming, dynamic time warping

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 have achieved state-of-the-art results due to improvements in data availability and computational power [1]. Convolutional Neural Networks (CNN) [2] in particular have achieved the state-of-the-art results in many areas of image recognition, such as offline handwritten digit recognition [3], text digit recognition [4, 5], and object recognition [6, 7].

Most recent successes in time series recognition have been through the use of Recurrent Neural Networks (RNN) [8] and in particular, Long Short-Term Memory (LSTM) networks [9]. Typically, CNN-based models have been used in the image domain, however, they have also been used for time series patterns. A predecessor to CNNs, Time Delay Neural Networks (TDNN) [10, 11] used time-delay windows similar to the filters of CNNs. CNNs were also used to classify time series by embedding the sequences into vectors [12] and matrices [13, 14].

CNNs use sparsely connected shared weights that act as a feature extractor and maintain the structural aspects from the input. In particular, these shared weights are linearly aligned to each corresponding window value of the input. However, the linear alignment assumes that each element of the input window correspond directly to each weight of the filter in a one-to-one fashion. It is possible that there is a more optimal alignment of the shared weights and the input values.

We propose a method of finding that alignment using dynamic programming, namely Dynamic Time Warping (DTW) [15]. DTW estimates the globally minimal distance between two time series patterns by elastically matching elements using dynamic programming along a constrained path on a cost matrix. While DTW is traditionally used just as a distance measure, we exploit the elastic matching byproduct of DTW to align the weights of the filter to the elements of the corresponding receptive field to create more efficient 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 by dynamically matching the weights to similar input values. Using the discovered dynamic weight alignment, we create a nonlinear matching to create more effective convolutions. Second, we demonstrate the effectiveness of the proposed method on multiple time series datasets including: Unipen online handwritten character datasets, the UCI Spoken Arabic Digit dataset, and the UCI Activities of Daily Life dataset and perform a comparative study to reveal the benefits of the proposed weight alignment.

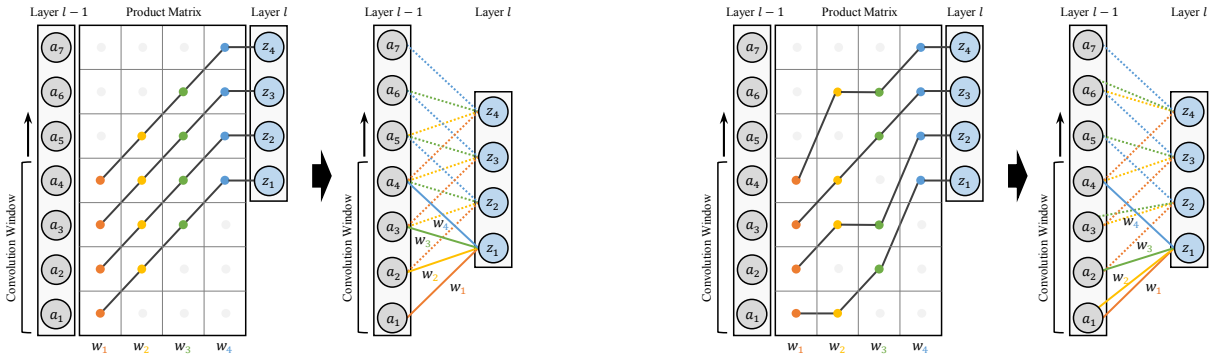
2. RELATION TO PRIOR WORK

Dynamic neural networks is an emerging field in neural model learning Dynamic Filter Networks (DFN) [16] use filter-generating networks to produce filters that are used depending on the input. Dynamic Convolutional Neural Networks (DCNN) [17] use dynamic k-Max Pooling to simplify CNNs for sentence modeling. Deformable Convolutional Networks [18] use deformable convolutions to relax the constraints of a traditional convolutional window. DTW-NNs [19] similarly use DTW as a nonlinear inner product for regular feed forward neural networks. The distinction between these models and the proposed method is that we use dynamic programming to estimate the optimal weight alignment within convolutions.

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. In this case, we define "optimal" as the globally minimal warping path determined by DTW. In other words, instead of the conventional linear inner product of a convolution, the convolutional filter weights and the input window values are dynamically matched to minimize the difference between similar features of the weights and the input values. Figure 1 demonstrates the difference between

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



(a) Convolution with linear weight alignment

(b) Convolution with dynamic weight alignment

Fig. 1: The comparison between a conventional linear convolution (a) and the proposed convolution with dynamic weight alignment (b). 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 dot is the product of the corresponding weight and input and the blue circle is the sum of the products.

a conventional convolutional layer with linear weight alignment and the proposed CNN with dynamic weight alignment.

3.1. Convolutional Neural Networks

A CNN is an artificial neural network which contains one or more convolutional layers. The key features of convolutional layers is that they have sparse connectivity and use parameter sharing. Specifically, 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 $z_j^{(l)}$ of a convolutional layer is defined as:

$$z_j^{(l)} = \sum_{i=0}^{I-1} w_i^{(l)} a_{i+j}^{(l-1)} + b^{(l)} \tag{1}$$

for each element j, where l is the convolutional layer, $l-1$ is the previous layer, i is the index of the filter, and I is the window size. We denote $w_i^{(l)}$, $a_{i+j}^{(l-1)}$, and $b^{(l)}$ as the shared weights, the previous layer activations, and the bias respectively. In other words, $z_j^{(l)}$ is the inner product of the shared weights $\mathbf{w}^l$ and each window of the previous layer $a_j^{(l-1)}, \ldots, a_{j+(I-1)}^{(l-1)}$. 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 noisy elements or feature translation and scale variance within the filter.

3.2. Dynamic Weight Alignment

The conventional inner product of a convolution acts much like a similarity function. Thus, the general idea is to align the weights so that there is a stronger activation to input windows that are similar but only slightly misaligned. To optimize the alignment of weights, we adopt a dynamic programming solution, specifically DTW.

3.2.1. *Dynamic Time Warping*

DTW is an asymmetric positive semi-definite similarity function that is 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 finds the total cost over an optimal warping path of a local cost matrix 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. Namely, 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'. Additionally, 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.

3.2.2. *Dynamic Weight Alignment with Shared Weights*

The forward pass calculation is done in two steps. First, DTW is calculated between the shared weights of each convolution and the receptive field window of the input. This is possible if we consider the weights of the convolution as the time series $\mathbf{p}$ and the window of the input as $\mathbf{s}$. The result is a mapping of the shared weights to the input values based on minimizing the L^2 distance between sequence elements.

Second, the convolution is calculated using the stored mapping. Namely, we propose using DTW to determine $\mathcal{M}_j$ and then calcu-

late the result of the convolution $z_j^{(l)}$:

$$z_j^{(l)} = \sum_{(i',j') \in \mathcal{M}_j} w_{i'}^{(l)} a_{j'}^{(l-1)} + b^{(l)}, \tag{3}$$

where $\mathcal{M}_j$ is the set of matched indices i' and j' corresponding to the index i of $\mathbf{w}^{(l)}$ and the index j in $\mathbf{a}^{(l-1)}$, respectively. When used in this manner, we create a nonlinear convolutional filter that acts as a feature extractor similar to using shapelets with DTW [20]. In addition, it is important to note that unlike a conventional CNN, the set of matched indicies $\mathcal{M}_j$ allows for duplicate and skipped values of $w_{i'}^{(l)}$ and $a_{j'}^{(l-1)}$.

The idea is that DTW will match similar features from the filter to the input and skip elements with a very high distance to the weights and perform small translations. Therefore, the process of aligning the weight using DTW is repeated for every stride of the convolution during all forward passes including during training and testing. Consequently, the alignment is only kept for the immediate forward and backward round and recalculated on the fly for subsequent iterations.

3.3. Backpropagation of Convolutions with Dynamic Weight Alignment

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

$$\frac{\partial C}{\partial w_i^{(l)}} = \sum_i \frac{\partial C}{\partial z_j^{(l)}} \frac{\partial z_j^{(l)}}{\partial w_i^{(l)}}, \tag{4}$$

where C is the loss function. In a conventional CNN, $w_i^{(l)}$ has a linear relationship to $z_j^{(l)}$, thus $\frac{\partial z_j^{(l)}}{\partial w_i^{(l)}}$ 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:

$$\frac{\partial C}{\partial w_i^{(l)}} = \sum_i \frac{\partial C}{\partial z_j^{(l)}} \frac{\partial \left(\sum_{(i',j') \in \mathcal{M}_j} w_{i'}^{(l)} a_{j'}^{(l-1)} + b^{(l)} \right)}{\partial w_i^{(l)}} \tag{5}$$

$$= \delta^{(l+1)} \sum_{(i',j') \in \mathcal{M}_j} a_{j'}^{(l-1)}. \tag{6}$$

where $\delta^{(l+1)}$ is the backpropagated error from the previous layer as determined by the chain rule.

4. EXPERIMENTS AND RESULTS

4.1. Datasets and Evaluation

We demonstrate the effectiveness of the proposed method by quantitatively evaluating the architecture and compare it to baseline methods for three diverse datasets.

The Unipen multi-writer 1a, 1b, and 1c datasets [21] are constructed from pen tip trajectories of isolated numerical digits, uppercase alphabet characters, and lowercase alphabet characters respectively. The UCI Spoken Arabic Digit Data Set [22] contains spoken Arabic digit patterns encoded using 13-frequency Mel-Frequency Cepstrum Coefficients (MFCC) in 10 classes. The UCI Activities of Daily Life (ADL) Recognition with Wrist-worn Accelerometer Data Set [23] is made of patterns from 7 classes of ADL actions. The Unipen and the UCI ADL datasets were divided into three sets for training, a test of 10% of the data, a training set of 90% of the data, and 50 patterns set aside from the training set for a validation set. The UCI Arabic data has a pre-defined division of the data with a speaker-independent training set and test set.

Table 1: Accuracy (%) on the evaluated datasets. The highest accuracy for each dataset is in bold.

	Unipen			UCI	
Method	1a	1b	1c	Arabic	ADL
Proposed	98.54	96.08	**95.92**	96.95	**90.0**
CNN	98.08	94.67	95.33	95.50	87.1
LSTM	96.84	92.31	89.79	96.09	81.4
SVM GDTW [24]	96.2	92.4	87.9	–	–
HMM CSDTW [25]	97.1	92.8	90.7	–	–
DTW-NN [19]	96.8	–	–	–	–
Google [26]	**99.2**	**96.9**	94.9	–	–
Tree Dist [27]	–	–	–	93.1	–
CHMM $\Delta(\Delta$MFCC) [28]	–	–	–	**98.4**	–
WNN [29]	–	–	–	96.7	–
GMM + GMR [23]	–	–	–	–	63.1
Decision Tree [30]	–	–	–	–	80.9

4.2. Architecture Settings

For the experiment, we implement a five-layer CNN. The first two hidden layers are convolutional layers with 50 nodes of the proposed dynamically aligned filters. In addition, we use batch normalization [31] on the results of the convolutional layers. The third and fourth 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.

The learning rate η_t at iteration t is defined as $\eta_t = \frac{\eta_0}{1+\alpha t}$, 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 $\alpha = 0.001$ for the convolutional layers and a static learning rate of 0.0001 between the fully-connected layers.

Given that the experimental datasets are made of sequences of different dimensions, the filters should correspond accordingly. The convolutional filters were of size 8×2 at stride 2, 6×13 at stride 2, and 12×3 at stride 4 for the Unipen datasets, the UCI Arabic dataset, and the UCI ADL dataset, respectively. A stride was used to reduce redundant information and decrease computation time. The experiment uses batch gradient decent with a batch size of 100, 50, and 5 for the three datasets respectively and for 60,000 iterations. The batch sizes were selected based on the size of the training sets and were chosen to iterate through epochs at generally the same rates. This is the reason for the very small batch of 5 used for the ADL dataset.

In the DTW implementation, we used the asymmetric slope constraint proposed by Itakura [32] and Euclidean distance as the local distance function $||\cdot||$ of Eq. (2).

4.3. Comparison Methods

We report classification results literature as well as evaluate the datasets on established state-of-the-art neural network methods.

To evaluate the proposed method, we compare the accuracy to current methods from literature. For the online handwritten character evaluations, we compare results from two classical methods, SVM GDTW [24] and HMM CSDTW [25], and two state-of-the-art neural network methods, DTW-NN [19] and Google [26]. For the spoken Arabic digits, there is one reported neural network solution using a WNN [29] as well as other models using a Tree Distribution model [27] and a Continuous HMM of the second-order derivative MFCC (CHMM $\Delta(\Delta$MFCC)) [28]. For the ADL dataset, we compare our results to the original dataset proposal [23] using a Gaussian Mixture Modeling and Gaussian Mixture Regression (GMM + GMR) and the best results of Kanna et al. [30] using a Decision Tree.

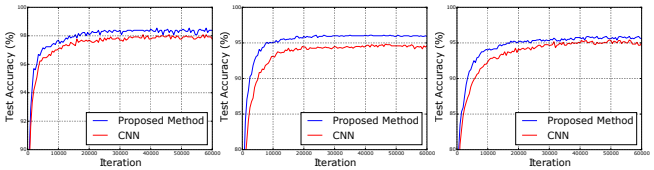
The evaluated baselines were designed to be direct comparisons for the proposed method. The LSTM is used as the established state-of-the-art neural network method for sequence and time series recognition and a traditional CNN is used as a direct comparison using standard convolutional layers. Both comparative 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, two fully-connected layers, and a softmax output layer was used. The second comparative evaluation was using a CNN with the same exact hyperparameters as the proposed method, but with standard convolutional nodes.

4.4. Results and Discussion

The results of the experiments are shown in Table 1. The results show that the proposed method surpassed all of the results of a conventional CNN as well as the LSTM. Furthermore, the results are competitive with the state-of-the-art methods despite many of them being tailored to the respective datasets and data types.

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 spacial 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.

The most important comparison is the conventional CNN with linearly aligned weights against the proposed method with dynamically aligned weights. In addition to the increased accuracy, we observed from Fig. 2 that compared to the conventional CNN, the proposed method achieves a higher accuracy during all parts of training



Unipen 1a Unipen 1b Unipen 1c

Fig. 2: Test accuracies of the Unipen online handwrittenof a conventional CNN and the proposed CNN with dynamic weight alignment.

but especially during the early stages. This indicates that the nonlinear alignment is able to optimize the weights efficiently.

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 by a CNN act like filter for feature extraction [2]. The purpose of using dynamically aligned weights is to warp the assignment of weights to their most similar corresponding inputs. In this way, noisy input values can be skipped and normally muted but relevant features are enhanced. This provides a more robust convolution.

4.5. Computational Complexity

In the case of the proposed method, the number of elements in the aligned sequences is equal to I and J, where I and J is the width of the filter and the input, respectively. Furthermore, the complexity of each DTW calculation is $O(IJ)$, which is required for every application of a convolutional filter. Thus, the computational complexity of the convolutional layer with dynamic weight alignment becomes $O(\frac{NIJ^2}{S})$, where N is the number of convolutional nodes and S is the stride. Compared with the standard convolution of a temporal CNN with a complexity of $O(\frac{NIJ}{S})$, this a relatively small increase in complexity compared to the overall network.

The per classification runtime for the traditional CNN was 0.036s, 0.092s, and 0.029s for the Unipen, ADL, and Spoken Arabic datasets, respectively. The proposed method had runtimes of 0.114s, 0.403s, and 0.078s, respectively. The networks were constructed in Python using Numpy with no GPU on a desktop computer with an Intel Xeon 2.6 GHz CPU. However, these speeds can be further optimized with the use of GPUs and deep learning libraries.

5. 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. In this way, the weights of the convolutional layer are aligned to maximize their relationship to the data from the previous layer. Furthermore, we show that the proposed model is able to tackle time series pattern recognition. We evaluated the proposed model on a variety of datasets to reach state-of-the-art results. This shows that the proposed method a viable feedforward neural network model for time series recognition and an effective method of optimizing the convolutional filter in CNNs. There is potential for this work to be extended to any CNN-based model.

6. REFERENCES

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

[2] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

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

[4] 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.*, 2016, pp. 464–472.

[5] 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, pp. 405–410.

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

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

[8] H. Jaeger, "Tutorial on training recurrent neural networks, covering bppt, rtrl, ekf and the "echo state network" approach," German National Research Center for Inform., Tech. Rep., 2002.

[9] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[10] 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.*, vol. 37, no. 3, pp. 328–339, 1989.

[11] K. J. Lang, A. H. Waibel, and G. E. Hinton, "A time-delay neural network architecture for isolated word recognition," *Neural Networks*, vol. 3, no. 1, pp. 23–43, 1990.

[12] Y. Zheng, Q. Liu, E. Chen, Y. Ge, and J. L. Zhao, "Time series classification using multi-channels deep convolutional neural networks," in *Int. Conf. Web-Age Inform. Management*, 2014, pp. 298–310.

[13] N. Razavian and D. Sontag, "Temporal convolutional neural networks for diagnosis from lab tests," *arXiv preprint*, 2015, https://arxiv.org/abs/1511.07938.

[14] 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.*, 2015, pp. 3995–4001.

[15] H. Sakoe and S. Chiba, "Dynamic programming algorithm optimization for spoken word recognition," *IEEE Trans. Acoustics, Speech, and Sig. Process.*, vol. 26, no. 1, pp. 43–49, 1978.

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

[17] N. Kalchbrenner, E. Grefenstette, and P. Blunsom, "A convolutional neural network for modelling sentences," in *Annu. Meeting Assoc. Computational Linguistics*, 2014, pp. 655–665.

[18] J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, and Y. Wei, "Deformable convolutional networks," in *IEEE Conf. Comput. Vision and Pattern Recognition*, 2017, pp. 764–773.

[19] 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*, 2016, pp. 265–270.

[20] L. Ye and E. Keogh, "Time series shapelets: a new primitive for data mining," in *Int. Conf. Knowledge Discovery and Data Mining*, 2009, pp. 947–956.

[21] 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*, vol. 2, 1994, pp. 29–33.

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

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

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

[25] C. Bahlmann and H. Burkhardt, "The writer independent online handwriting recognition system frog on hand and cluster generative statistical dynamic time warping," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 3, pp. 299–310, 2004.

[26] D. Keysers, T. Deselaers, H. A. Rowley, L.-L. Wang, and V. Carbune, "Multi-language online handwriting recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 6, pp. 1180–1194, 2017.

[27] N. Hammami and M. Bedda, "Improved tree model for arabic speech recognition," in *Int. Conf. Comp. Sci. and Inform. Technology*, vol. 5, 2010, pp. 521–526.

[28] 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*, 2012, pp. 1–5.

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

[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*, vol. 4, no. 3, pp. 1406–1413, 2016.

[31] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint*, 2015, https://arxiv.org/abs/1502.03167.

[32] F. Itakura, "Minimum prediction residual principle applied to speech recognition," *IEEE Trans. Acoustics, Speech, and Sig. Process.*, vol. 23, no. 1, pp. 67–72, 1975.