2018 24th International Conference on Pattern Recognition (ICPR)
Beijing, China, August 20-24, 2018

Multi-Frequency Decomposition with Fully Convolutional Neural Network for Time Series Classification

Yongming Han, Shuheng Zhang, Zhiqiang Geng*
College of Information Science & Technology, Beijing University of Chemical Technology
Engineering Research Center of Intelligent PSE, Ministry of Education in China
Beijing, China
*E-Mail: gengzhiqiang@mail.buct.edu.cn

***Abstract*—Fully convolutional neural network (FCN) has achieved state-of-the-art performance in the task of time series classification without any heavy preprocessing. However, the FCN cannot effectively capture features of different frequencies. Therefore, this paper proposed a novel FCN structure based on the multi-frequency decomposition (MFD) method. In order to extract more features of different frequencies, the MFD based on real fast Fourier transform (RFFT) is set as a layer of the FCN to decompose the original signal into n sub-signals of different frequency bands. And then the improved FCN fuse those features of different frequencies together to obtain time series classification. Finally, compared with the existing state-of-the-art methods, the proposed method is effectively verified through some datasets in UCR Time Series Classification archive.**

***Keywords*—Fully Convolutional Neural Network; Time Series Classification; Fourier Transform; Multi-Frequency decomposition**

I. Introduction

Time is a very important attribute in our world. A lot of time series data are produced every time and everywhere in reality, such as weather readings, financial recordings and industrial process [1]. Therefore, time series data analysis and classification is one of the most important tasks and has been widely concerned in the past few decades. Time series classification can be expressed as $S^i \rightarrow y^i$, S^i represent i-th series of length l: $S^i = (s_1^i, s_2^i, \cdots, s_l^i)$ is associated with a category $y^i \in \{0, 1, \cdots c-1\}$, c is the number of categories. How to classify new and unlabeled series based on the labeled is the main goal [1].

Nowadays, a large number of time series classification methods have been proposed. The distance-based method used the distance measure (e.g. Euclidean distance) to compute the similarity between two series, and used classifiers (e.g. SVM, K-NN, Decision Tree [2]) to get classification results. Dynamic time warping (DTW) [3] used a much more robust measure to replace the Euclidean distance and found the shortest path between two sequences as the distance through the dynamic programming. The combination of DTW and k-NN got excellent performance. Since the DTW did not consider the relative importance of the phase difference between the reference point and the test point, Jeong et al. proposed a method to penalize points with higher phase difference [4].

Shapelets, originally used for image analysis, were introduced into time series analysis for finding similar subsections between sequences. Shapelets are sub sequences of time series which can maximally represent the category. And a method based on information gain was proposed to find shapelets [5]. Hills et al. found the best k shapelets, and got k features by computing the distance between series and k shapelets [6], which was equivalent to embedding the original series into a k-dimensional space. Grabocka et al. proposed a learnable-shapelets method by converting the learning process to a mathematical formulation without candidates through a classification objective function [7].

Feature-based methods are used to extract features that represent time series patterns. Lin et al. transformed the time series into a discrete string of fixed length, and used those characters to calculate the distance [8]. Baydogan et al. selected multi sub-sequences from random locations and partition with random length to capture the local information [9]. Schäfer transformed the sequence to a string by the time window and the quantisation, and got histograms that represent substructures of sequence through Symbolic Fourier Approximation (SFA) [10]. Ensemble methods improved the performance for time series classification by integrating multiple classifiers with 35 different classifiers [11].

Recently, many deep learning methods provided an end-to-end approach that greatly reduced the preprocessing for time series classification. Cui et al. used a multi-scale CNN structure to transform the sequences into multi-scale branch and multi-frequency branch [12]. Many image-based methods encoded series into images and classify images [13], [14]. Hatami et al. converted time series to Recurrence Plots, and used the CNN for feature extraction and classification [15]. Compared with the deep multilayer perceptions (MLP) and the residual networks (ResNet), the fully convolutional networks (FCN) could obtain an excellent performance [16]. Long short term memory (LSTM) was used to extract the sequence feature for enhancing the performance of the FCN [17].

The existing methods based on the deep learning for time

978-1-5386-3788-3/18/$31.00 ©2018 IEEE

series classification are mostly drawn from its application on computer vision. However, there are some differences between pictures and signals. The pictures follow the nearest neighbor rule, and the neighboring pixels have a higher degree of correlation. Meanwhile, the signals can be regarded as the superposition of different signals, and the features are hidden in different presentation levels. Therefore, these methods above need to be modified to analyze the signals.

In this paper, we propose a multi-frequency decomposition (MFD) method based on real fast Fourier transform (RFFT) that can be added to neural networks as a layer. And then an improved FCN structure is designed to enhance the performance of fully convolutional networks. Moreover, more abundant features in different frequencies are obtained. Finally, compared with the existing state-of-the-art methods, the proposed method is effectively verified through some datasets in UCR Time Series Classification archive.

II. Proposed Method

A. Multi-Frequency decomposition based on RFFT

Fast Fourier transform (FFT) is one of the most important tools in modern digital processing, which efficiently computes the Discrete Fourier transform (DFT) and converts the discrete signal from the time domain to the frequency domain. Meanwhile, the signal is decomposed into multiple period signals of different frequencies, which represent the components of the original signal at different frequencies. Let $x \in \mathbb{C}^N$ be the complex-value signal of length N, and the FFT maps it to a complex-value signal $X \in \mathbb{C}^N$ of length-N at frequency domain.

$$X_n = \sum_{k=0}^{N-1} x_k W_N^{nk} \tag{1}$$

Where $n = 0, 1, \cdots, N-1$ represent different frequency, and $W_N = e^{-j(2\pi/N)}$. And the inverse transform of the FFT (IFFT) can be obtained by Eq. 2.

$$x_k = \sum_{n=0}^{N-1} X_n W_N^{-kn} \tag{2}$$

Although the FFT and the IFFT are designed to compute Fourier transform of a complex signal, the original signal is real in many applications [18]. The real-valued DFT of length-N can be computed using a length-$(N/2)$ complex-valued FFT [19]. With symmetry of complex-value, the DFT is divided into two parts, an even-indexed one and an odd-indexed. And then the full-length DFT is produced by combining them together. Let $x \in \mathbb{R}^N$, and RFFT is computed as follow:

$$\begin{aligned} X_n &= \sum_{k=0}^{N-1} x_k W_N^{nk} \\ &= \sum_{k=0}^{N-1} x_{2n} W_{N/2}^{nk} + W_N^k \sum_{n=0}^{N/2-1} x_{2n+1} W_{N/2}^{nk} \end{aligned} \tag{3}$$

The existing methods based on the deep learning only operate the signal in the time domain, and extract sequence features with temporal convolutions. However, the MFD can decompose the original signal into n sub-signals of different frequency bands. The output from the front layer of the neural network is accepted as the input, and each sequence of the input is decomposed into n sub-sequences, which is set as the output of the next layer. Therefore, the MFD extends the ability of the neural network to process the sequence data.

The inverse transform with different frequency components of the FordB dataset in UCR TSC Archive by using the MFD is shown in Fig. 1.

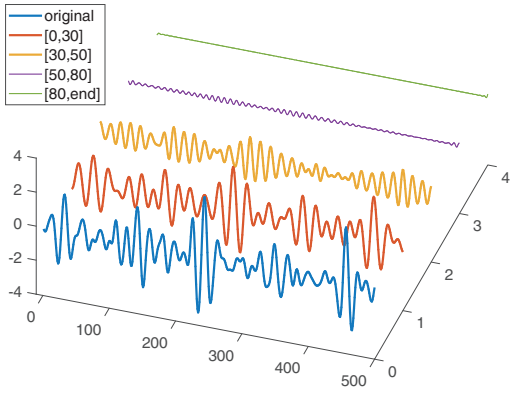


Fig. 1: Multi-Frequency Decomposition of FordB dataset

The sub-sequences of different frequency bands can represent the features of the original sequence at different frequencies. With the powerful feature extraction capability of neural networks, more abundant features in the sequence can be extracted. The MFD is designed as a layer in neural networks, and gradients can be calculated and propagated between layers.

Algorithm 1 Multi-Frequency Decomposition

Input:

$T \in \mathbb{R}^{L \times C}$: Output of the front layer

n: The number to be decomposed

Output: $H \in \mathbb{R}^{L \times n \times C}$

for each length-L signal $t \in T$ **do**
  $c \leftarrow RFFT(t)$, c is a complex vector of length-$(L/2)$
  $P \leftarrow$ Divide vector c into n parts
  for each part $p \in P$ **do**
    $q_i = \begin{cases} p_i & i \in index(p) \\ 0 & i \notin index(p) \end{cases}$, index $i \in [0, 1, \cdots, \frac{L}{2} - 1]$
    $r \leftarrow IRFFT(q)$
  end for
  Concatenate r together, and get $R \in \mathbb{R}^{L \times n}$
end for
Concatenate R together, and get the output $H \in \mathbb{R}^{L \times n \times C}$

The detailed procedures of the MFD are described in Algorithm 1. The input of the MFD layer is the output tensor $T \in \mathbb{R}^{L \times C}$ of the front layer which represents C different

sequences of length L. First, each sequence is transformed to a length-$(L/2)$ complex vector c at frequency domain by using the RFFT. And then, each c is divided into n parts, that represent components at different frequency. Meanwhile, each component is restored to a complex signal of length $L/2$, and the index of each part is remained unchanged and others are padded with 0. The inverse Real Fourier Transform (IRFFT) is performed on each complex signal to get a real signal of length L. Finally, all the transformed signals are concatenated together to get the output H of this layer, $H \in \mathbb{R}^{L\times n\times C}$. The output of the MFD is used as the input of the convolutional layer to extract features after frequency decomposition and enhance the ability of feature extraction.

B. Improved Fully Convolutional Networks Architectures

The FCN is used for the semantic segmentation on images by Long et al. [20]. A dense prediction needs to be accomplished for getting the category of each pixel in pictures. The traditional CNN for deposing the classification task has a fully-connected layer behind the convolutional layer, which maps the output of the last convolutional layer to a one-dimensional vector for classification. However, the final output of the network can be seen as a picture of categories, so the traditional CNN cannot accomplish the semantic segmentation on images. Fully-connected layer is directly replaced by the convolutional layer to solve this problem [20], and this structure is called fully convolutional network. Global Average Pooling replaces the traditional fully-connected layer in the CNN [21]. Fully-connected layers occupy a large part of parameters in neural networks, and are prone to overfitting. Therefore, the convolution layers are treated as a feature extractor to get feature maps. And then the average of each feature map is computed by Global Average Pooling. Finally, the resulting vector is fed directly into $softmax$ layer for classification to improve the generalization ability of the network. The Global Average Pooling is also computed to get a generic localization deep representation [22]. A single fully-connected layer is added between the Global Average pooling layer and the $softmax$ layer, and Class Activation Maps (CAM) is computed with the weights of this fully-connected layer.

$$M_c(x,y) = \sum_k w_k^c f_k(x,y) \tag{4}$$

where, c is category, $f_k(x,y)$ is the k-th feature map of the last convolutional layer, w_k^c is the weight corresponded by category c and map k in this fully-connected layer. The CAM of time series classification is also calculated to find out the contributing region in the raw data for the specific labels [16]. Combined with the global average pooling to the FCN for time series classification, the 1D convolution layers are used to extract feature in time series data and get the state-of-the-art result.

The MFD layer is added to the FCN for getting more abundant features of the sequence. And the representation of signals at different frequencies is easily exploited by neural networks. And then those signals of different frequencies are used as the input of the convolutional layer to extract features. Finally, all features are fused together for classification.

The improved network structure is described in detail as shown in Fig. 2.There are three convolutional layers for feature extraction, one MFD layer for frequency decomposition, one Global Average Pooling for feature mapping, and the last $softmax$ layer for classification. Each convolutional layer is followed by a batch normalization layer [23] and a $ReLU$ activation layer [24]. The detailed interpretation of the improved FCN structure is shown as below.

0) **Input:** The input of the whole network structure is the time series samples with size $L \times S$. In the traditional time series data, $S = 1$.
1) **Conv1D:** The first layer of the entire network structure is a 1-d convolution, which is used to process the original 1-d series data. The kernel size of this layer is 8 and the number of kernels is C. Therefore, The output of this layer is a tensor $L \times C$.
2) **MFD:** The second layer is the MFD layer, which decomposes the signal processed by the first layer into n sub-signals of different frequencies. At this layer, the tensor of $L \times C$ is converted to the one of $L \times n \times C$. This process can be considered that a one-dimensional signal of length L is converted to a two-dimensional signal of size $L \times n$ and the later can be treated as a two-dimensional signal. And then a batch normalization is followed by the MFD layer. The size of the sub-signals with the MFD is different, which needs to be normalized to enhance the features for processing by the next layer.
3) **Conv2D:** The third layer is a 2-d convolution. A little trick is used to implement the functions of 1-d convolution on different dimensions by 2-d convolution. In this layer, the size of kernel is $(k \times 1)$, that is, the size of the second dimension of the kernel is fixed to 1. Meanwhile, the frequencies of those signals are maintained, so the signals are processed at different frequencies.
4) **Conv2D:** The forth layer is also a 2-d convolution. But, the second dimension of the kernel is n which is the same as the number of components decomposed by the MFD. Therefore, the fusion of signals at different frequencies is achieved by this little trick, and a tensor of size $L\times 1\times C'$ is obtained as the output. Until now, the feature extraction process has been completed and C' feature maps are obtained.
5) **Global Average Pooling and Softmax:** Finally, the Global Average Pooling layer is used to map those features to 1-d vector of size C' by computing the average of each feature map [22]. And a fully-connected layer with weight matrix $N_c \times C'$ and activation function $softmax$ is used for classification. N_c represents the number of categories.

This proposed approach enhances the ability of the neural network to process the sequence data, so that the desired network structure can be designed flexibly, and be used as a part of the new network structure for the feature extraction.

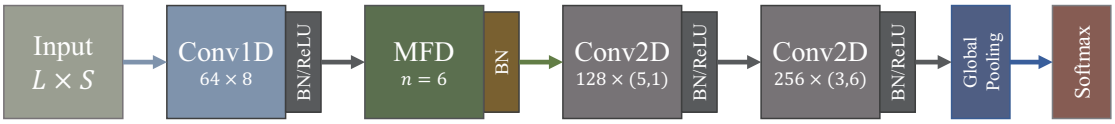
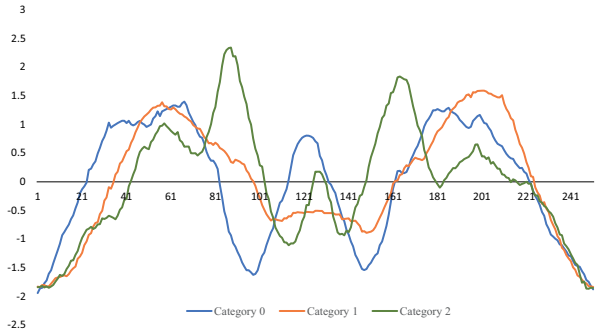


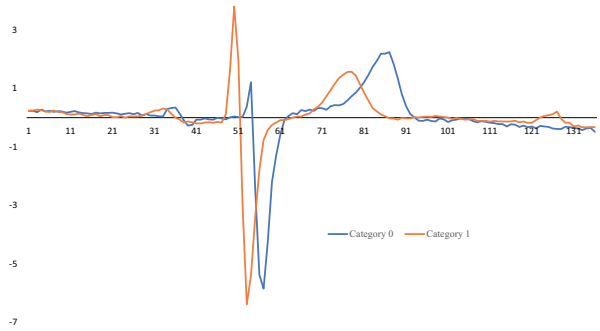
Fig. 2: The network structure of the improved FCN

TABLE I: Some Datasets in UCR TSC Archive

Dataset	L_s	N_c	N_{train}	N_{test}
Adiac	176	37	390	391
Coffe	286	2	28	28
ArrowHead	251	3	36	175
ECG200	96	2	100	100
FordA	500	2	1320	3601
FordB	500	2	810	3636
Beef	470	5	30	30
Haptics	1092	5	150	308
BeetleFly	512	2	20	20
BirdChicken	512	2	20	20
CinCECGTorso	1639	4	40	1380
ECGFiveDays	136	2	23	861



(a) ArrowHead



(b) ECGFiveDays

Fig. 3: Sequences of different categories in ArrowHead and ECGFiveDays datasets

III. Experiments and Results

A. Data sets

In this paper, the proposed method is validated with data sets in UCR Time Series Classification Archive [25]. There are 85 different datasets for time series classification in UCR TSC Archive, and those datasets are collected from completely different fields. The number of categories, the number of samples and the sequence length are different for each dataset. Furthermore, each dataset is split into a TRAIN partition and a TEST partition by default in its official website. Each algorithm uses the same training set and testing set to increase the comparability of algorithms. And 12 different datasets listed in Table I are selected to validate the proposed method. L_s, N_c, N_{train}, N_{test} represent sequence length, the number of categories, the size of training set and the size of testing set, respectively. Some sequences of different categories in ArrowHead and ECGFiveDays datasets are shown in Fig. 3.

B. Preprocessing and Evaluation measure

1) Preprocessing: Traditional methods require heavy pre-processing to extract features on the raw sequence data. The deep learning completes an end-to-end process [16], [17]. The only preprocessing in our experiment is normalization as shown in Eq. 5.

$$X_i' = \frac{X_i - \mu_{train}}{\sigma_{train}} \tag{5}$$

where, X_i and X_i' represent the original sequence and the normalized sequence, respectively. And μ and σ represent mean and variance of the train set, respectively.

2) Evaluation Measure: The number of samples in different categories is different in many cases. Therefore, the simple precision cannot fully represent the performance of algorithms. The ratio of categories in different datasets is described in Table II. The ratio is computed by Eq. 6.

$$ratio_c = \frac{N_c}{N} \cdot |c| \tag{6}$$

where, N_c is the number of samples in category c, and N is the number of samples in the whole train set or test set. $|c|$ is the number of categories.

Mean Per-Class Error (MPCE) [16] is used to validate the proposed model, and is defined as the arithmetic mean of per class error (PCE).

$$\begin{aligned} PCE_c &= \frac{e_c}{N_c} \\ MPCE &= \frac{1}{C}\sum_c PCE_c \end{aligned} \tag{7}$$

Note: e_k denotes the error rate of category c, and C is the

TABLE II: The ratio of categories in different datasets

Dataset	Category	Train	Test
ArrowHead	0	1	1.183
	1	1	0.907
	2	1	0.907
ECGFiveDays	0	1.217	0.994
	1	0.783	1.006
FordB	0	0.993	1.023
	1	1.007	0.977

number of categories in dataset. The MPCE considers the ratio of each category, and can make accurate measurements even if the number of samples in each category varies widely.

C. Implementation

The proposed model is implemented with *keras* [26] and *TensorFlow* [27] as backend. The detail structure is depicted in Fig. 2, and the model is performed at all datasets with parameters unchanged. *Adam* [28] is used to optimize the proposed model with initialized learning rate $1e{-}3$, and can be reduced during training with *ReduceLROnPlateau* (patience = 30, factor = $1/\sqrt[3]{2}$ [17], min_lr = $1e{-}5$). A L_2 regularization term $\lambda||w||_2^2$ with $\lambda = 1e{-}3$ is added in each convolution layer to improve the generalization ability. Finally, the batch size is set with 20, and each sequence is decomposed into 6 equal-sized and continuous frequency segments for simplicity in the MFD layer. [1]

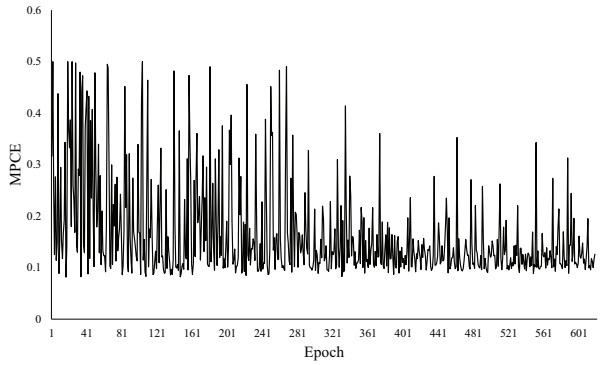
TABLE III: Performance comparison of different models

Dataset	**MFD-FCN**	FCN	ResNet
Adiac	**0.138**	0.143	0.174
Coffe	**0**	0	0
ArrowHead	0.148	**0.12**	0.183
ECG200	0.121	**0.1**	0.13
FordA	0.076	0.094	**0.072**
FordB	**0.078**	0.117	0.1
Beef	**0.1**	0.25	0.233
Haptics	0.453	**0.449**	0.494
BeetleFly	**0.05**	0.05	0.1
BirdChicken	**0**	0.05	0.1
CinCECGTorso	**0.129**	0.187	0.229
ECGFiveDays	**0**	0.015	0.045

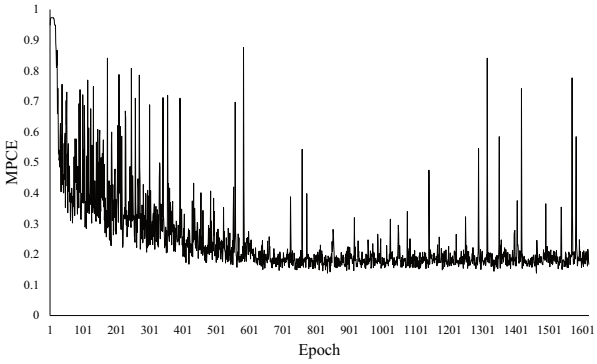
D. Result and Analysis

The results are showed in Table III. And the value of the MPCE on validated set of Adiac and FordB datasets during training are described in Fig. 4. The proposed method improves the performance of the FCN in time series classification. Among the 12 datasets, the performance of the proposed model is higher than the other two in 8 datasets, and second only to the highest results in the other 4 datasets. The improved FCN structure is higher than the original FCN structure in 9 datasets. Meanwhile, the results are very volatile during the training process. With the reduction of the learning rate by *ReduceLROnPlateau*, the network eventually tends to stable,

[1]The codes are available at https://github.com/zsh965866221/MFD-FCN



(a) MPCE of FordB dataset



(b) MPCE of Adiac dataset

Fig. 4: MPCE on validated set of some datasets during training

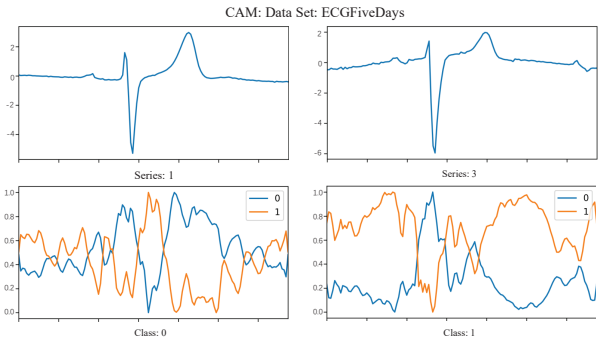


Fig. 5: The CAM on ECGFiveDays data set

but it is not efficient. Therefore, the stability of the network needs to be improved.

The CAM in section II-B is calculated to indicate the contributing region in the raw data for the specific labels. The result of the CAM of ECGFiveDays data set is shown in fig. 5. There are 2 different classes in ECGFiveDays data set, and we choose two sequences of different classes to show the performance of the CAM. The two pictures above are the raw sequence, and the following is the corresponding CAM, which represent the contribution of different classes of sequence at different locations. It can be seen from fig. 5 that, Class 0 is more inclined to the position where the original sequence has

large fluctuations, while Class 1 tends to be the flat ones.

IV. Conclusion and Future Work

This paper proposes a novel FCN structure based on the MFD method. The MFD based on the RFFT is added to the FCN as a layer to decompose the original sequence to subsequences of different frequencies. And then more features of different levels in the sequence data are extracted. Moreover, the improved FCN is used to classify time series and improve the classification performance. Finally, through some datasets in UCR Time Series Classification archive, the experimental results show that the MFD layer plays an important role in the feature extraction of the sequence data, meanwhile, this proposed model can achieve the significant improvement in the time series classification. The results of the CAM show that the added MFD layers do not change the transmission of category features in neural networks at different locations in the raw sequence.

In our future work, we will try to solve the problem of the volatility shown in Fig. 4 by using Bayesian approach [29]. Moreover, we can extend neural networks to the complex expression [30], and fuse Fourier transform in neural networks using the spectral representation [31].

V. Acknowledgment

This research was partly funded by National Natural Science Foundation of China (61673046, 61603025), the Natural Science Foundation of Beijing, China (4162045) and the National Key Research and Development Program of China (2017YFC1601800).

References

[1] E. Keogh and S. Kasetty, "On the need for time series data mining benchmarks: a survey and empirical demonstration," *Data Mining and knowledge discovery*, vol. 7, no. 4, pp. 349–371, 2003.

[2] N. M. Nasrabadi, "Pattern recognition and machine learning," *Journal of electronic imaging*, vol. 16, no. 4, p. 049901, 2007.

[3] E. Keogh and C. A. Ratanamahatana, "Exact indexing of dynamic time warping," *Knowledge and information systems*, vol. 7, no. 3, pp. 358–386, 2005.

[4] Y.-S. Jeong, M. K. Jeong, and O. A. Omitaomu, "Weighted dynamic time warping for time series classification," *Pattern Recognition*, vol. 44, no. 9, pp. 2231–2240, 2011.

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

[6] J. Hills, J. Lines, E. Baranauskas, J. Mapp, and A. Bagnall, "Classification of time series by shapelet transformation," *Data Mining and Knowledge Discovery*, vol. 28, no. 4, pp. 851–881, 2014.

[7] J. Grabocka, N. Schilling, M. Wistuba, and L. Schmidt-Thieme, "Learning time-series shapelets," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2014, pp. 392–401.

[8] J. Lin, E. Keogh, L. Wei, and S. Lonardi, "Experiencing sax: a novel symbolic representation of time series," *Data Mining and knowledge discovery*, vol. 15, no. 2, pp. 107–144, 2007.

[9] M. G. Baydogan, G. Runger, and E. Tuv, "A bag-of-features framework to classify time series," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 11, pp. 2796–2802, 2013.

[10] P. Schäfer, "The boss is concerned with time series classification in the presence of noise," *Data Mining and Knowledge Discovery*, vol. 29, no. 6, pp. 1505–1530, 2015.

[11] A. Bagnall, J. Lines, J. Hills, and A. Bostrom, "Time-series classification with cote: the collective of transformation-based ensembles," *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 9, pp. 2522–2535, 2015.

[12] Z. Cui, W. Chen, and Y. Chen, "Multi-scale convolutional neural networks for time series classification," *arXiv preprint arXiv:1603.06995*, 2016.

[13] Z. Wang and T. Oates, "Imaging time-series to improve classification and imputation," *arXiv preprint arXiv:1506.00327*, 2015.

[14] ——, "Encoding time series as images for visual inspection and classification using tiled convolutional neural networks," in *Workshops at the Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[15] N. Hatami, Y. Gavet, and J. Debayle, "Classification of time-series images using deep convolutional neural networks," *arXiv preprint arXiv:1710.00886*, 2017.

[16] Z. Wang, W. Yan, and T. Oates, "Time series classification from scratch with deep neural networks: A strong baseline," *arXiv preprint arXiv:1611.06455*, 2016.

[17] F. Karim, S. Majumdar, H. Darabi, and S. Chen, "Lstm fully convolutional networks for time series classification," *arXiv preprint arXiv:1709.05206*, 2017.

[18] H. V. Sorensen, D. Jones, M. Heideman, and C. Burrus, "Real-valued fast fourier transform algorithms," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 35, no. 6, pp. 849–863, 1987.

[19] J. W. Cooley, P. Lewis, and P. Welch, "The fast fourier transform algorithm: Programming considerations in the calculation of sine, cosine and laplace transforms," *Journal of Sound and Vibration*, vol. 12, no. 3, pp. 315–337, 1970.

[20] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 3431–3440.

[21] M. Lin, Q. Chen, and S. Yan, "Network in network," *arXiv preprint arXiv:1312.4400*, 2013.

[22] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba, "Learning deep features for discriminative localization," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2921–2929.

[23] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International Conference on Machine Learning*, 2015, pp. 448–456.

[24] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th international conference on machine learning (ICML-10)*, 2010, pp. 807–814.

[25] Y. Chen, E. Keogh, B. Hu, N. Begum, A. Bagnall, A. Mueen, and G. Batista, "The ucr time series classification archive," *URL www. cs. ucr. edu/~ eamonn/time_series_data*, 2015.

[26] F. Chollet *et al.*, "Keras," 2015.

[27] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin *et al.*, "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," *arXiv preprint arXiv:1603.04467*, 2016.

[28] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[29] B. Carpenter, A. Gelman, M. Hoffman, D. Lee, B. Goodrich, M. Betancourt, M. A. Brubaker, J. Guo, P. Li, and A. Riddell, "Stan: A probabilistic programming language," *Journal of Statistical Software*, vol. 20, pp. 1–37, 2016.

[30] C. Trabelsi, O. Bilaniuk, D. Serdyuk, S. Subramanian, J. F. Santos, S. Mehri, N. Rostamzadeh, Y. Bengio, and C. J. Pal, "Deep complex networks," *arXiv preprint arXiv:1705.09792*, 2017.

[31] O. Rippel, J. Snoek, and R. P. Adams, "Spectral representations for convolutional neural networks," in *Advances in Neural Information Processing Systems*, 2015, pp. 2449–2457.