2019 International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)

Gated Recurrent Neural Networks Empirical Utilization for Time Series Classification

Nelly Elsayed† Anthony S. Maida† Magdy Bayoumi⋆
†*School of Computing and Informatics*
⋆*Electrical and Computer Engineering Department*
†⋆*University of Louisiana at Lafayette*
†⋆Lafayette, Louisiana, USA

***Abstract*—Hybrid LSTM-Fully Convolutional Networks (LSTM-FCN) for time series classification has produced state-of-the-art classification results on univariate time series. This paper shows empirically that replacing the LSTM with a gated recurrent unit (GRU) to create a hybrid GRU fully convolutional network (GRU-FCN) can offer even better performance on many time series datasets. This resulted GRU-FCN model outperforms the state-of-the-art classification performance in many univariate time series datasets. In addition, since the GRU uses a simpler architecture than the LSTM, it has a simpler hardware implementation and fewer arithmetic components compared to the LSTM-based models.**

***Index Terms*—GRU-FCN, GRU, FCN, LSTM, time series classification, convolutional neural networks**

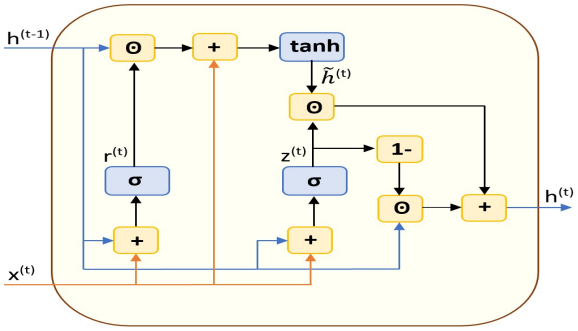

Fig. 1: Block architecture for an unrolled GRU.

I. Introduction

A time series (TS) is a sequence of data points obtained at successive equally spaced time points, ordinarily in a uniform interval time domain [1]. TSs are used in several research and industrial fields where temporal analysis measurements are involved such as in signal processing [2], pattern recognition [3], mathematics [1], psychological and physiological signals analysis [4], [5], earthquake prediction [6], weather readings [7], and statistics [1].

There have been several approaches to time series classification. The distance-based classifier based on the k-nearest neighbor (KNN) algorithm is considered a baseline technique for time series classification. Mostly, it uses Euclidean or Dynamic Time Wrapping (DTW) as a distance measure [8]. Feature-based time series classifiers are also widely used such as the bag-of-SFA-symbols (BOSS) [9] and the bag-of-features framework (TSBF) [10] classifiers. Ensemble-based classifiers combine separate classifiers into one model to reach a higher classification accuracy such as the elastic ensemble (PROP) [11], the shapelet ensemble (SE) [12], and the collective of transform-based ensembles (COTE) [12] classifiers. Convolutional neural network (CNN) based classifiers have advantages over other classification methods because CNNs provide the classifier with a preprocessing mechanism within the model. Examples are the multi-channel CNN (MC-CNN) classifier [13], the multi-layered preceptron (MLP) [4], the fully convolutional network (FCN) [4] and, specifically, the residual network (ResNet) [4]. The present paper focuses on the recurrent neural network based classification approaches such as LSTM-FCN [5], ALSTM-FCN [5], These models combine both CNNs and long short-term memory (LSTM) models to provide the classifier with both feature extraction and time dependencies through the dataset during the classification process. These models use some attention, squeeze, and masking algorithms to support LSTM learning due to its complex structure and data requirements.

This paper studies whether the use of gated-recurrent units (GRUs) can improve the hybrid classifiers listed above. We create the GRU-FCN by replacing the LSTM with a GRU in the LSTM-FCN [5] and keeping the rest components of the model unchanged to achieve a fair comparison between GRU and LSTM respecting the univariate time series classification tasks. Like the LSTM-FCN, our model does not require feature engineering or data preprocessing before the training or testing stages. The GRU has the ability to learn the temporal dependencies within the dataset. Moreover, the GRU has a smaller block architecture and shows comparable performance to the LSTM without the need for additional algorithms to support the model.

Although it is very difficult to determine the best classifier for all time series types, the proposed model seeks to achieve an equivalent accuracy to state-of-the-art classification models. Using the UCR benchmark [14] to compare our model with other state-of-the-art classification models, our model achieved higher classification performance on several time series examples compared to other state-of-the-art classification models.

978-1-7281-2980-8/19/$31.00 ©2019 IEEE
DOI 10.1109/iThings/GreenCom/CPSCom/SmartData.2019.00202

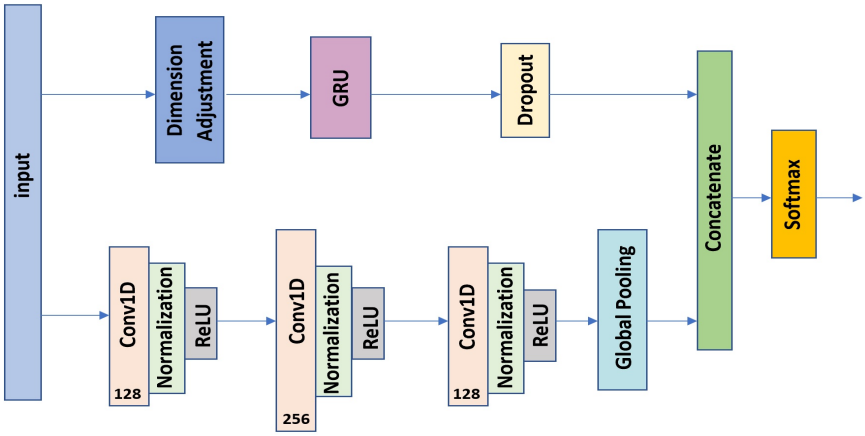

Fig. 2: The proposed model architecture based on [4], [5] architectures.

II. Model Components

A. Gated Recurrent Unit (GRU)

The gated recurrent unit (GRU) was introduced in [15] as another type of gate-based recurrent unit which has a smaller architecture and comparable performance to the LSTM unit. The GRU consists of two gates: reset and update. The architecture of an unrolled GRU block is shown in Fig. 1. $r^{(t)}$ and $z^{(t)}$ denote the values of the reset and update gates at time step t, respectively. $x_i \in \mathbb{R}^n$ is a 1D input vector to the GRU block at time step t. $\tilde{h}^{(t)}$ is the output candidate of the GRU block. $h^{(t-1)}$ is the recurrent GRU block output of time step $t-1$ and the current output at time t is $h^{(t)}$. Assuming a one-layer GRU, the reset gate, update gate, output candidate, and GRU output are calculated as follows [15]:

$$z^{(t)} = \sigma(W_z x^{(t)} + U_z h^{(t-1)} + b_z) \quad (1)$$

$$r^{(t)} = \sigma(W_r x^{(t)} + U_r h^{(t-1)} + b_r) \quad (2)$$

$$\tilde{h}^{(t)} = tanh(W x^{(t)} + U(r^{(t)} \odot h^{(t-1)}) + b) \quad (3)$$

$$h^{(t)} = (1 - z^{(t)}) \odot h^{(t-1)} + z^{(t)} \odot \tilde{h}^{(t)} \quad (4)$$

where W_z, W_r, and W are the feedforward weights and U_z, U_r, and U are the recurrent weights of the update gate, reset gate, and output candidate activation respectively. b_z, b_r and b are the biases of the update gate, reset gate and the output candidate activation $\tilde{h}^{(t)}$, respectively.

Similar to the RNN and LSTM, the GRU models temporal (sequential) datasets. The GRU uses its previous time step output and the current input to calculate the next output. The GRU has advantages, due to its lower memory requirements, fewer trainable parameters, and lower training time as compared to the LSTM.

B. Temporal Fully Convolutional Neural Network (FCN)

The fully convolutional neural network (FCN) [16]. utilizes weight sharing over grid-structured datasets such as time series [17], [18]. The convolutional layers within the FCN can learn to extract complex feature representations from the data with little or no preprocessing. The FCN consists of many layers of convolutional blocks that may have different or same kernel sizes, followed by a pixel-wise prediction map where each pixel corresponds to a classifier. For time series problems, the values of each convolution block in the FCN, are calculated as follows [4]:

$$y_i = W_i * x_i + b_i \quad (5)$$

$$z_i = BN(y) \quad (6)$$

$$out_i = ReLU(z) \quad (7)$$

where $x_i \in \mathbb{R}^n$ is the input 1D vector which represents a time series segment, W_i is the 1D convolutional kernel of weights, b_i is the bias, and y is the output vector of the convolutional block i. z_i is the intermediate result after applying batch normalization [19] on the convolutional block which is then passed to the rectified linear unit $ReLU$ [20] to calculate the output of the convolutional layer out_i.

III. Model Architecture

As stated earlier, our classification model replaces the LSTM with a GRU in a hybrid gated-FCN. Our model is based on the framework introduced in [4], [5]. The proposed architecture is shown in Figure 2 and has two parallel parts: a GRU and a FCN. Our model uses a three-layer FCN architecture proposed in [4]. We used also the global average pooling as proposed in [4], [5] which reduces the number of trainable parameters compared to the fully connected layer, without any sacrifice in the model performance. The FCN number of kernels is 128, 256, and 128 in each convolutional layer respectively. The size of each kernel is 8, 5, and 3 respectively. In addition, we used the GRU instead of LSTMs that were used in [5] models to reduce the number of trainable parameters, memory and training time. Moreover, we removed the masking and any extra supporting algorithms such as attention mechanism that were used in LSTM-FCN and ALSTM models [5]. The GRU is unfolded to 8 unfolds for all the datasets. The input was fitted using the concept used in [5] to fit input to a recurrent unit. We used the Adam optimization function [21] with $\beta_1 = 0.9$, $\beta_2 = 0.999$ and initial learning

TABLE I: Classification accuracy and rank for 44 time series datasets from the UCR benchmark [14].

Dataset	Classification Method and Testing Accuracy											
	GRU-FCN	***FCN***	***LSTMFCN***	***ALSTMFCN***	***ResNet***	***MCNN***	***MLP***	***COTE***	***DTW***	***PROP***	***BOSS***	***SE***
Adiac	**0.893**	0.857	0.859	0.861	0.826	0.769	0.752	0.767	0.604	0.647	0.780	0.627
Beef	**0.867**	0.750	0.833	0.800	0.767	0.633	0.833	**0.867**	0.633	0.633	0.800	**0.867**
CBF	**1.0**	0.992	0.995	0.994	0.994	0.998	0.860	0.999	0.997	0.998	**1.0**	0.990
ChrlorineCon	**0.998**	0.843	0.809	0.807	0.828	0.797	0.875	0.686	0.468	0.640	0.666	0.688
CinCECGTorso	0.876	0.813	0.809	0.807	0.828	0.942	0.842	0.936	0.651	0.938	0.875	**0.979**
Coffee	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	0.964	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**
CricketX	0.802	0.815	0.807	0.797	0.821	0.818	0.569	**0.846**	0.744	0.797	0.741	0.703
CricketY	0.812	0.792	0.817	0.815	0.805	**0.846**	0.595	0.833	0.744	0.844	0.792	0.674
CricketZ	0.817	0.813	0.810	0.825	0.831	0.858	0.592	**0.872**	0.744	0.844	0.754	0.723
DiatomSizeR	0.964	0.931	0.954	0.937	0.931	**0.977**	0.964	0.918	0.967	0.941	0.954	0.931
ECGFiveDays	0.993	0.990	0.988	0.988	0.955	**1.0**	0.970	**1.0**	0.768	0.822	**1.0**	0.945
FaceAll	0.915	0.929	0.940	**0.955**	0.834	0.765	0.885	0.895	0.808	0.885	0.790	0.753
FaceFour	0.864	0.932	0.943	0.943	0.932	**1.0**	0.833	0.909	0.833	0.909	**1.0**	0.966
FacesUCR	0.947	0.948	0.929	0.943	**0.958**	0.937	0.815	0.943	0.905	0.937	**0.958**	0.921
FiftyWords	0.749	0.679	0.729	0.716	0.727	0.810	0.712	0.809	0.690	**0.820**	0.699	0.712
Fish	0.977	0.971	0.977	0.965	**0.989**	0.949	0.874	0.971	0.823	0.966	**0.989**	0.943
GunPoint	**1.0**	**1.0**	**1.0**	**1.0**	0.993	**1.0**	0.933	0.993	0.907	0.993	**1.0**	0.940
Hapitics	0.529	**0.551**	0.548	0.542	0.505	0.470	0.461	0.512	0.377	0.416	0.464	0.393
InlineSkate	0.481	0.411	0.470	0.481	0.365	0.382	0.351	0.449	0.384	0.433	**0.489**	0.347
ItalyPower	**0.973**	0.970	0.963	0.960	0.960	0.970	0.966	0.964	0.950	0.961	0.947	0.947
Lightening2	0.754	0.803	0.803	0.787	0.754	0.836	0.721	0.836	0.869	0.885	0.852	**0.902**
Lightening7	**0.863**	**0.863**	0.836	0.822	0.836	0.781	0.644	0.753	0.726	0.767	0.658	0.726
MALLAT	0.962	**0.980**	0.968	0.968	0.979	0.943	0.936	0.964	0.934	0.950	0.942	0.908
MedicalImages	**0.801**	0.792	**0.801**	0.796	0.772	0.740	0.729	0.742	0.737	0.755	0.712	0.695
MoteStrain	0.924	**0.950**	0.939	0.936	0.895	0.921	0.869	0.915	0.835	0.886	0.927	0.887
NonInvThorax1	**0.963**	0.961	**0.963**	**0.963**	0.948	0.936	0.942	0.907	0.790	0.822	0.839	0.826
NonInvThorax2	**0.958**	0.955	0.957	0.952	0.951	0.940	0.943	0.927	0.865	0.888	0.899	0.882
OliveOil	**0.933**	0.833	**0.933**	0.899	0.867	0.867	0.400	0.900	0.833	0.867	0.900	0.867
OSULeaf	**1.0**	0.988	0.996	0.996	0.979	0.729	0.570	0.855	0.591	0.806	0.988	0.727
SonyAIBORobot	**0.983**	0.968	0.982	0.970	0.985	0.770	0.727	0.854	0.725	0.707	0.679	0.762
SonyAIBRObotII	**0.982**	0.962	0.978	0.975	0.962	0.930	0.839	0.924	0.831	0.876	0.902	0.934
StarLightCurves	0.975	0.967	0.976	0.977	0.971	0.977	0.957	0.969	0.907	0.921	**0.979**	0.907
SwedishLeaf	**0.984**	0.966	0.979	0.974	0.958	0.934	0.893	0.954	0.792	0.915	0.728	0.880
Symbols	**0.986**	0.962	0.984	0.984	0.872	0.951	0.853	0.954	0.950	0.951	0.968	0.917
SyntheticControl	0.996	0.990	0.993	0.990	**1.0**	0.997	0.950	**1.0**	0.993	0.990	0.970	0.967
Trace	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	0.820	0.990	**1.0**	0.990	**1.0**	0.950
TwoLeadECG	**1.0**	**1.0**	0.999	0.999	**1.0**	0.999	0.853	0.985	**1.0**	**1.0**	0.996	0.971
TwoPatterns	0.991	0.897	0.997	0.997	**1.0**	0.998	0.886	**1.0**	0.904	0.933	0.984	0.952
UWaveX	0.825	0.754	**0.849**	0.848	0.787	0.820	0.768	0.804	0.728	0.801	0.759	0.752
UWaveY	0.760	0.725	**0.767**	0.766	0.668	0.732	0.703	0.733	0.634	0.717	0.687	0.678
UWaveZ	0.763	0.729	0.797	**0.798**	0.755	0.768	0.705	0.735	0.658	0.710	0.688	0.654
Wafer	**0.999**	0.997	**0.999**	0.998	0.997	0.998	0.996	**0.999**	0.980	0.997	**0.999**	0.998
WordSynonyms	0.738	0.580	0.671	0.668	0.632	0.724	0.594	0.734	0.490	**0.774**	0.655	0.643
Yoga	0.910	0.902	0.888	0.887	0.858	0.888	0.855	0.887	0.836	0.879	**0.919**	0.841
no. best (out of 44)	**20**	8	9	6	7	6	1	8	3	4	12	4
Arithmetic AVG Rank	**3.193**	6.068	4.2954	4.295	6.227	5.636	9.488	5.625	9.875	7.363	6.693	9.238
MPCE	**0.0170**	0.0207	0.0182	0.0192	0.022	0.0240	0.0400	0.0226	0.0407	0.0299	0.0257	0.0299

rate $\alpha = 0.01$. The learning rate α was reduced by 0.8 factor every 100 training steps until it reached the minimal learning rate $\alpha = 0.0001$. Our model used categorical cross-entropy as the loss function [18].

The input to the model is the raw dataset. The FCN is responsible for feature extraction from the time series [4] and the GRU enables the model to learn temporal dependencies within the time series. Therefore the model learns both the features and the temporal dependencies which empowers the model to predict the correct class for each data entry.

IV. Method and Results

We implemented our model by modifying the LSTM-FCN [5] model implementation. The model is implemented using the Keras API [22] with TensorFlow backend. We used the UCR benchmark [14] datasets in our experiments. Each dataset in the UCR benchmark is divided into training and testing sets. The size, length, and number of classes for each dataset is provided by the UCR benchmark online source which can be found in [14]. We used 44 datasets of the UCR benchmark which are the most commonly used datasets in different classification models testing. We compared our GRU-FCN with different state-of-the-art time series methods such as FCN [4] which are based on fully convolutional neural network, LSTM-FCN [5], ALSTM-FCN [5], that are based on long short term-memory and fully convolutional neural networks, residual neural network model (ResNet) model [4], multi-scale convolution neural networks model (MCNN) [13], multi-layered perceptrons model (MLP) [4], collective of transformation-based ensembles model (COTE) [12], dynamic time warping model (DTW) [23], PROP model [11] which is based on elastic distance measures, BOSS model [9] that based on noise reduction in the time series representation, and the Shapelet ensemble based model (SE) [12]. Our model shows the overall highest accuracy classification rank and smallest mean per-class classification error (MPCE) compared to state-of-art classification models as shown in Table I.

We evaluated our model using the Mean Per-Class Error

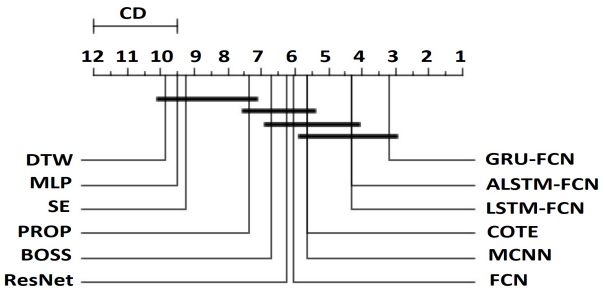

Fig. 3: Critical difference diagram based on models ranks arithmetic mean on the UCR benchmark datasets.

(MPCE) proposed by [4] to evaluate the performance of a classification method over multiple datasets. The MPCE is calculated based on the per-class error (PCE) as follows:

$$\mathrm{MPCE}_i = \frac{1}{M}\sum_{m=1}^{M}\mathrm{PCE}_m = \frac{1}{M}\sum_{m=1}^{M}\frac{e_m}{c_m} \tag{8}$$

where i is the classification model with error rate e_m when applied to a specific dataset m which consists of c_m class labels, and M is the total number of datasets that are applied.

Table I shows the MPCE value for our GRU-FCN and other state-of-the-art models on the UCR benchmark datasets. The results obtained by training and tested of the GRU-FCN, FCN, LSTM-FCN, and ALSTM based models. For the other models, we obtained the results from their own publications. Our GRU-FCN has the smallest MPCE value compared to the other state-of-art classification models. This means that generally, our GRU-FCN model performance across the different datasets is higher than the other state-of-art models.

Figure 3 shows the critical difference diagram [24] for Nemenyi or Bonferroni-Dunn test [25] with $\alpha = 0.05$ on our GRU-FCN and the state-of-art models based on the ranks arithmetic mean on the UCR benchmark datasets. This graph shows the significant classification accuracy improvement of our GRU-FCN compared to the other state-of-the-art models.

V. Conclusion

The empirical study of the GRU-FCN classification model shows that replacing the LSTM by GRU in classification problems enhances the classification accuracy without extra supporting algorithm requirements such as masking or attention algorithms. Thus, this paper deduces that GRU outperforms the LSTM in univariate time series classification problems with smaller architecture and computational requirements. Furthermore, the proposed GRU-FCN classification model achieves the performance of state-of-the-art models and has the highest average arithmetic ranking and the lowest mean per-class error (MPCE) through time series datasets classification of the UCR benchmark comparing to the state-of-art models.

References

[1] J. D. Hamilton, *Time series analysis*. Princeton university press, Princeton, NJ, 1994, vol. 2.

[2] H. Sohn and C. R. Farrar, "Damage diagnosis using time series analysis of vibration signals," *Smart materials and structures*, vol. 10, no. 3, p. 446, 2001.

[3] M. Gul and F. N. Catbas, "Statistical pattern recognition for structural health monitoring using time series modeling: Theory and experimental verifications," *Mechanical Systems and Signal Processing*, vol. 23, no. 7, pp. 2192–2204, 2009.

[4] Z. Wang, W. Yan, and T. Oates, "Time series classification from scratch with deep neural networks: A strong baseline," in *Neural Networks (IJCNN), 2017 International Joint Conference on*. IEEE, 2017, pp. 1578–1585.

[5] F. Karim, S. Majumdar, H. Darabi, and S. Chen, "LSTM fully convolutional networks for time series classification," *IEEE Access*, vol. 6, pp. 1662–1669, 2018.

[6] A. Amei, W. Fu, and C.-H. Ho, "Time series analysis for predicting the occurrences of large scale earthquakes," *International Journal of Applied Science and Technology*, vol. 2, no. 7, 2012.

[7] J. Rotton and J. Frey, "Air pollution, weather, and violent crimes: Concomitant time-series analysis of archival data." *Journal of personality and social psychology*, vol. 49, no. 5, p. 1207, 1985.

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

[9] 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.

[10] 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.

[11] J. Lines and A. Bagnall, "Time series classification with ensembles of elastic distance measures," *Data Mining and Knowledge Discovery*, vol. 29, no. 3, pp. 565–592, 2015.

[12] 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.

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

[14] Y. Chen, E. Keogh, B. Hu, N. Begum, A. Bagnall, A. Mueen, and G. Batista. (2015, July) The UCR time series classification archive. [Online]. Available: http://www.cs.ucr.edu/$\sim$eamonn/time_series_data/

[15] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[16] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural computation*, vol. 1, no. 4, pp. 541–551, 1989.

[17] Y. LeCun, Y. Bengio *et al.*, "Convolutional networks for images, speech, and time series," *The handbook of brain theory and neural networks*, vol. 3361, no. 10, p. 1995, 1995.

[18] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep Learning*. MIT press Cambridge, 2016, vol. 1.

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

[20] 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.

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

[22] F. Chollet *et al.*, "Keras," https://keras.io, 2015.

[23] 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.

[24] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *Journal of Machine learning research*, vol. 7, no. Jan, pp. 1–30, 2006.

[25] T. Pohlert, "The pairwise multiple comparison of mean ranks package (PMCMR)," *R package*, vol. 27, 2014.