Time Series Classification with Recurrent Neural Networks

Denis Smirnov[1,2] and Engelbert Mephu Nguifo[1]

[1] University Clermont Auvergne, CNRS, LIMOS, 63000 Clermont-Ferrand, France
[2] National Research University Higher School of Economics, Faculty of Computer Science, 101000 Moscow, Russian Federation

Abstract. Deep learning techniques showed promising results in time series classification. This work summarizes the achievements of deep neural networks in the problem of univariate time series classification and studies the application of recurrent neural networks to the problem. An experimental part evaluates the classification quality of different recurrent models over a collection of 85 UCR datasets and compares them with the existing feedforward baselines and between each other.

Keywords: Time Series Classification · Deep Learning · Recurrent Neural Networks.

1 Introduction

Time series classification (TSC) is a common time series analysis problem which requires restoring a functional dependence between the set of possible time series and the finite set of classes using a training set with known classes. There are several works which illustrate that deep neural networks are suitable for TSC problem and can outperform other algorithms [11], [7], [8], [5] which encourage further work in this direction. There exist deep learning models like recurrent neural networks (RNNs) that are designed specifically for processing sequential data, and thus could be applied for time series. Although some recent works use special kinds of RNNs as a component of resulting model for TSC problem [7], recent publications, which study the efficiency of RNNs as a self-sufficient approach for the problem, were not found.

The research objective of this work is to compare the results of TSC obtained by the application of different recurrent neural network architectures between each other and the current baselines.

2 Time Series Classification

2.1 Traditional approach for time series classification

One of the simplest but very powerful approaches – Dynamic Time Warping (DTW) – was introduced in 1978 in the context of speech recognition [9]. DTW

is a way to define the distance measure between two univariate time series, and it can be used with a k-Nearest Neighbors (k-NN) classifier. The main idea consists in calculating the distance between two time series not pointwisely, like it is done using Euclidian distance, but using the mapping between structurally similar points, which is time-invariant. Extensive experimental evaluation [1] showed that 1-NN DTW is a strong benchmark which cannot be beaten by many proposed algorithms. If beaten, the advantage is often not so significant, compared to the difficulty of the implementation or the computation.

Among more sophisticated approaches the best results are shown by Bag-of-SFA-Symbols (BOSS) [10] and Collective of transform-based ensembles (COTE) [2][1]. The first method is a multistage time series transformation procedure inspired by the bag-of-words model from text processing, while the second is an ensemble which aggregates the results of 35 classifiers which process four different types of time series representation types.

The publication of renewed University of California, Riverside (UCR) TSC Archive [3] in 2015 with more datasets available is an additional driver to research in this area because it allowed conducting more reproducible research with comparable results [1].

2.2 Time series classification with deep neural networks

One notable work on this topic is [5] which introduces Multi-Scale Convolutional Neural Network (MCNN). The proposed model uses convolutional layers to independently extract features from the three different representations of time series: the original one, the downsampled one and the smoothed with a moving average one. For the last two transformations, the multiple versions of time series are generated with different downsampling scales and moving averages window sizes. Features from these three branches are concatenated and processed with the another block of convolutional layers in order to obtain final prediction. The authors have conducted experiments comparing their model and multiple classical approaches on 44 UCR datasets where the proposed model showed best error rate 10 times while COTE did so 11 times and BOSS 15 times. The reported mean rank of the solution is 3.95 which is the second result after COTE.

The most fundamental study on TSC with deep neural networks was conducted by Wang et al. [11]. In this paper, the authors presented the results of an experimental study on 44 UCR datasets which compared the existing best approaches with three basic neural network models: multilayer perceptron (MLP), fully convolutional network (FCN) and residual network (ResNet). The goal of the study was not to beat the existing state-of-the-art results but to define a baseline for TSC with deep neural networks. The proposed baseline models had the same shape and parameters for all datasets and were not finetuned. Nevertheless, the proposed FCN was able to outperform COTE so this baseline turned out to beat the best non-deep learning approaches. Meanwhile, MLP with 1-NN DTW showed the worst result among the considered models.

Fazle Karim et al. in [7] proposed a new deep learning model for TSC with a branching structure: the first branch is exactly the convolutional part of the

model from the previously presented work by Wang et al.[11], the second branch is a Long Short-Term Memory (LSTM) block which receives a time series in a transposed form as multivariate time series with single time step. The output of two branches is concatenated and fed to a dense classifier. The experimental results on all 85 UCR datasets showed that this model outperforms the existing state-of-the-art methods on the most of the datasets.

Another paper on deep learning for TSC proposes not an end-to-end model but an approach for time series preprocessing [8]. The authors have trained a sequence-to-sequence autoencoder in order to extract meaningful features from time series. Authors have trained the model that consisted of the 3 recurrent layers of encoder and the 3 recurrent layers of the decoder on 18 datasets from the UCR Archive with time series length less or equal than 512. Using the obtained embeddings as features, SVM classifiers were trained for 30 other datasets separately. The accuracy achieved by these classifiers in the majority of the cases was higher than using the baseline approach which is a DTW classifier on the initial time-series.

3 Experiments and Discussion

One of the best models for TSC [7] uses the LSTM network as a component of their neural network architecture but this recurrent component takes a transposed time-series as an input (multivariate time-series of length 1), and thus the time-range dependencies in the data are not taken into account. Moreover, no works and experimental studies were found about the efficiency of LSTMs or the other recurrent networks as standalone classifiers for this task. In this work, it is proposed to train simple recurrent neural networks and LSTMs with a different number of layers and different layer widths. These models would be trained on the all of the 85 UCR datasets. Since the results of deep neural network baselines [11] are reported only on a subset of the archive, the models from that work should also be trained on all the data during the proposed experiment.

3.1 Data description

The data for this experiment is the UCR TSC Archive [3], which is a collection of 85 reference datasets for the TSC problem. Each dataset contains the predefined train and test sets of equal-length univariate time series. The data is real-valued, and there are no missing values.

The number of classes varies from 2 to 60. The time series length varies from 24 to 2709 observations, on average it is around 422. For 77 datasets, the size of the training set is less than 1000 time series (average is 432) and the test set is often larger than the training set.

3.2 Experiment setup

In this experiment for each of the 85 UCR datasets several deep neural networks with different architectures were trained. The train and the test sets of every

dataset were normalized by subtracting the train set mean and the train set standard deviation. No other data preprocessing techniques were applied.

The models were implemented using Keras 2 framework [4] with TensorFlow backend (Python 3.6). The models were trained on the Amazon Web Services (AWS) spot instances of two different types: GPU instances equipped with NVIDIA Tesla V100 GPU and CPU instances with 32 CPUs. The environment was the same for all the used instances - the Deep Learning AMI with Conda (Ubuntu) with preinstalled necessary frameworks.

In this experiment, twelve different deep neural network models were studied: MLP and FCN as presented in [11] and 10 different recurrent neural network architectures. Although Wang et al. also applied ResNet to TSC problem [11], this architecture achieved the results which are similar to the results of FCN, so the ResNet was not included in this experiment.

The studied models can be divided in four groups: feedforward networks from [11], the networks with simple recurrent hidden layers and a dense output layer, the networks with LSTM hidden layers and a dense output layer, and networks with simple recurrent hidden layers and a recurrent output layer. The feedforward networks were trained on the GPU instances, while the recurrent networks were trained on the CPU instances.

The proposed MLP consists of three hidden dense layers with 500 units in the each of them. The dropout is used after input layer and after the every dense layer with rates 0.1, 0.2, 0.3, 0.4. The activation function for hidden layers is Rectified Linear Unit (ReLU). In this experiment, the MLP and all other networks use softmax activation in the output layer, while the optimized loss function is a categorical cross entropy. Following the description presented in the original paper, the MLP was trained during 5000 epochs using the Adadelta optimizer with a learning rate 0.1. The authors have reported the random seed for this particular model, which was also used without changes. The same seed was used for all other models in this experiment. For this and all the following models, the batch size was set to 1/10 of the training set size if it is not less than 16, and strictly to 16 otherwise.

The FCN as proposed in [11] has three convolutional layers, containing 128, 256 and 128 hidden units with kernel sizes 8, 5, 3. As reported in the original work, the training was performed during 2000 epochs using the Adam optimizer with default parameters.

The other models in this experiment were networks with 1, 2 and 3 simple recurrent hidden layers with 128 neurons which use either a dense or a recurrent output layer and also RNN with one hidden layer of size 256. Besides, the models with 1 and 2 hidden LSTM layers of different sizes with a dense output were trained. In the considered models, all recurrent layers except for the last one (which is either the layer before the dense output or the recurrent output layer) produce a sequential output. In this experiment, the dropout was not applied to the recurrent models. The networks were trained using the Adam optimizer with the same default parameters, as they were used in the FCN, during 500 epochs.

In the rest of the paper, for the recurrent models the following naming convention is used: *typeOfHiddenLayers_hiddenLayerSizes_typeOfOutputLayer*. For example, *lstm_128_128_dense* refers to the network with two hidden LSTM layers of size 128 and a dense output layer. This architecture is shown on the figure 1.

4 Results and Discussion

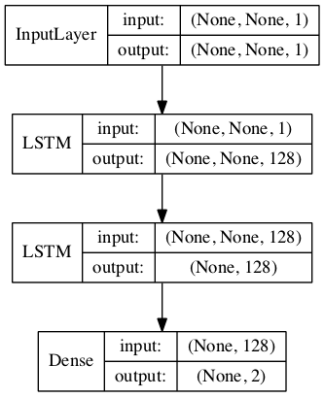

Fig. 1. Architecture of *lstm_128_128_dense* model for binary classification problem

The paper by Wang et al. [11] proposed an evaluation measure for the performance of classifiers on multiple datasets - Mean Per-Class Error (MPCE) which is an average of per-class errors (PCE) over all datasets (the lower, the better). PCE for each dataset is calculated as follows:

$$PCE = \frac{1 - accuracy}{number\ of\ classes}$$

In their paper, an evaluation was performed on the subset of UCR Archive which contained 44 of 85 datasets and reported $MPCE_{MLP} = \mathbf{0.0407}$ on the test sets for the MLP, while in this experiment using the same random seed a value of **0.0433** was obtained. This little discrepancy can be explained by the difference in the implementations of tensor operations in the underlying frameworks.

For the FCN they reported $MPCE_{FCN} = \mathbf{0.0219}$ on the same subset of UCR Archive which was the best performance in that study and was claimed to be the state-of-the-art result on some of the datasets. The random seed which was used for the training of the FCN was not reported by the authors, so it was not possible to exactly reproduce the obtained results. In our experiment, the mean per-class error of the FCN on 44 datasets was **0.0324**.

The resulting MPCE calculated on all 85 datasets in our experiment was **0.0751** for the MLP and **0.0539** for the FCN. These errors are higher, but the FCN still outperforms the MLP.

Among all the other trained models, somewhat comparable but still worse result was shown by *lstm_128_dense* (LSTM network with one hidden layer of width 128) - **0.0943**. All other models showed a higher MPCE.

These MPCE values are reported in order to provide the link to the previously published results. The calculation of the MPCE requires dividing by the number of classes, and thus results in lower values for the datasets with a high number of classes. In the rest of the paper, the accuracy and the weighted F1-score are used to compare the performance of the considered models. The average accuracy values and the average of weighted F1-scores for each model on the train and the test sets are shown in the table 1.

Table 1. The average accuracy and the average F-1 score achieved by each model on the train and the test sets of 85 UCR datasets. The best result in the each group of models is in **bold**, the best result among all the considered models is **underlined**.

	Average Accuracy Train	Average Accuracy Test	Average F1-score Train	Average F1-score Test
MLP	**0.975**	0.723	**0.974**	0.724
FCN	0.963	**0.801**	0.957	**0.792**
rnn_128_dense	**0.592**	**0.516**	**0.535**	**0.458**
rnn_128_128_dense	0.566	0.486	0.491	0.411
rnn_128_128_128_dense	0.563	0.475	0.485	0.399
rnn_256_dense	0.537	0.476	0.463	0.404
lstm_128_dense	0.766	**0.637**	0.739	**0.608**
lstm_128_128_dense	**0.804**	0.623	**0.777**	0.593
lstm_256_dense	0.750	0.613	0.718	0.581
rnn_128_rnn	0.592	**0.512**	**0.540**	**0.461**
rnn_128_128_rnn	**0.605**	0.505	0.539	0.437
rnn_128_128_128_rnn	0.562	0.493	0.479	0.409

None of the trained recurrent networks is able to achieve the same average results as the fully convolutional network. However, in order to claim that one model outperforms another on multiple datasets, it is not enough to compare averages, it should be done using a Wilcoxon signed rank test which is a non-parametric alternative of the paired t-test. For multiple models on multiple datasets, the Friedman test is recommended [6].

The pairwise comparison of the accuracies and of the F1-scores obtained by different recurrent networks on the test sets of the UCR datasets using the Wilcoxon signed rank test on the significance level of 1% allowed to derive the following conclusions:

1. Using the LSTM layers instead of the simple recurrent layers in the considered experiment **does** increases the classification quality on the test sets of the UCR datasets.

2. Adding a third layer of size 128 to the considered two-layer networks does not significantly affects the classification performance on the UCR datasets.

3. Changing the size of the hidden LSTM layer from 128 to 256 units in the considered network does not significantly affect the performance on the test sets of the UCR datasets in terms of the considered metrics.

4. Using the simple recurrent hidden layer of width 256 instead of 128 in the considered 1-layer network **decreases** the quality of classification on the test sets of the UCR datasets in terms of the considered metrics.

5. It was not observed that the use of the recurrent output layer instead of the dense output layer significantly affects the classification quality of the considered neural networks on the test sets of the UCR datasets.

Critical difference (CD) plot is a visualization technique for the Friedman test results [6]. Such a plot for the recurrent neural networks and MLP with FCN

built using the accuracy scores is shown in the figure 2. The axis of this plot corresponds to the average rank of a model. The models that are not significantly different from each other are connected with a line. Plot for the weighted F1-score shows the same results and is omitted.

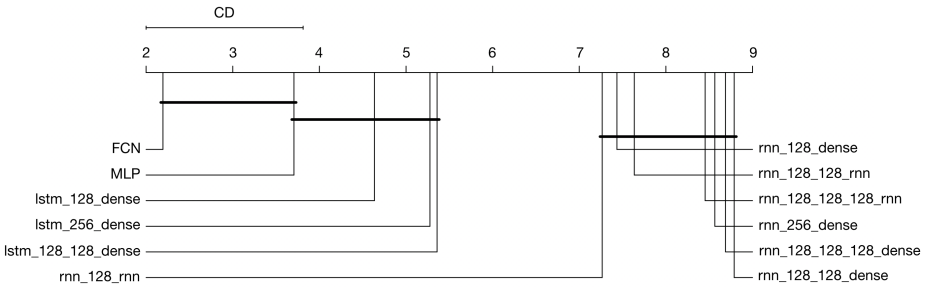

Fig. 2. Critical difference plot on accuracy ranks for considered models

We can notice that all considered simple recurrent networks are not significantly different from each other which is also true for all the considered LSTM networks. The difference between the MLP and all considered LSTM networks turned out to be not significant. At the same time, all considered recurrent neural networks are significantly worse than FCN.

5 Conclusion

The interpretation of the obtained results has shown that the use of the LSTM layers instead of the simple recurrent layers allows us to achieve a better classification quality on the UCR datasets for the considered models. No significant difference in the classification performance between using 128 and 256 hidden units in the considered one-layer LSTM network was found, while the classification quality in the similar setting for the simple recurrent neural network decreases. We might expect to obtain similar results for the same models on the data similar to the UCR Archive datasets.

Although the obtained performance of the recurrent neural networks is below the performance of existing convolutional models, RNNs are still a viable choice for TSC problems especially in the case where the input sequences have different lengths. While recurrent models can process such input out of the box, an average feedforward network would require to fix some sequence size and to use trimming or padding for the incoming data.

The variety of recurrent architectures is not limited to simple RNNs and LSTMs. Further work may involve the evaluation of gated recurrent units (GRUs) and bidirectional RNNs in the similar experimental setting.

Acknowledgement. This work was done as the master thesis of the first author at LIMOS in the context of the dual degree program of the University Clermont Auvergne (Clermont-Ferrand, France) and the National Research University Higher School of Economics (Moscow, Russia).

This work is funded by the French Labex project IMOBS3, and also partly supported by the CNRS PEPS program TransiXplore.

References

[1] Anthony Bagnall, Jason Lines, Aaron Bostrom, James Large, and Eamonn Keogh. "The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances". In: *Data Mining and Knowledge Discovery* 31.3 (2017), pp. 606–660.

[2] Anthony Bagnall, Jason Lines, Jon Hills, and Aaron Bostrom. "Time-Series Classification with COTE: The Collective of Transformation-Based Ensembles". In: *IEEE Transactions on Knowledge and Data Engineering* 27.9 (2015), pp. 2522–2535.

[3] Yanping Chen, Eamonn Keogh, Bing Hu, Nurjahan Begum, Anthony Bagnall, Abdullah Mueen, and Gustavo Batista. *The UCR Time Series Classification Archive*. `www.cs.ucr.edu/~eamonn/time_series_data/`. 2015.

[4] François Chollet. *Keras*. `https://keras.io`. 2015.

[5] Zhicheng Cui, Wenlin Chen, and Yixin Chen. "Multi-Scale Convolutional Neural Networks for Time Series Classification". In: *CoRR* abs/1603.06995 (2016). arXiv: `1603.06995`.

[6] Janez Demšar. "Statistical Comparisons of Classifiers over Multiple Data Sets". In: *J. Mach. Learn. Res.* 7 (2006), pp. 1–30.

[7] Fazle Karim, Somshubra Majumdar, Houshang Darabi, and Shun Chen. "LSTM Fully Convolutional Networks for Time Series Classification". In: *IEEE Access* 6 (2018), pp. 1662–1669.

[8] Pankaj Malhotra, Vishnu TV, Lovekesh Vig, Puneet Agarwal, and Gautam Shroff. "TimeNet: Pre-trained deep recurrent neural network for time series classification". In: *25th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*. 2017, pp. 607–612.

[9] Hiroaki Sakoe and Seibi Chiba. "Dynamic Programming Algorithm Optimization for Spoken Word Recognition". In: *IEEE Transactions on Acoustics, Speech and Signal Processing* 26 (1978), pp. 43 –49.

[10] Patrick Schfer. "The BOSS is concerned with time series classification in the presence of noise". In: *Data Mining and Knowledge Discovery* 29.6 (2015), pp. 1505–1530.

[11] Zhiguang Wang, Weizhong Yan, and Tim Oates. "Time series classification from scratch with deep neural networks: A strong baseline". In: *2017 International Joint Conference on Neural Networks (IJCNN)*. 2017, pp. 1578–1585.