UNIVERSITÉ Clermont Auvergne

LIMOS
LABORATOIRE D'INFORMATIQUE, DE MODÉLISATION ET D'OPTIMISATION DES SYSTÈMES

HIGHER SCHOOL OF ECONOMICS
NATIONAL RESEARCH UNIVERSITY

Time Series Classification with Recurrent Neural Networks

Denis Smirnov (UCA, LIMOS / Higher School of Economics)
Engelbert Mephu Nguifo (UCA, LIMOS)

3rd ECML/PKDD Workshop on
Advanced Analytics and Learning on Temporal Data

September 14, 2018
Dublin, Ireland

Outline

- Introduction
- Time Series Classification with Deep Neural Networks
- Experiment and discussion

Time Series Classification

Time series classification (TSC) — a 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.

- The publication of renewed University of California, Riverside (UCR) TSC Archive [3] in 2015 drives research in this area
- One of the oldest method, Dynamic Time Warping (DTW) sets a strong baseline for the task which is not easily beaten [1]
- Ensemble method COTE [2] is among the best algorithms for the problem
- Recent works illustrate that deep neural networks are suitable for TSC problem and can outperform other algorithms

[1] Anthony Bagnall et al. (2017) “The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances”. In: Data Mining and Knowledge Discovery 31.3
[2] Anthony Bagnall et al. (2015) “Time-Series Classification with COTE: The Collective of Transformation-Based Ensembles”. In: IEEE Transactions on Knowledge and Data Engineering 27.9
[3] Yanping Chen et al. (2015) “The UCR Time Series Classification Archive”

Deep Neural Networks

Deep learning algorithms — class of machine learning algorithms which learn hierarchical feature representations from input data.
Most commonly used model — artificial neural network with multiple layers.

- Deep Neural Network (DNN) may be considered as a directed graph of tensor operations over input data. For feedforward DNNs such graph is acyclic
  In order to use feedforward network for TSC, all time series should have the same length which is defined before training

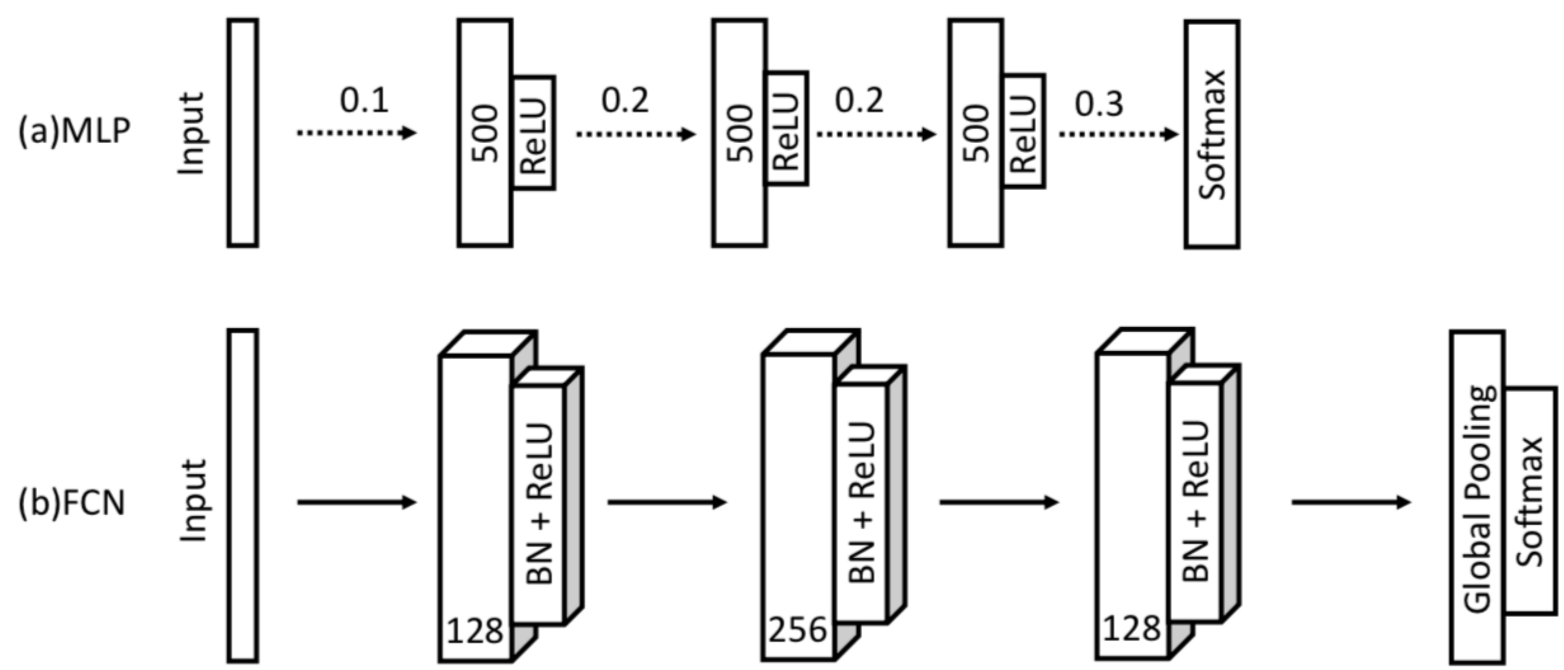
- Recurrent networks are specifically designed for processing sequential input. They have feedback connection in the layers, and thus can capture time-range dependencies

- Long Short-Term Memory (LSTM) is a special type of recurrent network which is capable of long-range dependencies

TSC with deep neural networks

Zhiguang Wang et al. [4] conducted experiments, comparing the performance of multilayer perceptron (MLP), fully connected convolutional network (FCN) and Residual Network (ResNet) with existing best methods on the subset of UCR Archive.

- The proposed FCN outperformed all other models
- COTE was 2nd-best, ResNet a little bit worse
- DTW and MLP showed worst results

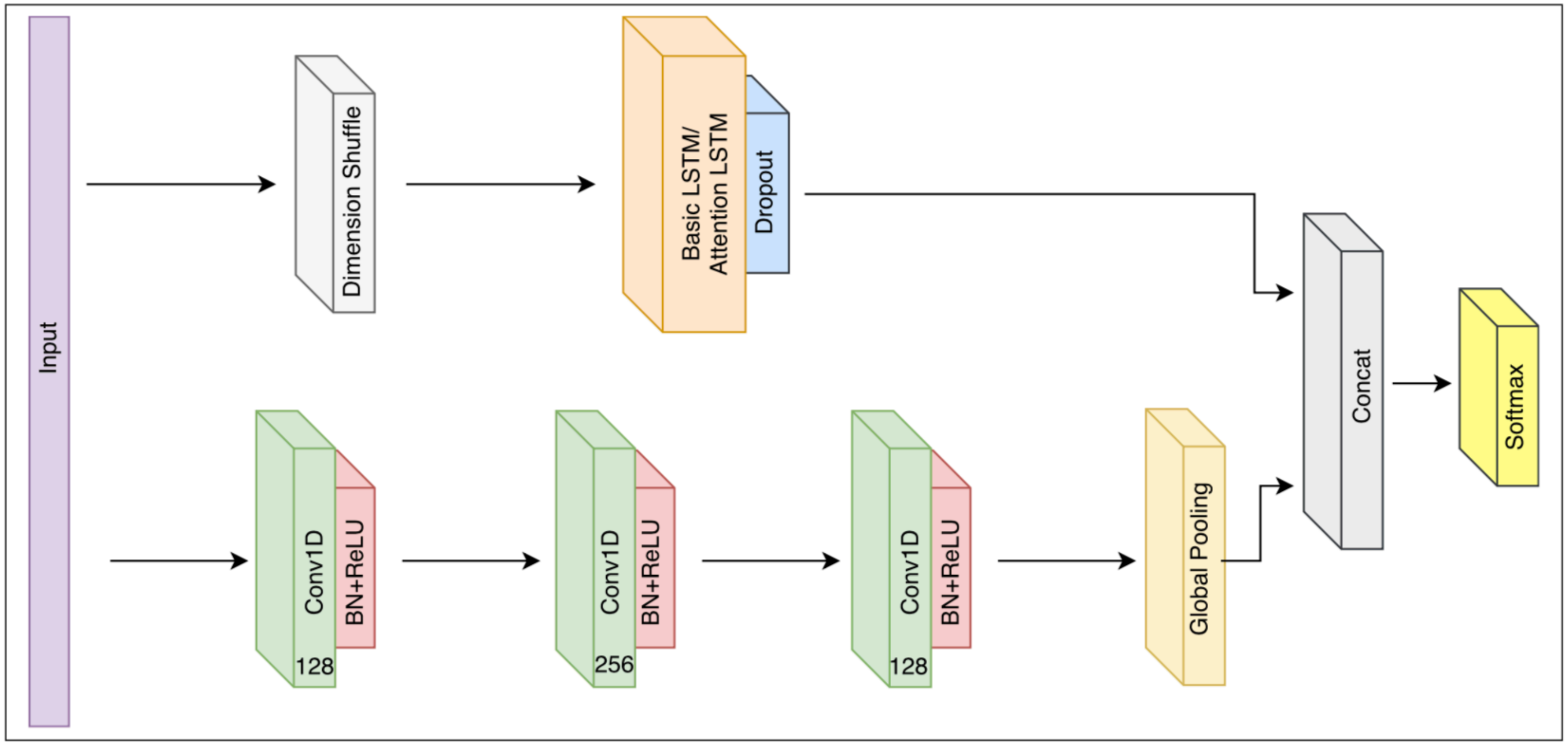


[4] Zhiguang Wang et al. (2017) “Time series classification from scratch with deep neural networks: A strong baseline”. In: 2017 International Joint Conference on Neural Networks.

TSC with deep neural networks

An extension of previously introduced FCN was proposed in [5] - a mixed architecture of LSTM with transposed input and FCN.

Such usage of LSTM does not take into account time-range dependencies in data. However, their experimental results on all 85 UCR datasets showed that this model achieves state-of-the-art performance on most of the datasets.



[5] Fazle Karim et al. (2018) “LSTM Fully Convolutional Networks for Time Series Classification”. In: IEEE Access 6

Experiment motivation

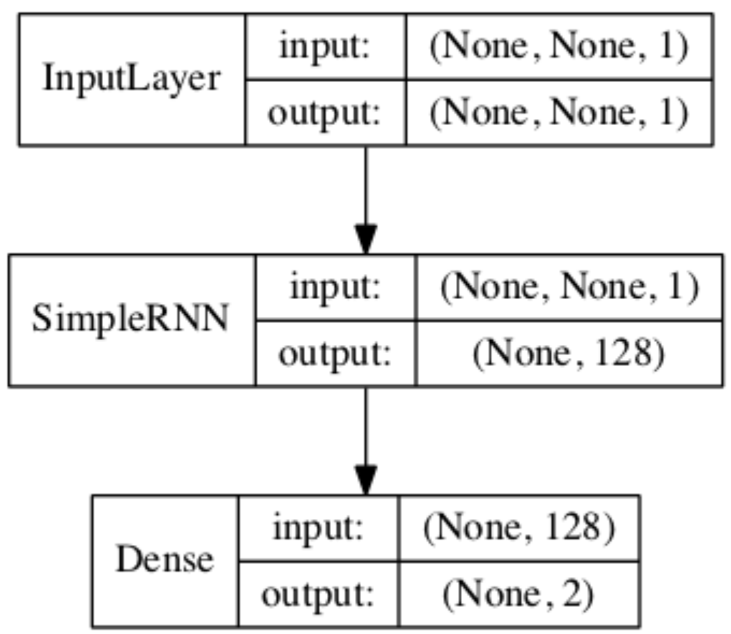
- No experimental studies were found about the efficiency of recurrent neural networks as a standalone classifiers for this task
- Experiment presented in [4] does not cover whole UCR Archive but only 44 datasets
- [4] also use their own metric - MPCE (Mean Per Class Error — mean ratio of error to the number of classes) which tends to show better values when number of classes is high

In my experiment I have reproduced MLP and FCN from [4] with the same learning settings and trained 10 different recurrent networks using all 85 UCR datasets.

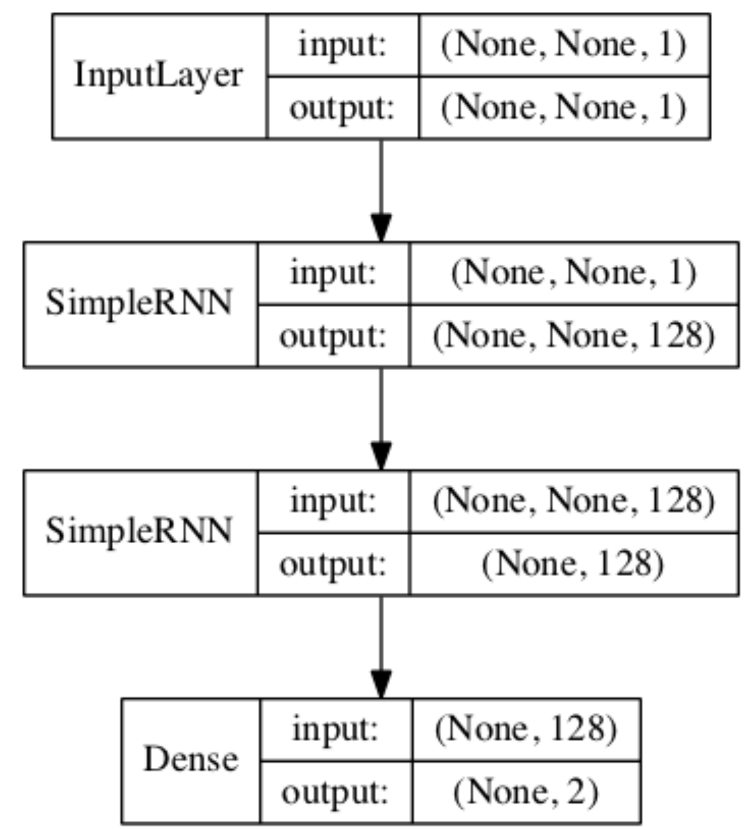
The models were compared using accuracy and weighted F1-score on the test sets.

Experiment setup

Simple recurrent networks with a dense output layer



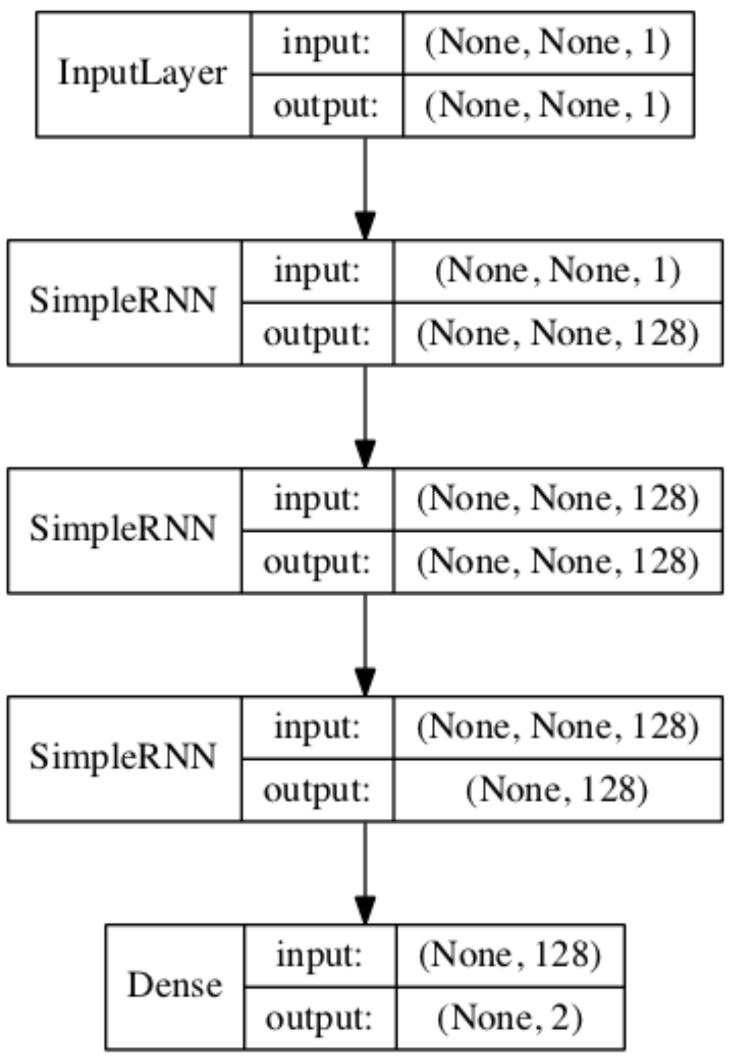
rnn_128_dense



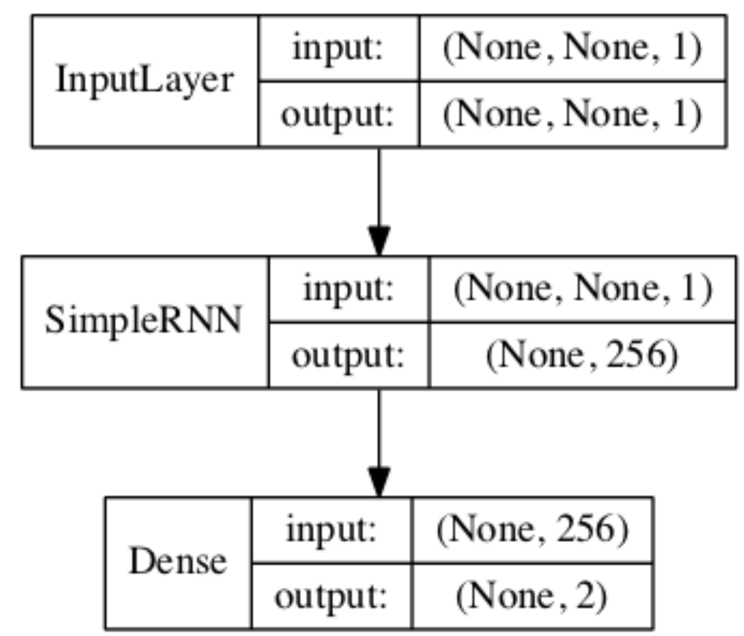
rnn_128_128_dense

Experiment setup

Simple recurrent networks with a dense output layer



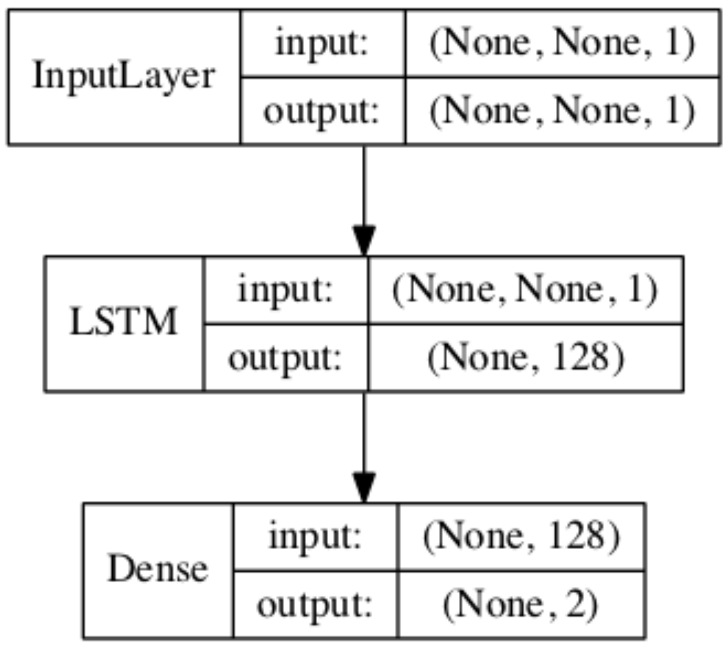
rnn_128_128_128_dense



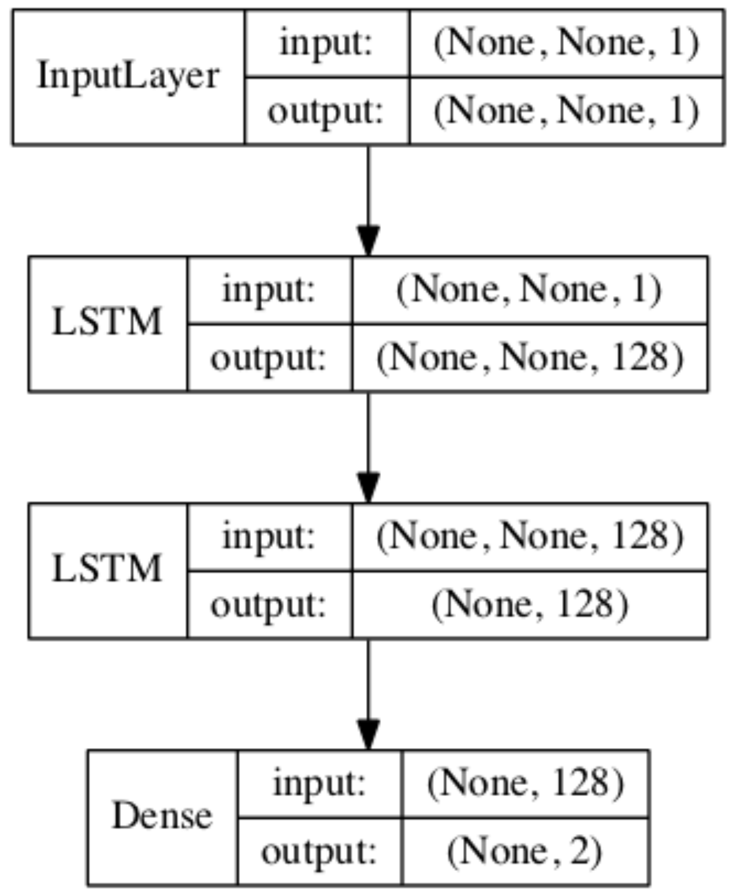
rnn_256_dense

Experiment setup

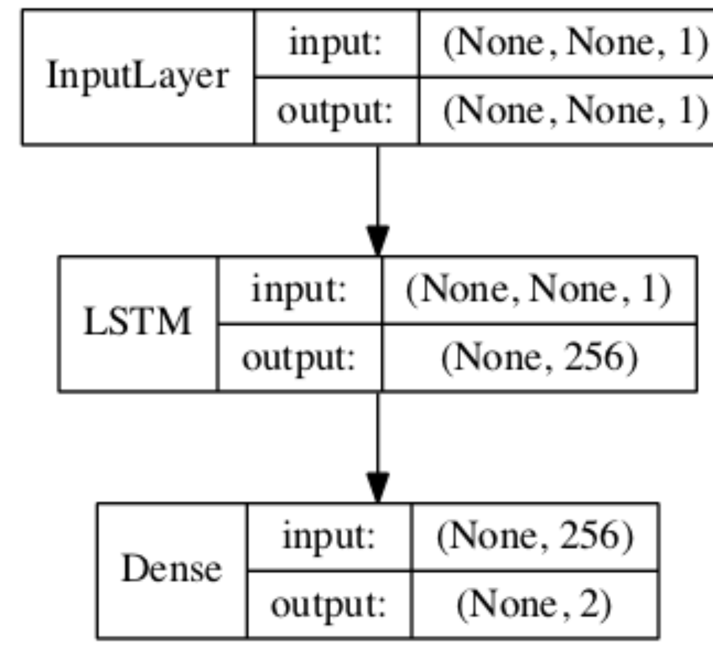
LSTM networks with a dense output layer



lstm_128_lstm



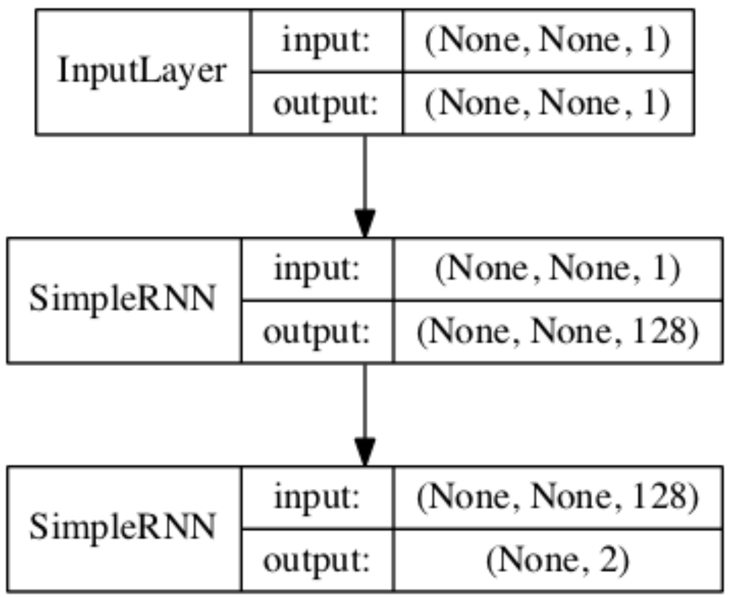
lstm_128_128_lstm



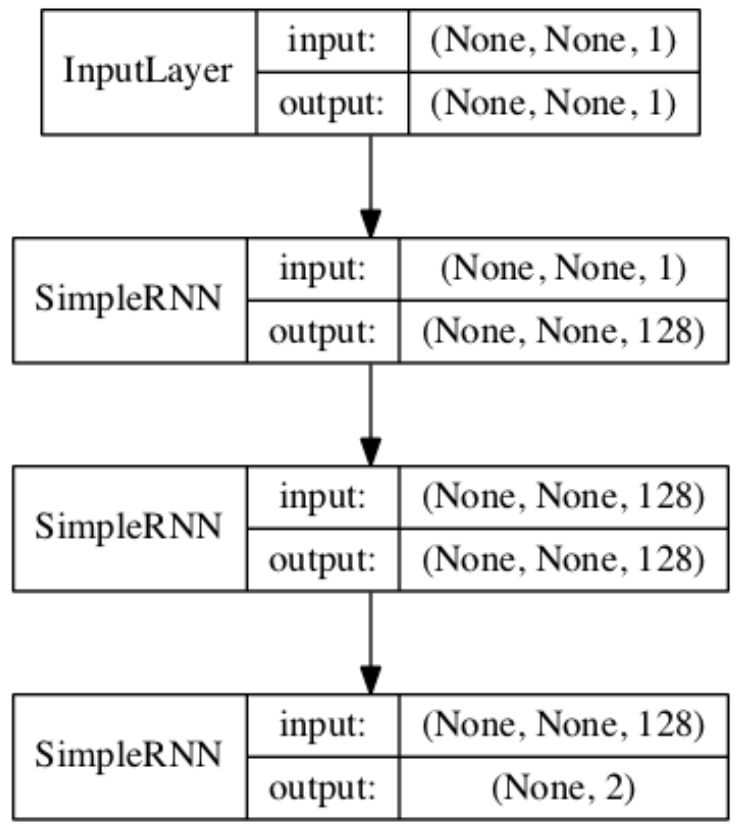
lstm_256_lstm

Experiment setup

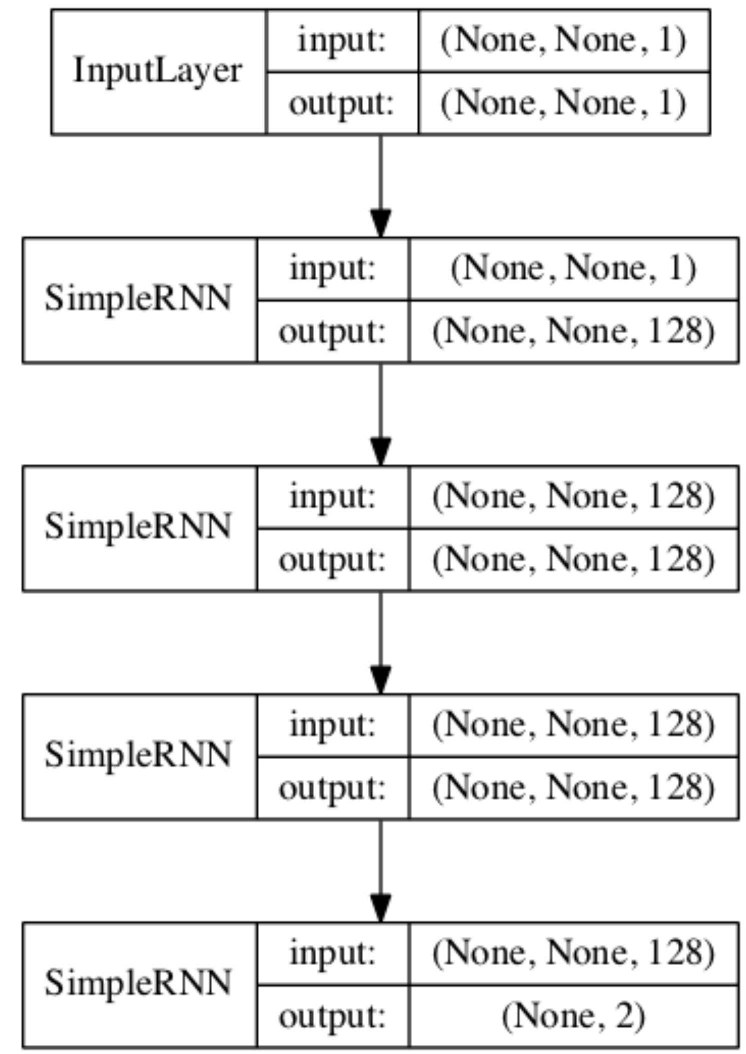
Simple recurrent networks with a recurrent output layer



rnn_128_rnn



rnn_128_128_rnn



rnn_128_128_128_rnn

Experiment setup

- Models were implemented using Keras 2 framework with TensorFlow backend (Python 3.6)
- Training all recurrent models with Adam optimizer for 500 epochs
- The models were trained on the Amazon Web Services (AWS). GPU instances with NVIDIA Tesla V100 for feedforward networks and CPU instances with 32 CPUs for recurrent networks
- Environment: the Deep Learning AMI with Conda (Ubuntu)
- Training time: ~ 80 hours of GPU instances and ~ 465 hours of CPU instances

Experiment outcomes

	Average Accuracy Train	Average Accuracy Test	Average F1-score Train	Average F1-score Test
MLP	**<u>0.975</u>**	0.723	**<u>0.974</u>**	0.724
FCN	0.963	**<u>0.801</u>**	0.957	**<u>0.792</u>**
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

Table: The average accuracy and the average weighted F-1-score over all 85 datasets

Experiment outcomes

Pairwise comparison of recurrent models using Wilcoxon singed rank test allowed to derive the following conclusions:

1. Using the LSTM layers instead of the simple recurrent layers in the considered experiment **does** increase the classification quality (both accuracy and weighted F1-score) on the test sets of the UCR datasets.

Experiment outcomes

Pairwise comparison of recurrent models using Wilcoxon singed rank test allowed to derive the following conclusions:

1. Using the LSTM layers instead of the simple recurrent layers in the considered experiment **does** increase the classification quality (both accuracy and weighted F1-score) 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 affect the classification quality on these datasets.

Experiment outcomes

Pairwise comparison of recurrent models using Wilcoxon singed rank test allowed to derive the following conclusions:

1. Using the LSTM layers instead of the simple recurrent layers in the considered experiment **does** increase the classification quality (both accuracy and weighted F1-score) 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 affect the classification quality on these 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.

Experiment outcomes

Pairwise comparison of recurrent models using Wilcoxon singed rank test allowed to derive the following conclusions:

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.

Experiment outcomes

Pairwise comparison of recurrent models using Wilcoxon singed rank test allowed to derive the following conclusions:

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.

Experiment outcomes

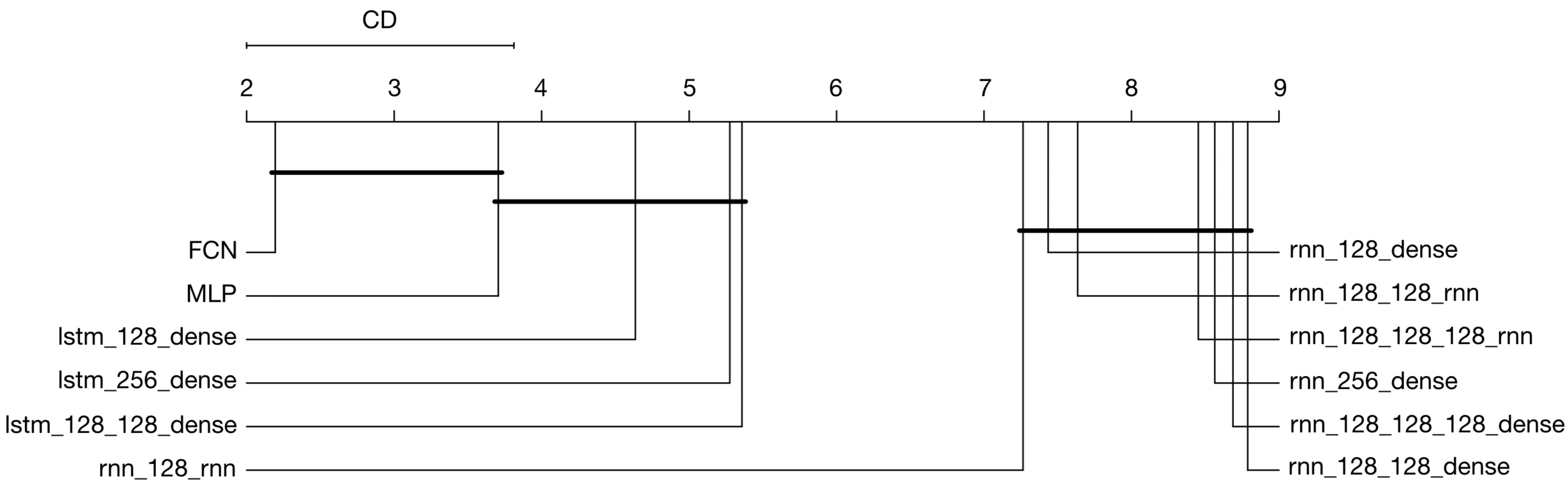


Figure: critical difference plot built on test accuracies

Experiment outcomes

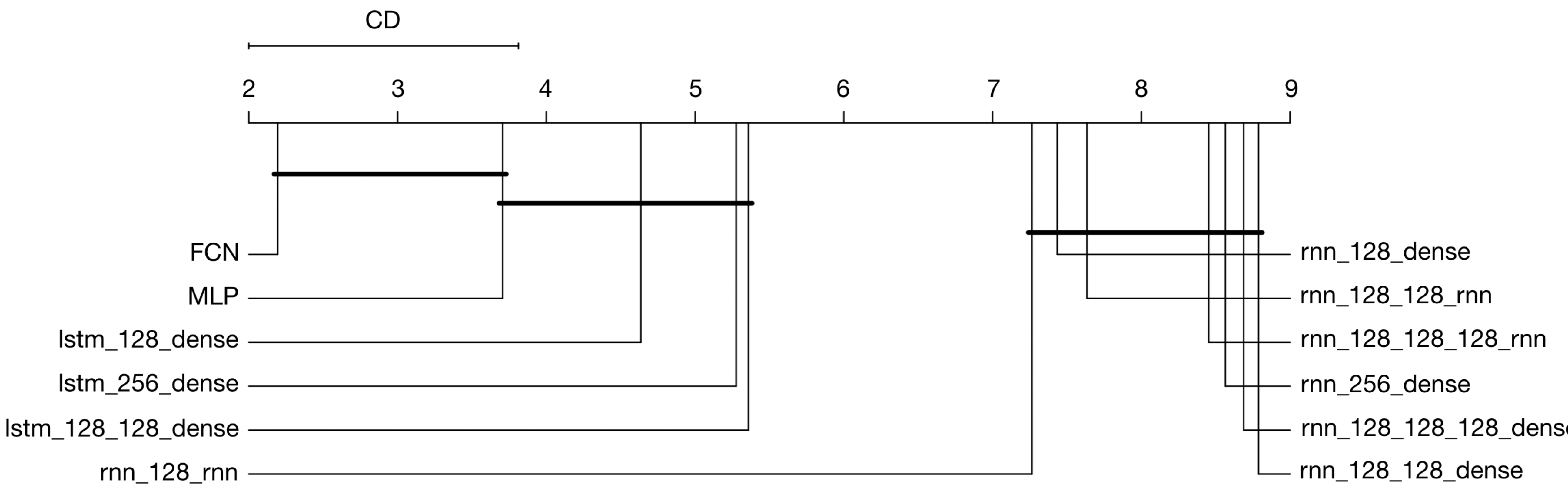


Figure: critical difference plot built on test accuracies

FCN performs significantly better than all considered recurrent architectures.

The results of FCN and MLP are not significantly different between each other.

Experiment outcomes

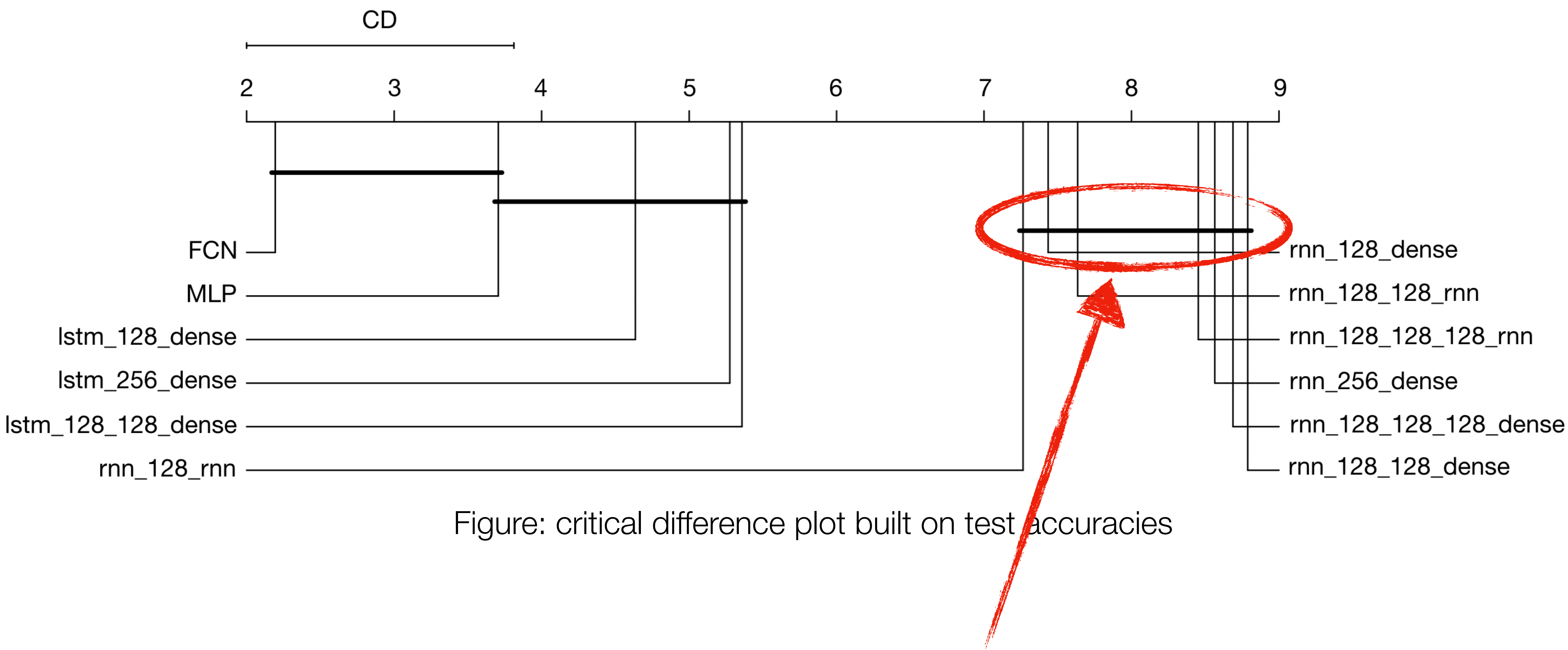


Figure: critical difference plot built on test accuracies

All considered models with simple recurrent layers showed significantly worse results than other architectures and their performance is not significantly different between each other

Experiment outcomes

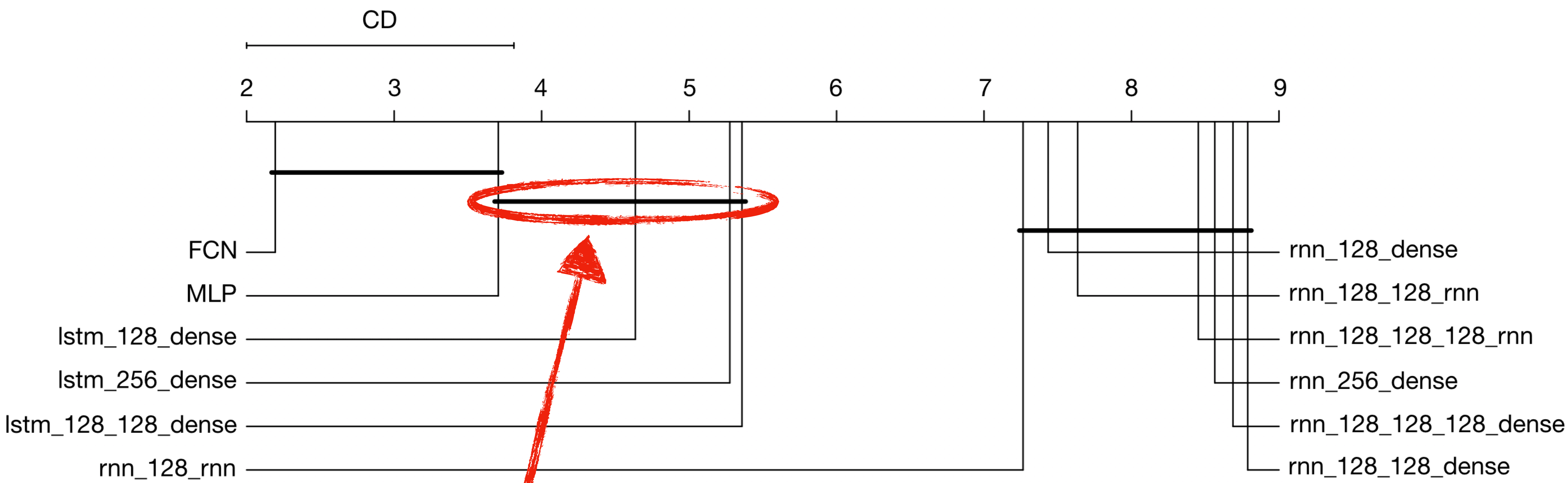


Figure: critical difference plot built on test accuracies

The difference between the performance of the MLP and all considered LSTM networks turned out to be not significant in this experiment.

Summary

- An experimental evaluation of 10 recurrent architectures and two previously proposed feedforward models was performed on all 85 UCR datasets
- Pairwise comparison of different recurrent settings allowed to derive several conclusions about their performance on UCR datasets
- Studied recurrent models were not able to outperform FCN but we believe that they are still a viable choice for TSC because of limitations of feedforward models

Thank you for your attention!

The source code is available at https://github.com/denis-smirnov/tsc-with-rnn

You can contact me at denis.m.smirnov@gmail.com

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)

[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)

[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] 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. (2017), pp. 1578–1585.

[5] 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.

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