Artificial Neural Network and Deep Learning

Homework 2
Timeseries Forecasting

Team Members: *Gabriele Marra, Matteo Miceli, Giuseppe Piccirillo*

1 Overview

The homework aims to predict the future samples of a time series.
In particular, the provided training multivariate time series consists in 68528 samples and 7 features.
We are supposed to build a forecasting model able to predict the next 864 samples.

2 Testing models

We decided to test and implement incrementally all the techniques and models taught during the course to see the different performances.

We could observe the quality of our models utilizing the Root Mean Squared Error (RMSE) as the main metric, both for each feature and an overall score.

In almost every model, we used a validation set split of 10% of the available dataset.

2.1 Direct

In the beginning, we tried to implement models similar to the one used during the lessons.

The first network is quite simple: in the first section, there are two *LSTM* layers, separated by *Dropout* layers, followed by two *Dense* layers along with a Convolutional (*Conv1D*) layer.

Later we keep improving the direct model trying different configurations and adding new layers. We obtained a better result with a *Conv1D* layer after each LSTM layer and inserting a *MaxPooling1D* layer between them, as shown in Figure 1.

2.2 Autoregressive

After experimenting with direct models, we wanted to try autoregression: we took some already built architectures, and we modified them to support the autoregressive method.
We noticed that the *RMSE* slowly decreased at each quarter of our goal samples, and it is a result we couldn't achieve with the direct method.

Unfortunately, even the best Autoregressive model we obtained was definitely worst than our best Direct model, so we decided to focus on other types of networks.

2.3 Attention implementation

We then tried to implement networks using a location-based attention mechanism. We noticed a performance improvement compared to the first direct models. However, this result has been overtaken by the latest Direct models we produced.
We also tried a transformer-like architecture or rather a self-attention mechanism but with even worse results.

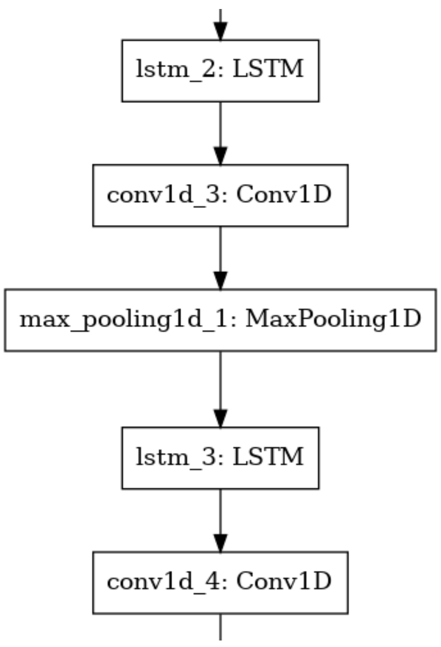


Figure 1: Section of model with LSTM, Conv1D, and MaxPooling1D

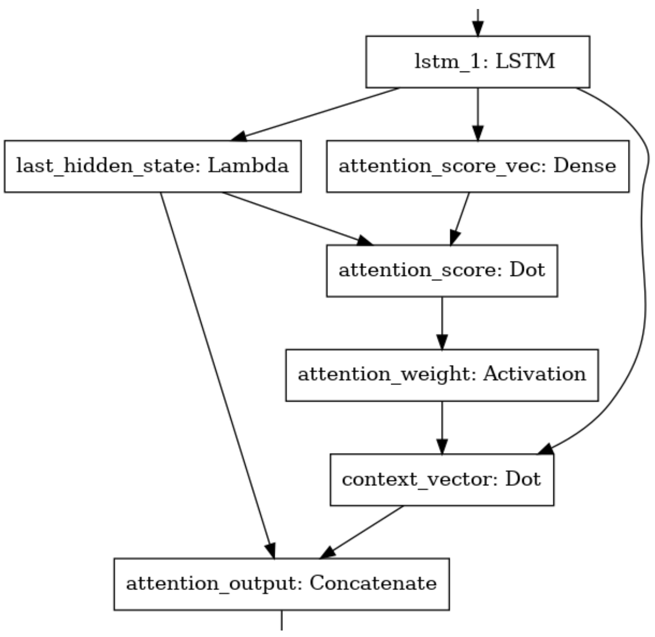

lstm_1: LSTM
last_hidden_state: Lambda
attention_score_vec: Dense
attention_score: Dot
attention_weight: Activation
context_vector: Dot
attention_output: Concatenate

Figure 2: Section of Attention model

2.4 Honourable Mentions

2.4.1 Evaluation

At some point during our testing, we noticed that a smaller RMSE on the validation test was not always reflected on a smaller RMSE on the CodaLab test set. For this reason, we tried other ways to evaluate our models before submitting them on CodaLab:

- using the RMSE of the test set (obtained from the last 10% of the original dataset) to get a better idea of the RMSE on the not normalized value
- not normalizing the dataset before training for the same idea of the point before, but this lead to a more difficult training procedure

2.4.2 Learning Rate Plateau

To improve the training and reduce overfitting, we implemented a plateau for the learning rate: when the learning stagnates for a specified number of epochs, the learning rate is reduced.

2.4.3 Patience

We noticed substantial different RMSE scores on the validation test between multiple executions of the same model (with identical values for hyperparameters). We also observed how the graph of the validation loss was not diverging that much before stopping due to EarlyStopping.
So, we tried to increase the patience using KerasTuner of both EarlyStopping and Learning Rate Plateau to not interrupt the training too early, obtaining the desired improvements.

2.4.4 No train/validation split

We tried to train our models also on all the dataset, without splitting in training and validation sets. To reduce the probability of overfitting, we set as the max number of epochs the average value of the last 30 training (with similar models). This method brought us a small improvement.

3 Final decision

After experimenting with all the models, our best model is a Direct forecasting model with the architecture shown in Figure 3 and Figure 4.

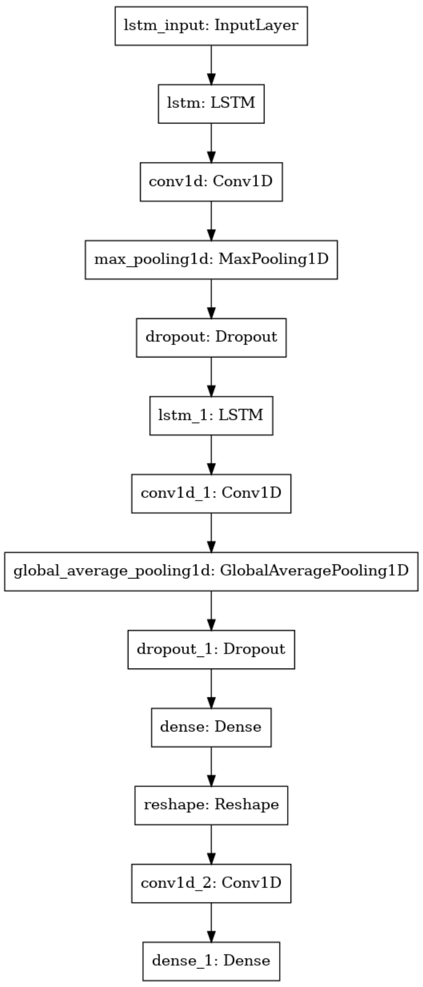
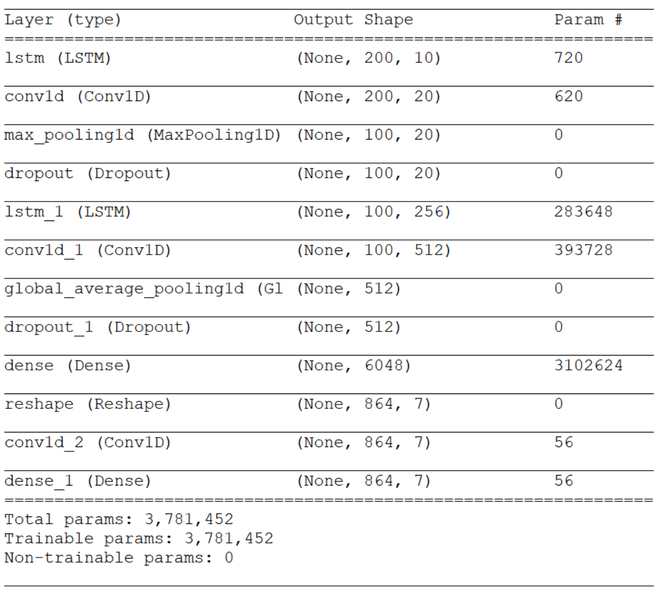


Figure 3: Best Direct model diagram



```
Layer (type)                 Output Shape              Param #
=================================================================
lstm (LSTM)                  (None, 200, 10)           720
_________________________________________________________________
conv1d (Conv1D)              (None, 200, 20)           620
_________________________________________________________________
max_pooling1d (MaxPooling1D) (None, 100, 20)           0
_________________________________________________________________
dropout (Dropout)            (None, 100, 20)           0
_________________________________________________________________
lstm_1 (LSTM)                (None, 100, 256)          283648
_________________________________________________________________
conv1d_1 (Conv1D)            (None, 100, 512)          393728
_________________________________________________________________
global_average_pooling1d (Gl (None, 512)               0
_________________________________________________________________
dropout_1 (Dropout)          (None, 512)               0
_________________________________________________________________
dense (Dense)                (None, 6048)              3102624
_________________________________________________________________
reshape (Reshape)            (None, 864, 7)            0
_________________________________________________________________
conv1d_2 (Conv1D)            (None, 864, 7)            56
_________________________________________________________________
dense_1 (Dense)              (None, 864, 7)            56
=================================================================
Total params: 3,781,452
Trainable params: 3,781,452
Non-trainable params: 0
_________________________________________________________________
```

Figure 4: Best model Summary

With a similar architecture, we initially obtained an RMSE on CodaLab of *4.0480*.
After tens of training on our local machine, we tuned all the hyperparameters and obtained the final model. The final RMSE score on CodaLab is *3.8090*.