Artificial Neural Networks and Deep Learning 2021

Homework 2 – Time Series Forecasting

Prof. Matteo Matteucci

Prof. Giacomo Boracchi

Group members:

Hamid Salehi 10759997

Hossein Mohammadalizadeh 10717597

Anastasia Cotov 10767910

Academic Year 2021/2022

Define the problem

In this challenge we are dealing with a multivariate time series task. The goal is to design and implement forecasting models to learn how to exploit past observations in the input sequence to correctly predict the next 864 future sample.

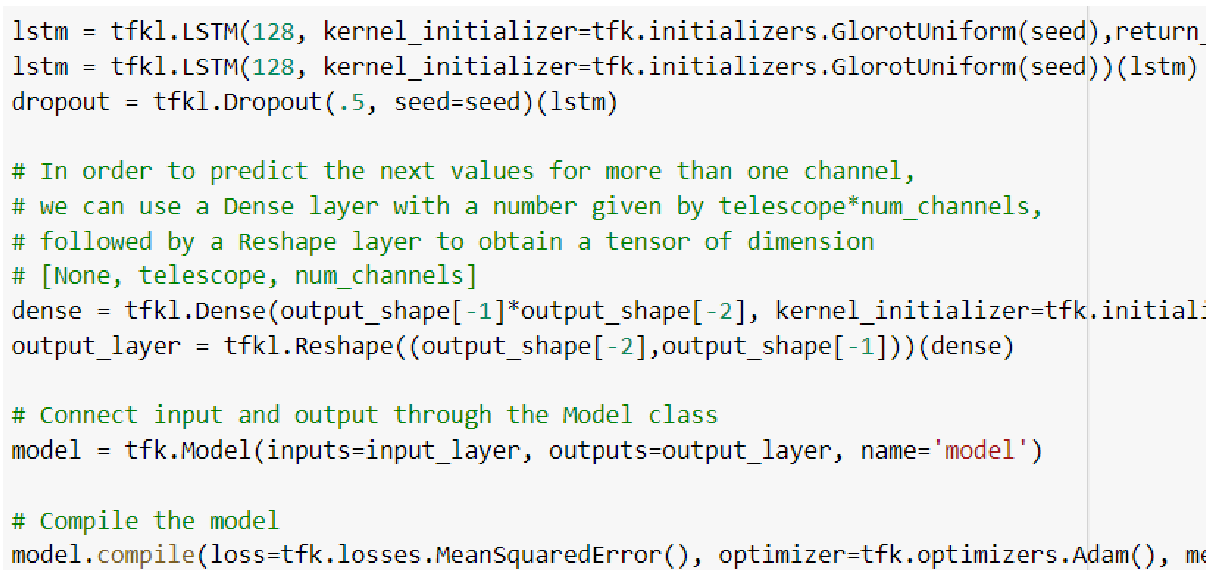
Big Picture

We have different ways to deal with sequences like hidden Markov models, linear dynamic systems. In this challenge we use Recurrent Neural Networks which can lead us to do forecasting in time series task. One of the most common ways to implement RNN is to use LSTM blocks because they can solve the issue of vanishing gradients. In addition to this, we have two different ways to tackle with this challenge:

1. Direct forecasting which means we forecast 864 samples at a time
2. Autoregressive forecasting in which model can forecast less than 864 samples but with the advantage of sliding windowing, at the end it can predict total samples that we want.

Code/Model

Our final model skeleton is represented in figure below. This is our base model and all tuning of hyperparameters is done according to it. From our experience on training challenge dataset the best model was the simple one with few bidirectional LSTM layers. To be mentioned that we trained model with convolutional and bi-LSTM layers together, however the result was a higher RMSE=8.3 on hidden test set.



```
lstm = tfkl.LSTM(128, kernel_initializer=tfk.initializers.GlorotUniform(seed),return_
lstm = tfkl.LSTM(128, kernel_initializer=tfk.initializers.GlorotUniform(seed))(lstm)
dropout = tfkl.Dropout(.5, seed=seed)(lstm)

# In order to predict the next values for more than one channel,
# we can use a Dense layer with a number given by telescope*num_channels,
# followed by a Reshape layer to obtain a tensor of dimension
# [None, telescope, num_channels]
dense = tfkl.Dense(output_shape[-1]*output_shape[-2], kernel_initializer=tfk.initiali
output_layer = tfkl.Reshape((output_shape[-2],output_shape[-1]))(dense)

# Connect input and output through the Model class
model = tfk.Model(inputs=input_layer, outputs=output_layer, name='model')

# Compile the model
model.compile(loss=tfk.losses.MeanSquaredError(), optimizer=tfk.optimizers.Adam(), me
```

Parameters of models

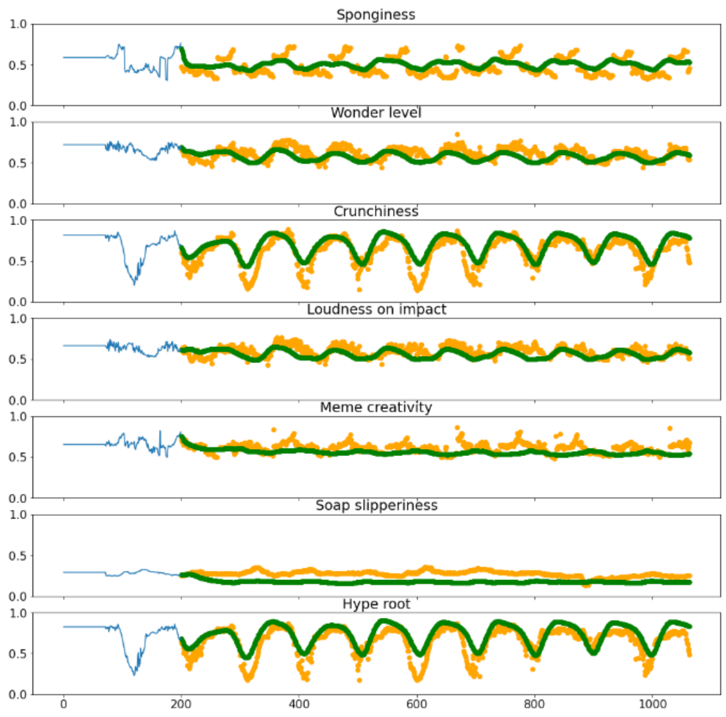
Similar to all machine learning pipeline, the first step is to create our dataset. In this task we are given signals with 68528 samples, so we must choose the number of samples (W) that we are going to predict based on them. The stride that we use is another parameter that we can change. As mentioned, in autoregressive method, we also can play with the number of samples that we want to predict (telescope).

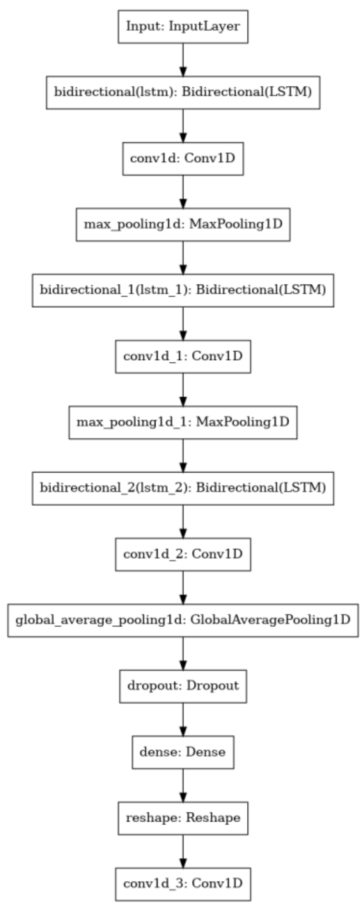
Besides, all the parameters used in LSTM and classic classifier are our other parameters. Here are some important models that use the simple model below but with different parameters:

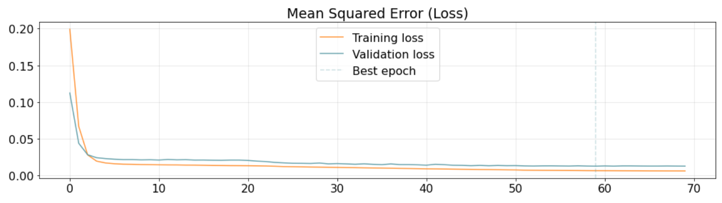
	MODEL1	MODEL2	MODEL3	MODEL4	MODEL5	MODEL6	MODEL7	MODEL8	MODEL9
Direct or Autoregressive	Auto	Auto	Auto	Auto	Auto	Auto	Auto	Auto	Direct
Window	576	576	576	576	576	576	576	576	1728
Stride	24	24	24	24	24	24	12	12	12
Telescope	288	288	288	288	288	288	288	288	864
Bidirectional LSTM or not	no	yes	yes	yes	yes	yes	yes	yes	yes
Number of LSTM layers	2	2	2	2	3	3	3	3	3
Number of units of LSTM	128	128	256	512	256	256	256	256	512
The amount of dropout	0.5	0.5	0.5	0.5	0.5	0.3	0.5	0.5	0.5
Number of classifier layers	1	1	1	1	1	1	2	1	1
Number of units of classifier	2016	2016	2016	2016	2016	2016	256, 2016	2016	6048
RMSE	4.25	4.2	3.96	3.86	3.8	3.83	3.92	3.63	4.96

Results

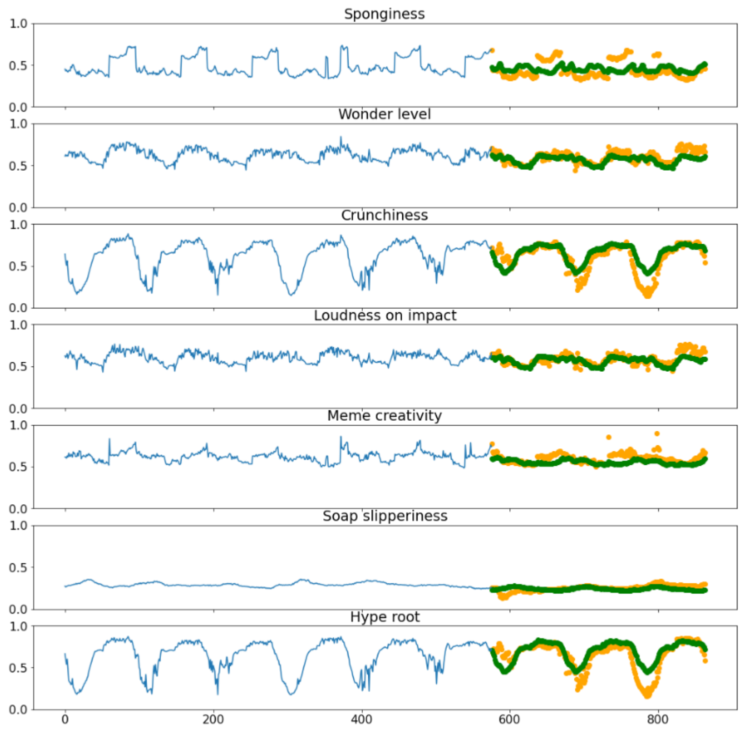
We started solving the challenge using a Conv1D layers with bi-LSTMs which performed good during the training, with loss below **0.05** and quite good prediction of time series represented below in plots (prediction is the green line). On hidden test set this model got **RMSE=11.33** which by tunning parameters did not go below **8.3**. Here we encountered overfitting problem.

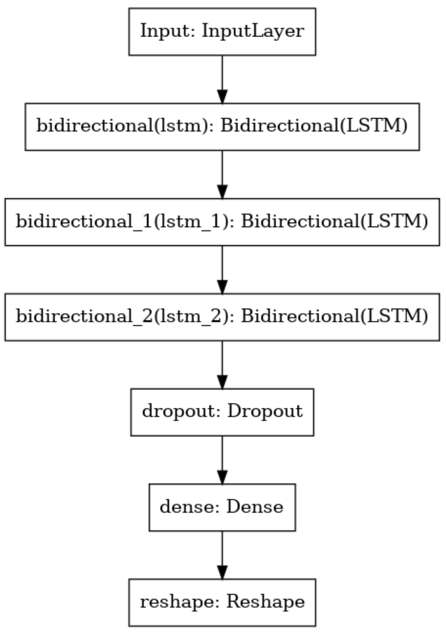


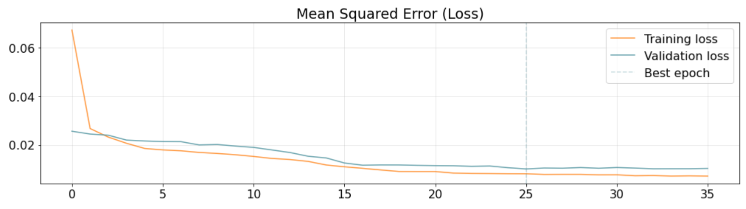

Input: InputLayer
bidirectional(lstm): Bidirectional(LSTM)
conv1d: Conv1D
max_pooling1d: MaxPooling1D
bidirectional_1(lstm_1): Bidirectional(LSTM)
conv1d_1: Conv1D
max_pooling1d_1: MaxPooling1D
bidirectional_2(lstm_2): Bidirectional(LSTM)
conv1d_2: Conv1D
global_average_pooling1d: GlobalAveragePooling1D
dropout: Dropout
dense: Dense
reshape: Reshape
conv1d_3: Conv1D



Our best model during this challenge turns out to be very simple, as we mentioned before our base model consists of few bi-LTSM layers. As we can see from pictures below during the training, model performs very well with time series prediction. Mean square error goes below **0.02** and during testing on the hidden set this model outperformed all the other models we have tried, having **RMSE=3.63.**




Input: InputLayer
bidirectional(lstm): Bidirectional(LSTM)
bidirectional_1(lstm_1): Bidirectional(LSTM)
bidirectional_2(lstm_2): Bidirectional(LSTM)
dropout: Dropout
dense: Dense
reshape: Reshape



Conclusion

For given task in challenge 2, we conclude that using more complex models with both Conv1D and bi-LSTM layers learnt to provide good performance during training time, but it gave us poor result during testing on hidden set, we ended up with overfitting. One of the reasons this happens is because of huge number of parameters we had to train with our model. To overcome this issue, we tunned hyperparameters, used early stopping, dropout layer ending up with a result of 8.3 RMSE.

Moving forward to less complicated models, implementing only bi-LSTM layers we avoid overfitting we took the same strategy as before: multiple tunning of hyperparameters, using early stopping, regularization. Our best result is 3.6 RMSE.

Solving this challenge made us to conclude that there is no best strategy for all types of time series prediction. The most complex model with a lot of layers does not guarantee a better performance. The best way to deal with problem is to analyze the given dataset, to try different combinations of layers in the model, to tune hyperparameters and, of course, use techniques to avoid overfitting or vanishing gradient issues.