Home work2

At the all steps, the validation size is 10 percent of the whole data, batch_size is 64, and also, we used min-max normalization.

Step 1: In this step, we tried some different models with random parameters for choosing the best architecture. We have used Bi_LSTM (Bidirectional LSTM), Bi_GRU (Bidirectional GRU), CNN, LSTM, GRU, and a combination of these layers. Our best results were with Bi_LSTM and LSTM layers.

During training, we used Autoregression (for example Telescope_size = 1) and reached poor results. The main reason for these results is that, we predict only 1 next step and we put it in the input set and we use this value as X for predicting the next Y. Actually, we have now a cumulative error. What does it mean? In prediction, typically, we have some error. We predict the next Y with error and put it in X set and this operation repeats over and over and leads to predicting the next steps with more errors. As a result of this, we tried mostly with large Telescope_size (mostly equal to test size).

We have chosen different Window_size; 20,50,100,200,400,500,800. The best results were reached with 400 or 500.

Using GlobalMaxPooling (GMP) and GlobalAveragePooling (GAP) in some cases worked better.

Having dropout after the LSTM layer sometimes helps us to get good performance and after several trains, we reached that maximum 0.2 dropout is good.

With choosing large Window_size maybe we face gradient vanishing. Although LSTM layers are made for dealing with vanishing the gradient, in long sequences it could happen still. For solving this problem, we used skip connection and we got good results with it.
Table 1 is presented a summary of some architectures with different parameters that shows why we selected these parameters (We tried a lot of architectures but in this table for summarizing we mentioned a few of those).

architecture	Test RMSE
Bi_Lstm(64)+Conv(128,3)+Bi_lstm(128)+Conv(256,3)+GAP+DR(0.5) / Windows(200) Telescope(864)	8.4667
Bi_Lstm(64)+Conv(128,3)+ Bi_lstm(128)+Conv(256,3)+GAP+DR(0.5) / Windows(500) Telescope(864)	8.0982
Bi_Lstm(64)+Conv(128,3)+ Bi_lstm(128)+Conv(256,3)+GAP+DR(0.5) / Windows(50) Telescope(1)	8.6078
Bi_Lstm(64)+Conv(128,3)+ Bi_lstm(128)+Conv(256,3)+GAP+DR(0.5) / Windows(20) Telescope(1)	10.0501

Bi_Lstm(64)+Conv(128,3)+ Bi_lstm(128)+Conv(256,3)+GAP+DR(0.5) / Windows(20) Telescope(9)	10.2858
Bi_Lstm(64)+ Bi_lstm(128)+GAP / Windows(800) Telescope(864)	7.8362
Bi_Lstm(64)+ Bi_lstm(128)+GAP / Windows(500) Telescope(864)	4.4480
Bi_Lstm(64)+ Bi_lstm(128)+GAP+ DR(0.2) / Windows(500) Telescope(864)	4.9064
Lstm(64)+lstm(128)+GAP/ Windows(500) Telescope(864)	4.2321
Bi_Lstm(64)+Bi_lstm(128)+GAP / Windows(500) with skip connection Telescope(864)	4.0171
Bi_Lstm(64)+Bi_lstm(128)+GAP+ DR(0.2) / Windows(400) with skip connection Telescope(864)	4.6925
Bi_Lstm(64)+Bi_lstm(128)+GMP/ Windows(400) with skip connection Telescope(864)	4.1797
Lstm(64)+ lstm(128)+GAP/ Windows(500) with skip connection Telescope(864)	5.5934
Lstm(64)+ lstm(128)+GMP+DR(0.2) / Windows(400) with skip connection Telescope(864)	3.8936
Lstm(64)+ lstm(128)+GAP+DR(0.2) / Windows(400) with skip connection Telescope(864)	3.8081
Lstm(64)+ lstm(128)+GAP+DR(0.2) / Windows(400) with skip connection and DR(0.2) Telescope(864)	**3.7356**

Table 1: Summary of trained models.

In the skip connection architecture, we feed input to different LSTM layers and finally we concatenate these parts. When we used skip connection, we reached the best results (3.7356 RMSE).

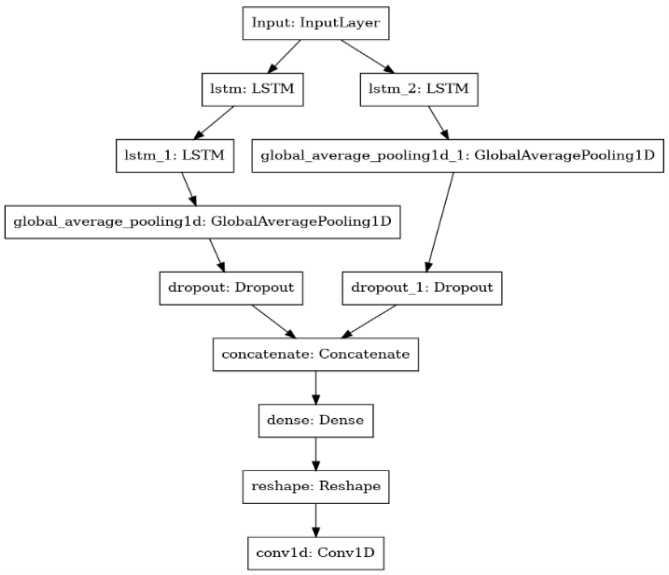

Figure 1: Architecture for skip connection.

Step 2: According to table 1 choosing big window_size could help the prediction but not necessarily. We computed Autocorrelation Function (ACF) for all features to see how each feature is correlated with a lagged copy of itself. (For simplification here the lag is 400).

feature 0→ corr=0.3740, feature 1→ corr =0.5529, feature 2→ corr =0.2935, feature 3→ corr =0.5784, feature 4→ corr =0.6165, feature 5→ corr =0.8675, feature 6→ corr =0.3002.

Predicting large telescope_size is difficult and the more telescope_size increases, the more autocorrelation decreases. We need a model that uses less window_size and computes less

amount of next time steps. We tried several different models and finally with telescope 288 (equal to 1/3 of the whole test set) and window_size 300 we got 3.9883 RMSE.

Step 3: Until now our best idea is that use large window_size and telescope_size but because LSTM layers bottleneck is long sequences, we want to use a transformer as a predictor. Attention mechanisms could apply to regression problems, for example [1],[2].
We applied a multi-head attention mechanism with different parameters and have gotten 3.8899 RMSE. As well as, the attention mechanism is faster than LSTM, when we trained this model, we faced this point.

```
input_layer = tfkl.Input(shape=input_shape, name='Input')
layer = MultiHeadAttention(num_heads=50 ,key_dim=10,)(input_layer,input_layer,input_layer)

flatt2 = tfkl.Flatten()(layer)
layer = tfkl.Dropout(.2)(flatt2)

dense = tfkl.Dense(output_shape[-1]*output_shape[-2], activation='relu')(layer)
output_layer = tfkl.Reshape((output_shape[-2],output_shape[-1]))(dense)

output_layer = tfkl.Conv1D(output_shape[-1], 1, padding='same')(output_layer)

model = tfk.Model(inputs=input_layer, outputs=output_layer, name='model')

# Compile the model
model.compile(loss=tfk.losses.MeanSquaredError(), optimizer=tfk.optimizers.Adam(), metrics=['mae'])
```

Figure 2: Multi-head attention mechanism architecture.

All in all, the best architectures for dealing with this dataset were LSTM with skip connection (telescope size 864 and 288), and multi-head attention mechanism. We selected hyperparameters randomly, but by applying hyperparameter tuning methods such as grid searching, we can obtain the lowest loss.

Finally, we represented both LSTM with skip connection and multi-head attention architecture but our best model was LSTM with skip connection.

References

1. **https://www.kaggle.com/miljan/stock-predictions-with-multi-head-attention**

2. Abbasimehr, Hossein, and Reza Paki. "Prediction of COVID-19 confirmed cases combining deep learning methods and Bayesian optimization." *Chaos, Solitons & Fractals* 142 (2021): 110511.