AN2DL Second Homework
Time Series Forecasting

Team Gamma

Rivi Gabriele 994131 10663569
Redaelli Mattia 995873 10622823
Ratzonel Ariel 995067 10631746

January 21th, 2022

1 Introduction

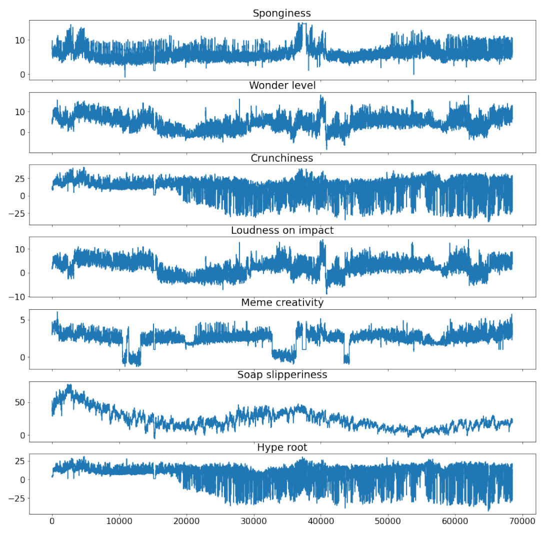
1.1 About the project

The goal of the project is to predict future samples of a multivariate time series. The TimeSeries Forecasting problem is characterized by analysing a sequence of data showing past interactions and behaviour, and eventually exploiting recurrencies and seasonalities the model can make a reliable prediction of how will be the future for a fixed number of timesteps. In this case, our model has to predict 864 samples after the 68528 given as training set.

1.2 Dataset details

- Length of the time series: 68528 timesteps
- Number of features: 7 We split the time series in this way:
  - `Training`: 80% of the length.
  - `Validation`: 10% of the length.
  - `Testing`: 10% of the length (last samples).

The dataset is normalized on the difference between the maximum and the minimum value, so: $X = \frac{X - min}{max - min}$ where X is the input sample and min and max are computed on the entire dataset.



2 Design choices

2.1 Direct vs Autoregressive Approach

The Timeseries Forecasting problem is characterized by having to predict future values basing the prediction on past information. There are two main techniques for this problem, similar but with a distinctive difference, characterized by a parameter called *Telescope*:

- **Direct**: The 864 samples to predict are all computed in a single shot, so the Telescope is set to 864.
- **Autoregressive**: The 864 samples are predicted in successive iteration, in batches of size equal to the Telescope. The most basic case is where Telescope is 1.

In our case, after widely trying out both approaches, we opted to use a mix of the two, an autoregressive technique using a Telescope of length different from 1. The loss function used to compute the error and on which the model is evaluated on the leaderboard is the `Root Mean Squared Error`.

2.2 Direct

The first model used `Conv1D` layers acting as feature extractor from the timeseries and let the processing be done by some Bidirectional `LSTM` layers. The difference of `BILSTM` from simple `LSTM` layers is that the timeseries is scanned both from left to right and from right to left.
Unfortunately, all this model could be able to predict was a straight line (probably due to the choice of window and stride). From there we continued using a direct approach but making the model simpler without convolutional layers and we experimented in finding which window and stride better performed on the task. The best results were obtained using a window comprised in the range of 500 to 2000 samples, with a stride in a range from 1 to maximum 10. Window and stride are hyperparameters, indicating how many samples are comprised in the "sliding window" that analyse the train set, and how much the window slides after each step of training. From this stage we got to a maximum score of 4.00 RMSE on the leaderboard.

2.3 Autoregressive

We passed to an Autoregressive approach, combining `Dense` layers with `BILSTM` layers. We started getting more accurate predictive model locally, thanks also to the use of `TimeDistributed` on the `Dense`. TimeDistributed apply the same layer to several inputs, and it produce one output per input to get the result in time. Also the Telescope now became an hyperparameter to tune, since we decided to opt for an hybrid approach. We valued different alternatives, all dividends of 864. The most effective came out to be 36,72 and 144. After some different architectures, we reached the score of 3.92 RMSE on the leaderboard. We tried to include some *Conv1D* layers to enhance the analysis, but didn't bring to notable results.

2.4 What didn't work

- `Keras Tuner`: For some time we used an hyperparameter tuning object to find out which was the best possible configuration for some hyperparameters in our model but it came out as being too much time consuming and didn't improve the result significantly with regards to a tuning done manually.
- `Window too little or too large`: keeping the window value out of the previously indicated range result in poor results and even straight lines in prediction.
- `Batch Normalization, GRU layer`

2.5 Final Model and Approach

In the end, what proved to make a significant difference was the implementation of an *Attention* layer. The model is as follows:

1. `Bidirectional LSTM, 512 units and return sequences + Dropout(p=0.3)`
2. `Attention Layer`
3. `RepeatVector`
4. `TimeDistributed Dense, 512 units + Dropout(p=0.3)`
5. `Bidirectional LSTM, 512 units and return sequences + Dropout(p=0.3)`
6. `Attention Layer`
7. `Dense + Reshape as seen in class to correctly format output tensor`

Hyperparameters: Window: 800 | Stride: 5 | Telescope: 144 | Batch Size = 128
We tried to combine this Attention Layer with various models, using different approaches and different architectures but in the end this model listed here was the best performing one on the leaderboard, with a score of 3.6204 *RMSE*.

3 Test Results

3.1 Training plots & Predictions

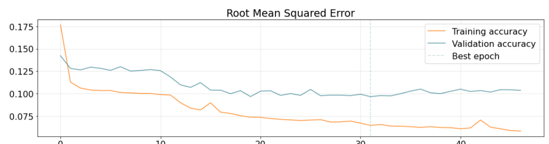


Fig. 1: RMSE

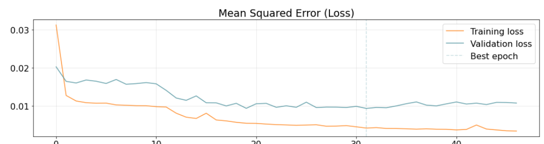


Fig. 2: MSE Loss

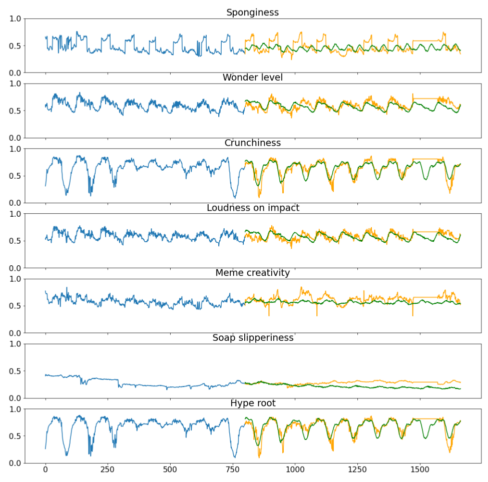


Fig. 3: Testing

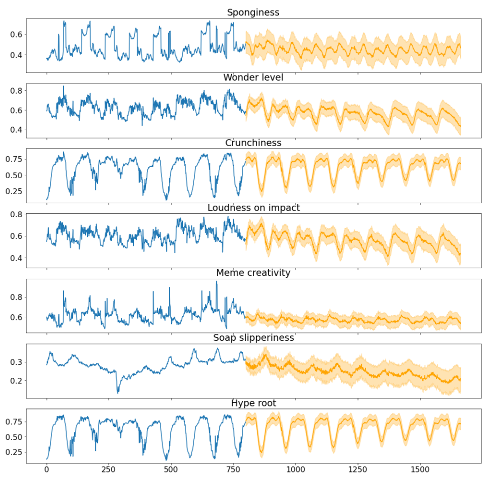


Fig. 4: Prediction