Time Series Forecasting

Marco Domenico Buttiglione, Luca De Martini, and Giulia Forasassi

Politecnico di Milano

January 21, 2022

1 Introduction

Time series forecasting is one of the applications of deep learning methods, it consists in trying to predict a range of future values for multiple variables based on their past behaviour. This task is usually approached with recurrent neural networks. In this case the task is to predict the next 864 values for a 7 variables time series.

In this project we have explored RNN approaches to forecasting using LSTM and GRU and attention only methods inspired from the Attention is All you Need Transformer.

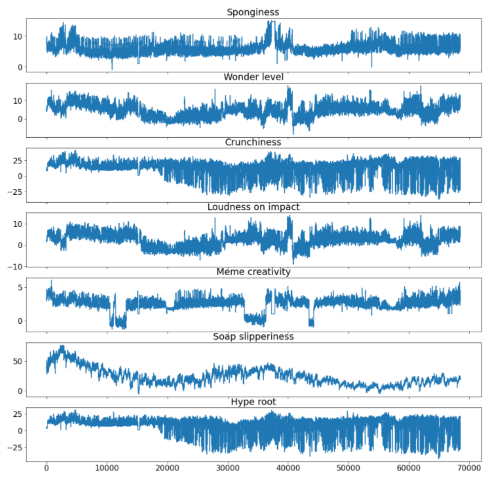

Figure 1: Inspect of a dataframe

2 Data analysis

The training dataset is a CSV file containing the floating point valued samples of a multivariate time series. There are 68528 samples for the following 7 features: sponginess, wonder level, crunchiness, loudness on impact, meme creativity, soap slipperiness and hype root. The features are not normalized and range across different scales as can be observed in Figure 1. By computing a Fast Fourier Transform we observed some periodic components in the signals (Figure 2).

The dataset has been initially divided in training, validation and test set, with a split of about 60000 samples for training, 4000 for validation and 4000 for tests. Later it was decided to remove the test set in order to increase the data available during the training. After splitting the data, we normalized each variable between 0 and 1. We also tried to augment the data by adding low variance (0.01 to 0.05) Gaussian noise to the signal to increase generalization and avoid overfitting. We observed slight improvement in validation performance in most cases.

3 Models and techniques

In this section we present the main techniques employed for our forecasting models.

3.1 LSTM

First, we tried simple and bidirectional LSTM models. We found the bidirectional models to perform worse than the forward only version. This is likely due to the nature of the prediction task, since we are interested in predicting the future, going through the data in inverse temporal order does not give much insight. Because of this, we switched to forward RNN models for all subsequent tests.

The architecture of the LSTM model starts with 2 convolutional layers with linear activation for basic feature extraction without adding too much complexity, followed by the LSTM layer from which the last output is taken as the latent representation of the input. Finally a fully connected produces the final prediction.

After obtaining satisfying results with LSTM models, we experimented with GRU models, but

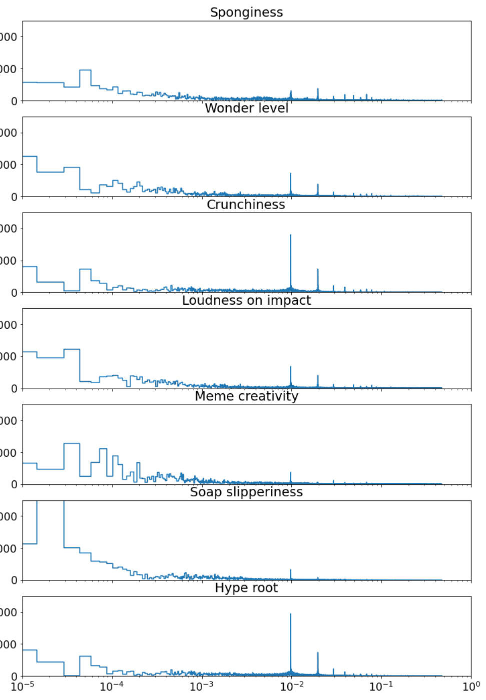

Figure 2: Discrete Fourier transform

we found the result to be slightly worse.

3.2 Transformer

We also experimented with Transformer models inspired from the Attention Is All You Need paper. The transformer model is most often used with text, in that case, the input is tokenized, then embedded and positional encoding is added. Since we don't have the high dimensionality problems related to text and the input is already in discrete time steps, we skip the tokenization and embedding.

The architecture chosen for the transformer layer consists in a linear convolution of the input to perform basic feature extraction, then a stack of encoders, a stack of decoders and a final linear convolution. The Encoders apply positional encoding, then perform multi-head self attention and a feed forward neural network with two hidden convolutional layers. The Decoders have the same structure, but also perform multi-head attention over the encoder outputs after performing self attention.

We experimented first with encoder decoder models, then with decoder only and encoder only. The decoder encoder and decoder only models seemed to perform very well during training, however produced useless results in prediction due to the nature of the timeseries problem associated with the teacher forcing method, which will be discussed in the Training 4 section. On the other side, the encoder only models performed very well, improving over the previous LSTM and GRU models.

4 Training

The training was performed tracking the accuracy of the model with cross-validation via hold-out. By measuring the validation accuracy and loss, we were able to keep track of potential overfitting over the training data and observe the generalization properties.

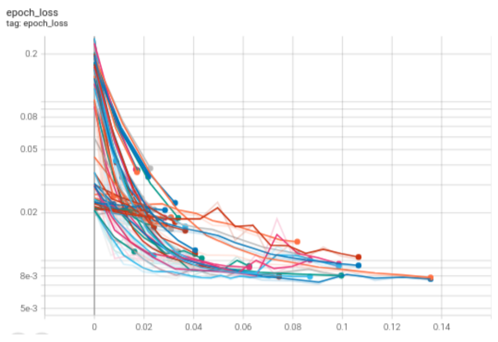

Figure 3: Epoch loss using HyperBand

4.1 Hyperparameter search

After experimenting with hand picked models, we performed hyperparameter search using `keras_tuner`: we defined a hypermodel building function, which can then be used to generate models with different hyperparameters following the HyperBand algorithm to select the best performing models. We used hyperparameter search both to validate hypothesis about order and choice of layers and to find the best performing combinations of numeric hyperparameters.

4.2 Configurations

During training we split the time series using a sliding window approach and tried to predict a the next values for each window. Across our experiments, we used different values for window size and stride and tried to predict a different numbers of samples (telescope).

For the stride we used either 16 or 32, as for the window size, we started with small windows (e.g. 256), but still greater than the fundamental periods observed in the FTT analysis, and then increased the value (e.g. 1024, 2048) this was beneficial for the transformer models, but for the LSTM and GRU models we saw no improvements.

We used different values for the telescope using autoregression to generate the required 864 samples. We tried a pure autoregressive approach using a telescope of 1, but this resulted in poor performance, this is likely due to the high frequency components of the signal being noisy, making it hard to predict the low frequency patterns if the loss is only computed on one sample. On the other side we obtained good results with a telescope of 32 for RNN models and 128 for the transformers.

4.3 Teacher forcing

The transformer models with decoders were initially trained using teacher forcing, feeding as input to the decoder the target signal shifted by one and using a lookahead mask. This proved to be problematic since it resulted in networks with excellent training loss, but terrible prediction loss. Our most likely theory is that due to the nature of the time series forecast problem, by having access to the shifted target, the network can obtain good training results by cheating and varying only slightly the decoder input, which will not work when the decoder input is inaccurate.

5 Results

This section shows the results of the most successful models used for the task, and the RMSE values are reported in the table below.

Model	RMSE
Transformer (encoder only)	3.6866
LSTM model	3.8241
GRU model	3.9163

Table 1: Results on remote test data

As we can see in the RMSE data 1, the best results were obtained with Transformer model encoder only.

In the experiments performed using LSTM, we obtained better results with simpler models. Looking at the figure 4, we can see that the model predicts almost all the features quite well, even if the ones that are more complex to predict are sponginess and wonder level.

As for the transformer models, using fewer encoder layers resulted in faster training time and generally led to best results. Multi head attention performed better than Luong style attention and applying positional encoding before each encoder layer resulted in faster training times.

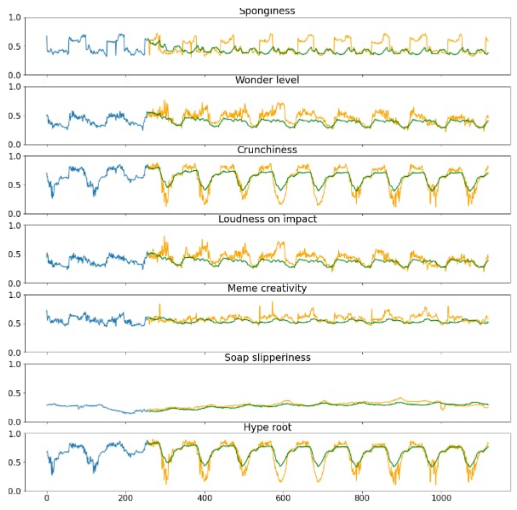

Figure 4: Example prediction with LSTM model

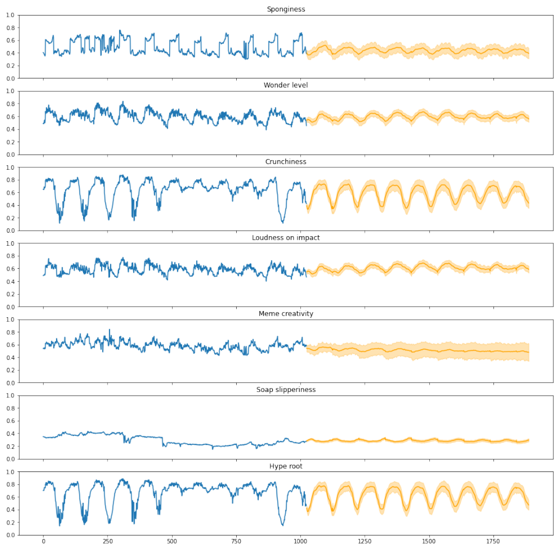

Figure 5: Example mae over Transformer (encoder only) prediction

6 Conclusions

From the experiments, we found that simpler models led to better results than more complex networks for this kind of data. For LSTM and GRU, we didn't find any benefit in bidirectional models. As for the transformer inspired models, fewer encoder stacks were more successful due to the quicker training time. Further improvements could be achieved with scheduled sampling for the transformer decoders models, which may overcome the teacher forcing problems.

Tools

- Tensorflow
- Keras