Second Challenge

Report

Artificial Neural Networks and Deep Learning

29/11/2021
A.Y. 2021/2022

Submitted by

Person Code	Names of Students
10582405	Castellano Sebastian
10578330	Ciapparelli Federica
10608654	Colapenna Francesca

POLITECNICO
MILANO 1863

1 Approach

We started our model development by referring to the practical reference notebook of the course and we created a model composed by a Bidirectional LSTM + 1D convolution + max pooling, followed by another Bidirectional LSTM + Convolution + GAP and finally dropout + dense layer and a final Convolution. We set the telescope to 864 and the backward window to 2000.
This model did not produce a very good result (RMSE=8.5), so we tried with an autoregressive model (also based on the course's practical notes), but it produced a worse result (RMSE=9.2). To keep our focus on a singular model type, we decided to not develop this autoregressive approach.
Going back the initial model, we tried substituting the LSTM with a GRU, setting the window to 1000 and removing some constant values from the dataset (see Data Cleaning, 2.a), obtaining RMSE=8.0.
At this point we decided to simplify the structure of the model, using only GRU + dense layer and we obtained a remarkable improvement, reducing the RMSE to 5.4.
The next step we added was the removal of some highly – correlated features (see Correlation, 2.a) and the RMSE further decreased to 4.8.
Lastly, we added two other dense layers to improve the performance of the model and substituted the GRU with a LSTM, getting a final RMSE = 4.1.
We also tried the same structure with a GRU instead of LSTM, but the error was higher, so we kept as final model the one with LSTM + 3 Dense layers.
Finally, our best model reached a value of RMSE equal to 3.7, obtained by adding some sinusoidal functions to help it predict periodical patterns in the time series (see Sinusoidal Timeseries, 2.a)

2 Final Model

2.a Pre-processing

- **Data Cleaning**

  - **Dataset Characteristics**
    We started by looking through the specifics of our dataset. Through the command dataset.describe(), we checked the main characteristics of each timeseries, and made sure that they didn't contain any NaN values.

  - **Correlation**
    Next, we checked the correlation matrix of our dataset. We discovered that Crunchiness and Hype Root showed a correlation of 99%, while Wonder Level and Loudness of Impact of 93%. Therefore we removed Crunchiness and Wonder Level from our training set, to avoid feeding redundant information to the model.

  - **Plot the timeseries**
    We drew a plot of the dataframe, to have an idea of the behavior of each series and to check for any outliers: something that stood out was that for certain intervals the points of all the timeseries became constant. This behavior seemed suspicious since most of the time series showed some sort of periodicity or at least had a much greater variance. We concluded that these may be missing values, and we removed them form the training set after.

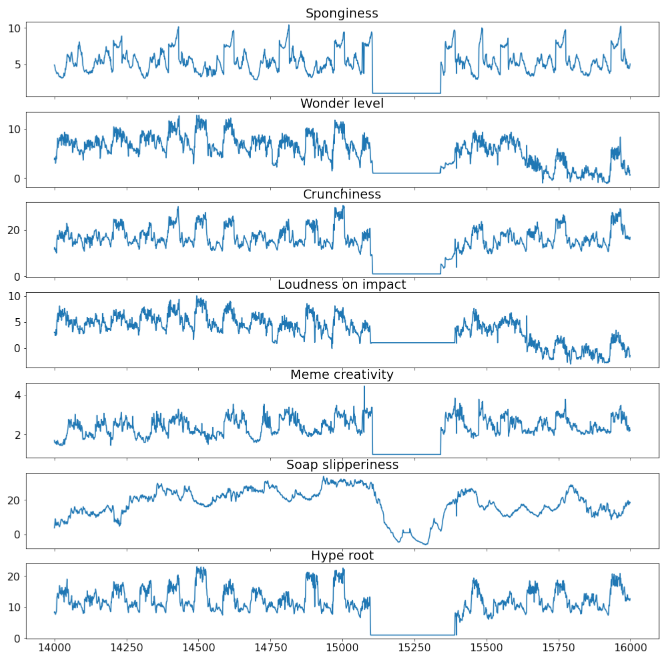


Figure 1: Plot of a part of the dataset. You can see the constant parts, which are probably missing values.

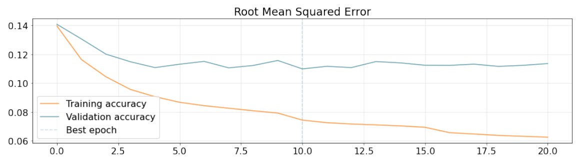
- **Data Preparation and Augmentation**
  - **Sinusoidal timeseries**
    When plotting the timeseries we noticed that almost all of them showed some sort of periodicity. To help the model to learn this behavior we tried adding some sinusoidal functions to represent the main periodical patterns expressed by the series. We found the right frequencies by utilizing the Fourier transform of the series.
  - **Normalization and missing values removal**
    We normalized the timeseries through a min-max normalization and we separated the data in sequences with a window of 1000. Then we removed all training sequence that contained any of the constant values identified before.

  The end result is a dataset with 14 timeseries split in 3514 different pairs of train/val sequences.

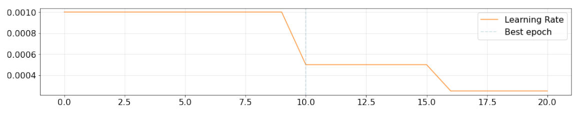
2.b Structure of the Neural Network

- **Structure**
  1) LSTM with 512 units
  2) Two dense layers with activation relu, 8000 e 7000 neurons respectively
  3) Dense output layer with 864 * 7 neurons.

- **Compiling/Fit of the model**:
  In order to prevent overfitting and get the most out the training, during the model.fit() we used early stopping and learning rate decay callbacks.
  As loss we used the mean squared error, as metric the root mean squared error, with Adam optimizer.



(a)



(b)

Figure 2: Evolution of the RMSE of Train/Validation set

Figure 3: Evolution of the learning rate

3 Conclusions

3.a Model performance

Overall we can affirm that generally the model performed quite well on most of the timeseries, being capable of understanding the seasonality and predicting the right peaks and valleys.
The predictions become less accurate in the parts with high volatility or jumps/discontinuities (as we can see in the Sponginess and Crunchiness graphs).

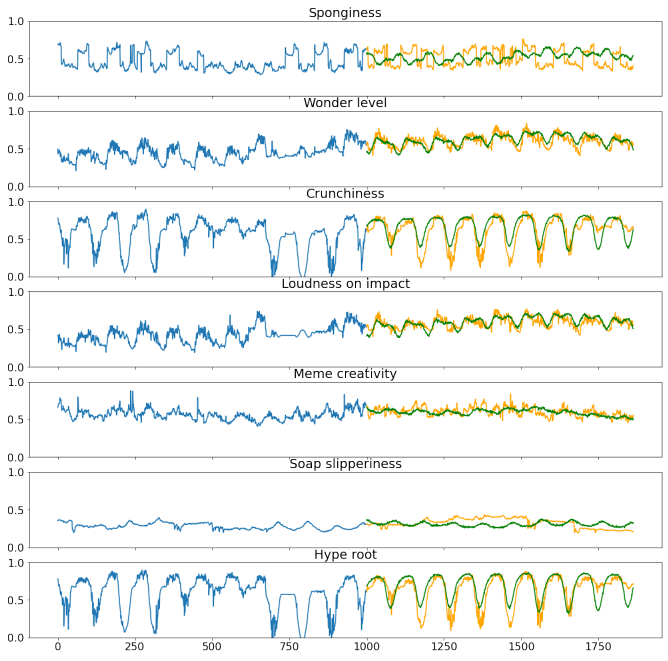


Figure 4: Model performance on the test set

3.b Possible Improvements

One thing we tried was to fill the missing values in the dataset with the predictions of the model to obtain a bigger training set. Unfortunately this didn't lead to the improvements we hoped for.
Secondly, to help with Sponginess (hard to predict due to it's "jumping" behavior) we tried to add a "one hot encoding" of the values above and below a certain threshold. We hoped this would help the model to identify where to "jump" but once again we didn't get any improvements unfortunately.
Nonetheless we do believe that these ideas, with some tweaks, could be useful to further improve the predictions of the model.