Artificial Neural Networks and Deep Learning 2021

Homework 2

Prof. Matteo Matteucci

Cristofaro Giulio (Personal Code: 10555172)
Shalby Hazem (Personal Code: 10596243)
Tiu Robert Andrei (Personal Code: 10652209)

POLITECNICO
MILANO 1863

1 Problem statement

The goal of this homework is the prediction of future samples in a multivariate time series. The predictions were only made for `3` fourths of the final test set, since `phase 2` never happened due to technical problems.

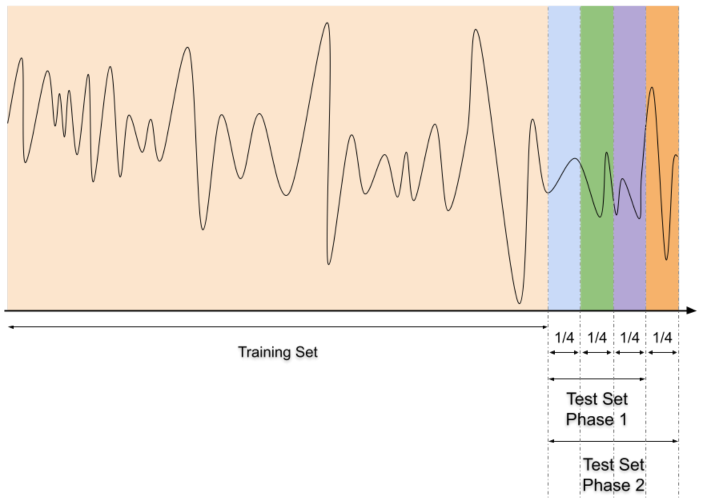


Figure 1: Organization of the dataset

The features associated with the input/output tensors are the following: `Sponginess, Wonder Level, Crunchiness, Loudness on Impact, Soap Slipperiness and Hype Root.` Clearly, those features hide the real world event behind the data, so supporting the model research through exploratory data analysis was quite limited. Instead of using scientific research to find models that best fit a certain real world task resolution, we looked for more general neural network architectures and experimented a lot through trial and error.

Here is a deeper look into the dataset:

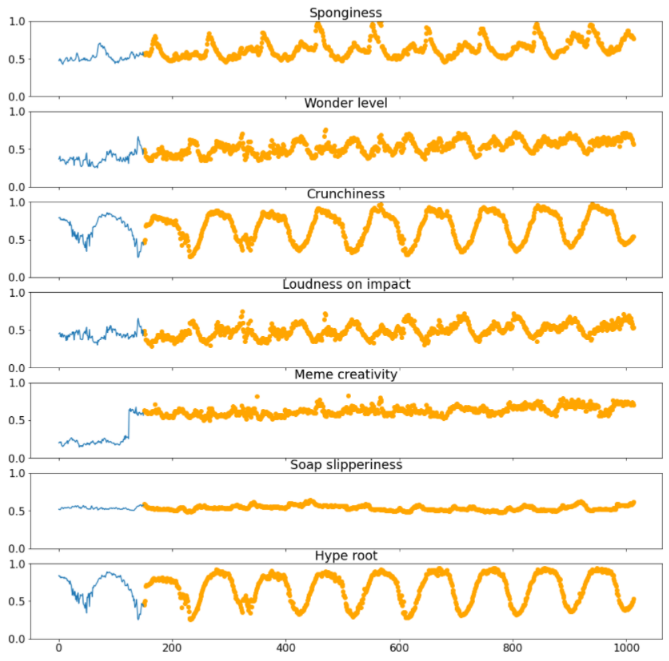


Figure 2: The orange points indicate the data which the model will have to predict

2 The process

2.1 Models

The first step of the process was to build the sequences that would be given as an input to the model. This is performed by selecting the first `864` samples from the data set as the first sequence, which is then shifted of `5` data points to create the second sequence, and so on. Each sequence is then divided into `150` samples (which we will call ”window”) to create the input for the model, with the remaining ones becoming the output.
As usual, in order to avoid `overfitting`, the resulting sequences will also be split into three parts: **`training, validation, test`**.

The neural network that led to the best results was the following:

- `Bidirectional LSTM` layer with `32` units
- `Convolutional` layer with a kernel of size `3x3` and `128` filters
- `One-dimensional max pooling` layer
- `Bidirectional LSTM` layer with `64` units
- `Convolutional` layer with a kernel of size `3x3` and `256` filters
- `One-dimensional max pooling` layer
- `Dense` layer
- `Reshape layer` (in order to obtain as an output a tensor of the correct size for the model testing)
- A final `Convolutional` Layer for output

This model was among the first ones that we tried, and it was quite sensitive to hyper-parameter tuning. Our first attempts at tuning the model did not lead to great results, with the first models getting a root mean squared error (`RMSE`) in the range of `8`. After further trial and error, we selected the parameters described above and reached a `RMSE` of around `3.96`.

One of the trials that we perform was to simplify the `LSTM` layers of our model, changing the `Bidirectional LSTM` of our best model to the simpler `LSTM` version, where we have only one `RNN` traversing the sequence left-to-right (`Forward`).

However, the outcome was a downgrade in performance compared to our previous more complicated models, both locally and also in the CodaLab platform.

We also experimented by changing the activation functions in all of our models, introducing a `Leaky ReLu` in order to fix the ”dying neuron” problem, but once again this didn’t prove beneficial at all, since the results were again worse than our best model. We eventually restored the classic `ReLu` activation function.

Since we seemed to have found a stall with improvements to the architecture, we focused on the training part.
We complemented the training with some old tricks: as we learned from the previous project, a dynamic handling of the learning rate during training can lead to better results, so we used once again the Keras callback `ReduceLROnPlateau` (factor = `0.5`) alongside `EarlyStopping` with patience set respectively to `3` and `6`.

Since the network achieved satisfactory results without unneeded complexity, we deemed this model as the final one.

3 Plots

Find below some plots that summarize the final outcome of the project.

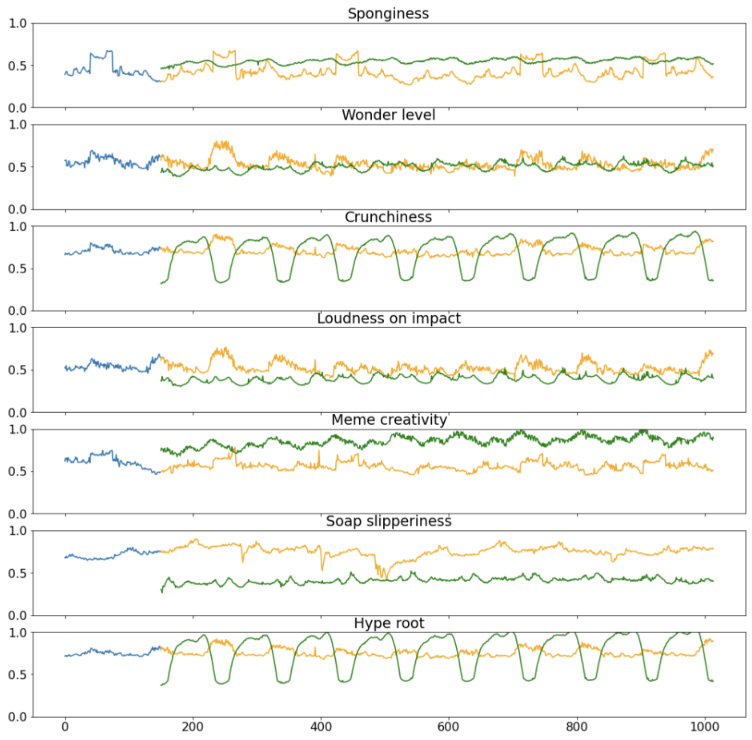
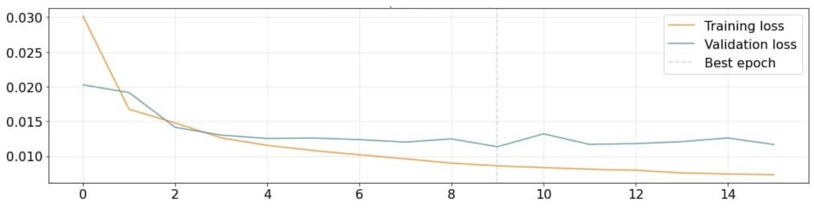
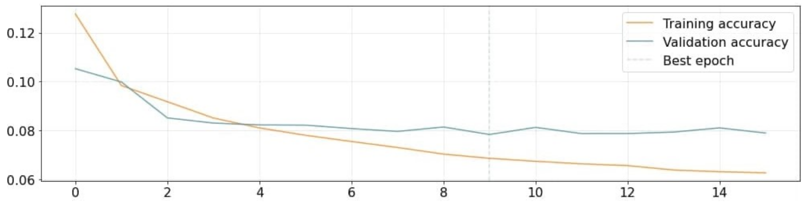


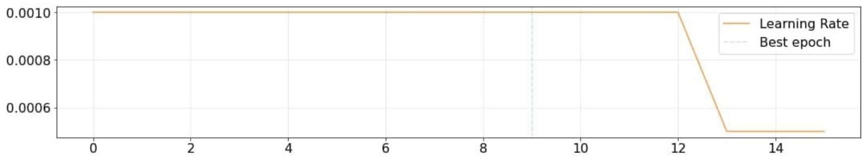
Figure 3: Final predictions computed by the model



(a) Mean Squared Error (Loss)



(b) Mean Absolute Error



(c) Learning Rate

Figure 4: Error metrics and learning rate for the training and validation sets