Politecnico di Milano

ARTIFICIAL NEURAL NETWORKS AND DEEP LEARNING PROJECT

MULTIVARIATE TIME SERIES FORECASTING

Authors:
Alessandro Polidori
Agnese Straccia
Andrea Alberto Bertinotti

A.A. 2021/2022

Index

1. Introduction

The challenge consisted in a multivariate time series forecasting problem, so we had to leverage multiple historical time series in order to predict the future trends of the different series.

We were provided with a training set composed of seven time series, each one regarding a different feature of a cartoon (or a meme). In particular these features were: sponginess, wonder level, crunchiness, loudness on impact, meme creativity, soap slipperiness and hype root.

The metric we had to use to measure the goodness of our prediction was the Root Mean Square Error (RMSE).

In all of our models we used the Mean Square Error (MSE) as loss function and Adam as optimizer.

As regulation methods we used Early Stopping and Dropout layers. Moreover, to ease the learning process we used the "reduceLearningOnPlateau" callback in order to progressively reduce the learning rate when no improvement was shown.

The challenge has been addressed implementing in parallel three main architecture paradigms based on 1DConvolution, LSTM/GRU and a mixture of the two.

2. 1D Convolution

One of the approaches was based on the use of 1DConvolution layers only. We started from a very simple model taken from TA classes based on direct forecasting and we obtained constant predictions, so the model was basically predicting the average value of future points.

At this point we started to experiment with several configurations of layers, kernel sizes, stride, dropout rate, window size and batch size. We found that using a reduced window yielded the best results (so from that point on, we used windows of 400/500 samples).

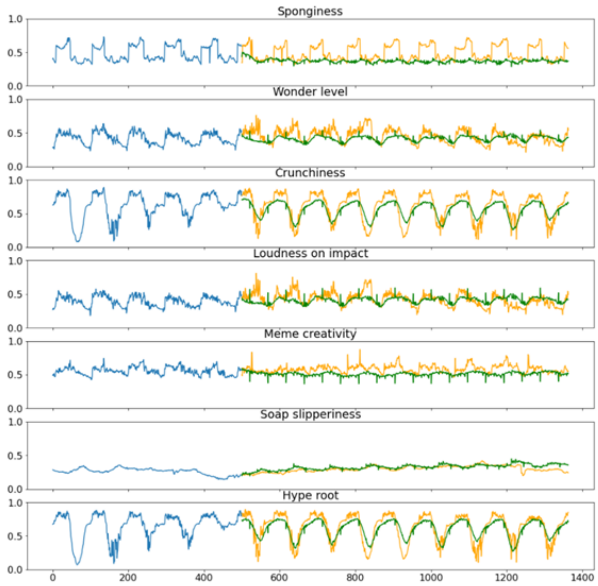
Once we obtained satisfying predictions on local tests, we tried to upload on codalab the direct forecasting model, obtaining an RMSE of 4.95.

We decided to switch to an autoregressive model, predicting one point in the future at a time, and we initially obtained worse results. We continued experimenting on autoregressive models until we got to a turning point. Predicting more than only 1

point in the future at a time led to better and better results. Eventually we found that the sweet spot was at 48 future points.

The best result on codalab, RMSE = 4.01, was obtained using a window of 500 points, stride 4 and predicting 48 points in the future at a time.

Best model local results using 1DConvolution only:




```
# Build the neural network layer by layer
input_layer = tfkl.Input(shape=input_shape, name='Input')

cnn = tfkl.Conv1D(32,5,padding='same',activation='relu')(input_layer)
cnn = tfkl.Conv1D(32,5,padding='same',activation='relu')(cnn)
cnn = tfkl.MaxPooling1D()(cnn)
cnn = tfkl.Dropout(.1, seed=seed)(cnn)

cnn = tfkl.Conv1D(64,5,padding='same',activation='relu')(cnn)
cnn = tfkl.Conv1D(64,5,padding='same',activation='relu')(cnn)
cnn = tfkl.MaxPooling1D()(cnn)
cnn = tfkl.Dropout(.1, seed=seed)(cnn)

cnn = tfkl.Conv1D(64,5,padding='same',activation='relu')(cnn)
cnn = tfkl.Conv1D(64,5,padding='same',activation='relu')(cnn)
cnn = tfkl.MaxPooling1D()(cnn)
cnn = tfkl.Dropout(.1, seed=seed)(cnn)

cnn = tfkl.Conv1D(64,5,padding='same',activation='relu')(cnn)
cnn = tfkl.Conv1D(64,5,padding='same',activation='relu')(cnn)
cnn = tfkl.MaxPooling1D()(cnn)
cnn = tfkl.Dropout(.1, seed=seed)(cnn)

cnn = tfkl.Conv1D(64,5,padding='same',activation='relu')(cnn)
cnn = tfkl.Conv1D(64,5,padding='same',activation='relu')(cnn)
cnn = tfkl.MaxPooling1D()(cnn)
cnn = tfkl.Dropout(.1, seed=seed)(cnn)

gap = tfkl.GlobalAveragePooling1D()(cnn)
```

3. LSTM Model

At the very first moment, we built a "direct forecasting" model with only 2 LSTMs of 128 units each, with a window of 200 points. The result obtained on CodaLab from that initial model was 4.92. From that time on, we tried to improve it by changing a little bit both the architecture and the hyperparameters. As in the 1DConvolution models, we focused our attention on stride, dropout rate, batch size and window size, in order to understand how the performance and the relative curves in output changed. After several experiments we started to reach satisfying results. One remarkable leap forward came after changing the two LSTMs into Bidirectional LSTMs. After that we kept the architecture as it was and we tried different combinations of hyperparameters. In fact, by changing the window and stride we noticed that the performance changed a

lot even if the change was very small. This has led to a further tuning and a resulting increase of the performance up to 4.12 (window of 400 points and a stride of 10).
The best results, however, were obtained, as with 1DConvolution, switching to an autoregressive model. We will discuss it in the last paragraph.

4. Other Attempts

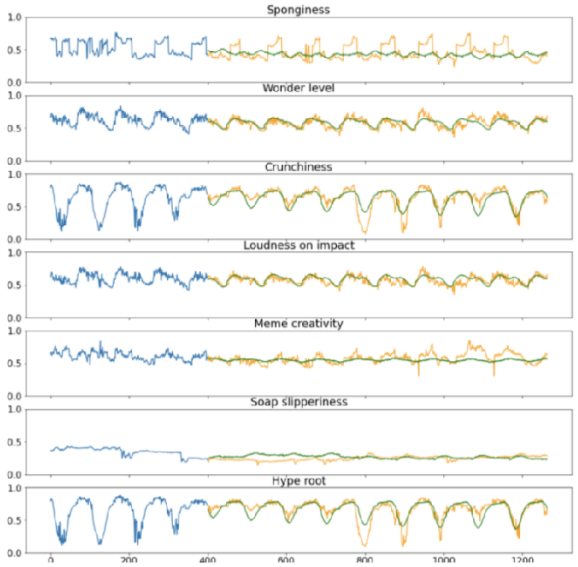
We tried to further improve the results by changing the architecture of the model. An attempt was to implement GRU layers, using them instead of the BidirectionalLSTM layers, and the result obtained was 4.35.

Moreover, we also endeavored to combine different types of layers, such as LSTMs and Conv1D, in a sort of "hybrid" model. We worked on many combinations of these layers trying to find the best combination, and the best sequence we have found was obtained composing a BidirectionalLSTM, a Conv1D and a LSTM layer. To prevent overfitting we placed a Dropout layer after LSTM and BidirectionalLSTM layers. We experimented with many changes in window, stride and test size to test this model, and the best result we were able to achieve was a RMSE equal to 4.64, exploiting the direct forwarding technique. Other combinations of layers, such as replacing the LSTM with another BidirectionalLTSM layer, the addition of another Conv1D, and even changing the order of the layers, led to worse results.

By the way, given that the best result of this "hybrid" model was not satisfying, we decided to focus on a more straightforward configuration of the model.

5. Best Model

Once we obtained satisfying results with “direct forecasting” LSTM-based models, we decided to apply autoregression to the model that performed better up to that point. After experimenting with different autoregression telescopes,we found that, as in 1DConvolution models, 48 points in the future at a time led to the best result. So the lowest RMSE on CodaLab (3.90) was achieved with a LSTM-based autoregressive model. Respectively, the following pictures show the details about the architecture composed of two Bidirectionals of 64 units each, followed by a DropOut layer and the prediction curves.




```
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
Input (InputLayer)           [(None, 400, 7)]          0
_________________________________________________________________
bidirectional (Bidirectional (None, 400, 128)          36864
_________________________________________________________________
bidirectional_1 (Bidirection (None, 128)               98816
_________________________________________________________________
dropout (Dropout)            (None, 128)               0
_________________________________________________________________
dense (Dense)                (None, 336)               43344
_________________________________________________________________
reshape (Reshape)            (None, 48, 7)             0
_________________________________________________________________
conv1d (Conv1D)              (None, 48, 7)             56
=================================================================
Total params: 179,080
Trainable params: 179,080
Non-trainable params: 0
_________________________________________________________________
```