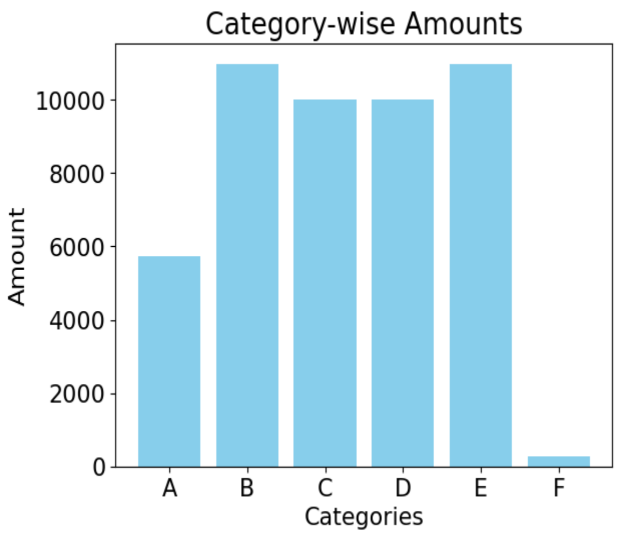
AN2DL - Homework 2
Time series Forecasting - TensorFlex

Sironi Alessandro, Stefanizzi Giovanni, Stefanizzi Tomaso, Villa Ismaele

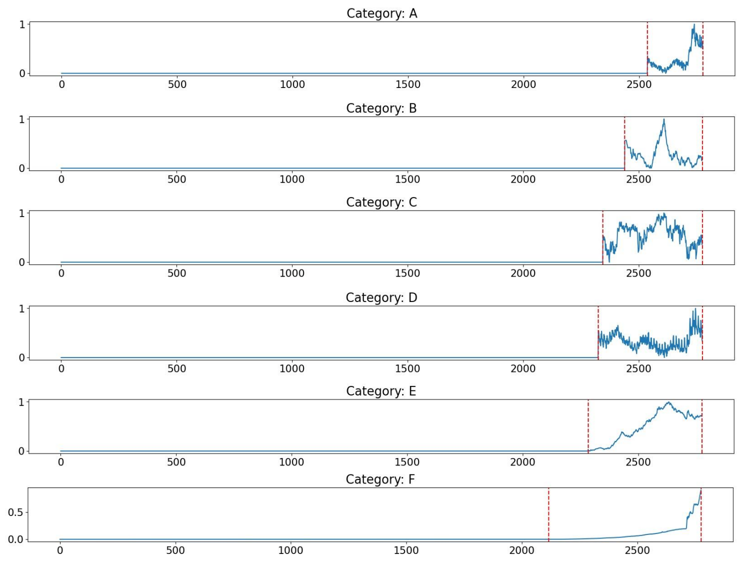
Dataset specification



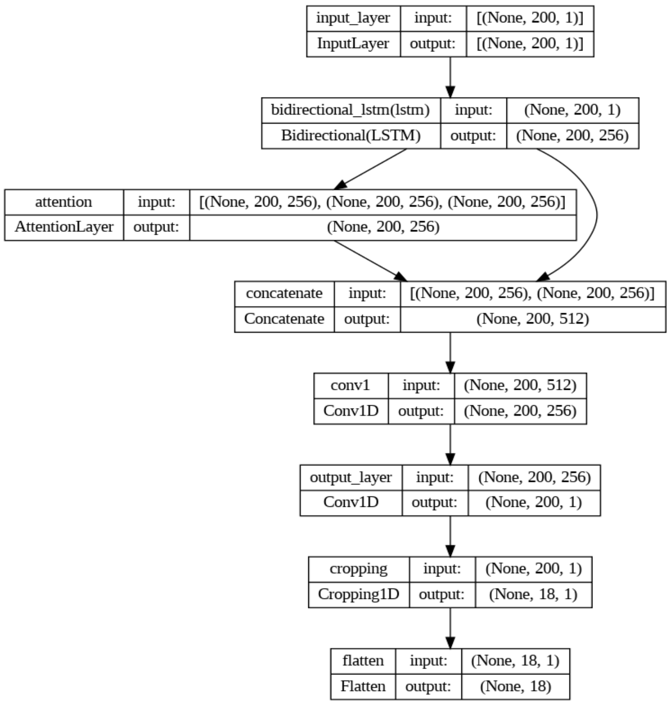
The second challenge's dataset consisted of three NumPy arrays: the training data (48000, 2776), valid periods (48000, 2), and categories (48000,) associated with the data.
These time series exhibit variable lengths, and padding was applied for handling the data. Consequently, the initial step with the provided dataset involved utilizing the 'valid_periods' array to filter out non-valid data segments.
The time series were divided into 6 different categories, meaning that they belong to different data sources, but we decided to use all of them as the test set on Codalab was hidden and probably composed of the whole types of categories.

Inspecting data

After visualizing and examining the data, we conducted an outlier search, yet no anomalies were evident. To streamline the training process, we opted to exclude a few time series that significantly differed in length compared to the majority.
As the data was normalized in the range (0,1) we tried to apply a RobustScaler to re-normalized the dataset. However the overall result was slightly worse than the one we achieved with the original data.



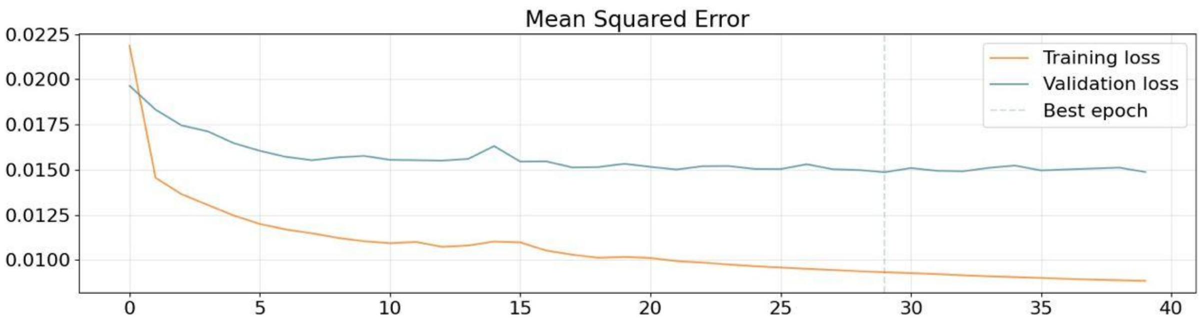
Building the model


input_layer | input: | [(None, 200, 1)]
InputLayer | output: | [(None, 200, 1)]
bidirectional_lstm(lstm) | input: | (None, 200, 1)
Bidirectional(LSTM) | output: | (None, 200, 256)
attention | input: | [(None, 200, 256), (None, 200, 256), (None, 200, 256)]
AttentionLayer | output: | (None, 200, 256)
concatenate | input: | [(None, 200, 256), (None, 200, 256)]
Concatenate | output: | (None, 200, 512)
conv1 | input: | (None, 200, 512)
Conv1D | output: | (None, 200, 256)
output_layer | input: | (None, 200, 256)
Conv1D | output: | (None, 200, 1)
cropping | input: | (None, 200, 1)
Cropping1D | output: | (None, 18, 1)
flatten | input: | (None, 18, 1)
Flatten | output: | (None, 18)

We employ an RNN structure of multiple layers, where the initial key layer consists of a bidirectional LSTM (Long Short Term Memory). This LSTM is adept at effectively assimilating and handling sequential data from past and future time steps, thereby enabling the model to gain a deeper understanding of the sequence's context.
We then wrote an Attention layer to enhance the model's ability to capture relevant information from the input sequences and selectively focus on relevant data. The aforementioned layer is not directly added to the sequence but concatenated with the input layer. Then two Conv1D layers are added as we are dealing with sequential data evolving in one dimension (time). And eventually we have a Cropping layer that is needed in order to forecast the right amount of points required by the challenge and a Flatten layer, for the same purpose.

Parameters tuning

After several training sessions, we managed to find the best parameters combination for our model. We tried 64,128 and 256 as batch size, but 128 was the one that worked better and an epoch value of 100 was sufficiently large to avoid early training stop.
Regarding the build sequence process we managed to create homogeneous sequences of 200 points each (as this was the required size for our prediction data) with a stride of 10, meaning that we made our window move forward 10 steps at a time.
The telescope was strictly set at 9 for the first phase of the challenge and 18 for the second one.



Custom Build Sequences

In order to build the windows used for the training and splitting data, we've employed a custom function to calculate the padding for each time series. A loop calculates the number of windows that can fit into the series with the given stride, and determines the required padding to ensure that the sequences align properly.
After creating the windows, we've considered eliminating from the dataframe the sequences which had a padding size much bigger than the majority. However, we've repeatedly seen worse results, so we opted not to do so.

Model Performance - Local test set (9 points forecasting)

This is a resume of the results we obtained during the first phase of the competition. After some attempts to shape the model as required by the competition we managed to get quite good results. However for some reason we didn't get such good MSEs in the second phase.

Model Name	MSE	MAE
conv_lstm_attention	0.00499344	0.05188869
conv_lstm_attention + conv_gru_attention	0.00557919	0.03157493
conv_lstm_only_valid_data	0.00576166	0.05432449
conv_lstm_full_data	0.00591252	0.05787633
conv_lstm_robust_scaler	0.00660196	0.05939539

Results

In conclusion we managed to get a Mean Squared Error of 0.01248039 and Mean Absolute Error of 0.07965311. MSE is the average of the squared differences between the predicted and actual values and gives more weight to larger errors than smaller ones, which means it penalizes outliers more harshly. MAE is the average of the absolute differences between the predicted and actual values and is more robust to outliers and noise. These values are calculated over a hidden test set of 60 time series over 18 points each.

Contributions

DATASET INSPECTION: *Sironi Alessandro, Stefanizzi Giovanni, Stefanizzi Tomaso, Villa Ismaele*

MODEL BUILDING: *Sironi Alessandro, Stefanizzi Giovanni, Stefanizzi Tomaso, Villa Ismaele*

PARAMETERS TUNING AND TRAINING: *Sironi Alessandro, Stefanizzi Giovanni, Stefanizzi Tomaso, Villa Ismaele*

DOCUMENTATION AND OVERALL CODE: *Sironi Alessandro, Stefanizzi Giovanni, Stefanizzi Tomaso, Villa Ismaele*