Deep learning homework 2

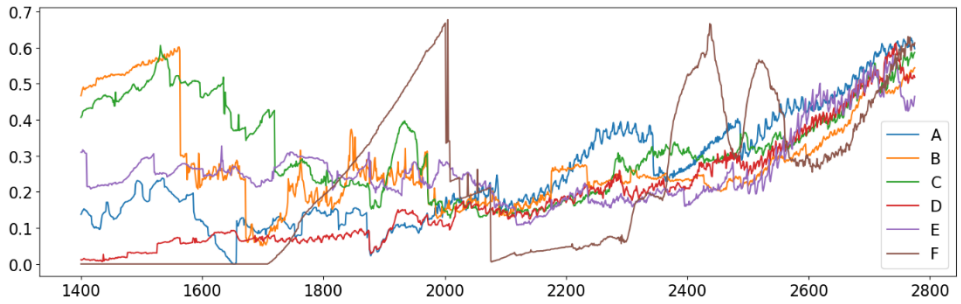
1. Introduction

For this homework, we were asked to forcast time series data based on monovariante time series database, composed of a single feature, belonging to six different domains.

To solve the time series forecasting problem, we were provided with a dataset containing 48000 time series of length 2776, along with their associated labels about categories.

2. Data preparation

Tackling with incomplete time series



Average of all the timeseries for each categories

The dataset was a set of 48 000 time series from 6 differents categories. The maximum length of the timeseries was 2776 but most of them were incomplete and padded with zeros. We decide to isolate the individual valid sequences of each time series and extract sequences from them using a sliding window approach allowing to generate a new dataset for each category containing 20 000 sequences of a given window size and corresponding prediction squences of a given size. The forecasting size is called “telescope” in our code.

Window size

To choose the window size, we used the autocorrelation function (ACF) on our time series: we were interested in the lags corresponding to the extrema of the ACF. Since these differ from one timeseries to another, we set the value at 100, which we felt was reasonable to cover their tendencies.

Robust scaling

The original dataset contained outliers sequences in each categories. We decide to try the robust scaler method to transform the data. This method reduced the importance of outliers on the dataset caracteristics [1]. Unfortunatly the results obtained on our final model where slightly worst with the robust scalar method. Therfore we devided to not use it anymore.

3. Building the model

It has been decided to train one model for each category to exploit their possible specificities and maximize performance. But our limited time and resources did not allow us to build different models for each category. The only thing that is distinguishing them is therefore the training data. Thus the following results were applied to all the six models.

Choice : Conv1D + Bidirectional LSTM + Dense

We investigated various forecasting models arhitectures with components such as LSTM, convolutional layers, and self-attentio. We experimented with both basic convolutional LSTM models and more intricate designs including transformers and attention layers.

The most successful model, according to our testing and analysis—which is presented in our "FinalNotebook" and condensed in the results Table 1. was *Model 12: Conv1D + Bidirectional LSTM + Dense* with the lowest MSE and MAE values 0.00698 and 0.054648, respectively based of our test set.

To our surprise, this model outperformed the others. Its simple architecture worked exceptionally well for our time series problem, balancing efficiency and complexity without the need for transformers or attention methods.

Layers structure

In our notebook named "FinalNotebook" we tried several layers structures including model with attentions and transformer. The model that gave the bests results was a simple convolutional LSTM with two one dimensional convolutions layers coupled with two max pooling layers and a final dense layer to create the output.

The results of testing different combinations of layers are shown in the table:

Index	Model	MSE	MAE
Model1	Only LSTM – 32 units	0.009904	0.065101
Model2	Only LSTM – 64 units	0.008178	0.058946
Model3	LSTM + Conv1D	0.00786	0.057928
Model4	Conv1D + LSTM + Conv1D + Dropout + Dense	0.007956	0.058067
Model5	LSTM + Conv1D + MaxPooling + Dense	0.008952	0.060758
Model6	Bidirectional LSTM + Conv1D + MaxPooling + Dropout + Dense	0.02095	0.115252
Model7	Bidirectional LSTM + Conv1D + MaxPooling + Attention + Dense	0.065147	0.215921
Model8	Transformer + Conv1D + Dense	0.292562	0.476914
Model9	Bidirectional LSTM + Attention + Conv1D	0.009114	0.063025
Model10	Bidirectional LSTM + Conv1D	0.011673	0.072231
Model11	Bidirectional LSTM + Attention + Transformer	0.008891	0.061596
Model12	Conv1D + Bidirectional LSTM + Dense	0.00698	0.054648

Table 1. MSE Comparison for Different Models

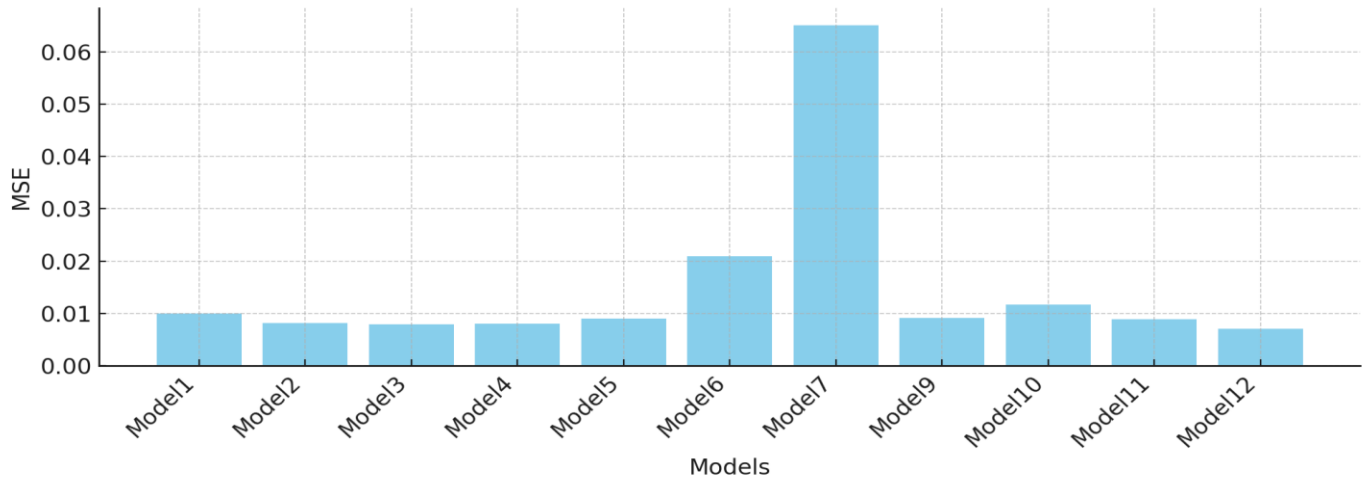


Figure 1 : MSE Comparison for Different Models (without model 8.)

The MSE comparison graph showcases the variance in model performance, with lower values indicating better predictive accuracy.

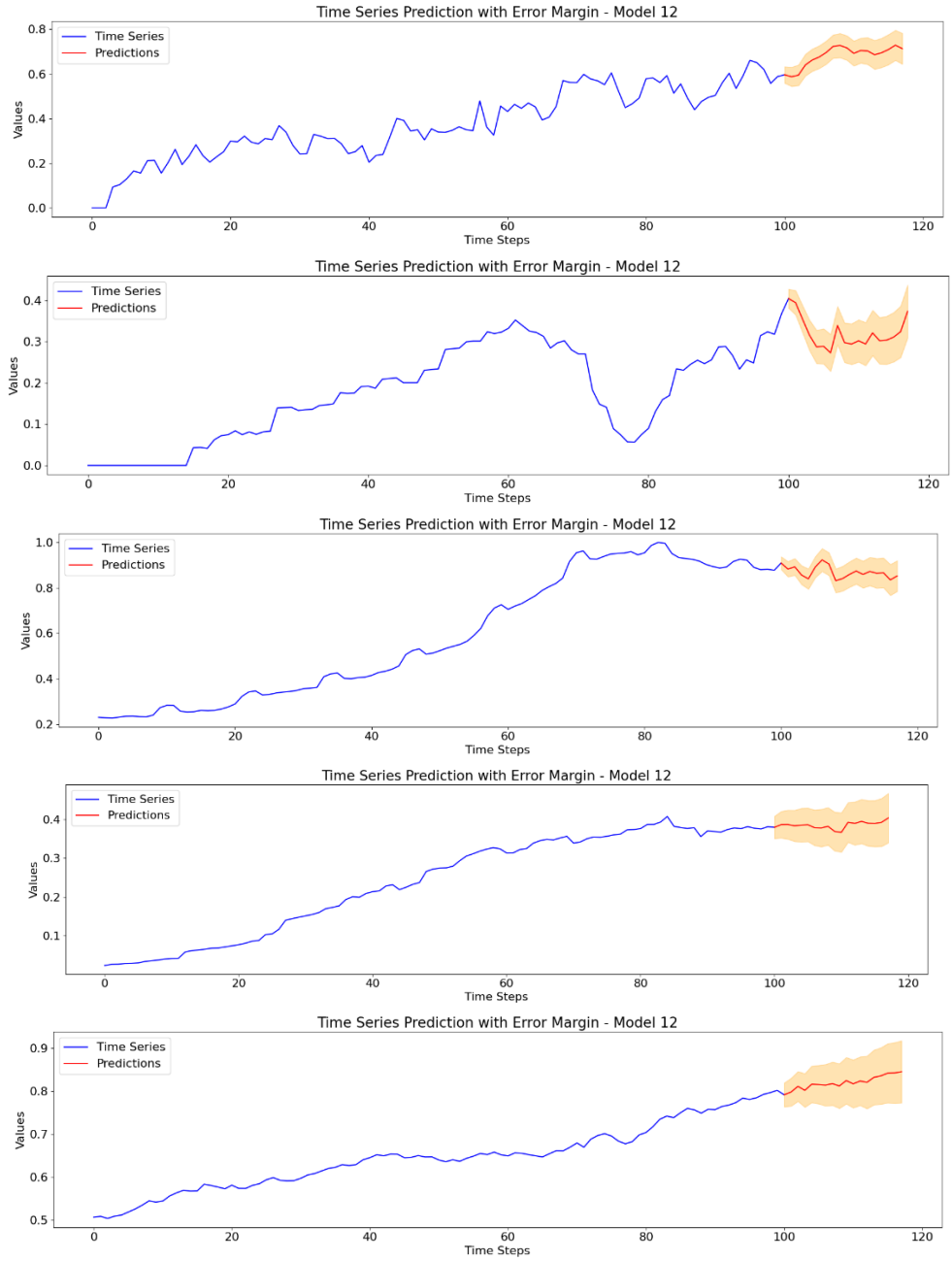
4. Final implemented solution

The final implementation was a simple convolutional LSTM model using robust scalar for training with a window of size 100.

5. Results

The results of the best model *(Model 12)* we trained are shown in the following graph.

Prediction graph with error margin



6. Conclusion

Our strategy for data preparation was selecting time series with the highest variance from individual valid sequences, a process that ensured the integrity and quality of our training data. After that we tested variour types of models using components such as LSTM, Conv1D and Attention and we found that Model 12: Conv1D + Bidirectional LSTM + Dense as the most effective architecture. The model's simplicity, combined with its ability to efficiently handle time series data, resulted in good performance, in our case demonstrating that complexity does not necessarily equate to better forecasting.

7. Contributions

- Filip Fabris - models testing, model comparison, sequences building, report
- Maxime Pellichero – data preparation, models testing, robust scalar, sequence building, report
- Jana Penic – preparing dataset, report
- Mai Lan Pho – auto-correlation, robust scalar, sequences building, report

8. References

[1] https://proclusacademy.com/blog/robust-scaler-outliers/
[2] https://keras.io/api/callbacks/early_stopping/
[3] https://keras.io/api/callbacks/reduce_lr_on_plateau/