ANDL Homework 2 – Report

Filippo Desantis – 10677781 (HPC)

Livia Giacomin – 10709077 (HPC)

Kamil Hanna – 10932895 (HPC)

22nd December 2023

Abstract

The problem we were presented with was to train a network to predict the next steps of some given time series. We explored different approaches, architectures, and data processing.

1. Data loading and analysis

First of all we looked at the dataset. It is made up of 48000 time series of 2776 columns each. We also were given a file that associated a category to each time series out of A, B, C, D, E, F and a file that, for every time series, provided a starting index and an ending index. This was needed because the series were naturally of different lengths and padding was used to allow numpy to work with homogenous vectors. We extracted the indicated section from each time series and inspected them to see what kind of data we were dealing with. As seen in Figure 1 the time series were all quite different, so we understood that we needed our network to generalize quite a lot.

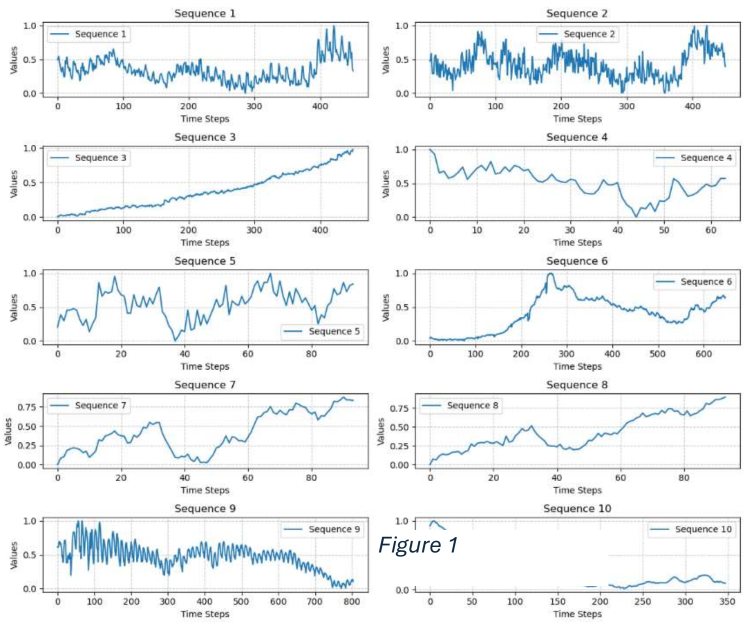

Figure 1

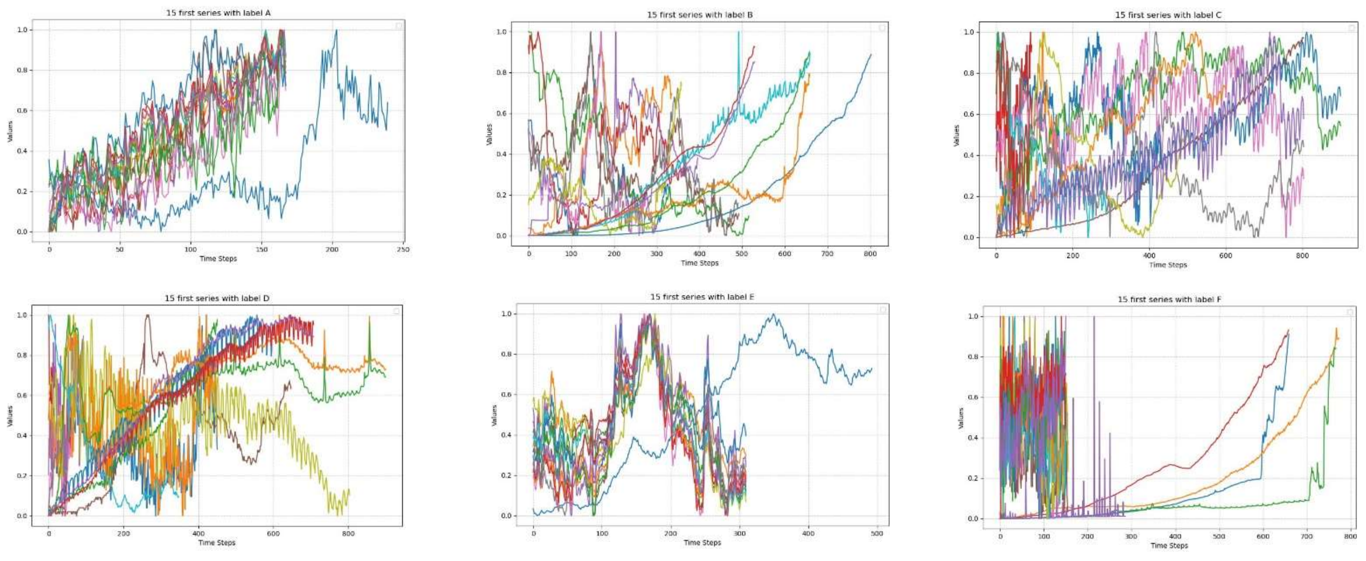

Figure 2

We thought of assessing the categories and using a classifier along with 6 models, each trained only on one class of series. However, as seen in Figure 2 in the previous page, the classes weren't very distinct and showed some substantial differences inside them, as if there were subclasses: we decided, given the time we had and the increased difficulty that combining 7 models would imply, to not classify and ignore the category labels. This was also backed by the fact that in the competition data, it was stated that the different categories didn't really mean much but were just a matter of collection.

The next problem was making the most of our data, specifically the fact that we needed to train our network to process inputs of 200 steps, but our input dataset provided many different lengths. We tried different approaches. First, we just took the sequences that were longer than target length and cut them accordingly. Then we noticed that this way we wasted too much data, so we turned to "splicing" the data in lengths, without overlap, padding the sequences that were shorter than target length. Then we also tried to make sequences with a window and a stride, trying different lengths and strides, but we noticed that this caused the model to overfit very fast and not give good results on the Coda Lab submissions. For example, we tried to train on sequences of 80 instead of 200, accordingly modifying the model.py file to take as input the last part of the test series, but this didn't get good results. So, we finally settled on the splicing as seen in Figure 3, with a sequence before splicing and after. We also settled on training with a target length of 218. We then split into a training set and a test set with proportion 0,2, and then we took the first 200 timesteps of each series as X_test and the last 18 as Y label, to test against to calculate loss.

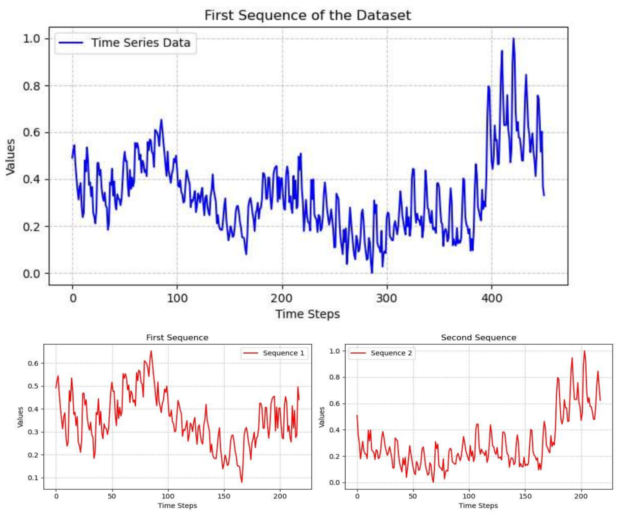

Figure 3

We then moved onto data augmentation techniques to increase our dataset. We tried using random scaling and scattering, as seen in Figure 4, but this didn't lead to relevant improvements in our results, at least not on the leaderboard. We also tried to normalize the data, applying the same normalization in the model.py and denormalizing. This also did not improve the results. The data augmentation that surprisingly worked was concatenating the same data multiple times, eventually changing the number of rounding digits.

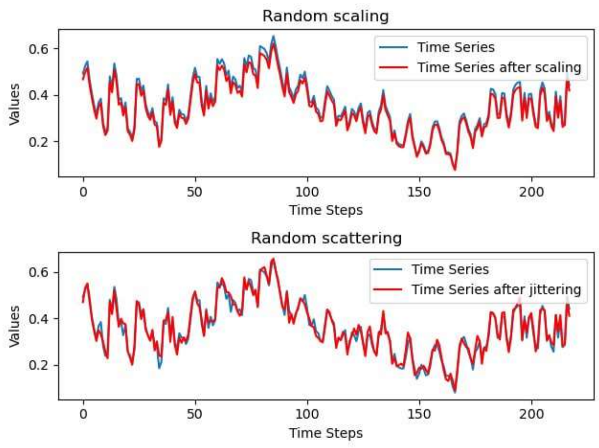

Figure 4

We had some issues with reshaping the vectors in the right way, but having solved that the rest went more smoothly.

2. Network architecture

First, we tried dynamic LSTM, which took in time series of different lengths even without the data processing, but we soon realized it could not be used with our model. So we considered it for optimization purposes and to better understand the problem. Then we tried two RNN architectures: GRU and LSTM. GRU did not give good results, not even adding Conv1 layers. Instead, the LSTM gave better results, but not with Conv1 layers. So, we removed them and ended up with a very simple LSTM architecture with three dense layers. We also tried some models that used autoregression and tried an attention-based model, but the results were never better than our simple model. We had seen that making it more complex made it worse, too slow, and too overfitting. So, the final model we chose to use is the one visible in Figure 5.

3. Hyperparameters

We tried different batch sizes, seeing that some were faster in executing or faster in learning, but soon the training stagnated. We settled on batch size 32. We also tried to use a cyclical learning rate, but this too did not improve the model. We chose to use mse to calculate loss, comparing to validation set. On our best model, we achieved around 0,01 mse locally and 0,009 on the leaderboard.

4. Sample predictions

In figure 6 we show our predictions for some time series, testing locally.

5. Conclusions

This project was a bit frustrating because we tried many things, even some elaborate things, but the simplest net with the simplest augmentations was still the best. We did not expect simple data duplicaton (with or without rounding) to achieve the results it did, but that was still the best we could get. It was at least interesting to write different numpy functions that spliced and made sequences out of different vectors, achieving different lengths.

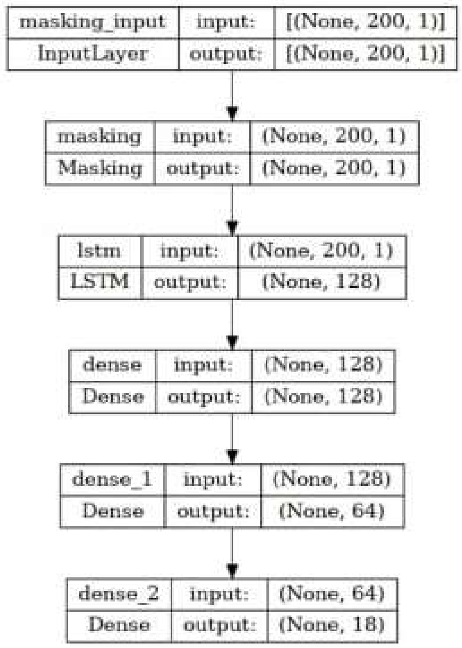
masking_input | input: | [(None, 200, 1)]
InputLayer | output: | [(None, 200, 1)]
masking | input: | (None, 200, 1)
Masking | output: | (None, 200, 1)
lstm | input: | (None, 200, 1)
LSTM | output: | (None, 128)
dense | input: | (None, 128)
Dense | output: | (None, 128)
dense_1 | input: | (None, 128)
Dense | output: | (None, 64)
dense_2 | input: | (None, 64)
Dense | output: | (None, 18)

Figure 5

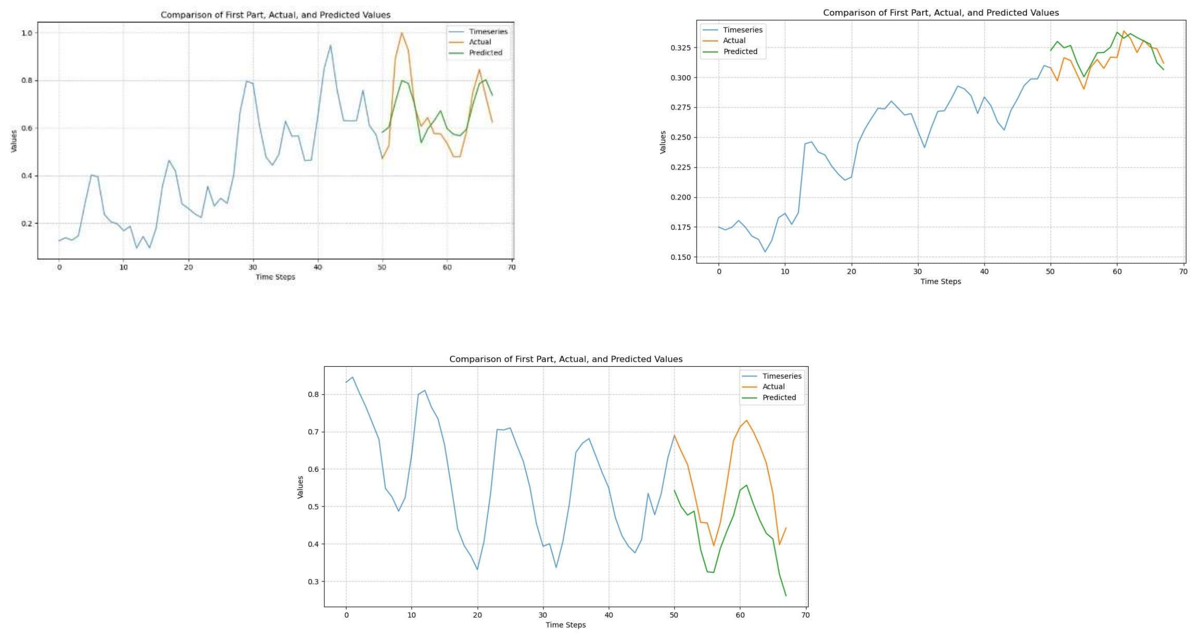

Figure 6

6. Contributions

As with the previous project, we worked in parallel, and all tried different approaches. Kamil focused a bit more on the net shape and preprocessing, Livia focused more on the data processing and Filippo focused more on the data analyzing and post-training analysis. Overall, we collaborated in most aspects of the challenge, taking turns submitting some commonly agreed-on and shared submissions. The report was written together, with Livia and Kamil mainly writing the text and Filippo obtaining the pictures and plots.