**Report of the group '3 Neurons 1 AI'** for the challenge regarding time series forecasting.

**Inspection.**
The provided dataset consisted of 48,000 time series belonging to six different domains. Our exploration began with a thorough examination of the data, where we initially visualized random sequences. Despite the apparent variability within the sequences, we found it challenging to draw conclusive insights.
Subsequently, our focus shifted to identifying distinctions among the six categories. Notably, only category 'F' exhibited notable differences in features; however, we acknowledged the possibility that this observation might be attributed to the limited number of elements within that category.
To dive deeper into the dataset, we conducted an analysis of the series lengths, recognizing its significance given the range from 24 to 2776.What was promising was the fact that the average length hovered around 200, aligning closely with the length of series in the test set.
Our exploration extended to the identification of outliers, employing both Root Mean Square Error (RMSE) from the overall mean and z-score calculations. However, we couldn't consider the values we found as outliers since they could be present in the test set as well.
In conclusion, our data inspection yielded limited conclusive findings, with the exception of sporadic observations of periodicity in certain series. We kept this in mind for possible preprocessing.

**Preprocessing**
After tackling homework1, we figured out that the real game-changer for a neural network is how you handle the data. Messing with different model setups doesn't shake things up as much, at least in the beginning phases. So, we went all-in on trying different ways to handle the data but, in the end, nothing gave better results than the provided dataset scaled between 0 and 1. Here's a rundown of the more interesting stuff we tried:

- Instead of using the original series we took the difference between one time step and the next, thinking it might work since it keeps the data intact and makes the output stationary[1].
- Toying with stationarity, we tried various transformations like square root, cubic root, and log. We even threw in some second-order differentiation and mixed and matched these transformations just to see what happens.
- Figured reducing noise in the sequences could help, so we played around with rolling mean smoothing, exponential smoothing, and polynomial interpolation with one or more polynomials, each on its own.
- Then we got into different padding methods while training, like ditching short sequences, padding with the sequence mean, flipping the sequence at the start, and slapping the first 12 time steps at the beginning. We stumbled on the 12-step thing because turns out many sequences had a 12-beat rhythm. More on that in the next section on our other attempts.

**Model selection**
To approach time series with a model that could have memory we had to choose between LSTM, GRU or RNN layers. In addition, we also tried an ESN layer, because of its use in time series forecasting. Through a few attempts we understood that GRU bidirectional layers were learning better in our models. We also tested different batch sizes, skip connections, batch normalization and dropout

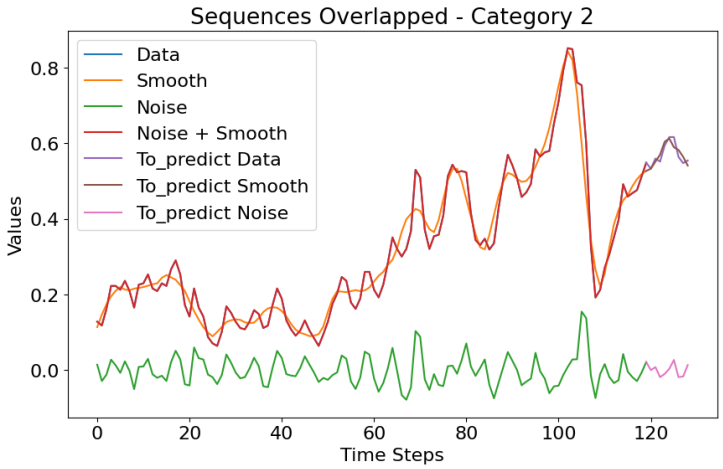
[1] We deemed this characteristic useful since, after some research on the topic, we thought it might simplify the underlying patterns and relationships in the data, making it easier to model and predict future values

layers but, surprisingly, simplicity reigned supreme and we discovered that a straightforward model tended to outshine the more complex ones. The thing that improved our model in a not negligible way was, however, the introduction of attention. We implemented the multi-head attention layer from Keras, fine-tuning the number of heads and the key dimension, and that's where we saw some noticeable improvements in results.

**Many interesting attempts**

Diving into the dataset, we examined numerous samples, and we had a lot of ideas on how to enhance feature extraction for a better-performing model and predictions. In particular, to measure model performance, we split the dataset into training and test sets, predicting on the latter along with predictions for different categories and their averages. Given the even distribution of the dataset across categories, it seemed like a fair approach.
All of the main ideas explored are listed below

- Adjusting the length of input sequences was an early consideration. Initially, models with sequences of 40-50 time steps performed well, but for predicting 18 steps into the future, longer sequences proved to be more effective.
- Experimenting with autoregressive forecasting using varying lengths didn't yield favorable results, possibly due to the importance of the last values in predictions.
- We explored the idea of applying smoothing to individual sequences, but surprisingly, this led to worsened results.
- Attempting time series decomposition into trend, seasonality, and noise, with different functions (see the following figure), didn't contribute to any major improvement although it seemed a promising route.

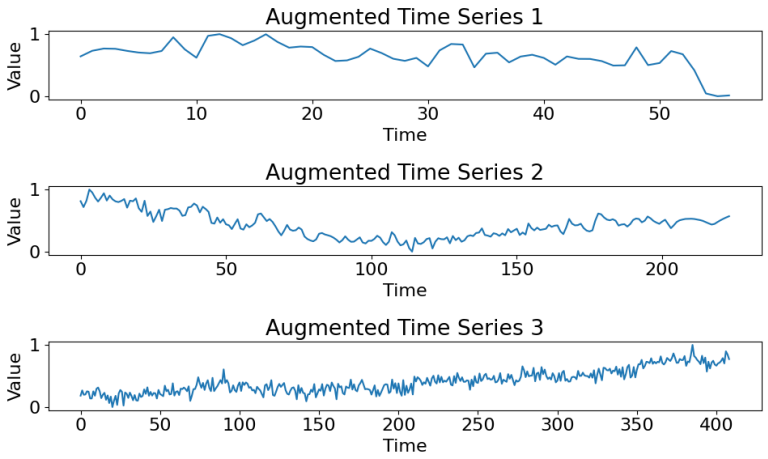



Even training without noise or making the data stationary by removing the trend did not yield positive results. We also delved into multivariate sequences, creating different features, but no combination led to improvement.

- Trying Fourier transform to exploit periodicity revealed intriguing insights. Series often exhibited high amplitude near zero, indicating constant areas. We noticed spikes at frequencies corresponding to 12 and 6 time steps. Although we experimented with training separate models for these groups, the reduced data hindered learning. Fine-tuning the best

model for this split didn't yield better results, so we stuck with a simpler approach using a single model.

- While multihead attention showed promise, we didn't stop there. We experimented with other attention techniques, including self-constructed attention blocks involving dense, repeat vector, and dot layers. Self attention and additive attention were also in the mix, exploring various avenues to enhance our model's performance.
- We tried  to implement Time2Vec in particular with the use of an embedding layer to convert time-related features into embeddings.
- Facing persistent challenges with category F results, we took a leap into data augmentation with the aim of upsampling this specific data. Initially, we experimented with various random combinations of four selected augmenters[2] from the tsaug library (the timeseries achieved were reasonably in line with all the others), but we didn't achieve any remarkable improvements for category 'F'.




Then we thought about doing something similar to the mix-up augmentation for image processing. The idea was straightforward: aiming for better generalization, we mixed random sequences from category F, maintaining a balanced approach by constraining the model with an even distribution of sequences. This approach was a deliberate attempt to inject diversity and balance into the training data for category F, with the ultimate goal of enhancing the model's performance for this particular category.

**Final model & Conclusions**

The final model we implemented, the one who worked best for us was a model trained on the dataset provided, without augmentation or upsampling. The model consisted of the time to vec embedding, with 2 bidirectional Gru layers and 3 convolutional 1D layers.
Although the result achieved didn't meet our expectations, we were still at a good point in the ranking and we learned a lot.

[2] Timewarp, Addnoise, Drift, Reverse

**Contributions**
Simone: visualization, build sequence structure, model tuning, fourier approach, differentiation, ensembling, padding, prediction

Andrea: model tuning, exponential smoothing, seasonal decomposition, autoregressive forecasting, upsampling, augmentation

Claudio: stationarity by transformations and differentiation, smoothing by autoregression and rolling statistics, attention techniques, model tuning