Second AN2DL Challenge Report

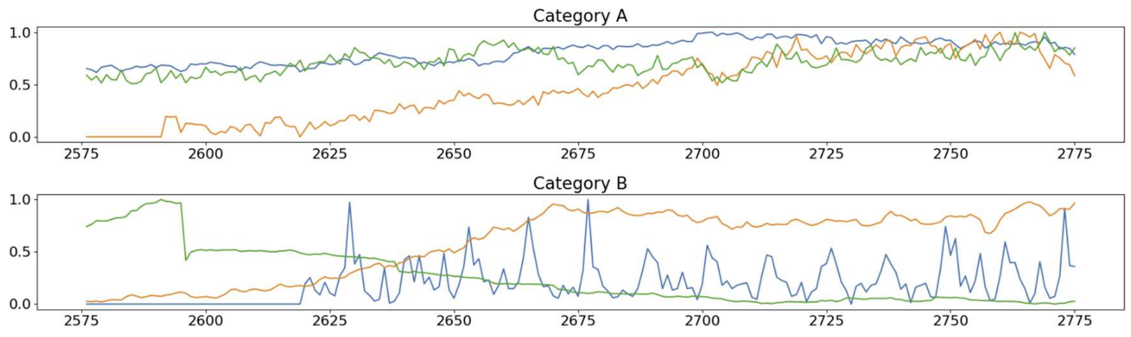
Francesco Ferrari, Roberto Cialini, Luca Bresciani, Francesco Pastori

Time series forecasting

The task is to develop a forecasting model that is able to predict several uncorrelated time series. The prerequisite is that the model exhibits generalisation capabilities in the forecasting domain, allowing it to transcend the constraints of specific time domains. This requires a model that, while specialised in forecasting, is not limited to predicting in a single or predefined time context. In the first phase it was required that our model predict 9 future values, in the second phase the values were 18.

Building the sequences

We began by creating the training samples for our model. First we transformed the dataset from numpy array to a dataframe and transposed it, for better usability. Next we proceeded by displaying some of the time series by grouping them according to category, as shown below.



Next we handled the creation of hopping time windows, which turned out to be very impactful on the model performance. We tried different ways of building time sequences using as much or as little padding, using a wider or narrower stride.

After several attempts, we realised that:

- The window size had to be set to 200, since it had to match the size of the series that we received as input when submitting the model to the platform;
- The best performance occurred by removing the series which had a length less than 200
- The best stride was 10, because with this size the model has a modest amount of windows that are not too similar to each other and did not overfit, which happens with a stride of 1 given the large amount of similar data.

In the building sequence algorithm we brought each sequence to be multiple of stride through the padding and then added the length of the telescope as padding, in this way we used all available timestamps but added the least possible padding.

Model Design

Our approach in this phase was very explorative and encompassed a wide range of possible solutions.

RNN [1]

The first try consisted in building a simple Recurrent Neural Network consisting of a layer of bidirectional GRU units. We settled on 64 units as it seemed the best compromise between trainable parameters and model performance. LSTM units were also tested, but we found that GRU units were more efficient.

Transformer Based Architecture [2]

We soon started researching other feasible solutions, by looking at the literature. We stumbled upon recent papers describing the use of transformer-like architectures for time-series forecasting purposes, like the Informer and Autoformer models. At first, they seemed like an optimal solution, but we quickly realised implementing such complex networks from scratch was outside our abilities and out of the scope of this task. However, as we gained more knowledge around the structure and features of transformers, we found that they could be applied to our task, by first overcoming some obstacles:

- Positional Embedding: we understood that we could not use the word and position embedding of a vanilla transformer intended for NLP problems, since our sample space was not finite and thus could not be easily contextualised in a vector mapping. This led to adapting a positional embedding algorithm to exclude the first part of the embedding, leaving us with information about time-dependence of our samples, which is crucial for accurate forecasting.
- Architecture: exploring the transformers solution, we struggled to translate the usual use case of the full transformer architecture, with all its features such as cross-attention, to ours. We started understanding that a decoder-only network was a feasible solution to emulate the generative capabilities of models such as GPT-2, which receive in input a prompt of vectorized tokens, and predict the next tokens in the sequence.

We proceed by creating a decoder network by adding a positional embedding layer, followed by 3 transformer decoders in series, and a global average pooling layer before the output to flatten the output shape.
With our local test set we obtained satisfying results, while testing against the hidden test set it gathered a result which was comparable to the RNN based model. We still hoped that this model would fare better in the final phase, as the transformer based architecture showed better performance than RNN as we increased the telescope size.

Hybrid Architecture: RNN + Attention Layer [3]

While researching materials about previous time-series forecasting efforts, we often stumbled upon models created by first having a RNN layer extract the time-series features, then putting an Attention Layer before the output, to preserve both the time dependent features extraction of the RNN and the context awareness of the Attention Layer, typical of transformer architectures. While this seemed like a great idea at first, and gave us a very satisfying result on the local test set, the result while testing against the hidden test set was very underwhelming, and we decided to focus on simpler ideas.

Autoregressive Prediction Model

For the second phase of the challenge, we tried to use the autoregression technique. The autoregression uses the model trained with telescope equal to 9 to predict the next 9 values and then takes the predictions and considers them part of the time series received as input by concatenating them to the latter, consequently going on to discard the oldest 9 values in the time series.This technique brought slightly lower performance than training the same model but with telescope equals to 18.

Conclusion

Model	First phase MSE codalab	Second phase MSE codalab
Gru 64 units	0,0056	0,0132
Transformers	0,0067	0,0108

From the results we infer that the model with transformers performs better with predictions farther in time while the model with GRU worsens by increasing the time horizon.

Improvements

Possible improvements to be implemented concern:

- optimization of the time embedding procedure regarding time series
- trying other building sequences algorithms to get different training sets. These new algorithms could exploit autocorrelation and use mean instead of zero as padding. It should be specified that many attempts have been made in this direction but were quickly abandoned because they did not bring improvements in testing, despite the fact that from a theoretical point of view they should have increased performance.
- try to set other parameters for the transformer in order to have a less complex model, because on the validation set the mse was always around 0.0075 but on codalab we were getting around 0.0112 this indicates overfitting perhaps due to a too complex model.

References

[1] - notebook name: “GRU dev.ipynb”
[2] - notebook name: “Transformer dev.ipynb”
[3] - notebook name: “Hybrid dev.ipynb”

Contributions

Every member of the group actively participated in every stage of the competition, both in researching new techniques and applying them in practice.
In particular:

- Francesco Ferrari experimented with and realised the transformer based models
- Luca Bresciani inspected the dataset in detail and worked on its representations
- Francesco Pastori experimented with sequence building and GRU models
- Roberto Cialini experimented with GRU and LSTM models

Francesco Ferrari 10651673
Roberto Cialini 10700005
Luca Bresciani 10692758
Francesco Pastori 10628982