Homework_2 – Report

Authors: E. Broggini, 10670120; E. Cabiati, 10698131; L. I. Pagliochini, 10707169; F. Zhou, 10679500.

1. *Introduction*

The task is to develop a forecasting model that is able to predict several uncorrelated time series. The prerequisite is that the model exhibits generalization capabilities in the forecasting domain, allowing it to transcend the constraints of specific time domains.

2. *Data Analysis*

As usual, we first inspected the data that was given: we started by plotting some time series form the dataset, also including the distribution over the different domains, as well as the mean length of the time series per domain. Regarding this matter, we readily highlighted a strong imbalance between the domains (specifically the fifth domain was strongly underrepresented, along with the first one, which was nonetheless the one with the highest mean length). Still, after arguing whether to balance the training using weights, given the challenge was not a classification problem, we shifted our attention rather than on the imbalance itself, on the very partition among time domains and how we could use it to our advantage.

3. *Regarding data manipulation, normalization and time domains…*

Firstly, each padding sequence was removed, and only valid time steps were kept.
As mentioned before, to deal with the different domains, we tried to further normalize the data, specifically by using a - custom implemented - robust scaler over a specific time domain. Still, by using it, the training process became slightly more instable and "dull", even after duly modifying the net and inference phase to take such normalization into account. Seeing no significant results, we opted to keep the data unchanged, while experimenting with ensembles of six different models, one for each domain, to see if this approach could have led to better results. In the end, simpler models still outperformed all the ensembles we tried.

4. *Dataset partitioning and sequence creation*

In general, we decided upon a partition of the dataset as follows: 20% of the whole time series set for the validation. What remained was further split into a 0.03% pool to generate a private test set, while the remaining 79.07% was used for training purposes. Regarding the creation of sequences, the selection of the stride hyperparameter was carried out by fine tuning over the very first models and was preserved throughout almost all our experiments with different models (that is, equal to 10, 5 for later models).

5. *Initial models*

The very first model developed was based on direct forecasting and made of a bidirectional LSTM layer followed by Conv1D, plain and simple. It soon became clear the need to add more complexity to the model, as it was not properly able to generalize. This model was soon followed by one with more convolutional layers (mixed with Average Pooling layers, to avoid the cropping on the last stage of the model, hoping to preserve as much information as we could). In this case, the model reached convergence during training, using 64 LSTM units, with a MSE value of 0.0103 on the private test set *(PTS; we chose to compute all the metrics over a private test set, where the inference was carried out over 18 time-steps*).
Increasing the number of units for the bidirectional layer only seemed to worsen the performance. Following changes to the architecture included using more Bilinear layers, as well as including more complex architecture for the CNN part of the net: none of these seemed to increase the ability of our model to generalize.
By plotting auto-correlation and partial auto-correlation graphs over the first time series of each domain, we highlighted a strong 'direct' relation between one time-step value and the successive 2/3 (the auto-correlation plot showed a relationship between the time steps that lasted longer, accounting for 'indirect' influences among these). As such, we soon switched to an autoregressive approach, using a lag of approximately 3 time-steps, which, given the same architecture, returned better results, with a MSE = 0.00977 on the *PTS.*
The switch from direct to autoregressive framework proved effective even on the first phase test set: it allowed us to move from a MSE of approximately 0.011 to 0.007, with a visible increase in performance.
Using this very same framework, we experimented with different ensembles, the reasoning behind being the following: creating a model for each time domain and letting it train over this should have led the model to learn patterns related to that specific time domain, something that may not have happened entirely using that very same

model over the entire dataset. In practice, though, this setting did not bring about good results: this model recorded a MSE with a value of 0.0107 over our *PTS* and a, rather disappointing, 0.099 over the first phase test set.

6. *Attention*

Unsatisfied with the results obtained by using simple LSTMs and Convolutional layers, we first approached the attention mechanism for time series prediction by trying to integrate it with the models we previously developed, as we found its 'selective focus' property over the input values quite appealing.
As such, we developed a new model, which included:

- *CNN* with Average Pooling layers for dimensionality reduction, to capture local patterns and enable hierarchical feature learning;
- *Bilinear LSTM* with 64 units, to equip the model with memory;
- *Custom Attention module*, for selective attention over the LSTMS inputs;
- Final short *dense layer.*

This last model showed a satisfactory prediction score compared to the previous models, scoring 0.0092 on the *PTS* and 0.0068 on the first phase test set.
We tried to further modify the model, to enhance its performance, but we were not able to notice sensible changes, as far as its score over the PTS was concerned. At the same time, we didn't want to spend more resources on this than necessary: the increase in performance alone was sufficient proof of the fact that attention could give our model something more. We then decided to further experiment on this, but with a different, more sophisticated framework.

7. *Final Model – Fully Connected*

After transitioning from direct to autoregressive forecasting, our focus shifted onto building a model able to capture tendencies, seasonality, and other time-related trends. This allowed to improve the accuracy of forecasts and offer contextual relevance to our predictions.
What was initially intended to be a simple benchmark for another, more complex model, ended up being the actual best model we could muster: a simple - 4 layered - fully connected, dense network.
The model which achieved the best score on the first phase test set is composed of 4 dense layers of respectively 256, 128, 64 and 32, plus an additional output layer of 3 neurons, with sigmoid activation function (for the normalized data, between 0 and 1). The incorporation of RELU/GELU activation functions was selected to introduce non-linearity and facilitate the model in capturing complex patterns in the data. Overall, a plain and simple architecture, and still, capable of generalizing more than what was supposed to be our most promising model.
Additional Batch Normalization (only for the first two layers) and Dropout layers where also included to consider the presence of outliers in the dataset, as well as to promote independent learning.
The overall performance assessment of this version on the hidden test set revealed an impressive MSE value of 0.0050 (first phase) but later fell short during the second phase, with a MSE of 0.012.
In another version, we trimmed the number of neurons in the model (from 256 to 32, and so on) and removed batch and dropout layers: the resulting MSE of 0.0096 on the *PTS* hinted at a worsening performance but, instead, did surprisingly well on both Codalab test sets (0.0051 for the first and 0.0099 on the second).
This emphasizes the importance of carefully considering the trade-off between model complexity and performance: even though the first version seemed more prone to follow accurately the trend of future samples, it indeed proved to be less effective than a – supposedly - less expressive model.

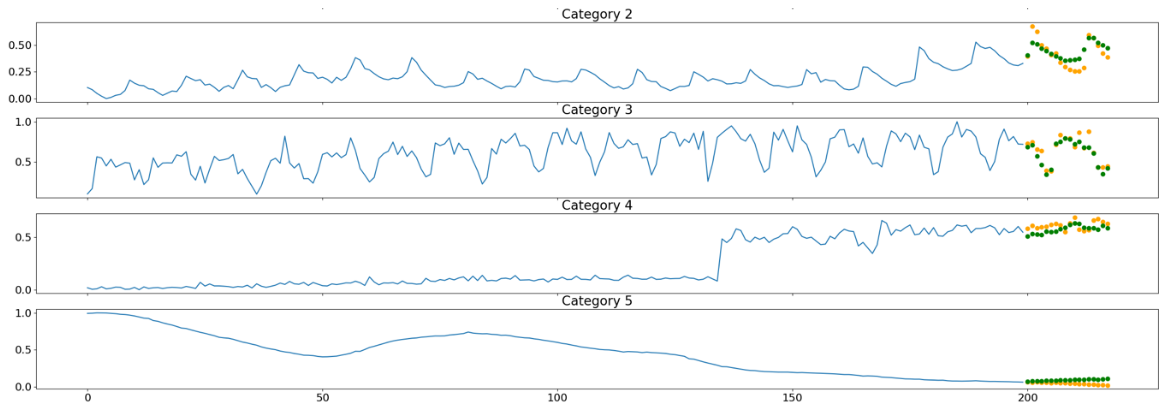

Figure 1- Predictions for the first 18 future time steps, for signals of different domains, as forecasted by the FC model (first version)

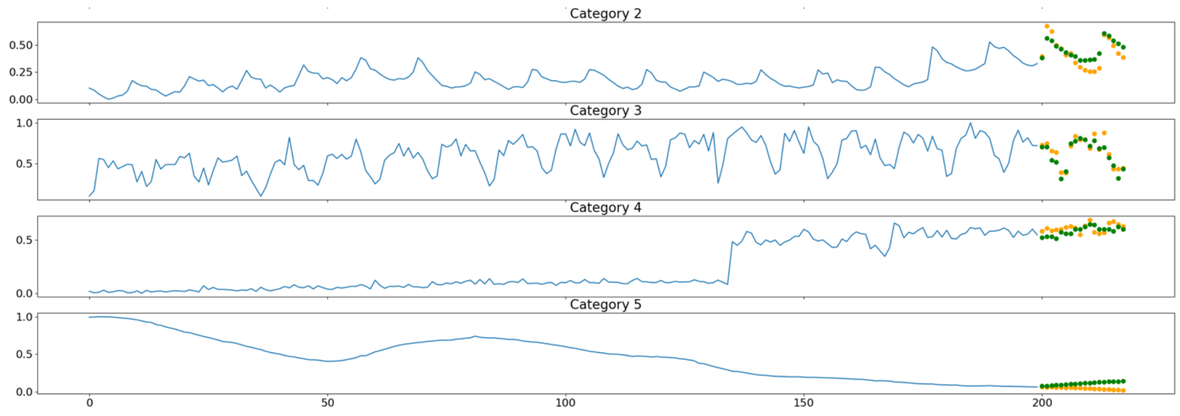

Figure 2 - Predictions for the first 18 future time steps, for signals of different domains, as forecasted by the FC model (second version))

8. *Further experiments and Honourable mentions*

Beside the aforementioned models, some more experiments were carried out on the following:

i. **Ensemble of Fully Connected**: the rationale behind this is the same as the one already exposed in the fifth paragraph[1]. While this approach had already brought to a worsening in performance compared to other alternatives, we still wanted to see whether this behaviour was due to particular sub-models chosen for the specific time domain. Still, again, this approach proved unsuccessful, even with - apparently - more performant models.

ii. **Transistor Encoder**: the inspiration for this model was taken from the "A Transformer-based Framework for Multivariate Time Series Representation Learning"[2] paper. The main blueprint for the model is the following:
  - Firstly, the input vector of 200 values (window over the time series) is projected onto a smaller dimensional space, using a Conv1D layer;
  - This projected tensor is then added to a learnable Positional Encoding tensor;
  - The embedded input it then fed into a stack of 3 Transformer Encoders with Multiheaded Attention. In this case, differently from the conventional encoder, Batch Normalization is used instead of Normalization Layers, while the Dense Fully connected net is substituted by a CNN, which provided a more stable trend during the training process;
  - Finally, the output of the encoder is fed to a last, fully connected and shallow network.

It is worth noting that, while this last model performed better on the PTS (with a MSE of 0.0072 and a MAE of 0.0527), it was still overshadowed by the fully connected model on both the first and last phase test set (on which it scored a MSE of 0.0105)

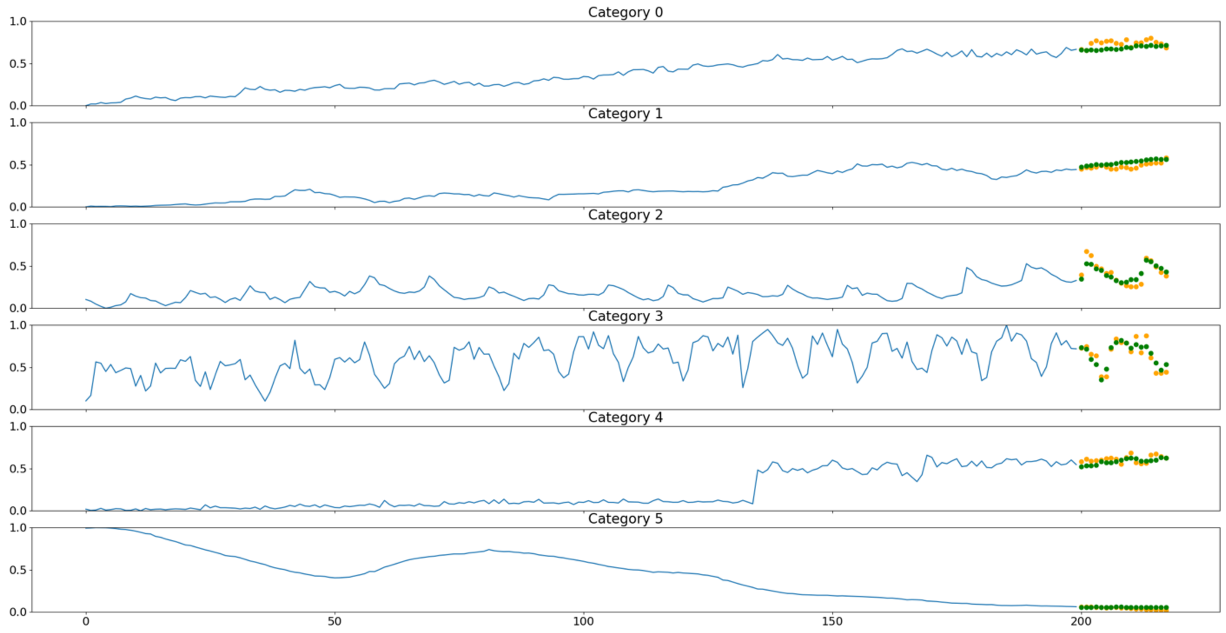

Figure 3 - Predictions for the first 18 future time steps, for signals of different domains, as forecasted by the Transformer Encoder model

9. *Contributions:*

E.Cabiati: research on Direct Forecasting, Autoregressive, Ensemble, custom Attention, Fully Connected and Transformer Encoder model
F.Zhou: basic LSTM along with Conv1D, some trials with basic Attention
L.I.Pagliochini: direct and autoregressive convolutional LSTM Models and testing on different normalization techniques
E.Broggini: analysed different models on Direct Forecasting, Autoregressive, then using Attention mechanism and finally Fully Connected

References:

[1] Ratnadip Adhikari, R. K. Agrawal. *Combining Multiple Time Series Models Through A Robust Weighted Mechanism*.
URL: https://arxiv.org/abs/1302.6595

[2] George Zerveas, Srideepika Jayaraman, Dhaval Patel, Anuradha Bhamidipaty and Carsten Eickhoff. *A Transformer-based Framework for Multivariate Time Series Representation Learning,* Dec 2020.
URL: https://arxiv.org/abs/2010.02803