Artificial Neural Networks and Deep Learning

Report - Challenge 2

Alessandro Ferri	Laura Colazzo	Ignazio Neto Dell'Acqua	Zanubia De Pasquale
10578145	10705522	10704869	10668387

A. Y. 2023/2024

1. Introduction

The primary goal of the second challenge was building and training a deep learning model able to perform time series forecasting. For this purpose, a dataset was provided, including a large number of univariate time series, i. e. time series composed of a single feature, each of them being assigned a specific category. The category associated to each sample was a single alphabetic character from A to F, for a total of six different time series categories available. From a semantic point of view, categories represented different domains and time series in the same category were not to be intended as correlated to each other, but rather collected from similar data sources. Hence, given the divers nature of the dataset, the predictive model was supposed to show generalization capabilities in generating 9 or 18 future data points, i. e. the model should have been able to disregard constraints on specific domains, thus being able to make accurate predictions on time series from different categories.

The structure of the report unfolds as follows: Section 2 provides insights on the preliminary analysis of the input dataset, exploring the trade-offs considered in the pre-processing phase. In Section 3, we dive deeper into the different architectures adopted to develop the solution to the challenge. Section 4, then, focuses on the development of an architecture comprising six different models, each of them being specialised in a specific category, together with the quantitative results achieved through it. The culmination of the exploration of solutions is presented in Section 5, where the architecture of the model that showed the best performance is detailed, together with the scores obtained.

Through these sections, we aim to provide a comprehensive understanding of our approach, challenges faced, and the ultimate success achieved in the pursuit of an optimal predictive model.

2. Data inspection and pre-processing

The first part of the analysis focused on understanding the structure of the dataset, which was already split in three portions: the training data, composed of nearly 50k time series, each of them padded with zeros to reach a uniform and fixed length of 2776 time steps across all the time series; the dataset of valid periods, reporting for each time series its actual starting and ending indexes, ignoring the initial padding; the category array, indicating for each time series the category it belonged to. The size of all these three datasets was the same, since given a row index you could have used it to check for data related to that single times series in all the datasets.

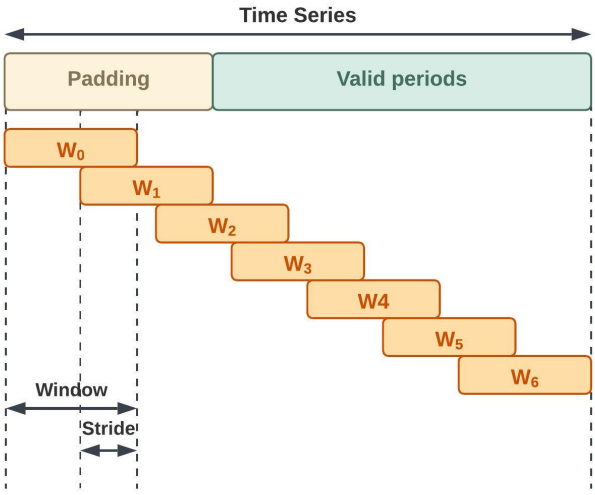
After inspecting for each category the number of time series associated to it, we realised that the number of samples in category F was dramatically low. Moreover, the statistics about the time series' lengths showed an average of almost 200 time steps: much smaller than the maximum registered value of 2776. The way we addressed these issues will be discusses in the following sections.

2.1. Dealing with padding

A preliminary naive strategy adopted to deal with padding (Figure 1) consisted in further extending the padded portion of each time series, in order to make its length a multiple of 200 time steps. This was also the size of the window used to split each time series in multiple input sequences to be fed to the model. Adopting this strategy allowed to obtain at least one sequence per time series. Then, when designing the architecture a masking layer was introduced after the input layer. This made possible to ignore the sequences of zeros prepended to most time series. However the masking layer proved to be the bottleneck of the architecture during training, making it last a few minutes per epoch. Moreover, it was in any case a non-optimal solution, since a lot of sequences containing only zeros were generated.

As a consequence, we decided to develop a more refined strategy (Figure 2) and to effectively use the information available about the start and end indexes of each time series to get rid of padding. Then, the problem related to having time series of different

sizes was delegated to a suitable windowing mechanism, later discussed.



2.2. Dealing with underrepresented categories

As we already mentioned, category F was underrepresented in the input dataset. To address this problem, we defined a custom strategy for making more fair splits between training and test set. As a general rule, we fixed the test set's size to 10% of the dataset size. We then chose the number of samples for each category to be the same across all of them in the test set (800 samples per category), except for the F category. The number of time series from the F domain was instead pondered on the actual availability of F samples, therefore it was set to 30.

2.3. Windowing strategy

The windowing strategy, adopted to split time series into sequences to be used as samples for training, involved a window size of 200. Moreover, in the beginning the telescope was fixed to 9, i.e. equal to the number of samples that we were asked to predict in the development phase of the competition. Later in the process, we increased the telescope to inspect the behaviour of our model on more advanced predictions, in preparation to the final phase of the challenge, which asked to make predictions of 18 samples in the future.

In the beginning, when we were still experimenting with the masking layer, the window was slid by 10 time steps at a time. This choice of making windows overlap came from the willingness to make the model learn from as many sequences as possible.

When we switched to the more refined strategy for dealing with padding, we kept these same parameters for windowing. The new strategy, as we mentioned before, consisted in completely discarding the padded portion of time series, whenever their length was actually larger than the window size, and in making windows start from the first valid time series' index. Then, the window was slid as long as its ending index fell inside the valid range of time steps. Hence, when performing the last valid slid, we generated the last sequence by making the end index of the window overlap with the last valid index of the time series. This allowed to end up with the majority sequences free from padded values.

Then, given the promising results obtained in training, we decided to gradually decrease the window stride, to further increase the number of sample the model was learning from. In this experimental part we found a value of 5 for the stride to be the optimal one.

3. Network architectures

While refining the window management, we decided to keep the model's architecture very simple, with just two LSTM layers of 64 memory cells each. This allowed us to understand how to deal with the generation of input sequences to the network, before experimenting with more sophisticated architectures. Only once the window management strategy was consolidated, we moved on to the design of a more complex network.

We then decided to explore two alternative strategies for building the architecture of the predictive model: one option was building and training six different models, each one specialized in one of the six time series domains; the alternative option was adding complexity to the base model we had already used in the very first phase of development. In the following sections, we will dive deeper into the details of these strategies, reporting the performances achieved with each tested architecture.

4. Ensemble of models

The idea behind having an ensemble of six models came from a very simple intuition: given that the test set on which prediction were done once the model was uploaded on Codalab also contained the category of each time series, we thought about training multiple models, each of them specialized in making predictions in a given domain. In this way, after checking the category of the sample time series given as input to the ensemble, it could have been redirected to the matching sub-model.

Pre-processing	MSE - CodaLab
Naive strategy	0.0079
Refined strategy	0.0065

Although this strategy seemed reasonable and our expectations on it were much higher, in the end it did not performed as good as initially thought. The alternative solution with a single model, indeed, proved to outperform the one with ensemble. The interpretation we were able to give to this result was that maybe there were some general patterns, independent from the time series domain, that the six specialized models were not able to capture.

5. Final model

The alternative solution to the ensemble of model, as we already discussed, consisted in adding further complexity to the base model used while experimenting with different pre-processing techniques.

The way such network was designed was in an incremental way. More in detail we firstly tried to introduce a third LSTM layer, added on top of the two layers of the base model. But given that no improvement was registered with this modification, in the end we kept only the two initial layers.

We later tried to combine LSTM layers with 1D convolutions, followed by a MaxPooling layer. The idea behind this strategy was that of letting Conv1D layers capture higher level patterns that LSTM layers could have subsequently correlated in the time domain. In doing so, we preferred to start with a relatively shallow network to try to check for simple patterns in input the sequences. Unfortunately, this experiment did not bring the desired results, making our base model still look like the best one that far. However, since we learnt from the very beginning of development that our input series were not excessively long on average and while visually inspecting the time series one by one we did not detect long-term patterns, it did not seem worth trying out deeper networks.

Given that any attempt at improving the score reached by the architecture with just two LSTM layers seemed to fail, we decided to focus on the fine tuning of that model.

Here we discuss about the optimal values found in this process, which are also the parameters used by our best-performing model on the local test set. The size of the test and validation sets remained 10% of the initial dataset's size, whereas the batch size was set to 128. Moreover, the architecture comprised two LSTM layers of 128 memory cells each, in contrast with our first model, where the number of cells was halved. Additionally the second recurrent layer had the return_sequences parameter set to False and a dropout parameter of 0.4. This last measure was taken to try to overcome the saturation of validation loss experienced in training. As a matter of fact, after introducing it, we noticed a general improvement of the performance, so we decided to keep it. The last layer of the network's architecture was a Dense layer, with a number of neutrons equal to the output size, i. e. the desired telescope's size. The model was trained using Adam as optimizer, choosing 1e-3 as value for the learning rate and adopting the Mean Square Error as loss function. Then, two callbacks were defined for the training procedure: Early Stopping and Reduce Learning Rate on Plateau. Both of them kept monitored the validation loss, but while the former had a patience of 12, the latter had this parameter set to 10. Additionally, we needed to a set a custom value of 1×10^{-4} for the min_delta parameter in Early Stopping, to better control the sensitivity to changes in the validation loss.

	MSE - Local test	MSE - CodaLab
Development Phase	0.0035	0.0048
Final Phase	0.0066	0.0104

5. Final phase

Given that our best model in the development phase performed significantly worse when submitted in the final phase, we thought about experimenting with one last technique to improve our prediction capability on a longer temporal horizon.

More in detail, we tried to implement an attention mechanism by adding an additional attention layer to the model we already discussed in the previous section. However, due to the limited duration of this phase, we were not able to properly fine tune the model as to obtain a better performance, despite the promising results in local trials.