AN2DL - Homework 2

Time Series Forecasting

Giorgio Miani, Francesco Micucci, Pietro Pasquini

22 December 2023

1 Introduction

The homework was about the Time Series Forecasting topic. So, the main goal was to implement a model that should have been able to predict the future based on past observations.
To achieve this task we proceeded in the following way:

1. We inspected the dataset
2. We implemented a first model (which we can consider our baseline) and analyzed its results when used for direct forecasting and for autoregressive forecasting
3. We proceeded with building a ResNet-style model with Conv1Ds
4. We tried a new model with a lower complexity compared to the one used so far
5. We trained out best model also with slight modifications of out starting dataset (removing short sequences and constant sequences from our starting dataset and using Robust Scaling)
6. We tuned the way we split the dataset in sequences with 200 time steps each in such a way as to obtain the best model possible

2 Data Processing

As soon as we started the project, the first thing we did was to inspect the dataset.
We immediately noticed that we had different amounts of data for each category. In particular, the categories B, C, D and E have more than 10000 time series, category A has approximately 5,000 time series, and category F has approximately 270 time series.
This skewness of the dataset let us think that a good idea could be to implement 6 different models, one for each category.
We also analyzed the sequences in the dataset in search of missing values but we didn't find any.
We checked how many time steps each time series actually consist of.
Worth of mention is the fact that we have time series with less than 25 time steps.
Moreover, plotting the time series, we also noticed that some of them maintain a constant value for a large amount of time steps.
After having captured all the possible information from the dataset, we splitted the time series in a training and a validation set. To do so, we shuffled the time series order (but not their timesteps) and, maintaining a proportion of 9:1, we assigned them to the 2 sets.
Moreover, we also splitted slightly variations of the original dataset in which we removed the time series having an amount of time steps below a certain tolerance and/or time series that maintain a constant value for a big amount of time steps.

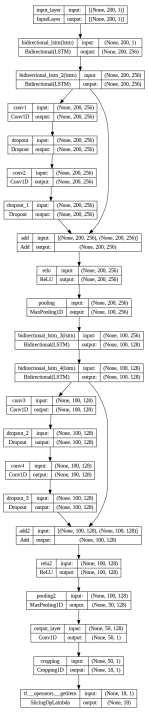
3 Starting Model

Once we completed the data processing, we proceeded by implementing a function (`build_sequence`). The idea behind this function is that, given some input time series, we extract from them sequences composed by a certain number of time steps (set through the parameter `window`) and the corresponding future predictions (the number of future timesteps is set through the parameter `telescope`).
We talked about this function because it has an important parameter that could be tuned: the stride. Given a certain time series, the stride represents the shift we have between the starting time step of one sequence and the starting time step of the next one. Choosing the stride in different ways, we may obtain sequences that may be partially overlapped or not overlapped.
So, given that function, we extract all the sequences that are needed for the training of the model.

The first model we used was the one coming from the laboratories composed of a bidirectional LSTM layer with 64 units, a 1D convolutional layer with 128 filters and kernel size equal to 3 and at the end a 1D Cropping layer to match the output shape.
We knew that it wasn't performant but we used it as a benchmark for our future models.
Obviously we tried it in the direct forecasting and in the autoregressive forecasting case and we immediately noticed that the autoregressive approach wasn't paying, indeed using the predicted time steps to predict the next ones gave us worse results compared to the one where we predicted immediately all the future time steps.

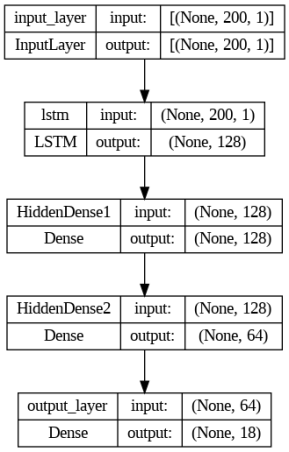
4 ResNet Model

After defining a baseline model from where to start, we began to increase the model complexity trying to reduce the validation loss of our model.
We started with a first draft of a ResNet-like model, which consisted of the baseline with a ResNet layer instead of the first conv1D. To be more specific, the ResNet layer was made of two conv1D layers with 256 filters each, followed by an Add layer for the residual connection, directly linked to the LSTM layer. This first complex model produced a slight decrease of the Mean Square Error on the test set.
Then we tried to further improve the performance of the network by adding another LSTM layer right after the first one and by inserting a MaxPooling1D layer right after the Add layer and we obtained a significant decrease of the MSE on the test set, going from 0.0071 to 0.0062.
We pushed further the complexity by first adding a Batch Normalization Layer after each conv1D layer inside the ResNet, which gave us worse results than before, and then we tried to add another ResNet Layer after the first one. This last change significantly increased the training time, so we had to increase both the batch size and the stride in order to train the model in a reasonable time. At the end the performance produced by this last model was similar to the best one so far, but the training time was lower then to the other models thanks to the bigger stride, so we decided to keep this as our best ResNet type of model.
In the meanwhile, we used one of the ResNet inspired networks to train 6 different models (one for each category) and at test time we predicted each time series with the associated model given its category. Even though it seemed a good idea for us, we achieved worse performances on the test set with this kind of network compared to the case in which all the categories were used to train just a single network.

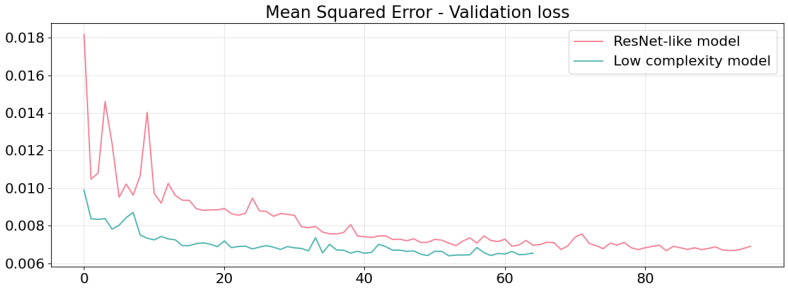


5 Low Complexity Model

Since increasing the model complexity wasn't paying, we decided to build a new model from scratch where we used the basic elements for a Recurrent Neural Network.
The new model, after several adjustments, was composed by an LSTM layer of 128 nodes, 2 Dense layers with ReLu activation function (composed respectively by 128 and 64 nodes) and an output dense layer used just to match the output shape.

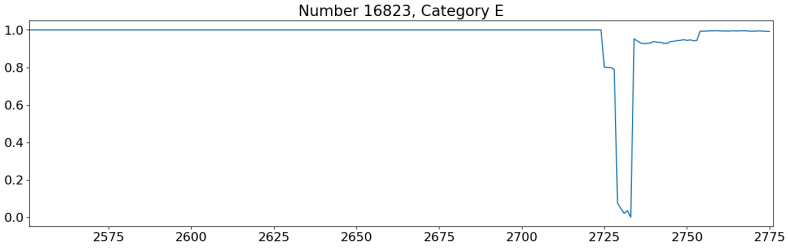


Using this new network we were able to reduce more the Mean Square Error on the validation set as can be see from the following figure:



6 Modified Dataset Analysis

Once we were not able to further improve our models we began training them with the variants of the original dataset created during the Data Processing phase. From the dataset analysis we first saw there were time series constant for a big amount of time steps as the following one:

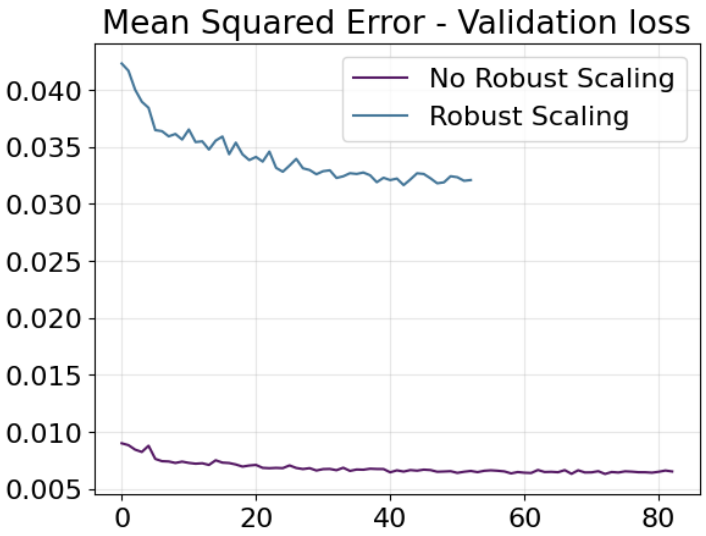


and we thought these time series were anomalies not relevant for our dataset so we tried removing them and training our model without them. This resulted in a little improvement of the MSE on the test set.

Then we realized that some time series had very few timesteps. We theorized the model didn't have enough steps in order to properly learn how to predict this time series, so we created a dataset without those and we trained our best models with these new sets. We weren't sure about the maximum length to remove, so we created four different datasets where we eliminated time series shorter than 25, 30, 35 and 50 timesteps respectively.
Unfortunately the results with these new data were not satisfying for both ResNet and Low Complexity Models.
Finally we combined the dataset with no constants with the ones without short sequences, but the performance still did not improve.

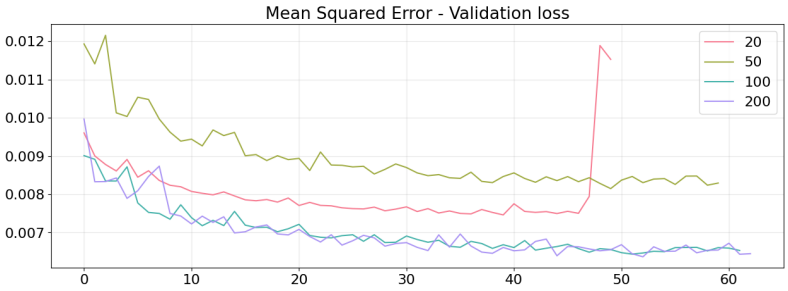
7 Robust Scaling

Given our initial data, we wanted to scale them using statistics that are robust to outliers.
For this reason we used robust scaling which consists of removing the median and scale the data according to the Interquartile Range.
Even though it seemed a good idea at first, when we completed the training of our best model with this scaling we immediately saw that the results were quite disappointing.
The mean square error on the validation set when using Robust scaling was much bigger than the mean square error achieved without the Robust Scaling as we can see from the following image:



8 Stride Tuning

As we said previously, the stride parameter used while building the sequences for the training of our models was important. Changing it will result in having more or less overlapped sequences for the training. In particular for our case, a stride equal to 200 would result in non overlapped sequences.
To tune this parameter we used our best model, and we built sequences using different strides. The results indicate that the best values for stride are 100 and 200 as we can see from the following image:



9 Final Model

To sum up, we built two different models: a more complex one with a ResNet-style structure and a simple one using just LSTM and Dense layers. We tuned on the stride trying to find the most performing one, and when we couldn't improve the model more we started working on the data. We analyzed it and we found possible outliers, such as short sequences or constant ones. Our first approach in order to deal with them was to remove them and train our best models on the new dataset. Then we tried to make the model robust to them using robust scaling. At the end our best model was obtained by combining the low complexity model with a stride of 100, trained on the original dataset.

Contribuitions

Similarly to the previous homework, we worked together on the whole project by meeting in person before or after lectures. We also split the training evely between us to utilize all the available resources.