Artificial Neural Networks and Deep Learning: Homework 2

Teka Kimbi Ntimanputu (10673197), William Stucchi (10661711) and Lorenzo Veronese (10654901)
{teka.kimbi, william.stucchi, lorenzo1.veronese}@mail.polimi.it

I. Introduction

Time series consists of a sequence of values of a certain quantity, each measured at a given timestamp. Hence, this type of data can be used to represent the evolution of quantities through time. One possible way to study time series is to do classification, which means to classify a sequence as belonging to one class or another. A further perspective in some scenarios is to use observed values to predict the future, i.e., how the quantity of interest will behave in the future timestamps. This last application is called forecasting.

In this work, forecasting is performed on a dataset of six uncorrelated time series. Due to the complexity of the data, deep learning (DL) techniques have been deployed: most of the state-of-the-art techniques were analyzed, including convolutional layers, Long-Short Term Memory (LSTM) and Attention mechanisms.

II. Dataset description

Due to the challenge format of the project, the dataset is split into three parts:

- local dataset, which can be analyzed and used to train the models;
- remote private dataset of phase 1, which is not accessible and is used to evaluate the first models. The telescope is of 9 future samples;
- remote private dataset of phase 2, also not accessible, which can be used a very limited amount of times, hence it is used for a final evaluation of the model. The telescope in this case is of 18 future samples.

The local dataset provides the distinction among six classes, A, B, C, D, E, F with respectively 5728, 10987, 10017, 10016, 10975, 277 sequences of various lengths, for a total of 48000. The low quantity associated with F immediately catches the eye: upsampling procedures were unsuccessful in solving this issue, hence they were not considered at the end. The remote datasets are composed of sequences of 200 timestamps, which indeed will be the input size of the models.

The local dataset comes as an array of shapes 48000×2776. To each sequence a couple of values are associated: the indices between these are the actual time series, while the others are just zero padding. Hence, the first step consisted of eliminating the padding, obtaining a dataset perfectly analogous to the initial one, also in terms of categories associated with the time series, but ready for future steps. It has been saved to avoid recomputing it every time a model has been trained. After that, the result has been shuffled and split between the train and validation set, with a ratio of 90-10% (conserving the distribution of the classes). No relevant changes were noticed when increasing the percentage of the validation set.

All time series are already normalized between 0 and 1, hence no step has been done towards this direction.

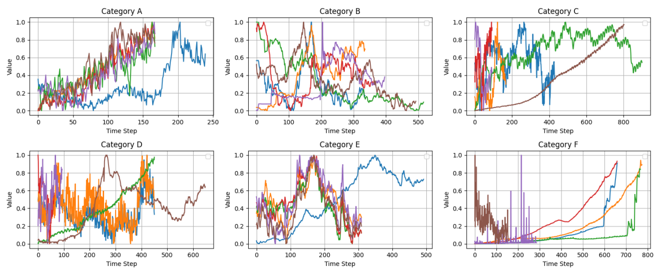

Fig. 1. Plots of the different time series for each category.

III. Experiments

This section describes the steps, trials and implementations that led to the final implementation that is shown in Section IV.

A. Autocorrelation and partial autocorrelation

An analysis of the time series autocorrelation has been performed to better understand the behaviour and possible relation between different timesteps of each time series, which was preparatory for wisely choosing the value of the stride for the window construction that will be explained in Subsection III-B. This study showed that in some time series, a sample is uncorrelated (i.e. the autocorrelation is statistically not significant) from the next ones from 5 timestamps after it, but in some other sequences, it may be 50 or even 100 timestamps after. This happens also inside a single category, making it hard to find relevant patterns.

To exclude the indirect relationships mediated by the intermediate values, also partial autocorrelation was performed, resulting most of the time in statistical significance only for the value directly after the current sample. This suggests that a good value for the stride could be 1, but this is impractical from a computational perspective since the quantity of data would increase too much. For this reason, the lowest value considered for the stride has been 5.

B. Windows-based approach

To prepare the dataset described in Section II for the training, a window-based approach was performed. From each time series, some couples of input and target sequences are obtained, each of 200 and 18 samples respectively. 200 fits all remote datasets, while 18 has been chosen as a telescope to produce general models, applicable to both the remote dataset of phase 1 (with telescope 9; in this case, the output is just cut to the first timestamps as this did not show any relevant effect on the performance) and of phase 2. A padding of zeros was applied at the start of the time series in the cases in which the window was overflowing. Regarding the stride, the best value turned out to be 5, confirming the partial autocorrelation analysis of Subsection III-A. This value is a good compromise between the performances of the models, which seem to increase as the stride decreases, and the training time.

To avoid losing the order of the windows coming from the same sequences, no shuffling has been performed after this step.

C. Data augmentation

Some techniques of data augmentation have been tested to increase the generalization capabilities of the model. In particular, jittering has been applied in two ways:

- sum of Gaussian noise: to each timestamp of a time series, a value extracted from a normal distribution is extracted and added to it.
- product of Gaussian noise: this is the same as above, but the extracted value is multiplied instead of summed.

All resulting values going below 0 or above 1, were cut to 0 and 1 respectively. Gaussians were set with a standard deviation of 0.1 so as not to change too much the values of the time series. To avoid the excessive presence of augmented data, around 10% and 20% of the training set has been transformed (note that the resulting time series are added to the training set, they don't substitute the original ones). Anyway, these methods did not change or even worse the performances of the models, hence they were discarded after the very initial steps. This may be because they did not represent the data, causing corruption that only damaged the ability of the model to learn from the input data.

D. Convolutional and LSTM models

Various models have been tested, with different characteristics and peculiarities. The first one has 1 bidirectional LSTM layer and 1 Conv1D layer, both with 128 units. This model has shown interesting performances, with a test mean squared error (MSE) on the phase 1 dataset of 0.0081. Further improvements were made by increasing the model complexity, for instance considering two consecutive bidirectional LSTM layers and two consecutive Conv1D layers, but they were not noteworthy. A noticeable improvement was recorded when these two layers were alternated with different patterns bringing to a phase 1 test MSE of 0.0068. This may be due to the fact that the convolutional layer can extract small time window features, while LSTM can work on bigger sequences: by combining these two components and interleaving them, the result is a both narrow and wide understanding of the input sequence.

Experiments with the starting learning rate, learning rate reduction on a plateau and early stopping parameters have been performed to improve the performances of each model, as well as modifications of the number of units of each layer. As it is explained in Section III-B, many experiments about stride have been performed, resulting in 5 as the best trade-off between performance and training time.

In addition, various configurations by adding fully connected layers at the end of the models were tested to grasp intricate patterns within the sequential data. However, in all of these experiments, it has been observed an increasing behaviour in the validation loss values during the training phase. This is a symptom of overfitting: since no regularization technique was effective, fully connected layers were discarded for the subsequent steps.

E. Class-based models

Another approach was to exploit the information provided by the categories and deploy one model for each of them: in fact, each sequence was associated with its category in both the local and remote datasets. Hence, six models were trained, each one using only the time series of its category. Then, in the prediction phase, it was necessary to just choose the right model according to the input sequence category and apply it to perform the prediction.

Similar techniques to the ones described above (Subsection III-D) were applied to create the models, hence convolutional and LSTM layers, taking particular care not to overfit, due to the low amount of data. This was particularly evident in the case of class F, which indeed resulted in the simplest model.

This approach resulted in pretty poor performances. The validation loss was seldom able to reach the values of the models trained on all classes and this was evident also in the remote dataset of phase 1, where at most 0.0082 of MSE was reached. This may be an index of the fact that the various time series have some connections with each other, hence a model trained on all data can perform better than six distinct models. Another explanation for this behaviour is to be found in the low amount of data on which the single models are trained in this case.

A possible evolution of this approach would have been to train a model on all categories and then fine-tune it on a single one to produce again six models. Anyway, due to the unsuccessful results of the previous method, this track was not implemented.

F. Self-attention mechanism

To enhance the model's capabilities to discern and prioritize relevant temporal information within the sequential data, the self-attention mechanism was employed. This mechanism enables the model to dynamically emphasize relevant elements within a time series, allowing each time step to consider the importance of others in the context of the entire sequence. Its implementation involves transforming the input sequence through convolutional layers to generate attention scores, which are the unprocessed values that indicate the relevance of each element in a sequence concerning the others. These scores are then normalized using a softmax activation to obtain attention weights. After that, the weighted inputs are derived by an element-wise multiplication of the original inputs with the attention weights. By integrating this mechanism with the models described before (III-D, III-E) the model achieved a MSE of 0.0051 and a test mean absolute error (MAE) of 0.0506 on the phase 1 dataset. Being the best result, attention was the core point of the final model.

IV. Final Model

Based on the results outlined in Section III, the final model was derived. Omitting the utilization of data augmentation techniques, as they did not yield performance improvements, a core role has been given to the self-attention mechanism. The final model architecture comprises 1D convolutional layers and two Bidirectional LSTM layers, each housing 64 units. The inclusion of the self-attention mechanism empowered the model to dynamically prioritize essential elements within the sequence, playing a significant role in its improved performance. The structure of the model is also shown in the table below. The model was trained with a batch size of 1024 samples, mean squared error as the loss function and employing the Adam optimizer, with a starting learning rate of 0.001. The learning rate was dynamically adjusted throughout training, reducing it when plateaus (patient of 10) were encountered. Early stopping with patient 12 helped avoid overfitting. The training procedure is shown in Figure 2.

Layer (type)	Output Shape	Param. #
Input	(None,200 , 1)	0
Conv1D	(None,200 , 1)	128
BiLSTM	(None, 200,128)	49664
BiLSTM	(None, 200,128)	98816
Conv1D	(None,200 , 1)	129
Conv1D	(None,200 , 1)	2
Flatten	(None,200)	0
Activation	(None,200 , 1)	0
Reshape	(None,200 , 1)	0
Multiply	(None,200 , 128)	0
Output	(None,200 , 1)	385
Cropping	(None, 18 , 1)	0

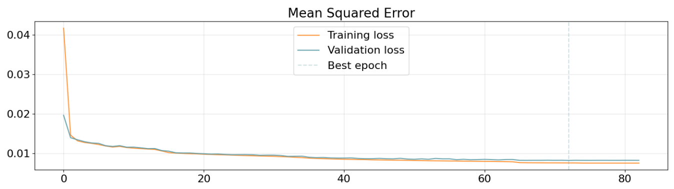

Fig. 2. Training loss (orange) and validation loss (blue) during the training of the final model.

This model has shown superior performance compared to the other attempts, exhibiting a MSE of 0.01069675 and a MAE of 0.07432986 on the phase 2 remote dataset (on phase 1: 0.0.00537269 and 0.05185162).

V. Conclusions

In this work, a model has been proposed to perform forecasting on various time series. Several approaches and models were tested, but the best configuration showed to be the combination of convolution, LTSM and self-attention. This led to a test MSE of 0.0107 and a test MAE of 0.0743 on phase 2 of the competition.

VI. Contributions

We worked on the project in a pretty parallel way, with each team member taking the initiative to try novel tracks and implementations. At the end of every iteration, each one shared its results, issues and doubts with the others to find together the best following steps. All members equally contributed to the work.