Long Short-Term Memory and Attention Mechanism for Time Series Forecasting

Gianluca Canzii, Leonardo Cesani, Matteo Colella, Leonardo Spinello
gianluca.canzii@mail.polimi.it, leonardo.cesani@mail.polimi.it, matteo1.colella@mail.polimi.it, leonardo.spinello@mail.polimi.it

***Abstract*—This report aims to show the work of the group *Backbropagation* carried on to provide a solution for a forecasting problem for time series coming from different categories. The proposed forecasting model uses a double attention mechanism and LSTM blocks to provide a prediction for 18 time steps in the future.**

I. Introduction to the Problem and First Analysis

The project, developed under the second homework provided by the "Artificial Neural Networks and Deep Learning" course offered by Politecnico di Milano, aims to build a forecasting model for different time series labelled with capital letters, as show in Fig. 1. The provided dataset is composed of 48000 labelled time series.

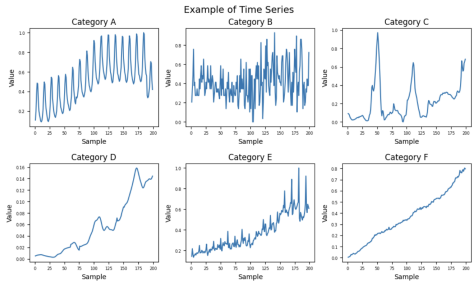

Fig. 1. Examples of the time series in the dataset

A. Dataset Analysis

In order to train the model it was necessary to split the dataset in subsequences of a suitable dimension for the input shape of the net. Indeed, our objective was to get sequences of 209 elements in the first phase; later, we decided to keep them also for the second phase, since we used *autoregression* to predict 18 values in the future as shown in Section II-E2. We exploited the information contained in the dataset concerning the *valid periods*, i.e. the initial and final indices of the useful values for each one of the 48000 sequences. With a further analysis we realized that there were still subsequent zeros at the extremities, as shown in Fig. 2: we decided to remove them, since we assumed the absence of additional information coming from them.

Further investigating the dataset, we realized that it suffered from a skewed distribution of the categories,

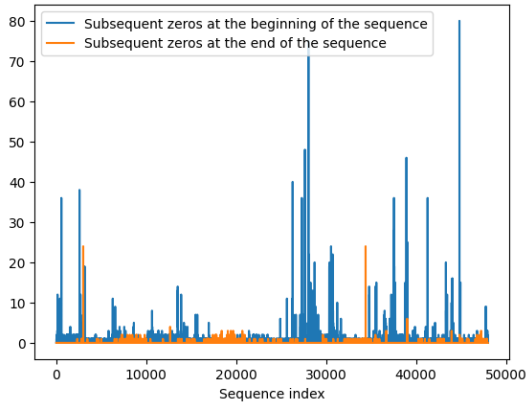

Fig. 2. For each sequence, number of subsequent zeros at extremities

as it can be seen in Fig. 3. This result held also after splitting the sequences in subsequences, since the different categories contain on average series of the same length. This makes it more likely to train a model prone to encourage better predictions for the most represented categories.

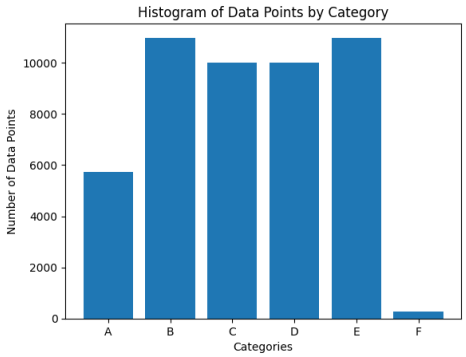

Fig. 3. Number of sequences for each class

B. Padding or cropping?

We could face the problem of dividing the sequences in subsequences in two ways: we could remove the ones shorter than 209 (our *target length*), throwing away also the excessively short pieces remaining after the crop, or impose a *threshold length* and discard only the series shorter than this value, padding with zeros the missing points. Our choice was based partly on analysis

of the dataset: we plotted the amount of lost points with respect to the chosen *threshold length* under which we take away the series and the remaining parts, as shown in Fig. 4. From it we can see that choosing to crop the sequences to multiples of 209, we renounce to about 25% of the points. As concluded later with the results during inference, the trade-off between the number of (padded) subsequences and the density of useful information throughout the whole sequence is in favor of the latter.

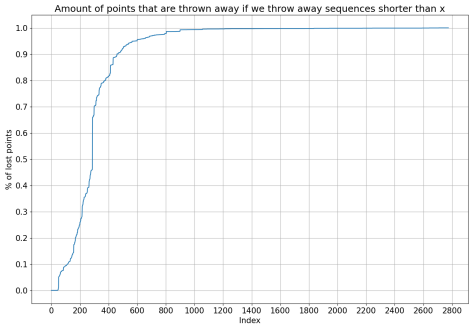

Fig. 4. For each *threshold length* under which we discard the subsequences, percentage of lost information

II. The Forecasting Model

In this section we present the model we have built to solve the task.

A. Model

The model under consideration seeks to emulate the DA-RNN architecture introduced by Qin et al. in 2017 [3]. This architecture proposes a dual-stage attention mechanism, enabling the network to focus on specific features in the time series data. In the DA-RNN framework, an encoder-decoder paradigm is employed: the encoder handles input time series data and contextual information, while the decoder utilizes the acquired representations for making predictions.

B. Attention

The first architecture we have built uses as attention mechanism the dot product between the input of the network and the output of the encoder, represented by an LSTM layer, proposed by Hochreiter et al. in 1997 [1].
This approach resulted not to be very effective probably due to its simplicity, forcing us to explore more advanced attention mechanisms:

1) Squeeze and Excitation (SE) mechanism: inspired by Hu et al.'s work in 2017 [2], this mechanism aims to address channel-level importance discrepancies within the feature representation. It consists of two modules—'squeeze' and 'excitation.' The 'squeeze' module condenses spatial dimensions via average pooling to a single value, while the 'excitation' module employs a bottleneck structure to determine channel-wise scaling weights, incorporating non-linearities to manage complexity. The output passes through a sigmoid activation, scaling the values based on channel importance.
2) Convolutional Block Attention Module (CBAM): in this attention block, proposed by Woo et al. in 2018 [4], the *channel* attention highlights varying channel importance by leveraging global max and average pooling separately, followed by a shared bottleneck network to merge feature vectors. The resulting output is passed through a sigmoid layer and directly influences the module's input. Conversely, the *spatial* attention focuses on informative regions, applying pooling operations across channels, followed by a convolutional layer to identify spatially relevant areas. The output, also passed through a sigmoid activation, directly affects the previous module's output.

From this moment forward, we worked on crafting two networks: a DA-RNN featuring SE blocks and another DA-RNN with the CBAM blocks, given their closely matched validation mean squared error on the validation set.

C. Category Information Input

To enhance our models' performance, we chose to include an input that includes also category information. Allowing the network to handle class information, we first encoded the category label using one-hot encoding before passing it to the network.
The information about the class was then concatenated with the output of the first LSTM layer. A sketch of our architecture is showed in Fig. 5.

After tuning the hyperparameters of the architecture, such as the number of neurons in the encoder and the decoder, we achieved a MSE on the online evaluator of 0.0046.

D. Ensemble Model

In order to make the predictions of our model more robust we decided to implement an ensemble method with the two architectures. The ensemble model performs a weighted average of the two outputs, based on a parameter that depends on the category and on the performance achieved on each of them during inference:

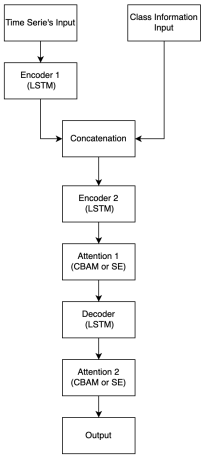

Fig. 5. The block diagram representing our network

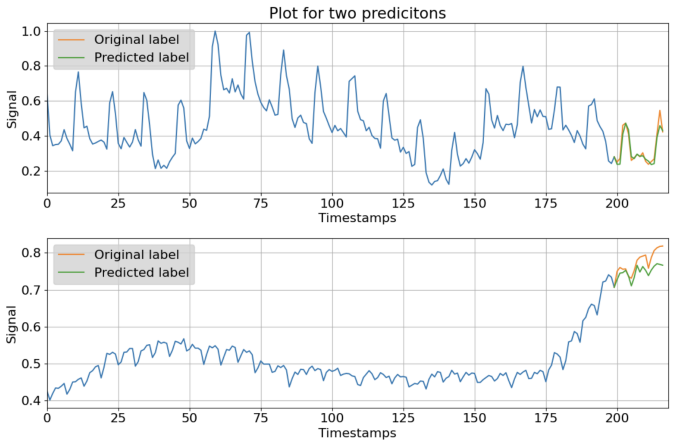

Fig. 6. An example of the prediction the model performs

after hypertuning this value, the overall model achieved a MSE of 0.0045 on the online evaluator.

E. Final phase

In the final phase the length of the output changed from 9 to 18. Hence, we needed to adapt the output shape of the model.

1) Transfer learning: We trained a model exploiting transfer learning, that copied the best performing models and added a final dense layer of 18 neurons. This trial did not work as expected, being able to achieve an MSE of 0.0095. Therefore we made an attempt with autoregression.

2) Autoregression: The final best result has been achieved merging the ensemble outputs of subsequent inferences, appending the first results to the test set, and performing a new prediction to obtain the 9 remaining time steps. Specifically, we managed to get an MSE equal to 0.00746.

3) Fine tuning: We thought that fine tuning the two models for each of the categories could lead to some improvements. Despite our expectations, however, we obtained a worse MSE, probably due to some sort of overfitting on the specific categories.

III. Final Results and Conclusions

A. Results

In the Fig. 6 it is shown an example of the prediction the model is capable of.

B. Future improvements

The best ensemble model we obtained got one of the best results in the competition. However there are still some improvements from which the model could benefit. For example, it is possible to apply a clustering algorithm on the categories to identify similarities that would allow to create models specialized in forecasting future values for groups of different categories. This could avoid the problem of a skewed dataset: indeed, in our case study, generating a forecasting model just for category *F* is not recommended due to the lack of sequences belonging to this class. A less refined approach that could be employed, but that we didn't carry until the end due to the focus on other techniques, is the use of a poorly trained *classifier*. The idea behind that is suggesting the categories that contain sequences that share so many similarities to be able to fool a neural network classifier. A good method to get results for this scope is the confusion matrix, hereafter reported in Fig. 7. It is evident that the category *F* is the most isolated one, which could be a problem since it is the least represented, causing the training of a tailor made regressor for it to be difficult. This method however heavily depends also on the training of the classifier, so it would be better to consider other alternatives.

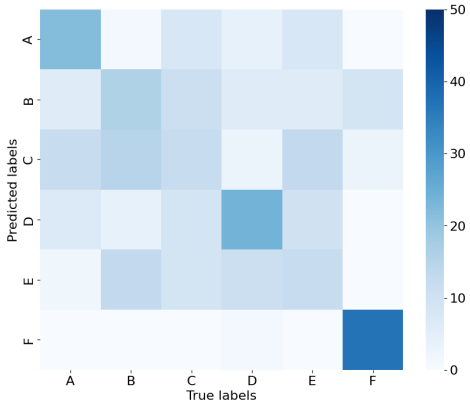

Fig. 7. The confusion matrix obtained inferencing with the classifier

IV. Contributions

Team members' focus summarized here:
Gianluca Canzii: data analysis and preprocessing, autoregression
Leonardo Cesani: data analysis and preprocessing, model implementation, model testing
Matteo Colella: model implementation, hyperparameters tuning, testing
Leonardo Spinello: model implementation, experiments on models

References

[1] Sepp Hochreiter and Jürgen Schmidhuber. "Long short-term memory". In: Neural computation 9.8 (1997), pp. 1735–1780.
[2] Jie Hu et al. "Squeeze-and-Excitation Networks". In: (2019). arXiv: 1709.01507 [cs.CV].
[3] Yao Qin et al. "A Dual-Stage Attention-Based Recurrent Neural Network for Time Series Prediction". In: (Apr. 2017).
[4] Sanghyun Woo et al. "CBAM: Convolutional Block Attention Module". In: (2018). arXiv: 1807.06521 [cs.CV].