POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Executive Summary of the Homework

AN2DL Second Challenge Report

Laurea Magistrale in Computer Science and Engineering

Author: Marco Grazi, Davide Esposito, Gabriele Giusti

Advisor:

Co-advisor:

Academic year: 2023-2024

---

1. Introduction

The second assignment for the Artificial Neural Network and Deep Learning challenge was a forecasting problem given an original dataset of 48000 time series, having different valid intervals specified by a validity array, and each belonging to one of 6 categories. From the first to the second phase, the number of values we had to predict went from 9 to 18, testing the consistency of our model.

2. Data Inspection

In the first data inspection step, we performed a data type conversion to "float32," made sure the data was normalized, and checked for any anomalies, such as nulls and duplicates. Subsequently, time series samples were analyzed, grouped by each category, examining their trends to identify any matches or features associated with the specific category of interest. From an initial analysis based on the direct visualization of time series, no pattern or characteristic associated with the respective category emerged. Furthermore, from quantitative analysis, we observed a significant imbalance in the number of time series in class F compared to the others. These latter two results are the main reasons why we decided not to work on each category separately but instead maintain a single dataset.

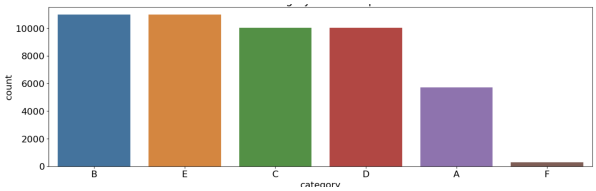

Figure 1: Count of time series for each category

Further analysis was dedicated to the autocorrelation of time series to identify the maximum lag where the data remained significantly correlated. This aimed to extract, in average, useful information on the autocorrelation window. Additionally, an analysis focused on trend and anomaly detection in time series was conducted, yielding no particular results usable in subsequent phases.

3. Data Preprocessing

Having observed that the dataset is made up of time series of different lengths, greater attention was paid to the construction of the sequences for each time series, which were then used to train the network. In particular, sequences with a fixed window size were considered, managing the construction of sequences based on the size of the time series. The reconstruction for time

series with a length shorter than the window size involves adding padding equal to the difference between the window size and the time series length, resulting in a single sequence. However, for all the other time series with a length greater than the window size, padding is added to make them multiples of the window size, plus an additional padding to make them multiples also of the stride. This procedure allows for an optimized construction of sequences, minimizing the added padding and thus maximizing the quality of information passed to the network.
Regarding the window, a size of 200 was used, which corresponds to the length of the sequences in the test set. As for the stride, after a series of trials, we observed that a stride of 5, significantly increasing the number of samples, notably improved performance on the validation set. Following this configuration, the network was trained with approximately 500,000 samples.

4. Model investigation and performance discussion

During the span of this challenge we tried and compared many different types of models, in a progressive way for the most part, starting from a Dense baseline used as a scale-setter for our judgement of the results.

LSTM vs GRU: the initial design of our model used LSTM recurrent units, since we recognized that a 200 time steps long sequence would be difficult to process for a SimpleRNN. Certainly the last values of the input sequence was extremely important for coherent forecasting, if for nothing else just to ensure continuity, but we thought that information on the behaviour in the past could be valuable to correctly predict that same behaviour in the future. As noted in one of the AN2DL course's slide, more than 100 steps long sequences are hard even for LSTM, and our results showed that the simpler GRU cell was actually better performing. Further tests brought us to discover Bidirectional added to the performance of GRU cells, and that the latter maintained similar results when using more than 16 units.

CONV1D: as seen in the Exercise session of the AN2DL course, the use of Conv1D layers was instrumental in extracting more general features that could then better inform our predictions. We tried all-Conv1D models, with some success, even using Residual blocks structure, but in the end we found that a combination of Bidirectional GRU and Conv1D layers worked best. As for kernel sizes we hypothesized that the convolution could do a better job at identifying good features if it had a large enough receptive field on the sequence values. Too small of a kernel might on the contrary hinder the model capability to recognize larger scope behaviours. We thus tested different kernel sizes, also based on the autocorrelation results we got during data inspection. In the end a kernel size of 15 yielded the best results.

CATEGORIES: We noticed that in the submission *model.py* file the *predict()* method had among its input parameters a categories vector, that classified each input sequence. We thought this was useful information and devised two approaches: - using 6 models each specialized on one category of time series, using the categories input to sort the input sequence to the right model.
- create one model, taking both the time series and the category as inputs.
After a few iterations of each approach, however, we did not get the expected results and thus abandoned this idea.

ATTENTION and TRANSFORMERS: as a final experimentation we tried using attention mechanisms at first, and then implementing a transformer. This proved to be kind of difficult, not so much in the actual code but in understanding exactly how the inputs, outputs and the chosen hyperparameters interacted. The Attention mechanism was implemented as a custom layer at first, following the Bahdanau version explained in the slides, and then outsourced to the keras function *Attention()*. To implement Transformers we found an implementation tailored specifically for time series which we were able to work with with just little adjustments. Unfortunately, As for the categories part, we were able to achieve good but not exceptional performances, and in the end the GRU-Conv1D combination remained

our best performing option.

5. Final setup and Autoregressive Forecasting

Our final setup used the bidirectional-GRU/Conv1D combination model design that achieved good performance on 9 steps predictions, reaching an MSE on test set of 0.00473.

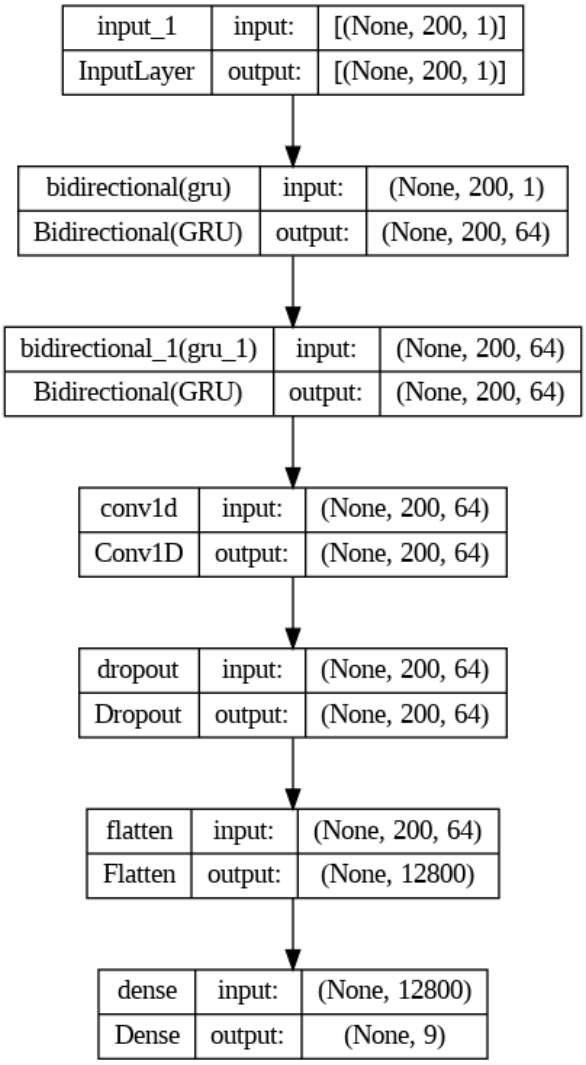

Figure 2: Final model configuration

Figure 3: training and validation loss of the final setup

For the second phase of the challenge, to predict the next 18 steps (instead of the 9 in the first phase), we implemented autoregressive forecasting using the best model previously identified. We thus used our model on the original input to produce the first 9 time steps, and then concatenating them with the last 191 steps of the input sequence. The resulting new input was then fed to our model to produce the next and final 9 steps. This proved to be more effective than training a new 18-step forecast model, obtaining an MSE on the final test set of 0.00902.

6. Conclusions

In comparison to the first task of the challenge, we have become more aware of how data preprocessing, and thus the quality of data used for training, affects the final performance in relation to the complexity of the implemented architecture.

7. Useful Material

[1] AN2DL lecture slides and recordings
[2] Keras Documentation sites: link
[3] Time Series Data Visualization: link
[4] Forecasting: Principles and Practice: link
[5] Tips for training RNNs: link
[6] SeqtoSeq and Attention: link
[7] Attention applied to time series: link
[8] Transformer for time series: link

8. Contributions

- Gabriele Giusti: data inspection.
- Davide Esposito: data preprocessing and testing of different model configurations.
- Marco Grazi: testing of different architecture and autoregression forecasting.

In general, we all shared code and models as soon as they were thought to be good performing and frequently calling each other to share, compare, discuss and coordinate our findings and tasks.