Artificial Neural Network and Deep Learning Homework 2

Paolo Piccirilli, Marco Riva, Federico Sarrocco, Alessandro Vacca

December 22, 2023

1 Introduction

The second homework of the course was about time series forecasting: the challenge was to build a model that predicts future values based on past observations. using a dataset composed of different time series observation. The key aspect of this homework was the fact that the dataset was composed of many series with specific characteristics for each one, therefore the main goal was to find a model able to properly generalize. We begun by analyzing the provided dataset, which consisted of 48.000 univariate time series. The time series were accompanied by a category of belonging (six in total) and a validity window. We observed that very different series could belong to the same category.

2 Development

We began by questioning the utility of creating an additional model for identifying the type of time series, as the provided categories seemed not useful for training. Instead, we focused on constructing models without considering the original categorization.

To standardize the length of all series, we fixed the window size at 200 data points. While many series were shorter, removing them significantly reduced the dataset. Therefore, we applied padding using 'edge' mode to extend shorter series to the chosen window size.

2.1 Data Processing

We partitioned the time series into distinct sets for training, validation, and testing. Within each series, we isolated both input and target sequences, with the former comprising 200 data points and the latter consisting of 18 points. Our decision to extract a single sequence from each time series stemmed from observing that nearly 85% of valid data exhibited lengths below 300 data points. The data values were uniformly distributed across the [0,1] range, obviating the need for further preprocessing steps.

2.1.1 Augmentation process

To expand our training dataset, we implemented augmentation techniques exclusively on the training set. Our augmentation approach involved random alterations, including:

1. Addition of random noise within the range of [-0.05, 0.05].
2. Application of scale factors to the time series, varying between [0.8, 1.2].
3. Introduction of random offsets ranging from [-0.2, 0.2].

However, we observed that the application of data augmentation did not universally enhance model performance; hence, we selectively applied it based on its effectiveness for each model.

2.2 Models

We concurrently developed various models, each optimized using the MSE or MAE loss function. Notably, we observed a slight enhancement in performance when employing the MAE metric.

Additionally, we experimented with augmenting the models by incorporating the one-hot vector of categories as supplementary input. However, this augmentation did not consistently improve performance across all models, leading us to selectively apply it to specific models.

Furthermore, certain models integrated a straightforward sample normalization strategy, as proposed by [3]. This method involved subtracting the sequence's final value before encoding and subsequently restoring it after decoding.

To fine-tune our models, we conducted hyperparameter searches using the Optuna framework [1] across the entire model set, aiming to identify the optimal hyperparameter configurations.

2.2.1 LSTM model

Our initial experiment involved a simple model comprising only two layers—an LSTM layer and a dense output layer. No additional data processing beyond what was essential for network evaluation was applied to the training data, as described in the previous section. Despite its simplicity, this model delivered outstanding results and stands as our best-performing single model in the challenge. Subsequent attempts at data augmentation or alternative data splitting techniques for training yielded inferior performance compared to the original attempt.

2.2.2 RESNET model

Utilizing residual blocks with skip connections, this model adopts a Residual Network architecture, employing 1-dimensional convolution instead of 2-dimensional from [2] to accommodate the data's dimensionality. Although stacking five residual blocks yielded the best performance, surprisingly, simpler architectures consistently outperformed more complex ones.

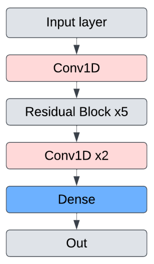

Figure 1: RESNET model structure

2.2.3 LSTM Encoder-Decoder with skip connections model

This model structure combines bidirectional LSTM layers with skip connections, allowing the model to reconstruct sequences by integrating insights from input and processed sequences before making predictions.

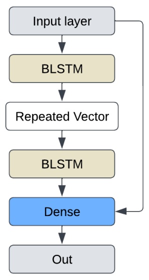

Figure 2: Encoder-Decoder model with skip c.

2.2.4 Attention model

The model architecture comprises bidirectional LSTM layers. An attention mechanism is integrated to prioritize relevant segments within input and LSTM-processed sequences. This mechanism assigns varying weights to different parts of sequences, emphasizing key information. The model also employs skip connections, merging insights from these sequences before predicting output. It concludes with dense and output layers, compiled using mean squared error.

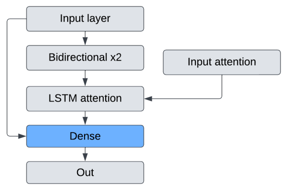

Figure 3: Attention model structure

2.2.5 Convolutional model

This model begins with convolutional layers for feature extraction, followed by bidirectional LSTM units for sequence understanding. Skip connections merge insights from input and processed sequences. The output layer predicts sequences, incorporating the last input value.

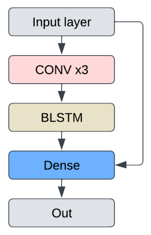

Figure 4: Convolutional model with skip connections

2.2.6 Autoregressive models

We experimented with all developed models in autoregressive mode. However, this approach generally yielded worse performances, primarily due to divergence, as referenced in Figure 5.

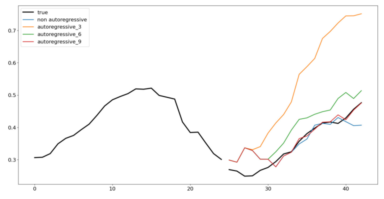

Figure 5: Comparison between autoregressive and non-autoregressive models

3 Ensemble models

We explored ensemble methodologies to harness the strengths of multiple models, in order to improve the performances of the best performing simple model that reaches an MSE of 0.0096.

3.1 Simple averaging

Pooling predictions from various models through a straightforward averaging technique yielded promising results. Despite achieving an MSE of 0.0094, this approach didn't maximize performance potential.

3.2 Autoregressive ensemble

Expanding our ensemble strategy, we incorporated autoregressive versions of each model, spanning different window lengths (3, 6, and 9). While individual autoregressive models showed degraded performance, combining them in an ensemble outperformed simple averaging, achieving an MSE of 0.0091.

3.3 Weighted average

Our exploration delved deeper into optimizing ensemble techniques by deriving model weights from the local test set values. Using the softmin function, we weighted the predictions from all models, including their autoregressive counterparts. This meticulous approach ended up in our most successful ensemble method, registering an MSE of 0.0090.

3.4 Meta-Learning

We experimented with a neural network approach for meta-learning, seeking to learn optimal combinations from the models. Unfortunately, this technique underperformed with an MSE of 0.01.

4 Final results and conclusions

Our ensemble techniques demonstrate the power of combining diverse models for enhanced accuracy. Notably, simpler individual models consistently outperformed complex ones, emphasizing simplicity's advantage in accurate time series prediction as also stated in [4]. Conversely, complex models lacked comparative predictive capacity, affirming the strength of simplicity within our ensemble.
In conclusion, our iterative approach and ensemble strategies significantly improved time series prediction, highlighting both the effectiveness of ensemble methods and the superiority of simpler models in this domain.

5 Contributions

After writing the functions for splitting, augmentation and sequence building we decided to work individually on different models, aiming at optimizing each of them, with the goal of ensembling them later. Tasks have been carried out as indicated in the following list:

- Models development: Federico, Alessandro, Paolo, Marco
- Complementary coding: Paolo, Marco
- Ensemble: Federico, Alessandro
- Report: Federico, Alessandro, Paolo, Marco

References

[1] Takuya Akiba et al. "Optuna: A Next-generation Hyperparameter Optimization Framework". In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.* 2019.

[2] Kaiming He et al. *Deep Residual Learning for Image Recognition.* 2015. arXiv: `1512.03385 [cs.CV]`.

[3] Shengsheng Lin et al. *SegRNN: Segment Recurrent Neural Network for Long-Term Time Series Forecasting.* 2023. arXiv: `2308.11200 [cs.LG]`.

[4] Riccardo Ughi, Eugenio Lomurno, and Matteo Matteucci. *Two Steps Forward and One Behind: Rethinking Time Series Forecasting with Deep Learning.* Apr. 2023.