ANNDL II Challenge:
Time Series Forecasting Report
TheCEOs

Balzarini Filippo - 10719101, Benelle Francesco - 10727489, Cavicchioli Michele - 10706553

December, 2023

Abstract

This report investigates diverse deep learning models, including concatenation of LSTMs, Autoencoders, ResNet-LSTM hybrid, sequence to sequence with Luong Attention, and Wavenet for forecasting future steps in a single-feature general time series. Through comprehensive testing and optimization, we evaluate the performance of each model on the challenging forecasting task. Results highlight the unique contributions of individual architectures. This study not only advances our understanding of deep learning applications in time series forecasting but also provides practical insights into the strengths of specific models.

1. Introduction

In this report, we detail the intricacies encountered in addressing the second challenge of the Artificial Neural Network & Deep Learning course. Our project focused on the development of a versatile neural network tasked with forecasting a single feature in a generic time series, irrespective of the source type supplied to the network.

2. Operations on Dataset

2.1. Dataset Inspection

The dataset consists of time series data sourced from six different origins. Each dataset entry comprises the time series itself, the corresponding validity interval, and a label specifying the source of that particular interval.

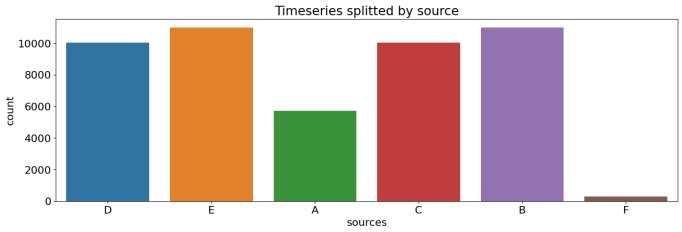

Figure 1: Time series group by source

Length	Count	Percentage
$(0, 50)$	2017	04.20%
$[50, 100)$	13378	27.87%
$[100, 150)$	3132	06.53%
$[150, 200)$	6416	13.37%
$[200, 2776]$	23057	48.04%

Table 1: Distribution of time series by their length

2.2. Data Processing

To effectively train our neural network, meticulous preprocessing of the dataset was imperative, involving careful considerations for the proper splitting based on the optimal configuration of the *window size*, *telescope*, and *stride*.
In determining the *window size*, a value of 200 was chosen, aligning with the input size of the test set. A padding strategy has been implemented to accommodate the fact that a significant proportion (51.96%) of the provided time series had lengths less than 200. All such time series were padded to meet the window size requirement. Subsequently, the dataset was partitioned using intervals dictated by the chosen *stride*, ensuring the retention of the last y-steps as specified in the selected *telescope*. This meticulous preprocessing allowed for a harmonized and standardized input structure for the neural network training process. The final parameters chosen are:
- *window size*: 200
- *stride*: 10
- *telescope*: 18

2.3. Data Normalization Strategies

Given that the initial dataset had undergone normalization procedures, we explored alternative normalization techniques beyond the conventional scaling between 0 and 1. Our experimentation involved the application of the *RobustScaler*[5], designed to enhance resilience against outliers. Additionally, we investigated an alternative normalization approach, scaling the data between -1 and 1. However, our analysis yielded no conclusive evidence supporting the utility of this specific normalization scheme for our time series forecasting problem.
Our exploration of diverse normalization methods showed the importance of enhancing model robustness and performance.

2.4. Data Augmentation Techniques

During the first days of the challenge, few types of Data Augmentation were tried, such as Gaussian Noise and Jittering, but since the performance of the models seemed to be worse, in the end, we chose to avoid train-time Data Augmentation.

3. Models

In this section, we outline the models we built and evaluate their performance using the MAE and MSE metrics. To accommodate the differing forecasting requirements of the two challenge phases:

1. Some models directly predict 18 steps, with output adjustments in *model.py* for the initial phase. During the *Developement Phase*, the outputs of the predict function of these models were cropped in order to predict only the first 9 steps.
2. Other models were built to predict 9 samples at a time. To use them even in the *Final Phase* we modified the predict function with an *autoregressive* approach to predict the 18 steps in 2 blocks of 9 samples each.

We noticed that the best results obtained was given by networks that directly predicted 18 steps.

3.1. Stacked LSTMs model

One of the fundamental models employed in our time series forecasting study involves the concatenation of Long Short-Term Memory (LSTM) networks. This model leverages both a bidirectional LSTM layer to enhance the comprehension of the input sequence and a single LSTM layer to capture crucial features inherent in the time series.
We incorporated a dropout layer and batch normalization layer to fuse these LSTM layers seamlessly. The final layer of the network consists of a Dense layer featuring 18 neurons, utilizing a *linear activation function*.
During the training phase, we applied Robust Scalers, as implemented in the scikit-learn library[5], to ensure the model's robustness and stability.
Employing Robust Scalers during training fortified our neural network, enabling it to proficiently handle time series data that had not undergone normalization with Robust Scalers. Remarkably, the model exhibited resilience and adeptness in processing time series data normalized within the conventional range of 0 to 1, demonstrating its versatility and robust generalization capabilities.

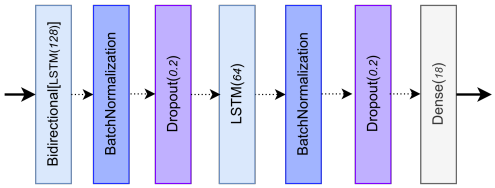

Figure 2: Concatenated LSTMs model

3.2. Convolutional LSTM

This was one of the first and also one of the best performance-wise models we developed. Such architecture was trained two times with slightly different training sets: in the first case the training set included every non-empty window, in the latter case we filtered the training set to include only windows that were at least half-full, therefore, from a window size of 200 we included only windows with at least 100 time steps. The comparison of these two models showed that this architecture produces better results when trained with windows at least half-full. We also tried to manipulate the number of samples required for a window to be in the training set; such final comparison determined that 100 was the optimal number.

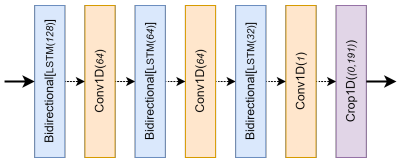

Figure 3: Convolutional LSTM model

3.3. Autoencoder with LSTM

Since the power of sequence to sequence architecture, we tried to use was an *Autoencoder* adapted to use *LSTMs*. The concept behind this model is the same one behind a usual purely convolutional autoenconder; compress the input in a latent dimension through an encoder, and use a decoder to decompress such compressed information to produce the output. The adaptation comes in the use of LSTMs, which enable the model to *remember* past values and so to capture features in time. With this model we also used *weight decay* [2] to prevent overfitting.

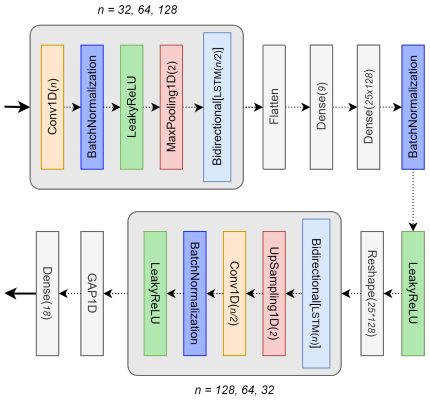

Figure 4: Autoencoder with LSTM model

3.4. Resnet with LSTM

Another type of Architecture that we tried to implement was a ResNet-like structure [1] made Conv1D [6] Layers in the Residual Blocks instead of the more classical Conv2D Layers. This model differs from the original ResNet also

because of the presence of two Bidirectional LSTM Layers, one right after the Input Layer, one right before the Output one. Though this model was one of the most complex between the models that we tried up to that moment, it was also the one that gave us the most scarce results both during training and testing.

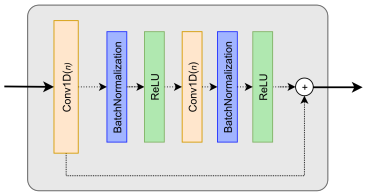

Figure 5: ResNet's Residual Block

3.5. Sequence to Sequence with Luong Attention

Taking inspiration from autoencoders' architectures, we attempted to enhance performance by incorporating the *Luong Attention Mechanism*[3]. This attention mechanism involved computing alignment scores a between the source s and the target t (encoded and decoded hidden states):

$$a_t(s) = \text{softmax}(h_t^T h_s)$$

The context vector **c** was then computed as the weighted average of the source states using attention scores, and it was concatenated with the source state before being fed into a fully connected layer. Despite these efforts, the network proved to be suboptimal for forecasting, a conclusion supported by findings in other studies [7].

3.6. WaveNet

Regardless of the success of WaveNet in generative raw audio tasks (Oord et al., 2016) [4], our attempt to adapt it for time series forecasting during the initial competition phase produced results below our expectations (*MSE: 0.008110, MAE: 0.060144*). This led to a pivot towards alternative models.

3.7. Results comparison

In this section, we will show the comparisons of the training and the results of our best networks, respectively, in terms of a graphical view and the obtained MSE and MAE in the submissions.

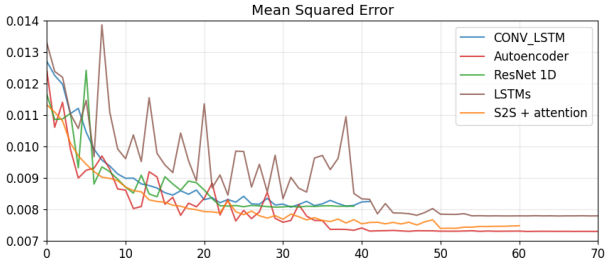

Figure 6: Training MSE comparison

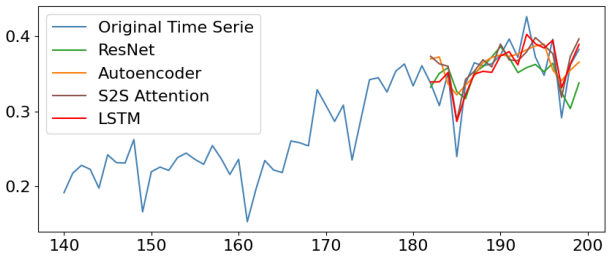

Figure 7: Predictions comparison

Model	MSE	MAE
Concatenated LSTMs	**0.00917**	**0.06876**
AutoEncoder	0.01048	0.07057
Convolutional LSTMs	0.01069	0.06895
Seq2Seq Attention	0.01113	0.07495

Table 2: Results obtained with the test on 18 future predictions

3.8. Hyperparameters Tuning

- *Batch Size*: Noticing that with smaller batch sizes the training went smoother and the overfitting was reduced despite the training time, in the end, we opted to keep the batch size to 256.
- *Patience*: Given that the training time was high and some of the models tended to overfit after a few epochs we opted to set a patience of 10 epochs, while saving the model's training checkpoints and reducing the learning rate, as explained in the following section.
- *Learning Rate*: Starting from a Learning Rate of 0.001, we choose to reduce it with patience of 8 epochs by a factor of 0.2 down to a minimum of $1e-5$.
- *Total number of Epochs*: the total number of epochs was 150, despite that the final epoch was never reached.
- Other *Hyperparameters* such as the Dropout Rate or the number of Filters are specified in every model above this section (3).

4. Conclusions

After having evaluated many different Architectures for time series forecasting, we can conclude that stacked LSTMs models are the best suited for this kind of task. Thanks to this challenge we had advanced our understanding in Deep Learning models for time series forecasting, being able to get practical insights into strengths and weaknesses of each model, having analyzed such a variety of them, from LSTMs to Sequential Models, including Sequence to Sequence models and Attention mechanisms.

5. Contributions

All the members of the team contributed equally to the project. Here is a list of the features that each member contributed the most to:

- **Filippo Balzarini - filomba**: Testing new kinds of models, Hyperparameters and Robust Scalers.
- **Francesco Benelle - Benels**: Analysis of the Dataset, Testing new kinds of models and Hyperparameters optimization.
- **Michele Cavicchioli - el_miki**: Analysis of the dataset, building of the windows, testing of new models and Hyperparameters optimization.

References

[1]H. Choi, S. Ryu, and H. Kim, ≪Short-term load forecasting based on resnet and lstm≫, in 2018 ieee international conference on communications, control, and computing technologies for smart grids (smartgridcomm) (2018), pp. 1–6, 10.1109/SmartGridComm.2018.8587554.

[2]I. Loshchilov and F. Hutter, *Decoupled weight decay regularization*, 2019.

[3]M.-T. Luong, H. Pham, and C. D. Manning, *Effective approaches to attention-based neural machine translation*, 2015.

[4]A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, *Wavenet: a generative model for raw audio*, 2016.

[5]F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, ≪Scikit-learn: machine learning in Python≫, Journal of Machine Learning Research **12**, 2825–2830 (2011).

[6]M. Zabihi, A. B. Rad, S. Kiranyaz, S. Särkkä, and M. Gabbouj, *1d convolutional neural network models for sleep arousal detection*, 2019.

[7]A. Zeng, M. Chen, L. Zhang, and Q. Xu, *Are transformers effective for time series forecasting?*, 2022.