Artificial Neural Network & Deep Learning

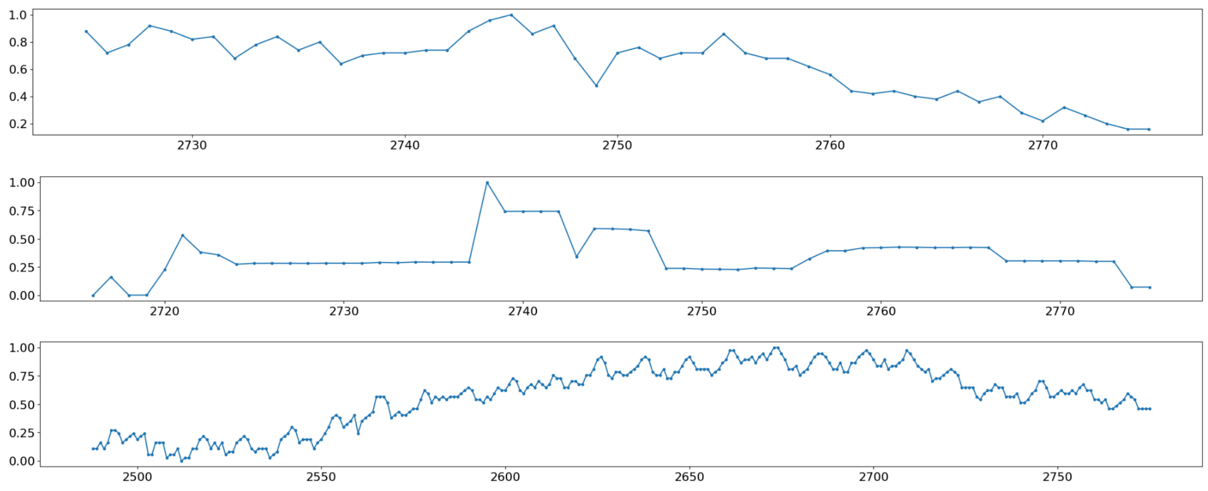
Time Series Forecasting Homework - Second Challenge

Introduction

The task was to predict future samples of input time series using neural networks capable of generalizing across different temporal contexts. The goal of the challenge was to predict 18 steps ahead of 60 univariate time series provided as input. The provided dataset consists of 48,000 time series, each padded to 2,776 for portability. Two additional files were provided: the first one containing the start and end of the validity period for each time series, and the second one containing the category to which each time series belongs. The categories (labeled A, B, C, D, E, F) correspond to different contexts. The time series belonging to each category are not to be understood as related to each other, but only as collected from similar data sources.

Data Inspection

After loading the dataset, we randomly plotted some time series to understand their nature, that is, to analyze the presence of correlation between signals belonging to the same category. As shown in the figure below, even within the same category, time series exhibit significant variations in both seasonality and trend patterns.



Work environment set-up

After inspecting and understanding the data, we proceeded to build a training environment for the networks. We set a random seed to ensure reproducibility and then divided the data sets in three parts: training, validation and test sets. The division was as follows: 80% for training, 10% for validation, and 10% for testing. We took care to preserve the distribution between the sets, ensuring an equal distribution of categories.

We then proceeded to split the dataset and normalize the length of the sequences in the training, validation, and test sets. We chose a window size of 200 for the input sequences. The target sequences (the telescope) varied in length from 1 to 9 (18 for the second phase) with multiples of 3. We sampled the series with a sampling window of ‘window+telescope’ (and also with a specific stride, typically set to 20). The subsequence covered in the telescope was extracted and used as the target for the prediction. Sampling was done only on the valid period of the series. Padding was applied only to subsequences with a minimum number of valid elements.

Architectures

Baseline

As a baseline for comparing the performance of our models, we created two simple models: the persistence model, which repeats the last *n* samples, and the constant model, which repeats the last sample *n* times.

A simple hand-crafted model was then used as the baseline architecture for some experiments, consisting of an LSTM followed by three convolutional layers ending with a final global average pooling. This resulted in a mean square error of 0.00705 on the test set. To improve performance, we replaced the LSTM layer with a bidirectional layer, doubling the number of parameters. This change resulted in slightly improved performance, with a mean square error of 0.00655. An alternative approach was to use the GRU layer instead of the bidirectional LSTM. The results showed that the performance was almost identical, with a result of 0.00651, but with fewer parameters. These experiments highlight the importance of exploring different recurrent layer architectures for the problem at hand. Despite having fewer parameters, the GRU layer performed almost as well as the bidirectional LSTM layer. This indicates that it efficiently captured the temporal dependencies in the time series data. Therefore, the additional complexity introduced by the Bidirectional layer may not be necessary for this specific task, and the GRU's ability to effectively model sequential dependencies was sufficient.

Model	**MSE**
Baseline - LSTM and 3 convolutional layers with final GAP	0.00705
BiLSTM and 3 convolutional layers with final GAP	0.00655
GRU and 3 convolutional layers with final GAP	0.00651

We conducted experiments to identify an ensemble model that surpassed our existing models. We began by averaging the predictions of the three previously described models, which resulted in enhanced performance compared to each individual model. The test set achieved an MSE of 0.00638. We then explored more complex ensemble models, using 10 networks trained on one or more categories. Again, averaging the predictions yielded the best results, while training the ensemble using transfer learning and adding a final dense layer produced the worst performance, as demonstrated in the table below.

Ensemble Models	**MSE**
Ensemble of the 3 initial networks (LSTM, BiLSTM and GRU)	0.00638
Ensemble of 10 networks: the 3 initial ones, an LSTM trained only on some categories (A, B, C) and a network with a GRU and 3 CNV layers for each one of the 6 categories, trained separately, using an average as final combination	0.00628
Ensemble of the 10 networks combining them with a final Dense Layer	0.00663

From these experiments, we can say that averaging predictions from different models can improve the overall predictive accuracy of our models by reducing individual model biases and exploiting complementary strengths. All these values were assessed on the first test set, while on the second phase we submitted only the Ensemble that previously gave us the best results, obtaining 0.0149.

Selection Between Different Models

To leverage the diversity of time series contexts while maintaining predictive power across categories, we developed multiple models with a shared architecture, each trained on a distinct category. These models were constructed upon the foundation of the best-performing model to date. The architecture comprises an input layer, a GRU layer with 128 units, and three convolutional layers, each with 128, 64, and 18 units, respectively. The models were trained for 200 epochs using AdamW optimizer.

During the prediction phase, the models were dynamically selected by a script in the *model.py* file depending on the category to which the time series to be predicted belonged. However, this set of models did not perform well on the test platform. In fact, it performed worse than a model with the same architecture trained on the whole dataset not considering the category attribute. This result suggests that there is low correlation between signals belonging to the same category.

Dilated Causal Convolution

Another architecture we built was based on applying dilated causal convolution with an exponential increasing dilation rate, as done for the wavenet architecture[1]. We first applied a layer of 128 LSTM cells to the input sequence followed by a cascade of six 1D convolutional layers, with causal convolution, 32 filters each and an exponential increasing dilation rate from 1 to 32. We actually built two variants of this network architecture. The first was designed to directly predict the next 9 samples of the series, which was achieved by adding a bottleneck convolutional layer and a global average pooling (GAP) layer. The network was trained with a batch size of 64 and for a maximum of 70 epochs. While training we also implemented early stopping and a reduced learning rate callback. The training results for this network were a 0.0056 and 0.0059 mse for the training and validation set respectively and the evaluation on the test set yielded a mse of 0.00611.

Subsequently, we created an autoregressive variant of the network by modifying the top layer to accommodate autoregressive prediction. Specifically, we removed the GAP layer and added a dense layer with one output neuron, allowing us to predict only the next step in the time series. This second network was trained with the same training parameters as the first one and achieved a MSE of 0.00586 on the development test set. Adapting this network to the second phase of the challenge required simply changing the autoregressive prediction steps from 9 to 18. In this final test set, the network achieved a MSE of 0.0122, more than double the previous MSE on the first test set.

To fine-tune the autoregressive telescope, we tried changing its window. We trained the same model with unchanged parameters on a window of size 3 and then 6. The results of the model on our local test set were very similar, so we submitted the one with a telescope of size 3, which marginally improved the score of the previous model, with a single-element telescope, bringing the MSE down to 0.0112.

Hyperparameter Tuning

One of the best performances on the second test set was achieved by tuning the parameters of a simple model with Keras Tuner. The model was the one consisting of a GRU and three convolutional layers that ended with a GAP. By adjusting the units of the layers, we were able to achieve a result of 0.0119 on the prediction of the future 18 data points. On the same network architecture, we also tried autoregressive prediction using Keras Tuner, but direct prediction worked much better. In conclusion, not only on this network, but also on the other architectures, hyperparameter tuning could further improve the behavior.

Contributions

- **Irfan Cela** (@IrfEazy - 10694934): model ensemble trials, hyperparameter tuning, experiments on different network architectures
- **Alessandro Cogollo** (@Frigobar123 - 10571078): dynamic selection between different models trained on different categories, various testing and tuning on simple networks (GRU + 3 conv + GAP)
- **Fabio Lusha** (@AlexOberstein - 10882532): experiments on dilated causal convolutional networks, fully-CNN networks, gated CNNs, autoregressive models
- **Bianca Christiana Savoiu Marinas** (@BexRedpill - 10684465): experiments on different recurrent layers and on different model ensembles, hyperparameter tuning, experiments on different architectures and causal convolution

References

[1] Aäron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. *Wavenet: A generative model for raw audio.*