Univariate Time Series Forecasting
Artificial Neural Networks and Deep Learning - A.Y. 2023/2024

Luigi Pagani*, Flavia Petruso† and Federico Schermi‡
M.Sc. Computer, Mathematical, and High-Performance Computing Engineering Politecnico di Milano - Milan, Italy
*Email: *luigi2.pagani@mail.polimi.it, †flavia.petruso@mail.polimi.it, ‡federico.schermi@mail.polimi.it*
*Student ID: *10677832, †10544566, ‡10776811*
Codalab Groupname: "RandNet"

1. Introduction and project goal

The provided dataset is composed of 48000 time series, each one associated with a category, ranging from A to F. Each time series has a unique length, but to facilitate data manipulation, sequences have been padded with zeros before their starting time. Consequently, the data are presented in the form of a 48000 by 2776 matrix. Additionally, the dataset includes vectors indicating the starting and ending times for each series. The provided time series had already been scaled (by row) from 0 to 1.

2. Data exploration and preprocessing

The first step we undertook was data inspection. We examined the distribution of time series lengths to assess their variability and the potential imbalance between categories, as depicted in **Figure 1**.

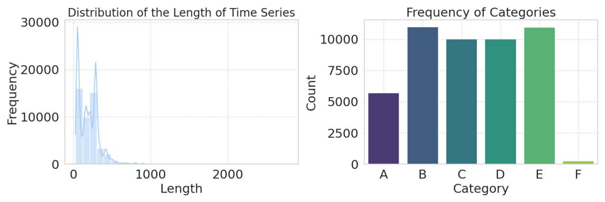

Figure 1. Distribution of the length of time series (left) and frequency of time series by category (right).

It appeared that most of the time series have a length lower than 1000 time points. The figures also highlight an under-representation of class F. In our exploration for potential class differences, we visually inspected numerous time series from diverse categories, as documented in the `dataset_analysis.ipynb` and `further_dataset_exploration.ipynb`.

Additionally, we computed the distribution of values by class and their boxplots. Our analyses unveiled a largely consistent distribution of data across categories, except for category *A*, which demonstrated a slightly higher median than the others. Consequently, after experimenting with several models with and without the inclusion of category labels and observing comparable performances, we opted to proceed without incorporating them.

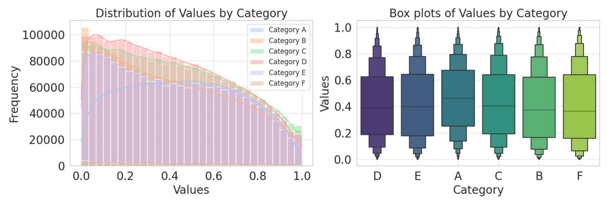

Figure 2. Distribution of time series by categories (left) and associated boxplots (right).

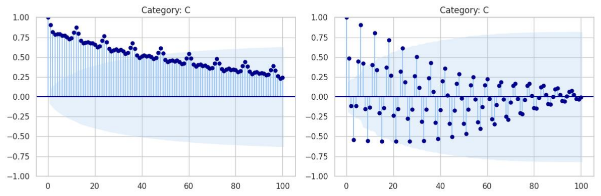

Figure 3. Autocorrelation for two samples from the category *C* over 100 lags. The left figure reveals a clear 12-step (possibly monthly) trend, whereas the right one has a more complex (and possibly weekly-like) trend.

Then, we moved to the investigation of seasonality patterns. To do so, we relied on the **autocorrelation** function, which revealed some seasonal trends, especially every 7 and 12 timestep, possibly revealing daily and monthly data exhibiting weekly and yearly trends, respectively, as also shown in **Figure** 3. We used this information to set the minimum window size during our experimentation always greater or equal to 12, to allow the model to learn its seasonal components.

Finally, since the task required to predict 18 values, we decided to remove the sequences that were shorter than 4*18=72 (n $\approx$ 12000), as we planned on having time series long at least three times the predictions.

Finally, to identify potential outliers computed the interquartile range and removed the data with IQR $>$0.95 or $<$0.05, leading to a final number of 34876 time series.

We split the data into 0.9 training and 0.1 validation sets, using the latter as a test set.

We then constructed sequences for training, selecting a window of 200 and a stride of 6 (12 in the transformer and in some augmentation trials) and introducing padding before sequences shorter than the desired window. The y sequences were set to 18 points, as the required prediction length.

3. Comparison of Neural Network Architectures

To create a baseline model when comparing results, we started by building a vanilla LSTM and then tried more complicated architectures, incorporating seq2seq models with attention mechanisms. The Transformer model, with a multi-head attention mechanism, was also included.

The details of each architecture are described in the homonymous notebooks in the *models* folder. Some additional architectures that have been tried but yielded suboptimal performances are in the folder *Other_experiments_and_models*. Here, we report the main aspects of the best models:

- Vanilla LSTM (128)
- LSTM(256)seq2seq+attention
- LSTM(64)seq2seq+attention+Time2Vec(dim=3)
- CONV1D(32)_LSTM(128)seq2seq+attention
- Transformer+multi-head attention(4)

3.1. Performance Comparison

As can be seen from **Table 1**, the different models yield comparable validation accuracy.

Model	**Validation MAE**
Vanilla LSTM	0.0574
LSTM seq2seq+attention	0.0554
LSTM seq2seq+attention+Time2Vec	0.0564
CONV1D_LSTM seq2seq+attention	0.0561
Transformer+multi-head attention	0.0605

TABLE 1. MEAN ABSOLUTE ERROR OF SELECTED MODELS IN VALIDATION.

3.2. Attention Mechanisms in Sequence Modeling

We introduced attention mechanisms to allow models to focus selectively on important segments of our input sequences [1] and improve the robustness of our predictions. This approach needs to be considered especially for sequences with a high proportion of zeros due to padding, or missing values, like in our dataset.

Furthermore, this technique also allows us to work with longer sequences, whereas traditional LSTM models have poorer predictions in those cases.

Additionally, our Transformer model presents a multi-head attention mechanism, which allows it to concurrently process different positions of a single sequence, providing a more nuanced understanding of complex hidden time patterns [3]. In our transformer model, we selected 4 heads for our multi-head attention mechanism.

3.3. Time2Vec Embedding in Seq2Seq and Transformer Models

The Time2Vec [2] method is based on representing time as a high-dimensional vector and has proven effective in capturing complex temporal patterns in machine learning models.

For its implementation into our seq2seq models, and especially LSTM, we settled for a Time2Vec embedding dimension of 3, based on some early exploration and heuristic considerations. This implementation yielded encouraging performances (see **Table** 1). However, implementing Time2Vec embeddings in transformer models requires a more careful approach. When we tried replacing the standard positional embeddings with Time2Vec, it led to significantly prolonged training times. This suggests that while the concept holds potential, it requires further exploration and more reasoned implementation, particularly in transformer models.

4. Data augmentation

Some of the models were separately run with data augmentation. Namely, we decided to add some jitter with normal distribution and standard deviation chosen as 0.01*standard deviation of each time series (row) of the training data, ultimately triplicating the number of available sequences. We performed this augmentation approach in our CONV1D_LSTM seq2seq model+attention with a decrease of 0.006 MAE, on the LSTM seq2seq model+attention, with an increase of 0.008 MAE, and the LSTM time2vec seq2seq model+attention, with an increase of 0.01 MAE. The complete implementation is in the folder *augmentation_studies*, inside the *models* folder.

4.1. Final model

Based on the test performances, we selected CONV1D LSTM seq2seq+attention as our final model.

Layers
Masking()
(CONV1D(32)X2+BatchNormalization()+MaxPooling1D())x2
LSTM (128, return_sequences, return_state)
RepeatVector(1)
LSTM (128, return_sequences)
Bi-LSTM (128/2, return_sequences)
Attention Mechanism (Dot(axes=[2,2])), activation = softmax
Dropout(0.5)
Context Vector (Dot(axes=[2,1]))
Concatenate()
Flatten()
Dense (18)

TABLE 2. CONV1D LSTM SEQ2SEQ + ATTENTION

5. Shared parameters, metrics, and approaches

- **Loss function**: due to the predictive nature of the task, we used the *mean squared error*, but we use the validation *mean absolute error* as a metric, due to its interpretation as an average percentage error.
- **Optimizer and learning rate**: we used Adam as optimizer, and we set 10^{-3} (10^{-4} in the transformer) as a reasonable standard value, which was then complemented with the callbacks.
- **Callbacks**: Early stopping based on validation loss, with patience 20 (10 in the transformer and augmentation studies) and with reduced learning rate on a plateau with minimum learning rate 10^{-5}, patience 10 (5 in the transformer and augmentation studies), and factor 0.5.

6. Results and performance

The best model on the test set was the CONV1D_LSTM, whose performance on the different datasets is shown in **Table** 3.

Mean absolute error on different datasets	
Train	0.0478
Validation	0.0564
Test	0.0752

TABLE 3. PERFORMANCE OF THE FINAL MODEL

There seems to be some overfitting of the training set, which emerges especially in the test set. A simpler model or stronger regularization techniques might help overcome this limitation.

7. Final considerations and Potential improvements

As can be seen by the comparable validation accuracy of models of different complexities, an important focus of this challenge was in the dataset preprocessing and augmentation. A first, important potential improvement concerns a more robust study on the results of the augmentation, which we explored only partially. Further augmentation techniques could include, for instance, time warping or permutation of small windows of data (without completely breaking the periodicity).

A further improvement concerns the realization of a robust scaling based on the median and the interquartile range, to improve the regularization of the model.

Another aspect would involve investigating more deeply the categories, which we only explored in the early stages of our analysis. We could introduce in the model weights inversely proportional to the categories.

Finally, one aspect we did not explore due to time constraints is a more thorough investigation into optimizing Time2Vec for transformers.

References

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. “Neural machine translation by jointly learning to align and translate”. In: *arXiv preprint arXiv:1409.0473* (2014).

[2] Seyed Mehran Kazemi et al. “Time2vec: Learning a vector representation of time”. In: *arXiv preprint arXiv:1907.05321* (2019).

[3] Ashish Vaswani et al. “Attention is all you need”. In: *Advances in neural information processing systems* 30 (2017).

Contributions

- **Luigi Pagani**: model conceptualization (LSTM+attention, transformer with multi-head attention, Time2Vec embedding), model development (LSTM+attention, LSTM+attention+Time2Vec), report writing.
- **Flavia Petruso**: model conceptualization (LSTM+attention, CONV1D_LSTM+attention), data inspection and preprocessing, model development (Vanilla LSTM, CONV1D_LSTM+attention, LSTM+attention, LSTM+attention+Time2Vec), report writing.
- **Federico Schermi**: data preprocessing, model conceptualization (LSTM+attention, transformer with multi-head attention), model development (LSTM+attention, transformer with multi-head attention), report writing.