AN2DL First Homework

Filippo Buda, Giacomo Carugati, Giuseppe Gentile

Team Name: King Gieck & the Filo Gent

December 22, 2023

1 Task Description

The report describes the methods and architectures adopted for a time series forecasting task.
Each sample belongs to one of 6 possible categories, namely 'A','B','C','D','E','F', and has variable length.
No information was provided to explain where the data was collected from and what quantity is measured by each entry.
The aim was to forecast the next 9 time steps of original time series in the first phase and 18 in the second.

2 Dataset Description

The provided dataset was composed of 48000 monovariate independent time series divided into categories. The length of the entries ranged from 0 to 2776 (timesteps). For time series shorter than 2776, zero padding was used, and in order to know which time series were padded and by how much, a `valid_periods` array was provided. The category information was stored in a `category` array as long as the number of time series.

3 Data Preprocessing

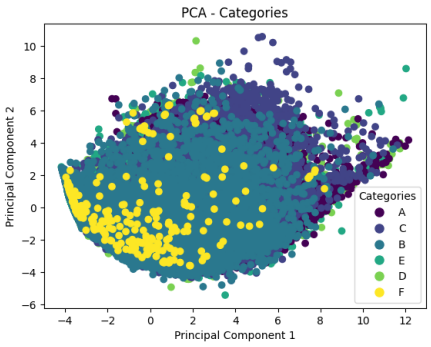
A high variability in the length of the samples was observed; big differences were also noticed in values range and trend of the time series, both intra and extra categories, as well as many noisy entries.
This called for preprocessing work, by using statistical tools and hand-crafted techniques.

3.1 Autocorrelation Analysis

Autocorrelation measures the correlation between the values of a time series and a lagged version of itself.
The lags at which the correlation is computed were considered only if statistically significant.
This information has been used first to discover noisy samples: if a sample shows no statistically significant lags under an autocorrelation analysis it's safe to assume the sample comes from noise and, therefore, can be eliminated from the dataset.
In addition, the significant lags for each time series have been investigated to decide the optimal value for the `window` value.

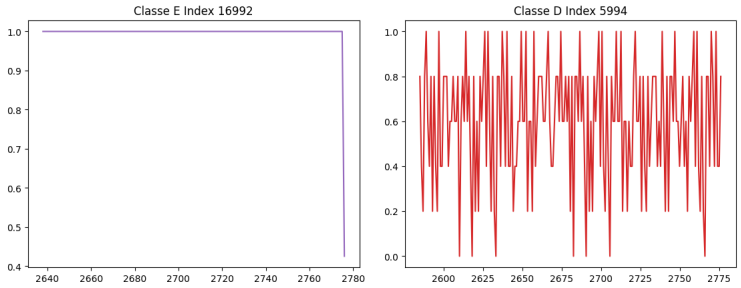
3.2 Category's Relevance Analysis

To acknowledge whether categories had any role in time series distribution, it was necessary to project the sequences of the time series into a smaller space.
Dimensionality reduction has been carried out on the original data, finding 13 components necessary to explain 90% of the variability. As can be seen, across different categories plotted over the space of the first two principal components, no distinct separation across classes can be observed. In light of this, we decided to discard the category information from the dataset that will be used to train our models.



3.3 Hand-crafted Methods

In order to spot meaningless time series, samples that didn't show more than 15 different unique values were removed.
Examples down here:



4 The Refined Dataset

To obtain more data and avoid training on long inputs, we divided each time series in sequences of the same length.
Previous analysis suggested the optimal window to live in a small value range (between 50 and 100). However, setting a `stride` of 10, due to the trade-off consisting in having a smaller window, but longer training time per epoch, the `window` size has been fixed to 200. Thus, series smaller than this have been ignored in order not to introduce additional padding.
Finally, each sample has been normalized and divided in overlapping sequences.
Same considerations are applied for the creation of local test sets.
All implemented models, therefore, have been trained on at least one-hundred thousand sequences, which is considered an acceptable amount of samples for deep learning models.

5 Implemented Neural Networks

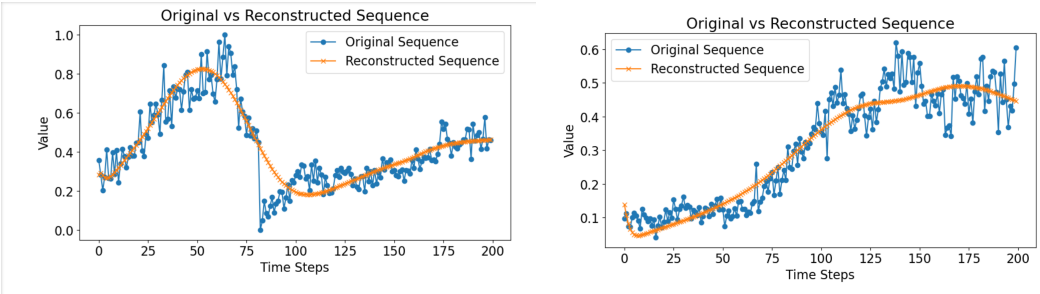
After some hyperparameter tuning, all the models ended up being trained using a `batch` size of 64 and a `learning rate` of 1e-3 with EarlyStopping and ReduceLROnPlateau callbacks to fight and hopefully avoid overfitting.
Every model has been trained to forecast the next 18 time step, independently from the phase of the challenge.

5.1 Experiments

In the following, models that reached a poor performance in the first phase of the challenge will be presented.

5.1.1 Cluster-specific Models

The proposed methodology entails encoding time series data into a lower-dimensional space using the encoder part of an autoencoder trained model, followed by clustering the resulting embeddings using k-means. This clustered representation is then utilized to train individual forecasting models for each cluster.
A simple forecasting model with biLSTM units is trained for each cluster, letting each model learn different weights.
Upon receiving a new input sequence, the architecture determines the cluster to which it belongs to and utilizes the corresponding trained model for prediction.
The assumption that distinct clusters in the embedding space correspond to temporal patterns across the entire dataset proved to be overly simplistic in this context, reaching MSE on hidden set of 0.0102. Another reason of the ineffectiveness of this idea may be the low amount of clusters chosen (only four).
However, this allowed to have a deeper understanding of how autoencoders behave in this scenario, and visualize how it reconstruct time series in a shallower and smoother representation:



5.1.2 ResNet

The idea was to create a skip connection between LSTM/Conv1D layers in order to pass information through different levels of the network.
However, no improvements were obtained; this type of models performed indeed in a similar way with respect to architectures implementing simple RNN layer stacked one above each other with the same hyperparameters.

5.1.3 Attention Mechanism

Another attempt was made by trying to implement a simple encoder-decoder architecture with attention mechanism without time embedding (although Time2Vec was taken into consideration), but it did not manage to obtain satisfying results.
This was probably due to some oversight in the implementation and in the preprocessing of the data for the decoder input.

5.2 Successful Models

The following subsections describe evolutions of the models that better performed in terms of MSE.

5.2.1 Baseline Model

The first model implemented was designed to output the last nine values it has seen as input.
This way a simple prediction model was built, and used as baseline performance for the next architectures.

5.2.2 Conv-RNN

The first proper forecasting model implemented was built using a combination of a one-dimensional convolutional layer and a bidirectional LSTM layer.
The convolutional layer was placed on top of the LSTM one in order to project the time series in input in a dimension where temporal patterns were easier to spot. The model directly predicted all the required timesteps as output.
It obtained relatively good performances, achieving a MSE of 0.0066052745 on first's phase test set.

5.2.3 Autoregressive Model

Leaving the direct forecasting behind, the autoregressive model started giving better results.
For this model the same architecture as its direct forecasting counterpart has been adopted and a parameter `autoregressive telescope` has been introduced (set to 3); the network was thus trained to predict the 3 following timesteps. The subsequent predictions were then concatenated, leading to a result on Codalab's hidden set of 0.0053.
Later on, the architecture underwent some smaller changes: the introduction of a GAP-1D layer before the output and the substitution of the cropped Conv1D output layer with a 3 units Dense layer, yielding to an improvement of the MSE on the hidden test of more than a tenth of thousandth (0.00519).

5.2.4 Final Model

Given the good outcomes of the autoregression approach previously explored, this has been further developed.
The major difference was the introduction of a GRU layer instead of the biLSTM (keeping the same amount of units).
Experimental data shown in table proved right our hypothesis that a more streamlined architecture can be beneficial in this scenario.

Cell-type	# of units	Optimizer	Loss	Val Loss	MSE
BiLSTM	64	Adam	0,0038	0,0041	**0,01079**
BiLSTM	64	AdamW	0,0036	0,0039	**0,01062**
GRU	64	AdamW	0,0039	0,0041	**0,01088**
GRU	128	AdamW	0,0034	0,0037	**0,00988**
GRU	256	AdamW	0,0028	0,0036	**0,00993**

The gating mechanisms of the GRU selectively update relevant information of long-range dependencies in time series, improving the predictive accuracy of biLSTM, which may encounter challenges in capturing intricate temporal patterns. Furthermore, `AdamW` optimizer was preferred over `Adam`, as suggested by external works ([4]) and validated by experiments.
The final model reached on the codalab dashboard MSE: 0.009703 and MAE: 0.066892.

6 Contributions

- Filippo Buda: autocorrelation analysis, dataset cleaning, baseline model, Conv-RNN, attention mechanism attempt
- Giuseppe Gentile: statistical analysis on dataset, cluster-specific model, autoencoder, final model
- Giacomo Carugati: data refinement, autoregressive models, local inference, final model

References

[1] https://danijar.com/tips-for-training-recurrent-neural-networks/#:~:text=Recurrent%20networks%20need%20a%20quadratic,to%20a%20ResNet%20or%20DenseNet

[2] https://machinelearningmastery.com/how-to-develop-lstm-models-for-multi-step-time-series-\forecasting-of-household-power-consumption/

[3] https://colab.research.google.com/corgiredirector?site=https%3A%2F%2Fstatisticsbyjim.com%2Ftime-series%2Fautocorrelation-partial-autocorrelation%2F

[4] https://paperswithcode.com/paper/llama-2-open-foundation-and-fine-tuned-chat