Artificial Neural Networks and Deep Learning - Second Challenge 2022

**Time Series Classification**

Nicola della Volpe, Alessandro Pindozzi, Jana El Khoury

Politecnico di Milano

December 20, 2022

**1. Introduction**

A challenge was set by the course coordinators in order to carry out the classification of samples in multivariate time series format. The time series contains data belonging to 12 classes. The goal is to correctly map the data in the features to their corresponding labels, over time. In our case, we had a training set of 2429 samples of size 36 x 6. Time series classification is one of the applications of deep learning methods. Different models, such as LSTM and Conv1D, and preprocessing configurations have been carried out.

**2. Data Analysis and Preprocessing**

The first thing done was to load the dataset with the numpy.load() function. Then we tried different preprocessing techniques.

The normalization of the dataset was not effective since after training the models later on, the results were worse than without normalization.

From that we tried to interpolate both on the axis of time and features, in this sort of augmentation, applying a linear interpolation. Regarding the y-axis, we interpolate in such a way that, between each pair of points in the original training set, three new points are obtained, so that the size of 36 is increased to 141. The obtained results were better.

Instead, for the z-axis at first we tried to remove some features, providing to the models training sets whose shape was 2429 x 141 x 5 or 2429 x 141 x 4, but we got worse results.

Then we decided to increase the number of features interpolating so as to add only one new point between each pre existing pair of points, getting a new size of 11. We noticed that the results were improving.

The splitting in the train and validation set was done by using train\_test\_split() from the data science library scikit-learn, with a ratio of 0,2.

The last thing that led to quite good results was using class weighting in order to balance the ones that contained less samples.

**3. Models and techniques**

We run different models in particular BiLSTM, Conv1D, Resnet for time series and finally GRU. On those models we tried different training configurations and different preprocessing in order to obtain the best result possible.

**3.1 BiLSTM**

A Bidirectional LSTM, or biLSTM, is a sequence processing model that consists of two LSTMs: one taking the input in a forward direction, and the other in a backwards direction. This model generally performed quite well in this task, but in our case it overfits and the results were not so interesting.

**3.2 ResNet**

ResNet achieves state-of-the-art performance in object detection and other vision related tasks. We explored the ResNet structure since we were really interested to see how the very deep neural networks perform on the time series data. Obviously, the ResNet overfits the training data more easily because the dataset is relatively small and lacks enough variants to learn the complex structures with such deep networks.

Nevertheless the performances are better than the ones obtained with LSTM, even if the accuracy on the validation set swings a lot.

**3.3 GRU**

Gated recurrent units (GRUs) are a gating mechanism in recurrent neural networks. The GRU is like a long short-term memory (LSTM) with a forget gate, but has fewer parameters than LSTM since it lacks an output gate. As expected the results are similar to the BiLSTM.

**3.4 Conv1D - Best model**

With a 1D Convolutional layer we achieved the best result. We used a couple of convolutional layers composed of 256 neurons each, a MaxPooling layer, a Gaussian Noise layer and a couple of dropout layers in order to deal with overfitting. Finally a GlobalAveragePooling layer was added in order to flatten the output for the dense classifier. As simple as effective, it remarks the no free lunch theorem.

**4. Training**

To carry out the training efficiently, we used the validation accuracy of the model to keep track of the performance. The model was trained for 3000 epochs and a patience of 300 epochs, which was required because a lot of time was needed before the model started learning. We added a kernel, bias and activity regularizers in order to apply penalties on layer parameters or layer activity during optimization to be summed into the loss function.

**5. Results**

The results obtained for each of the models and techniques used are gathered below, noting the accuracy values both on validation and test sets.

| Model | Val Accuracy | Test Accuracy |
| --- | --- | --- |
| BiLSTM | 0.6317 | -\* |
| Resnet | 0.7225 | 0.6996 |
| GRU | 0.6235 | -\* |
| Conv1D | 0.7401 | 0.7102 |

*Table 5: Results of trained networks*

*\*The model was not scored on test set since the result on the accuracy was not higher enough*

**6. Conclusion**

After training with the different models, the best model was the Conv1D, obtaining the highest accuracy. Augmentation techniques were further tried, however nothing led to improvements..

Finally, we can conclude by saying that Convolutional Neural Networks are great tools for time series classification, according to our experiments.

**7. Tools Used**

* Google Colab
* Tensorflow
* Jupyter Notebook
* Keras