Stefano Fumagalli 10628587
Lara Ferro 10622035
Alberto Noris 10634510

Artificial Neural Networks and Deep Learning

Time Series Classification

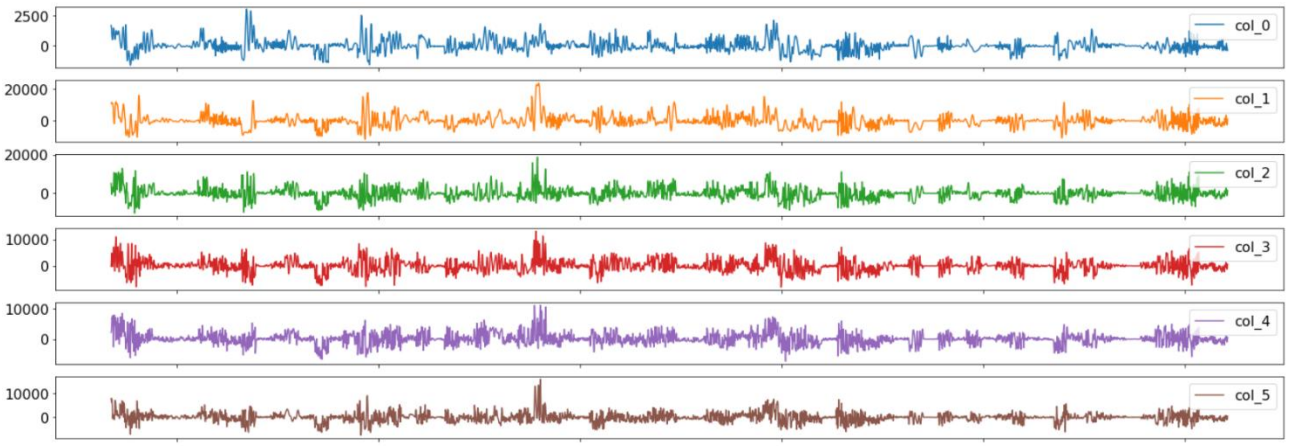
For this challenge we had to correctly classify time series belonging to twelve different classes, provided us in a zip folder. We had at disposal 2429 samples with size 36x6.

Moreover, the training samples per class were not balanced.

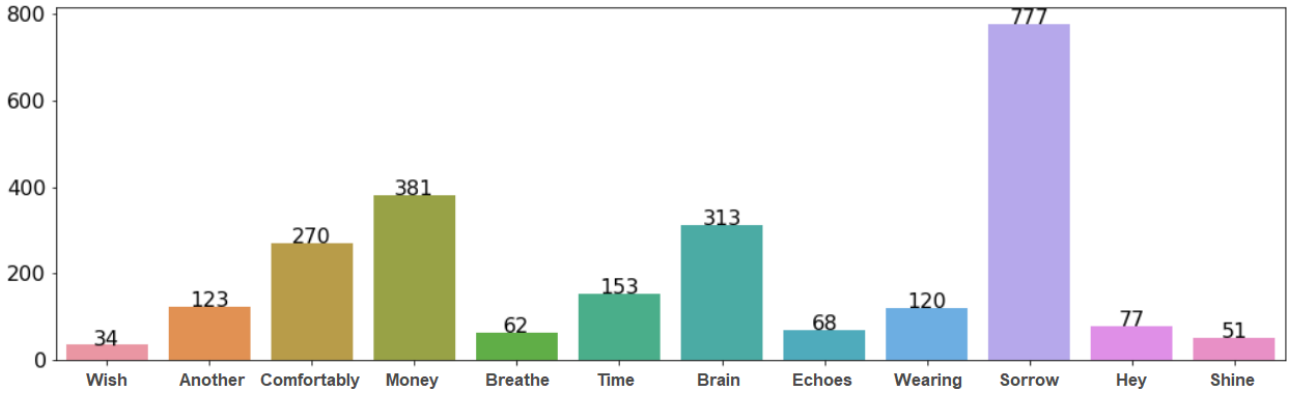
Facing this problem, in order to predict the correct class label, our team decided to organize the flow of work in this manner: develop a basic LSTM model from scratch and gradually proceeding towards more advanced models.

In the end, we had four different solutions for the problem:

- LSTM model
- Bi-LSTM model
- 1D-convolution model
- ResNet from scratch with Conv1D layers



Sample of time series from class "Hey"



Time series distribution

LSTM Model

To start with a custom LSTM model, we split the given dataset in two parts: 80% for the training set and 20% for the validation set to perform cross validation, early stopping and prevent overfitting. Then, we created a simple model following the steps from the laboratory lessons with 2 LSTM layers, dropout to prevent overfitting and 2 dense layers to classify the time series. We trained this first classifier obtaining a result of 60% on the test set.

Starting from this initial net, in order to improve the performance, we decided to change some parameters to adapt the model to our samples: we increased the number of hidden neurons in each layer to 256 and increased the dropout rate to 0.5. We also tried different batch sizes and different activation functions, obtaining our best result with batches equals to 128 and ReLu function. Furthermore, a technique that helped us increasing the performance was the use of class weights, to deal with unbalanced classes. With this model, we obtained a result of 66% on the test set.

Bi-LSTM Model

As second attempt we tried with Bi-LSTM, building a simple model with Bidirectional layers instead of LSTM. We tried different hyper parameters tuning finding out that an increase in the number of hidden neurons, together with the elimination of class weights gave us better results. The result of this model was 68%.

Conv1D Model

Then, we tried a different model based on convolutional layers as seen in the laboratory classes. After few attempts and hyper parameters tuning, we found out that we obtained best performances with a model composed by 3 blocks of 1D convolution, max pooling and dropout for the feature extracting phase, followed by a global average pooling and 2 dense layers as classifier.

In order to achieve better performance, we focused on the data applying different preprocessing techniques to the training part of the dataset: we first tried to normalize between 0 and 1 on each feature, both per sample and on the entire time series, but the fitting on this format didn't get any improvements. Therefore, we tried instead to standardize with zero mean and standard deviation at one, both per sample and on the entire time series as for normalization and, in this way, we discovered that the model learnt well with standardization on the entire set. To do this, we changed the shape of the dataset and by means of StandardScaler we fitted the training dataset and applied the transformation both on training and validation sets and then we reshaped again the data to the original dimensions.

We tried to modify the window size, decreasing it to 18 and doubling it to 72 but this didn't work as expected, resulting in a worse accuracy.

Another attempt was data augmentation: we exploited tsaug library to double the number of samples in our dataset by adding a small noise to each of the duplicated one. We also tried to perform cropping, drifting and time warping on our time series, but didn't work well as adding noise.

Since our model was very slow in learning we decided to increase the total epochs of training to 400 and the patience to 50 epochs as best tradeoff between high accuracy and low overfit. In this way we obtained our best model that reached 73% on the test set.

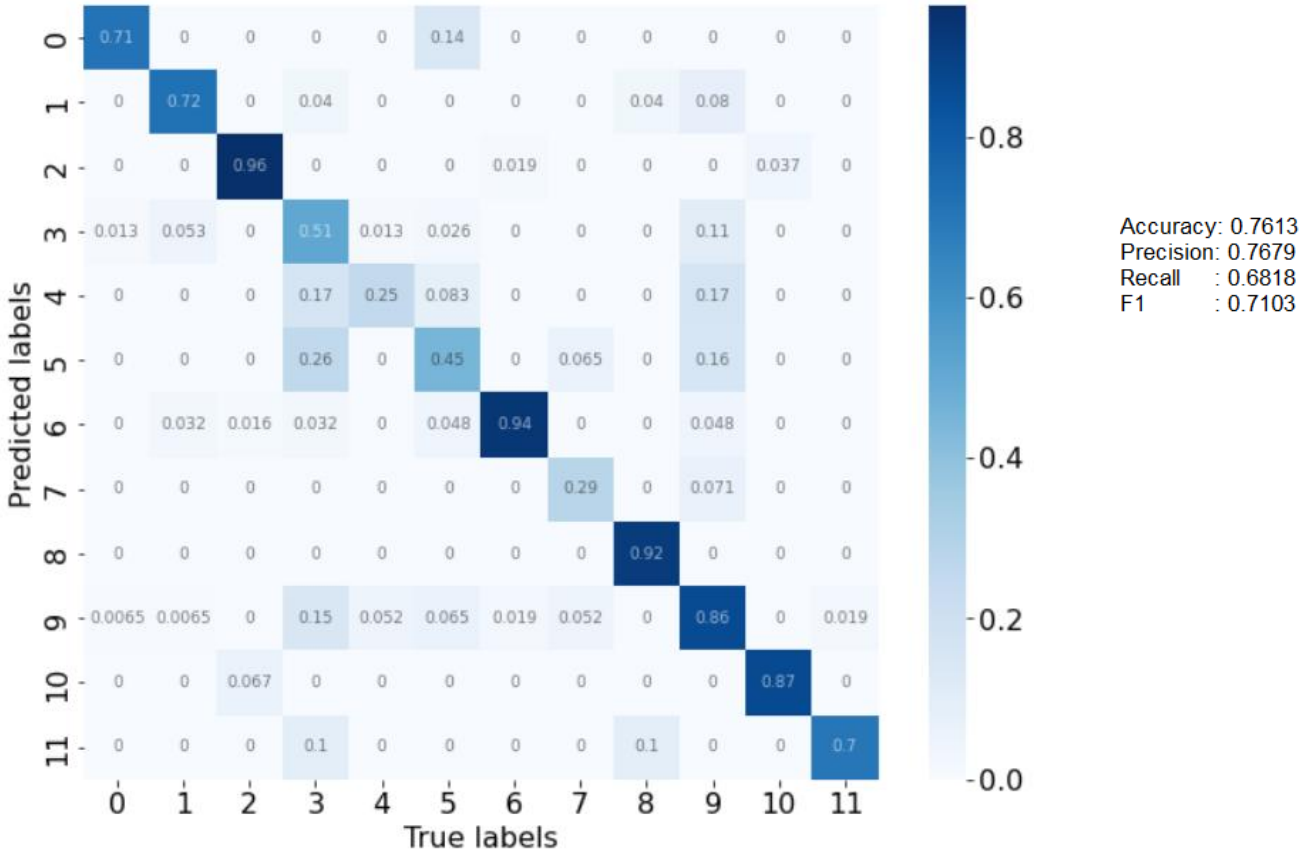
As final attempt to improve our Conv1D model we tried to train it without some features, removing them one at a time or in pairs, or merging them to create new features but these didn't change the performance.

ResNet Model

To explore different solutions, we built a residual network from scratch using three blocks composed by 1D convolutions and ReLu activations, merging the connections with an add layer and a top net composed by dense and dropout layers. We started to develop this basic model without any preprocessing or augmentation techniques, adding them one at a time to find out which one worked better with this model. After several tuning attempts, we ended up with a model with data augmentation and data standardization on the whole dataset that reached the same performances of 1D convolution model.

Searching on internet for advanced techniques, we also found a paper in which 2D convolutional ResNet was used to classify time series [source]. We decided to implement this variation obtaining again a result of 73% on the test set.

Finally, we concluded by submitting the 1D-convolution model giving us an accuracy of 73.04% on the test set of the first phase and an accuracy of 72.95% in the second phase, with the following confusion matrix referred to the validation set:



Confusion matrix of the best model