Time Series Classification

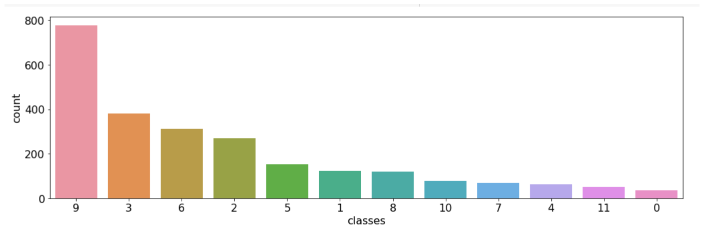
Leaderboard nickname: NotRunningCode
Students: Federica Baracchi, Federico Camilletti, Patricks Francisco Tapia Maldonado

1 Introduction

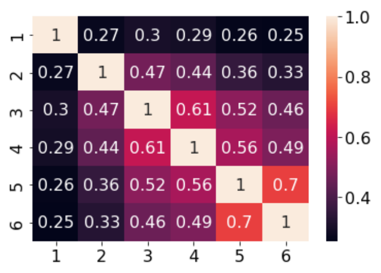
The aim of the homework was to correctly classify samples in the multivariate time series format. During the challenge several models was exploited to achieve the best result in terms of accuracy.
In this report firstly the data augmentation problem is described and then the five models used: ResNet, Bilateral-LSTM, Ensemble learning and K-fold cross-validation. The last section is then spent to describe the 1D-CNN model which was used to achieve the best result.

2 Data

The dataset has 2429 samples each one composed by 36 timestamps and 6 features, moreover the classes are 12. The image below shows that the dataset is unbalanced with respect to the classes so an augmentation of the dataset was necessary.



The first step was aimed to obtain a data augmentation consistent to the time series task. Different approaches were tested applying a scaling factor and a random noise to each sample of the training set. Moreover the shifts of the timestamps were used in order to feed the training with more data consistent with the idea of the time series.
The best data augmentation were provided mixing the shifts and the random noise. An external library was imported in order to coherently shifts randomly the timestamps doubling the training set. Then a further shift adding a random noise was provided to the whole training set. The model then was feed with 7772 samples.



To better investigate the dataset, and not knowing its nature, the correlation between the 6 features was analyzed: it is possible to notice how the features are particularly related to each other, except for feature 1. Our idea was to try to reduce the number of features, using the technique of **"Principal component Analysis"**: to find principal components as a linear combination of the original features, so reduce the number of variables of a data set, while preserving as much information as possible. In our case, however, it did not turn out to be very useful, so all 6 features were used.

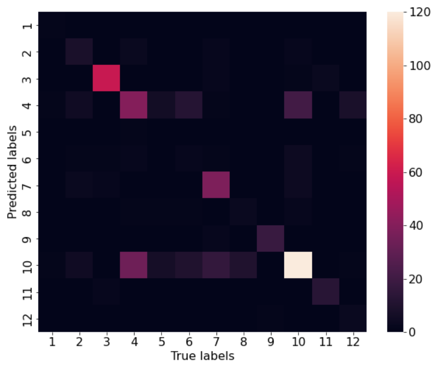
3 Models

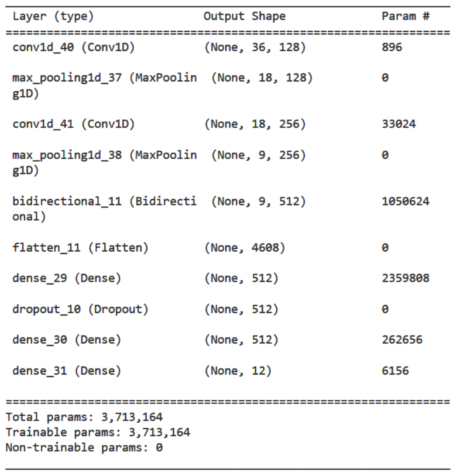
3.1 ResNet

Since the 1D convolutional model was successful we used ResNet. It was implemented using the reshape filter of Keras to realize a kind of image from each pixel. The resulting model has a total of 510,988 parameters (508,428 trainable and 2,560 non-trainable). The accuracy on the validation set during training reached 75%, but in the hidden test of phase one scored around 69%. We tried to improve the results, in particular by trying to avoid overfitting, without significant results.

3.2 Bilateral-LSTM

During the challenge, we tried to create models belonging to the class of long short-term memory models. We tried both a normal LSTM and a bilateral LSTM. However, the results of both of them were below the average of our models.

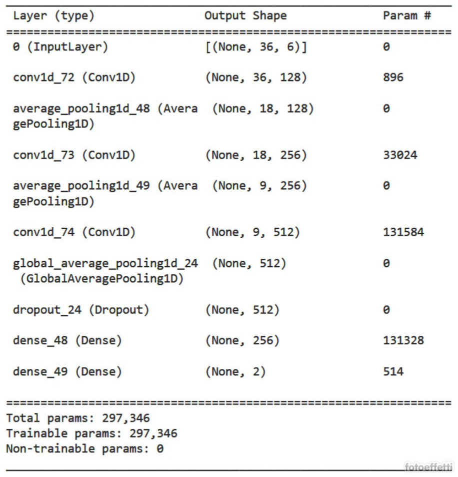




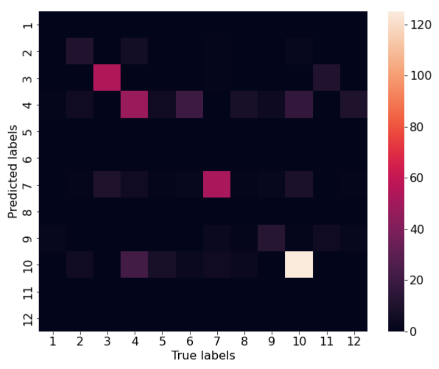
```
Layer (type)                Output Shape              Param #
=================================================================
 conv1d_40 (Conv1D)         (None, 36, 128)           896

 max_pooling1d_37 (MaxPoolin  (None, 18, 128)         0
 g1D)

 conv1d_41 (Conv1D)         (None, 18, 256)           33024

 max_pooling1d_38 (MaxPoolin  (None, 9, 256)          0
 g1D)

 bidirectional_11 (Bidirecti  (None, 9, 512)          1050624
 onal)

 flatten_11 (Flatten)       (None, 4608)              0

 dense_29 (Dense)           (None, 512)               2359808

 dropout_10 (Dropout)       (None, 512)               0

 dense_30 (Dense)           (None, 512)               262656

 dense_31 (Dense)           (None, 12)                6156

=================================================================
Total params: 3,713,164
Trainable params: 3,713,164
Non-trainable params: 0
_________________________________________________________________
```

3.3 Ensemble learning

Since the confusion matrixes of the models were similar and it was evident that some classes weren't recognized at all we tried another approach with ensemble training. The ensembled model was realized combining our best model which was a 1D convolutional model. We realized and trained a single of these models for each class, then we combined the submodels into a unique model. This method did not solve the problem of unrecognized classes since the submodels performed in the same way as can be observed by looking at the confusion matrix.



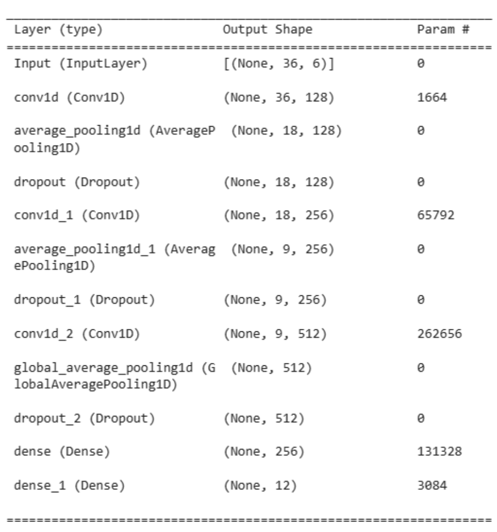
```
Layer (type)                Output Shape              Param #
=================================================================
 0 (InputLayer)             [(None, 36, 6)]           0

 conv1d_72 (Conv1D)         (None, 36, 128)           896

 average_pooling1d_48 (Avera  (None, 18, 128)         0
 gePooling1D)

 conv1d_73 (Conv1D)         (None, 18, 256)           33024

 average_pooling1d_49 (Avera  (None, 9, 256)          0
 gePooling1D)

 conv1d_74 (Conv1D)         (None, 9, 512)            131584

 global_average_pooling1d_24  (None, 512)             0
  (GlobalAveragePooling1D)

 dropout_24 (Dropout)       (None, 512)               0

 dense_48 (Dense)           (None, 256)               131328

 dense_49 (Dense)           (None, 2)                 514

=================================================================
Total params: 297,346
Trainable params: 297,346
Non-trainable params: 0
_________________________________________________________________
```



3.4 K-Fold Cross Validation

During our work the K-Fold Cross Validation was tested to resampling procedures generally used to evaluate machine learning models. The dataset is divided into k subsets of equal size, and the model is trained k times, and each time a different subset is used as a validation set. At the end of the K iterations, the results are averaged to obtain the final result. The displacement of the method lies in the weight of the computation, proportional to the choice of k(in our case k = 5/10). However, in our case, after testing it on various models, it proved to be more efficient for some but less for others, so it was decided not to use it for the final model.

4 Final Model: 1D-CNN



```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 Input (InputLayer)          [(None, 36, 6)]           0

 conv1d (Conv1D)             (None, 36, 128)           1664

 average_pooling1d (AverageP  (None, 18, 128)          0
 ooling1D)

 dropout (Dropout)           (None, 18, 128)           0

 conv1d_1 (Conv1D)           (None, 18, 256)           65792

 average_pooling1d_1 (Averag  (None, 9, 256)           0
 ePooling1D)

 dropout_1 (Dropout)         (None, 9, 256)            0

 conv1d_2 (Conv1D)           (None, 9, 512)            262656

 global_average_pooling1d (G  (None, 512)              0
 lobalAveragePooling1D)

 dropout_2 (Dropout)         (None, 512)               0

 dense (Dense)               (None, 256)               131328

 dense_1 (Dense)             (None, 12)                3084

=================================================================
```

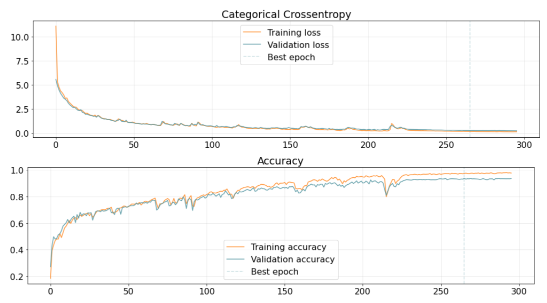
The best model in terms of error reduction and accuracy was chosen: 1D Convolutional neural network. The model was built using convolution and the average pooling technique was repeated three times. In particular for convolution the kernel size = 2 and the filter size: 128, 256, 512. At each convolution, it was used the "relu" as activation function because it proved to be the most efficient. In the end, to obtain the feature vector of our data, global average pooling (taking the average of each feature map) was used instead of "flatten".

To reduce the **overfitting**:

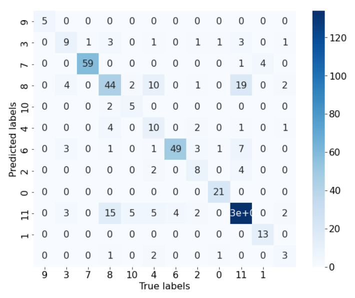
1. the dropout technique was used 3 times, turning off 20% of neurons.
2. in FC layer added a L2 regularization term (**ridge regularization**) with $\lambda = 0.01$

$$E(w) = E_p(w) + \lambda \sum_{i=1}^{n} w_i^2$$

The loss's curve evaluated on the train test, is very close to that evaluated on the validation set.



By looking the confusion matrix we can investigate which are the best-identified classes: It is possible to see that the best-recognized classes are classes 11, 7, and 6 which, as seen above, are some of the smaller classes. While it is possible to notice that, although class 11 is the best identified, the model sometimes confuses class 11 with class 8 and vice versa.


Predicted labels
True labels