ANNDL2022: Homework 2

Giovanni
Giuseppe
Luca

Introduction
The goal of this project is to realize a multivariate time series 12 classes classification, based on 2429 instances of 6 features collected for 36 consecutive time instants (input structure: $(2429, 36, 6)$).
We decided to produce several networks, starting with the ones seen during the lab sessions of the course, and then trying to mix the structures and preprocess the dataset in different ways.

Data preprocessing
Due to the amount of very unbalanced data, in opposition with our previous homework we now decided to split our dataset only into train and validation, avoiding the creation of a separate test set, and therefore only relying on the Codalab tests for unbiased evaluations of our performances.

Two streets were carried out in parallel to oppose this unbalancing: in some notebooks, we used a multiplicative function to balance all the classes, while in others we applied the *train_test_split()* function to obtain stratified train and test datasets. Our theoretical basis about this approach is that a network trained on balanced data will be more robust in results shown even on unbalance sets, while a stratified train could develop some "default choices" to deal with doubtful classifications of complicated instances, exploiting the unbalancing of the train set (which clearly is assumed to resemble the real phenomenon distribution), but is riskier since it is far less robust with respect to inconsistencies in the classes frequencies between train and test set, potentially providing disastrous results.

While in the first challenge we went with the first and more standard approach only (that is, assuming that no distribution is exploitable a priori), during this challenge we observed that the validation results obtained with stratified train and validation were preserved in the Codalab tests and were better than the ones obtained with the balancing of the classes by almost a 10% margin. This exploitation of the test set, even if not too theoretically soundly, was necessary to acquire the lacking info about at least the distribution of the classes for the phenomenon studied.

We also standardized our dataset features before providing them to our networks using either MinMaxScaler(), StandardScaler() and RobustScaler(). A normalization layer was included in some network to perform data normalization in different dimensions (features and windows). The latter seems to not improve our results.

First Models
During the lab sessions, we saw three different networks suited for time series classifications: a Vanilla Long Short-Term-Memory (LSTM) Neural Network, a Bidirectional-Long-Short-Term-Memory (BiLSTM) Neural Network and a 1D Convolutional Neural Network. In a first moment we analyze the LSTM and BiLSTM models reaching (after many tries and the help of *Keras Tuner*) 0.67 of accuracy in the codalab test set with a model with 3 bidirectional layers for features extraction, GAN, and 2 dense layers with 128 nodes. This result was reached with a balanced dataset.

1D Convolutional Network
In the second phase we model to the more promising 1D-Convolutional Network, where we were able to use the skills acquired in the first challenge better. Different models have been tried to improve the already good results of the initial model (70% on stratified validation set), but most of them gave us the same results as the initial one. In our attempts we noticed that our models have not negligible variance, so we decided to train the same model more than once to limit this phenomenon (*with kerasTuner executions_per_trial*). In our best attempt we reached 0.75 of accuracy in the stratified validation set with a network with the following characteristics:

- 1D Conv layer with 256 filters and convolution window size 5 and padding “same”, activation = ”relu”
- Max Pooling Layer
- 1D Conv layer with 512 filters and convolution window size 3 and padding “same”, activation = ”relu”
- Global Average Pooling Layer
- 2 Dense layers with 128 nodes, activation = ”LeakyReLU”
- 1 Dense layer with 12 nodes and activation = ”Solftmax”

Small L1L2 regularization and dropout layers are used in the fully connected part which allows to delay the overfitting.

Switching from *StandardScaler* to *RobustScaler* let us gain some percentage points in accuracy. We decided to submit this last model reaching 0.735 of accuracy on Codalab.

We also tried to mix our best performing 1D convolutional structure with some LSTM layers: our hope was to obtain a net capable of extracting useful elaborations of the original six features thanks to the filters of the convolutional layers, but also to exploit the powerful memory of LSTM layers to connect temporally that information.

The structure chosen had 2 consecutive blocks of 1D conv + LSTM + 1D MaxPooling, followed by a single block 1D conv + LSTM (now that the time dimension is fixed to 9) and then 2 dense feed-forward layers. Unfortunately, even varying largely both the number of parameters (from less than 100’000 to more than 4 millions) and the regularizations adopted, we could never really cross the 70% accuracy even in our validation set.

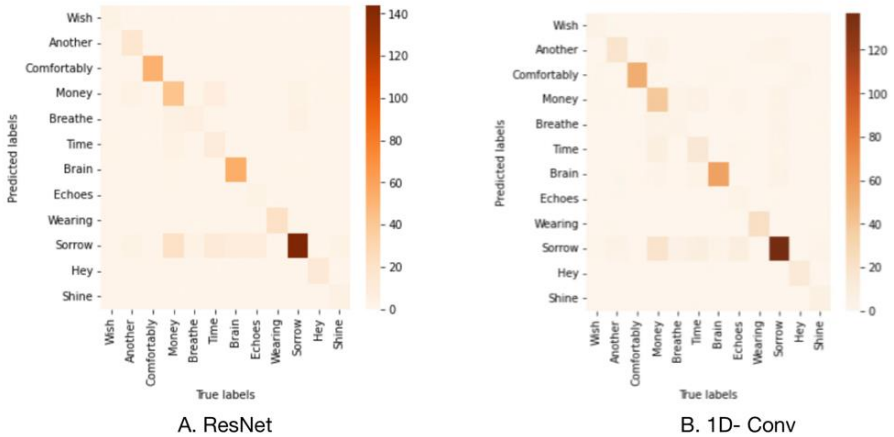
ResNet

Like what we did in the first challenge, we used a famous deep learning model, ResNet, to improve the results obtained from the model with only Conv1D layers. The network is made up of three blocks, each of those with a residual connection to the next one, followed by a Global Average Pooling and a softmax classifier. At the beginning, the results did not meet expectations, so some modifications to the original structure were made. The first thing that improved the accuracy of the model was to get rid of the BatchNormalization layers and the addition of a Dropout after each Conv1D layer: these two changes gave an accuracy of 70% reducing the overfitting. Then, the next improvements have been removing a Conv1D layer from each block, where originally there were three, and using the same kernel size of 5 for each Convolutional layer, instead of 8 and 5 for the first and the second Conv1D of each block, respectively: these further changes lead to an accuracy of 71%. The final modifications that gave an accuracy of 73.94 % on the final test set on Codalab, the best result for our ResNet model, were to increase the number of residual blocks from 3 to 6 and to add a Dense layer of 256 neurons between the GAP layer and the final softmax classifier.

Other networks

The following architectures have been tested: InceptionTime, which is inspired by the Inception-v4 architecture, and Transformer of the 1D Convolutional Network, even if some small changes have been

Figure 1



A. ResNet B. 1D- Conv

applied to the original structure. The performances of the two on the provided dataset did not match those of the ResNet model and due to the time constraint, we focused mostly on the well performing ones.

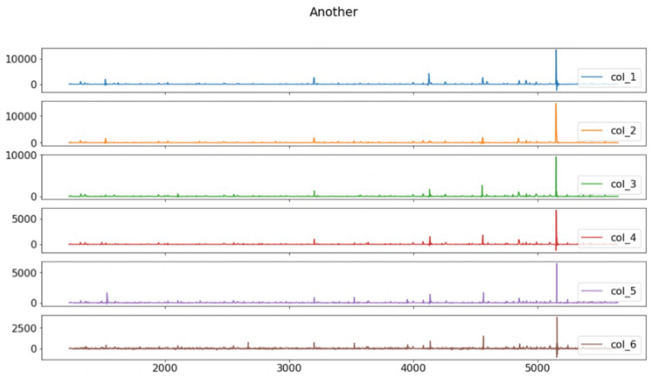
Data Augmentation

We tested also several ways to augment our data: we tried to create more instances for the training set by building sequences using different strides which will likely be a successful idea all those times the data is generated by partitioning longer sequences. Another idea was to tune the windows size to try to catch a more significative series (i.e. reducing the number of instants considered to a subset of the original 36) but both did not lead to a significative improvement with our most successful architectures

We also tried to add noise to our dataset, both by means of a gaussian noise applied to each feature and calibrated separately and introducing a random noise proportional to the value of each feature for each individual instant (i.e., a random noise modifying the value in a range of $\pm 20\%$, which therefore is proportional to the magnitude of the single value itself).

Plotting the samples of each class for each channel in a ordered way and we noticed that the six features were quite similar [figure 2] in the context of the same class, so we tried to prune the ones with the largest range of values to try to improve the normalization process. This did not lead to an improvement.

Figure 2



The standardization attempts were all disappointing, keeping the performance unchanged (or in some cases worsening it) in most of the tests, except for a slightly better result obtained considering RobutScaler in some models. The augmentations attempts have been a failure too: the data seems to not be obtained from longer sequences by segmentations, since our method of construction of new sequences always worsen the performances (with little differences between using zero padding or not), while the application of noises on the amplitude of the features to create new samples left our strongest nets unaffected.

Final Considerations

A deeper correlation analysis between data and maybe some specific knowledge about the phenomenon dealt with could help in future to increase the performance obtained by us so far: it is in fact interesting to observe that, with respect to the previous challenge, our lack of some sort of a priori knowledge forced us to move more blindly, which highlight the importance of a well though approach even when applying cutting edge models such as those provided by the Deep Learning field.