POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Artificial Neural Networks and Deep Learning 2022 - Second Homework

MSc in Computer Science & Engineering and Mathematical Engineering

Authors: Leonardo Gori, Lorenc Leci, Marco Rizzo **Team: LooneyFine-tunes** **Academic year: 2022-2023**

1. Abstract

This report includes the resume of the attempts that have been conducted during the development of a supervised learning-based model for the classification of time series.

2. Dataset

The development task consists in the classification of multivariate time series which are divided into 12 classes. The given dataset has a shape of 2429x36x6, i.e. it contains 2429 time series sampled in 36 time instants, each one having 6 recorded features. As reported in Table 1, the dataset is characterized by heavy imbalance, highlighted by the 2 extreme cases of classes 'Wish' and 'Sorrow' containing respectively 34 and 777 samples.

ID	Class	# Samples
0	Wish	34
1	Another	123
2	Comfortably	270
3	Money	381
4	Breathe	62
5	Time	153
6	Brain	313
7	Echoes	68
8	Wearing	120
9	Sorrow	777
10	Hey	77
11	Shine	51

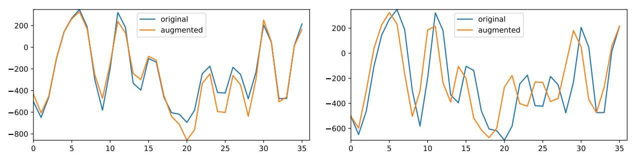
Table 1: Number of samples per class.

In order to test the generalization ability of the later models, we decided to split the dataset in training and validation (80% / 20%). For augmented strategies we decided to use as percentage split of 75% / 25% in order to have more samples in the validation set.

3. Data pipeline

In this section we present the procedures used to preprocess the data and to overcome the unbalanced dataset.

Data augmentation In the development we analysed many data augmentation techniques at dataset-level.
In particular, we tested the techniques described in [3] to augment the minority classes. Among the various transformations, we privileged those techniques which did not shift the time series, i.e. jittering, scaling and magnitude warp as it is shown in Figure 1a, since they proved to lead the model to better performances w.r.t. those transformations that shifted the values of the time series, i.e. window slice, permutation and time warp as it is shown in Figure 1b.



(a) Magnitude warp (b) Time warp

Figure 1: Comparison of two tested augmentation techniques.

In order to overcome the imbalance of the dataset, we augmented classes 2,3 and 6 with magnitude warp, doubling their size in the training set. Furthermore, we augmented classes 0, 1, 4, 5, 7, 8, 10, 11 contained in the training set with a combination of jittering, magnitude warp and scaling, making them eight times larger. The validation set was kept untouched. However, the augmentation procedure has shown to be effective only with some specific models.

Feature Correlation As illustrated in Figure 2, features 4-5 present high positive correlation of 70.03%, while features 2-3, 3-4 and 2-4 have a moderate positive correlation, respectively of 61.22%, 56.49% and 51.60%.
Having state that, for every network we tried training it both on the whole and on the reduced dataset (without features 3-4), but with the last dataset we got

slightly worse results on the validation w.r.t. the first one. So we kept all the features for training purposes.

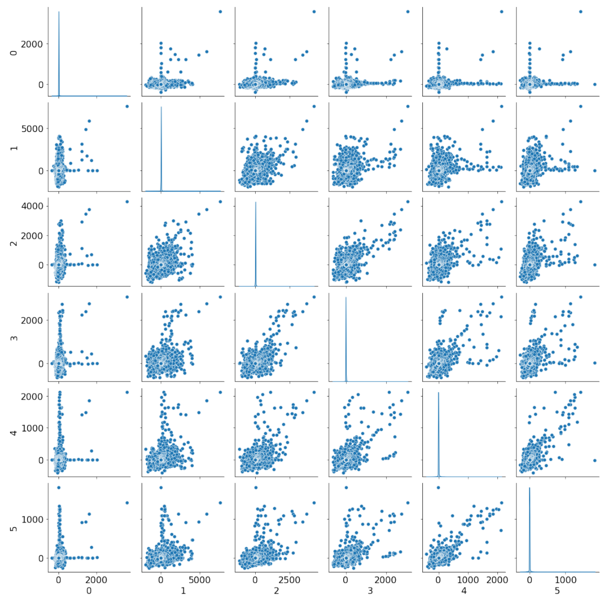


Figure 2: Correlation plot of the features.

Scalers In order to preprocess our data, we firstly reshaped the 3-dimensional numpy array into a 2-dimensional one. Then we tried to apply different transformation tecnhiques such as Zscore, MinMaxScaler, StandardScaler and RobustScaler. The last one uses statistics that are robust to outliers and it was the scaling that performed better for most of the models, except for the autoencoder one, for which we used StandardScaler.
StandardScaler was fitted on the training set, and then applied on both training and validation sets. Instead for the RobustScaler, we noticed that applying it on the whole dataset before splitting, unexpectedly got us better perfomances in all the cases except for the FCN classifier and the ResNet-style model customized with Conv1D (which will be described in section 4). For this reason we kept these two different strategies for the two different scalers.

4. Neural Network

We started using simple architectures as a first attempt:

1. Long Short Term Memory (LSTM) Neural Network
2. Bidirectional Long Short Term Memory (BiLSTM) Neural Network
3. 1D Convolutional Neural Network

In order to compile the models we used the CategoricalCrossentropy as the loss function of our models, the Adam optimizer with different learning rates and as metrics the "val-accuracy". We trained the model on 200 epochs, setting batch_size to 32 and using EarlyStopping and ReduceLROnPlateau with the aim of maximizing the validation accuracy.

The best epoch of each Neural Network tested so far got us a validation accuracy respectively of 69.96%, 67.28% and 73.25%. For this reason, we decided to focus on the model having the best validation accuracy: 1D-CNN.
We improved the last model by adding a GaussianNoiseLayer, a Conv1D layer and decreasing the kernel sizes of the Conv1D layers. Furthermore, we used as optimizer RMSprop with learning rate of 7e-4 and rho equal to 0.9. This led us to 74.27% on the validation accuracy.

After trying to improve our hand-made model, we started studying the state of the art of time series classification models, searching for new strategies to apply.

ResNet-style models We start building two ResNet-style models, one with Conv1D layers and another one with LSTM layers inspired by [4] and we removed the BatchNormalization layer (which usually got us worse results). We tested these model both on the dataset without augmentation and on the model with augmented data. The last strategy on the ResNet-style model with Conv1D Layers got us to an accuracy of 74.67% on the validation set.

Fully Convolutional Networks (FCN) We also tested all the models reported in [2] and the one that gave us better results was the Fully Convolutional Networks classifier. For this reason, we decided to improve its performances by modifying its structure. In particular, we removed the BatchNormalization and left the model with two Convolutional layers. In this way, we got 72.35% as validation accuracy. We also applied a Gaussian Noise layer as the input layer and used weighted categorical cross-entropy [1], but all the combinations of these two got to us worse performances.

Transformer model with Conv1D layers We build a Transformer architecture applied to timeseries inspired by the Keras-GitHub repository. It is composed of a transformer-encoder part followed by a 1D Convolutional Neural Network. This model was optimized using RMSprop with the parameters set as before and led us to 74.07% as validation accuracy.

Ensemble models We tried ensembling our best models, averaging their outputs, but this attempt had not improved our result.

Autoencoder As a further attempt, we came up with the idea of creating a specific autoencoder for each classes, thinking that they were indipendent with each others. So we tested a classifier composed by an ensemble of 12 autoencoders. The classification criteria was provided by the autoencoder with the minimum loss function. We evaluated its accuracy on the validation set according to the same criteria, and noticed that most of the samples were predicted to be of class 0 or 9 most of the times. So we decided to discard this approach.

5. Results

Our best model in the first phase (Transformer Model) did not performed as well in the second phase. The best one on the final test set came up to be the Resnet 1D Conv Model.

Model	Train	Validation	First Test	Second Test
Conv1d	0.7530	0.7428	0.6989	0.6847
Transformer	0.7452	0.7407	0.7316	0.6965
FCN costum	0.8213	0.7235	0.7243	0.7183
Resnet 1D Conv	0.9887	0.7467	0.7182	0.7238

Table 2: Best Models Accuracy Comparison.

6. Final Model

The structure of the final model is described in details in the code. Its visual representation is the following one:

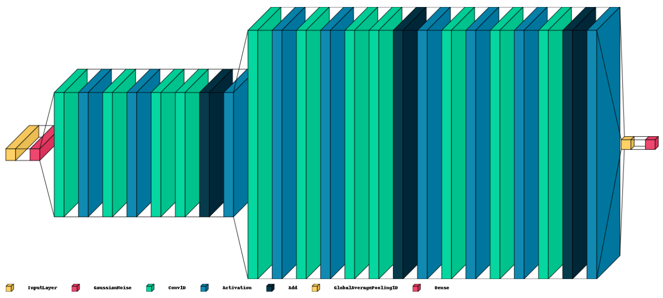


Figure 3: A visual representation of the structure of the model.

The model has reached an accuracy of 74.67% on our validation set, of 71.82% in the test set of the first phase of the competition and a final accuracy of 72.23% in the second phase. We can observe the ability of our model to generalize reasonably well considering that the size of the test set of the second phase was 66% of whole test set w.r.t. the 34% of the phase 1.

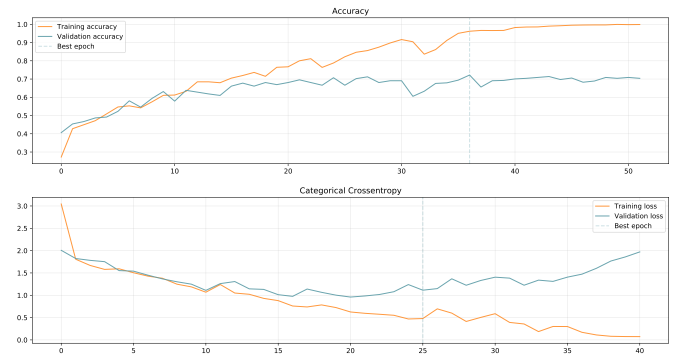


Figure 4: Plots of the Categorical Crossentropy and Accuracy of the ResNet 1D Conv Model.

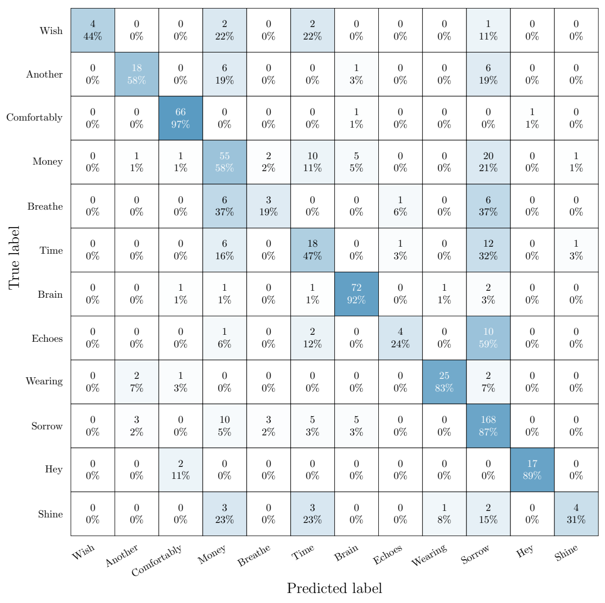


Figure 5: Confusion matrix of ResNet 1D Conv Model on the validation set.

References

[1] Dong Chen, Yuzhen Lu, Zhaojian Li, and Sierra Young. Performance evaluation of deep transfer learning on multi-class identification of common weed species in cotton production systems. *Computers and Electronics in Agriculture*, 198:107091, 2022.

[2] Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar, and Pierre-Alain Muller. Deep learning for time series classification: a review. *Data mining and knowledge discovery*, 33(4):917–963, 2019.

[3] Brian Kenji Iwana and Seiichi Uchida. An empirical survey of data augmentation for time series classification with neural networks. *Plos one*, 16(7):e0254841, 2021.

[4] Zhiguang Wang, Weizhong Yan, and Tim Oates. Time series classification from scratch with deep neural networks: A strong baseline. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 1578–1585, 2017.