AN2DL - HOMEWORK 2 REPORT

Alessia Lindemann	Michelangelo Olmo Nogara Notarianni	Sara Paratico
M.Sc. Biomedical Engineering	M.Sc. Biomedical Engineering	M.Sc. Biomedical Engineering
ID: 10862854	ID: 10625064	ID: 10626459
"AlessiaLindemann"	"olmonotarianni"	“sara.paratico”
Codalab group: “Typed leaves”		

Multi-class Time Series Classification problem

Time series classification uses supervised machine learning to analyze multiple labeled classes of time series data in order to classify series belonging to a new, independent data set. This homework was developed in order to classify time series shaped (n_samples, 36, 6) belonging to 12 classes named after some of the famous Pink Floyd’s songs. We tried to treat the data as a set of audio signals: we couldn’t listen to any reasonable sound from those, but we obtained some tools for the classification from the analysis in the frequency domain. In this report, we will try to clarify the steps we made and justify our choices related to this problem.

1) Our dataset

Our dataset consisted of 2 .npy files, named respectively “x_train” and “y_train” and containing 2429 samples (36x6 numpy arrays) with the associated integer labels (0-11). We associated the 36 rows of each signal to its values in time and the columns to 6 different channels/features.

The labels led us to try audio analysis tools: zero crossing rate, spectral features, etc. as features extractors.

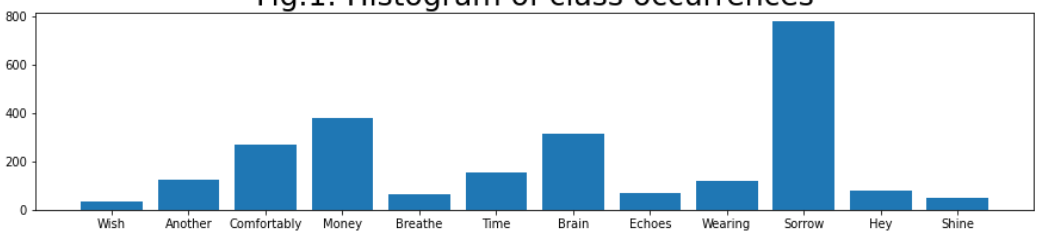
```
{0:'Wish',
1:'Another',
2:'Comfortably',
3:'Money',
4:'Breathe',
5:'Time',
6:'Brain',
7:'Echoes',
8:'Wearing',
9:'Sorrow',
10:'Hey',
11:'Shine'}
```

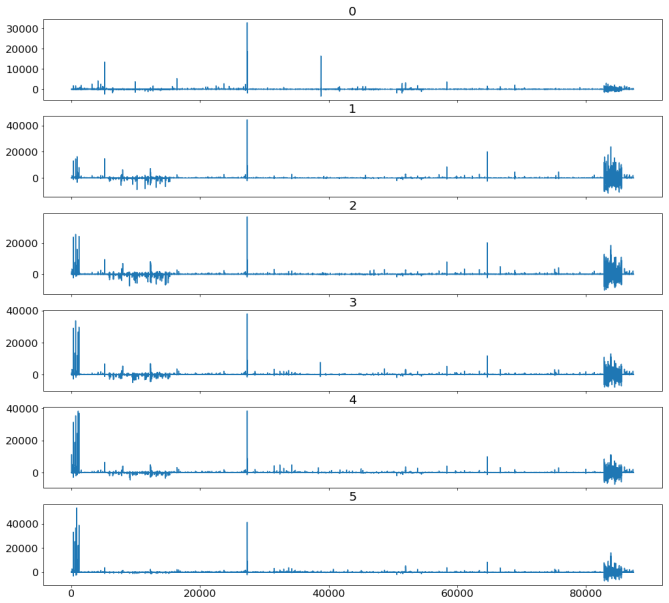
In the figure below (Fig.1), the histogram with occurrences shows that classes are not balanced. We introduced class weights to balance the training, but it didn’t seem to improve our results, therefore we excluded it.

1.1) Preprocessing

As we can see in Fig.2, the dynamics of the signals change a lot from sample to sample. The different channels may contain the same signal but scaled or delayed, or may not. We found our networks to learn better from the 6 channels together rather than analyzing each separately. The amplitude changes over several orders of magnitude in different channels but also from sample to sample, therefore we decided to normalize the input data. Normalization in [0,1] range was not improving our results in any way (considering the samples and/or the channels as a whole or not); we found better results on BiLSTM and CNN networks when standardizing each sample separately, i.e. with 36-long time windows with zero mean and unitary standard deviation as input. We also tried to aggregate and break up samples (respectively to 72 and 18 time steps each). Longer samples contain more features and are classified way better (up to +5% accuracy on our validation subset), but we only could construct them knowing the labels a priori, whereas padding techniques didn’t work.

Fig.1: Histogram of class occurrences





1.2) Analysis in the frequency domain

Z. Nasrullah and Y. Zhao [3] used spectrograms to perform music artist classification. Samples (some minutes long songs sampled at 44.1 kHz) are divided in time windows, and from each segment the square of the Fourier transform is computed. We tried to treat our samples as 36 time steps windows of some audio signal, computing their Mel spectrograms and using these as input for a 2D-CNN, which didn't work - the network could not learn this data at all. Still, we added to our initial 36x6 data set two channels containing respectively the magnitude and the angle (ρ,θ) of the discrete Fourier transform DFT of the mean value of the other 6 channels. These two new features are computed on the single x sample only, so we could use them both for classifying our validation subset and the independent test set, with a great improvement in terms of accuracy (up to +6% on the test set with a BiLSTM network)

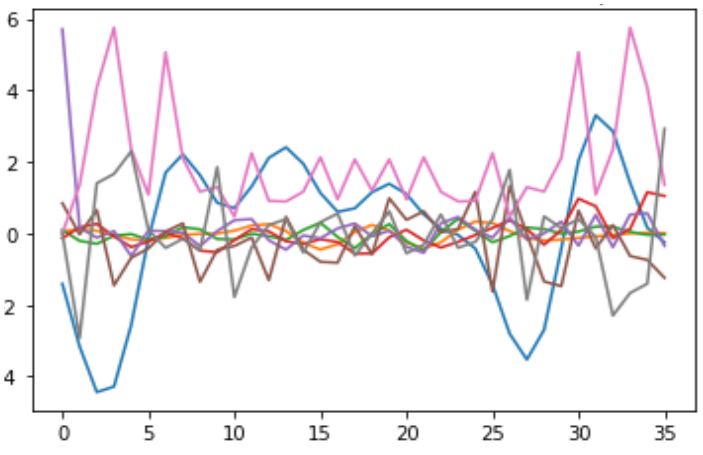


Fig-3: sample normalized with abs(fft) in pink and angle(fft) in blue

2) Our neural network

For this competition, we mostly used a 1D convolutional neural network. Since we tested several different networks, also thanks to the Keras Tuner library, we decided to report here only the results obtained from single variations from the final network, unless otherwise specified, following our layers from the input to the output and not necessarily our actual attempts in temporal order. An example of hyperparameter tuning for a BiLSTM model is given in Fig. 4.

2.1) Model

We started comparing the most common models shown during the course lectures: LSTM/BiLSTM networks and 1D-CNNs. In the end we tried to follow the hints and designed an attention based model. Simple LSTM and Transformer models (architecture from Attention is all you need[4]) didn't reach a 65% accuracy on the validation subset, even after long tuning sessions of hyperparameters search performed with bayesian optimization.
In the end, we decided to use the 1D-CNN as feature extractor because it gave us the best results with less operations. Our final architecture is:

- **1D-Convolution** with 512 neurons, kernel size = 3, ReLU activation function, padding 'same'
- **MaxPooling** layer
- **1D-Convolution** with 512 neurons, ReLU activation function, padding 'same'
- **GlobalAveragePooling layer:** we used a GAP layer to connect the features maps obtained to the fully connected network, which is known to give great generalization results; it reduced the dimension of the feature maps, decreasing the overall computational cost of the model.
- **Dropout layer** with rate 0.3**;** dropout is known to be a good regularization technique to reduce overfitting.

- **Dense layer** with 1024 units and ReLU activation function.
- **Dropout layer** with rate 0.3
- **Output layer**: 12 neurons with softmax activation function.

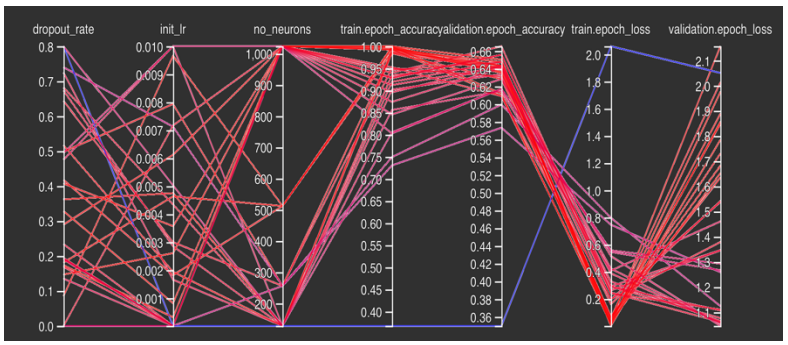


Fig.4: example of tuning results for dropout rate, initial learning rate and the number of neurons in a biLSTM architecture with Tensorflow

2.2) Training

The final model has 1.337.356 trainable parameters. We used categorical cross entropy as loss measure and accuracy as metric for stochastic gradient descent method with Adam optimization algorithm. The dataset was split in 80% for training and 20% for validation. We used learning rate scheduler "reduceLROnPl" and Early Stopping callback, both referred to accuracy loss. The network was fed with batches of 32x(36x8) samples at each step.

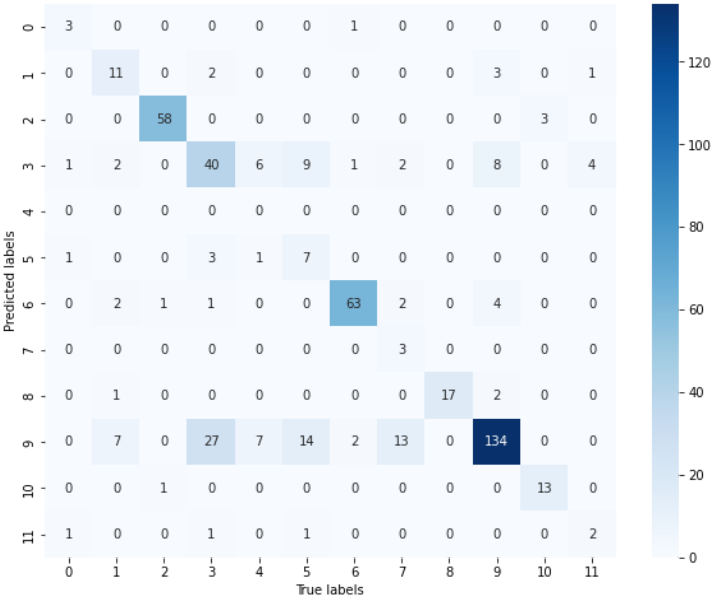
3) Results

The results on training and validation sets were:

- loss = 0.2951
- accuracy = 0.9022
- validation_loss = 1.0769
- validation_acc = 0.7119

The confusion matrix in the image was constructed with predictions on our validation set. The model classified the test sets with percentage accuracies:

- 72.07% (development phase)
- 73.07% (final phase)



4) References

- [1] https://www.tensorflow.org/api_docs
- [2] *Towards Effective Time Series Classification of Multi-Class Imbalanced Learning,* K Priyadarsini et al., 2021
- [3] *Music artist classification with convolutional recurrent neural networks,* Z.Nasrullah, Y.Zhao, 2019
- [4] *Attention is all you need*, A. Vaswani et al., 2017