Machine Learning project: Urban Sound 8k Classification with neural networks

Letizia Molinari ID 959194

January 13, 2022

Abstract

The aim of the following project is the training of neural networks for sound classification on the Urban Sound 8k dataset. In particular, Mel-Frequency Cepstral Coefficients (MFCCS) and Melspectrograms have been extracted, and Recurrent Neural Networks, Convolutional Neural Networks, Feedforward Neural Networks and LeNet-5 Convolutional Neural Networks have been implemented in order to compare the performance of different feature extraction methods and network architectures.

1 Introduction

The automated classification of environmental sounds is becoming nowadays a very interesting topic, since its usage can be helpful in everyday life, in order to implement for instance systems of remote surveillance or home automation. Environmental sounds are various non-human sounds, which differ to speech and music sounds since they do not have a common structure, while music sounds have. Since neural networks have proven to be able to handle huge quantities of data to model complex features, they can be applied for facing this challenging task. The deep learning approach for environmental sound classification applied in this project is the following one: perform normalization operations on the audio files, extract features from them and pass these features to the classifier in order to perform classification. In this project, Mel-frequency cepstral coefficients and Melspectrograms have been extracted from the audio data, and Recurrent Neural Networks, Convolutional Neural Networks, Feedforward Neural Networks and Le-Net5 Convolutional Neural Networks have been implemented.

2 Dataset Description

The analysis has been performed on the Urban Sound 8k dataset, which is a collection of 8732 excerpts of urban sounds labeled with ten classes: air conditioner, car horn, children playing, dog bark, drilling, engine idling, gun shot,

jackhammer, siren and street music. An audio file is presented as a numpy array, where each entry is a discretized sample having a sampling rate of 44100 Hertz. Each audio file approximately has a duration of four seconds. The dataset is given with ten folds, and the models will be trained on folds 1, 2, 3, 4, 6 and will be tested on folds 5, 7, 8, 9, 10.

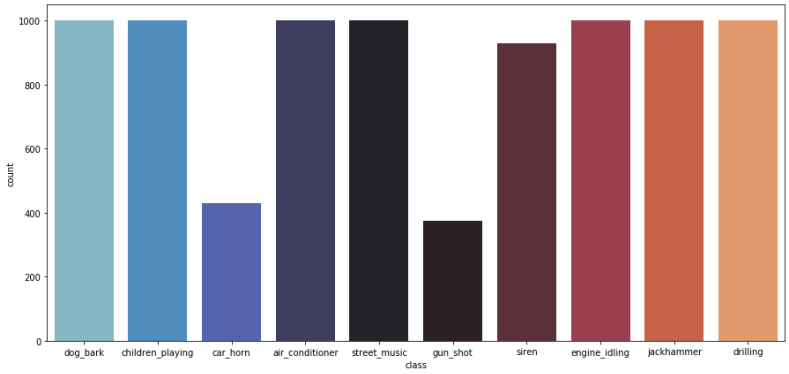


Figure one: Class distribution

By looking at the bar plot in figure one, it is possible to observe that the dataset is not perfectly balanced, since not all the classes have the same distribution.

2.1 Feature Extraction

In the following report, two kinds of features have been extracted:

- Mel-frequency cepstral coefficients(MFCCs), which are the coefficents that make up a mel-frequency cepstrum, which is the representation of the short-term power spectrum of a sound. It has been decided to extract 50 Mfccs with the librosa.feature.mfcc() function.

- Melspectrograms, which are spectrograms whose frequencies are converted to the mel scale, which is a unit of tone proposed by Stevens, Volkmann, and Newmann, according to which equal distances in tone sound equally distant to the listener. It has been decided to extract 50 mels, by applying librosa.feature.melspectrogram() function. Then the librosa.amplitude-to-db() has been implemented in order to convert the amplitude spectrogram to a deciBel scaled spectrogram.

Before extracting features, the librosa.util.normalize() function has been applied to the audio files, in order to perform normalization.

2.2 Feature Visualization

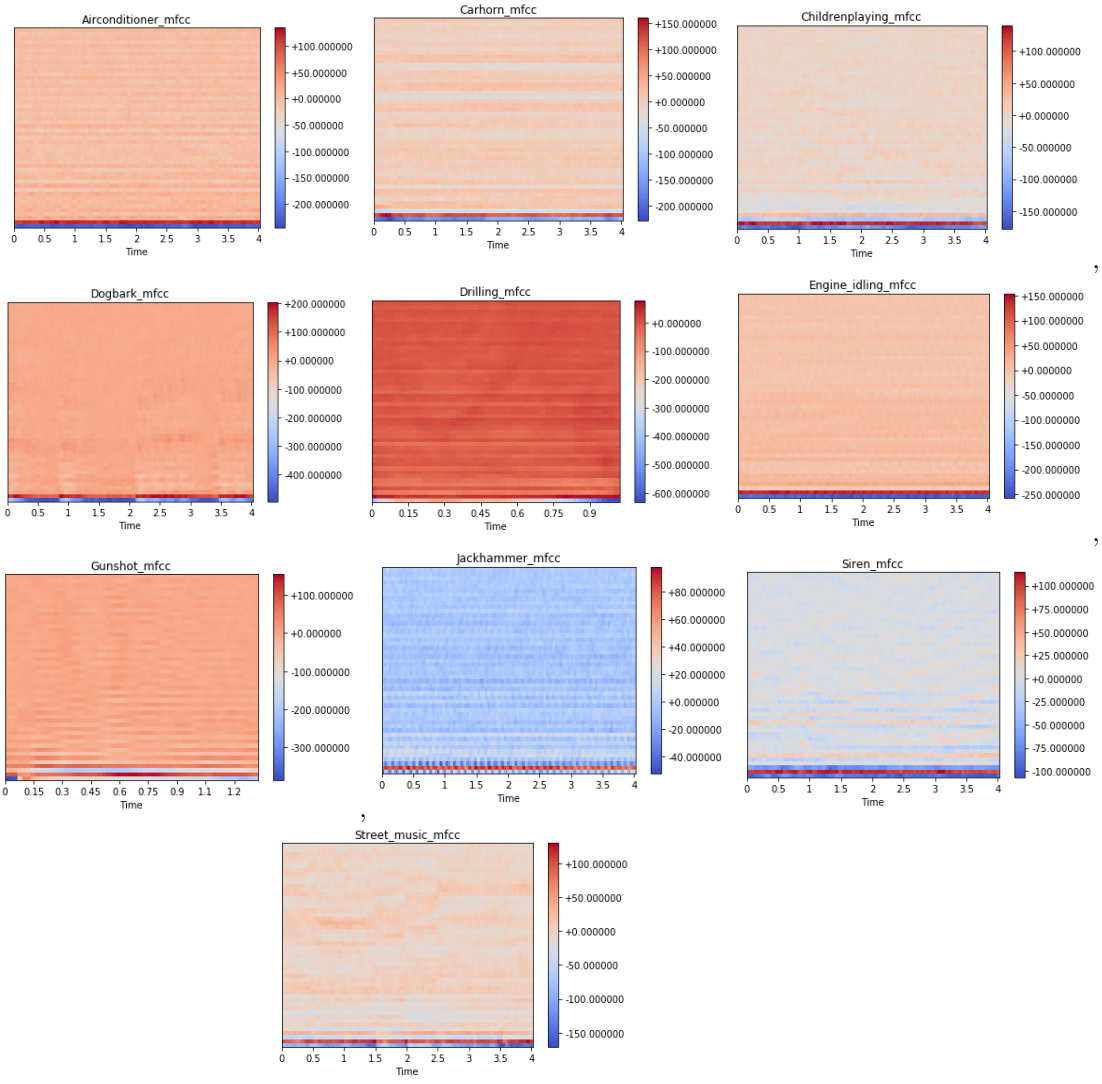


Figure two: Visualization of the Mfccs for the different classes of urban sounds

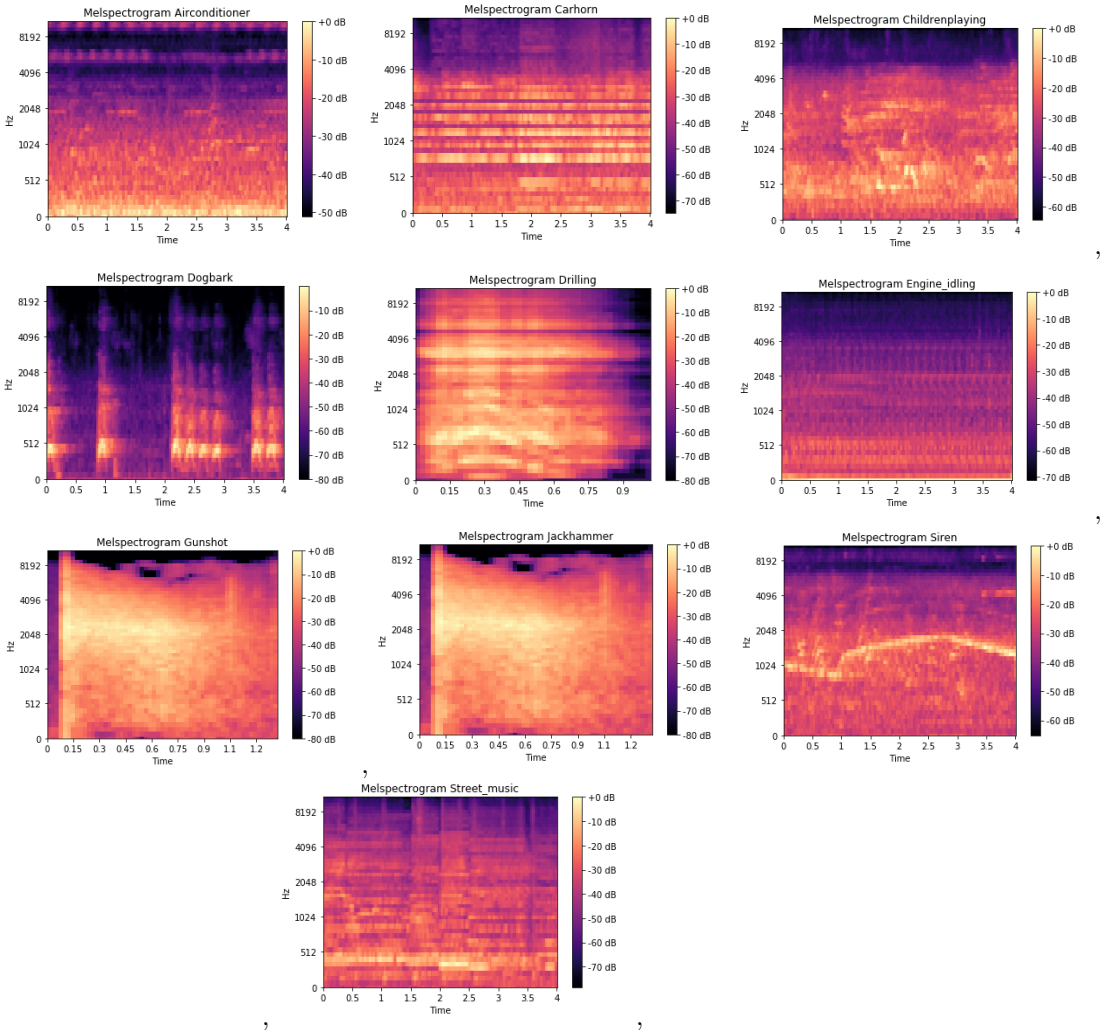


Figure three: Visualization of the Melspectrograms for the different classes of urban sounds

By looking at the images in figures two and three, it is possible to visualize the mel-frequency cepstral coefficients and the melspectrograms for each class of audio files. By looking at these figures, it is possible to observe that each class of audio files presents different traits.

3 Methodologies implemented

In the following project four different network architectures have been implemented:

- Feedforward Neural Networks(FNN), which are a class of artificial neural networks in which the connections between the nodes do not form a cycle and, during the classification operation, there is not feedback between the layers. As a consequence, the output is independent from the previous outputs. They are the simplest forms of neural networks, since information only goes in one direction;

Layer(type)	Output Shape	Param#
flatten_2 (Flatten)	(None, 8700)	0
dense_7 (Dense)	(None, 256)	2227456
dense_8 (Dense)	(None, 128)	32896
dense_9 (Dense)	(None, 64)	8256
dense_10 (Dense)	(None, 10)	650

Table one: structure of the FNN implemented in this project

- Recurrent Neural Networks(RNN), which are a class of artificial neural networks which do not only use the information available at time t in order to make a prediction, but they also use the information coming from the prediction at time t-1. As a consequence, the output of these networks depends on prior elements within the sequence;

Layer(type)	Output Shape	Param#
lstm (LSTM)	(None, 50, 128)	155136
dropout (Dropout)	(None, 50, 128)	0
dense (Dense)	(None, 50, 128)	16512
dense_1 (Dense)	(None, 50, 64)	8256
dropout_1 (Dropout)	(None, 50, 64)	0
dense_2 (Dense)	(None, 50, 48)	3120
dropout_2 (Dropout)	(None, 50, 48)	0
flatten (Flatten)	(None, 2400)	0
dense_3 (Dense)	(None, 10)	24010

Table two: structure of the RNN implemented in this project

- Convolutional Neural Networks(CNN), which are a class of artificial neural networks which take an image as input, assign learnable weights and biases to the various aspects in the image and are able to differentiate an image from another one;

Layer(type)	Output Shape	Param#
conv2d (Conv2D)	(None, 48, 172, 16)	160
max_pooling2d(MaxPooling2D)	(None, 24, 86, 16)	0
dropout_3 (Dropout)	(None, 24, 86, 16)	0
conv2d_1 (Conv2D)	(None, 22, 84, 32)	4640
max_pooling2d_1(MaxPooling2D)	(None, 11, 42, 32)	0
dropout_4 (Dropout)	(None, 11, 42, 32)	0
conv2d_2 (Conv2D)	(None, 9, 40, 64)	18496
max_pooling2d_2(MaxPooling2D)	(None, 4, 20, 64)	0
dropout_5 (Dropout)	(None, 4, 20, 64)	0
conv2d_3 (Conv2D)	(None, 4, 10, 128)	73856
max_pooling2d_3(MaxPooling2D)	(None, 2, 10, 128)	0
dropout_6 (Dropout)	(None, 2, 10, 128)	0
flatten_1 (Flatten)	(None, 2560)	0
dense_4 (Dense)	(None, 256)	655616
dropout_7 (Dropout)	(None, 256)	0
dense_5 (Dense)	(None, 256)	65792
dropout_8 (Dropout)	(None, 256)	0
dense_6 (Dense)	(None, 10)	2570

Table three: structure of the CNN implemented in this project

- LeNet-5 Convolutional Neural Networks, which are the first Convolutional Neural Networks which have been introduced for image classification. It has been decided to use this network architecture in order to compare it with the Convolutional Neural Networks.

Layer(type)	Output Shape	Param#
conv2d_4(Conv2D)	(None, 50, 174, 32)	320
max_pooling2d_4(MaxPooling2D)	(None, 25, 87, 32)	0
conv2d_5 (Conv2D)	(None, 23, 85, 32)	9248
max_pooling2d_5(MaxPooling2D)	(None, 11, 42, 32)	0
conv2d_6 (Conv2D)	(None, 9, 40, 64)	18496
max_pooling2d_6(MaxPooling2D)	(None, 4, 20, 64)	0
flatten_3 (Flatten)	(None, 5120)	0
dense_11 (Dense)	(None, 512)	2621952
dropout_9 (Dropout)	(None, 512)	0
dense_12 (Dense)	(None, 10)	5130

Table four: structure of the LeNet-5 CNN implemented in this project

For all of the architectures which have been implemented, it has been decided to adopt the 'Categorical Crossentropy' as a loss function for the training and it has been chosen Adam as optimizer, with a learning rate of

$$1^(e-3) \tag{1}$$

. In all the layers except for the last one the ReLu (Rectified Linear Unit) activation function has been used. The Softmax activation function has been adopted in the last layer. It has also been chosen a validation split of 0.2. In order to evaluate the performance of the models on the training and validation sets, the Accuracy measure has been selected, which is the percentage of samples which have been correctly predicted by a certain model. The performance of each model in each test fold has again been evaluated by the Accuracy score and by the Standard Deviation. It has been set a batch size of 24 for each of the four models and a number of epochs equal to 50.

4 Results

Overall, eight different combinations of network architectures and features have been implemented: feedforward neural networks with mel-frequency cepstral coefficients as input, recurrent neural networks with mel-frequency cepstral coefficients as input, convolutional neural networks with mel-frequency cepstral coefficients as input, LeNet-5 convolutional neural networks with with mel-frequency cepstral coefficients as input, feedforward neural networks with melspectrograms as input, recurrent neural networks with melspectrograms as input, convolutional neural networks with melspectrograms as input and LeNet-5 convolutional neural networks with with melspectrograms as input.

Model	Network Architecture	Input	Trained parameters
Model 1	Feedforward NN	Mfccs	$2,269,258$
Model 2	Recurrent NN	Mfccs	$207,034$
Model 3	Convolutional NN	Mfccs	$821,130$
Model 4	LeNet-5 NN	Mfccs	$2,655,146$
Model 5	Feedforward NN	Melspectrograms	$2,269,258$
Model 6	Recurrent NN	Melspectrograms	$207,034$
Model 7	Convolutional NN	Melspectrograms	$821,130$
Model 8	LeNet-5 NN	Melspectrograms	$2,655,146$

Table five: summary of the various network architectures and inputs implemented

Model	Fold 5	Fold 7	Fold 8	Fold 9	Fold 10
Model 1	0.465	0.465	0.465	0.465	0.465
Model 2	0.240	0.240	0.240	0.240	0.240
Model 3	0.609	0.609	0.609	0.609	0.609
Model 4	0.572	0.572	0.572	0.572	0.572
Model 5	0.464	0.464	0.464	0.464	0.464
Model 6	0.239	0.239	0.239	0.239	0.239
Model 7	0.585	0.585	0.585	0.585	0.585
Model 8	0.482	0.482	0.482	0.482	0.482

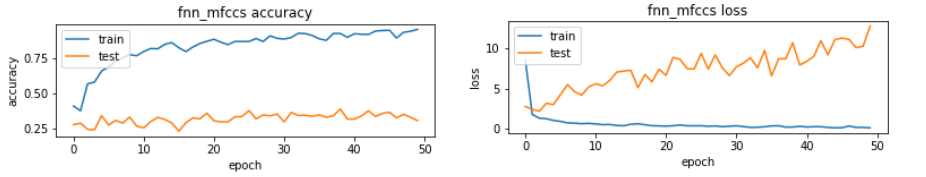
Table six: accuracy scores in each test fold

Model	Average Accuracy score	Standard Deviation
Model 1	0.465	0.0
Model 2	0.240	0.0
Model 3	0.609	0.0
Model 4	0.572	0.0
Model 5	0.464	$5.55e-17$
Model 6	0.239	$2.776e-17$
Model 7	0.585	0.0
Model 8	0.482	$5.55e-17$

Table seven: average accuracy score across the test folds and standard deviation

4.1 Comments on the results

By looking at the results reported in tables six and seven, it is possible to observe the accuracy scores in each test fold and the average accuracy score. It is possible to state that the accuracy score is more or less the same in each test fold. It is possible to observe that Model 3, that had Mfccs as input and a Convolutional Neural Network architecture, is the one having the highest average accuracy score, which is 0.603. Then it is possible to observe that model 7, in which Convolutional Neural Networks have been implemented, but with Melspectrograms as inputs, has an accuracy score of 0.585. Then, model 4 and model 8, which had a LeNet-5 Convolutional Neural Network architecture and, respectively, Mfccs and Melspectrograms as inputs, have average accuracy scores of 0.572 and 0.482. Then, model 1 and model 5, which have respectively Mfccs and Melspectrograms as inputs and a Feedworward Neural Network architecture, have average accuracy scores of 0.465 and 0.464. Model 2 and model 6, which are those having a Recurrent Neural Network architecture, are those with the lowest average accuracy score: it is equal to 0.240 for model 2 and it is equal to 0.239 for model 6.



,

Figure four: accuracy and loss of model 1

By looking at figure four, it is possible to observe the training and validation accuracy and the training and validation loss for model 1 across the epochs.It is possible to state that this model has a low accuracy in the test set and meets overfitting.

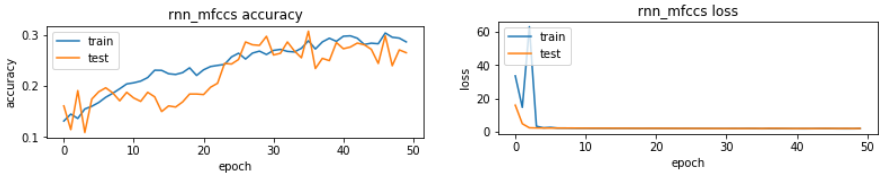


Figure five: accuracy and loss for model 2

By looking at figure five, it is possible to observe that the training an test accuracy across the epochs are quite the same, but they are very low: their accuracy score is less than 0.30, while the loss goes down.

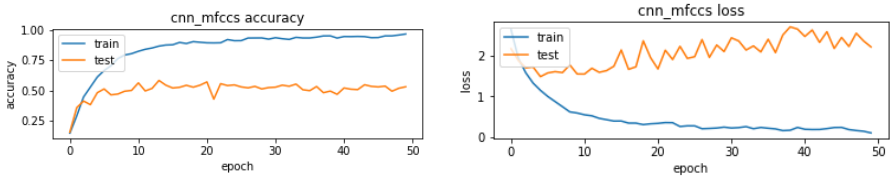


Figure six: accuracy and loss for model 3

By looking at figure six, it is possible to observe that the accuracy in the training set is high, while in the test set it is around 0.50. For what concerns the loss, it is possible to observe that there is a difference of two points between the training loss and the test loss.

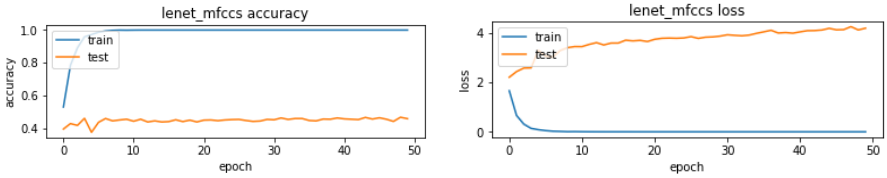


Figure seven: accuracy and loss for model 4

By looking at figure seven, it is possible to observe again that the accuracy across the epochs is quite high for the training set, but it is never over 0.40 for the test set. It is also possible to observe that this model suffers from overfitting: while the training loss decreases across the epochs, it does not happen the same in the test set.

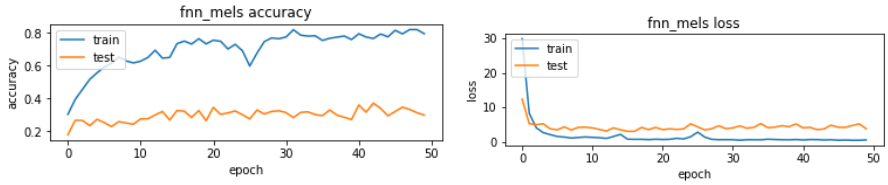


Figure eight: accuracy and loss for model 5

By looking at figure eight, it is possible to observe that this model does not have a high performance on the test set and does not suffer from overfitting.

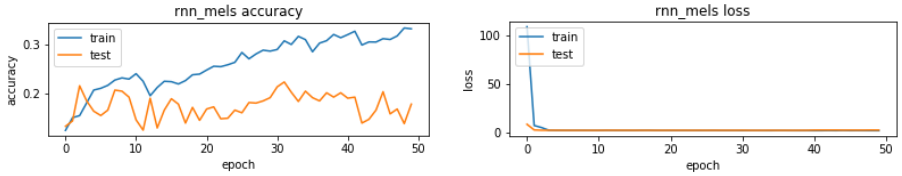


Figure nine: accuracy and loss for model 6

By looking at figure nine, it is possible to observe that the performance of this model is quite bad, and the loss is low.

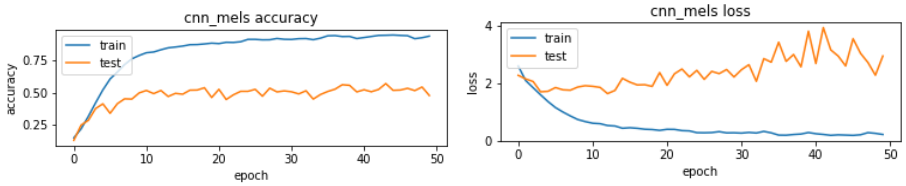


Figure ten: accuracy and loss for model 7

By looking at figure ten, it is possible to observe that the performance of this model in the test set across the epochs is around 0.50. For what concerns the loss, it is possible to observe that there is a difference of two points between the training loss and the test loss.

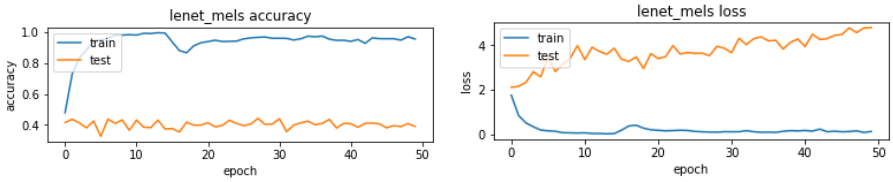


Figure eleven: accuracy and loss for model 8

By looking at figure ten, it is possible to say that the accuracy in the test set is around 0.40, while it is very high in the training set across the epochs. For what concerns the loss, it is possible to observe that the model suffers from overfitting: while the training loss decreases across the epochs, it is not possible to say the same for the loss in the test set.

5 Conclusions

By looking at the various models implemented, it is possible to conclude that the performances of these models is not very high: in particular, it never exceeds a score of 0.60. It is possible to observe that models two and six, in which Recurrent Neural Networks have been implemented, have the worst performance for both the features which have been used as inputs. By looking at models one and five, which have a Feedforward Neural Network architecture, it is possible to state that they do not have a good performance, but while model one suffers from overfitting, it is possible to say the contrary for model five. For what concerns models four and eight, it is possible to affirm that their performance is

better if compared to the one of the previous two models, but they suffer from overfitting. In the end, it is possible to say that models three and seven, in which Convolutional Neural Networks have been implemented, are those with the highest accuracy score in the test set if compared to the other network architectures, and the discrepancy between training and test loss is of two points. The performances of the eight models with respect to the two kinds of features which have been extracted are more or less the same, except for the case of models one and five: while model one suffers from overfitting, model five does not. It is possible to conclude that the performances of these models are not very elevated, even though hyperparameters have not been tuned, so probably the performances could be improved by a regulation of the hyperparameters. This could be considered an open challenge for subsequent works.

6 Declaration

I declare that this material, which I now submit for assessment, is entirely my own work and has not been taken from the work of others, save and to the extent that such work has been cited and acknowledged within the text of my work. I understand that plagiarism, collusion, and copying are grave and serious offences in the university and accept the penalties that would be imposed should I engage in plagiarism, collusion or copying. This assignment, or any part of it, has not been previously submitted by me or any other person for assessment on this or any other course of study.

7 References

- https://urbansounddataset.weebly.com/urbansound8k.html
- https://www.justinsalamon.com/uploads/4/3/9/4/4394963/salamon$_u$*rbansound*$_a$*cmmm*14.*pdf*

- https://en.wikipedia.org/wiki/Mel-frequency$_c$*epstrum*
- https://medium.com/analytics-vidhya/understanding-the-mel-spectrogram-fca2afa2ce53
- https://www.sciencedirect.com/science/article/pii/S1877050917316599
- http://conference.scipy.org/proceedings/scipy2015/pdfs/brian$_m$*cfee.pdf*