UNIVERSITÀ DI PISA

Computer Engineering

Intelligent Systems

Project Documentation

Matteo Guidotti

Academic Year: 2020/2021

Table of Contents

1 Introduction

The project consists of several tasks. The ultimate goal is to design and develop an intelligent system that measures a person's emotional state using signals recorded by sensors and to develop an emotion classifier that uses images of faces as input.
The tasks I decided to carry out are:

- **task 3.1**: Design and develop two MLP networks and two RBF networks that accurately estimate a person's valence and arousal levels.
- **task 3.3**: Design and develop a fuzzy inference system to estimate a person's arousal.
- **task 4.2**: Perform fine-tuning of a pre-trained CNN to accurately classify a person's emotions based on facial expressions.

2 | Data Preprocessing

Before starting to work with the provided data it is necessary to pre-process the dataset, it was performed as follows:

- Removal of non-numeric data and infinite values by means of the function *isinf* of MATLAB
- Removal of outliers using the function *rmoutliers*, by means of the default median method
- Balancing of the dataset among the different values of *arousal* and *valence*
- Features selection
- Normalization of data

All these steps are performed in the *data_preparation.m* file. The most interesting part of this process is constituted by the last steps.

2.1 Data Balancing

The dataset is composed by samples and each sample contains biomedical signals: to each different set composed by 54 biomedical signals (that we will call features) correspond a value for arousal and a value for valence. The possible values are 7, so that we are able to divide the dataset according to 7 classes, for arousal and valence levels. In the following the distributions of the samples among the classes are represented in two histograms.

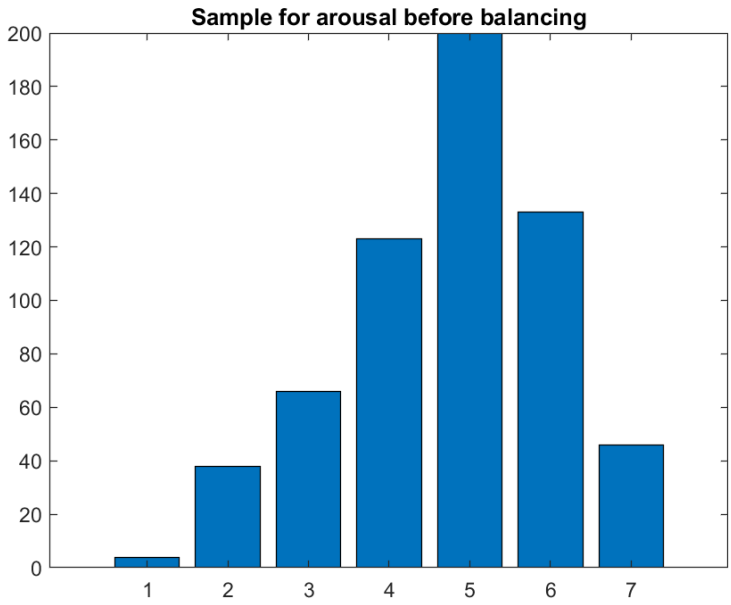

Figure 2.1: Classes distribution for arousal before balancing

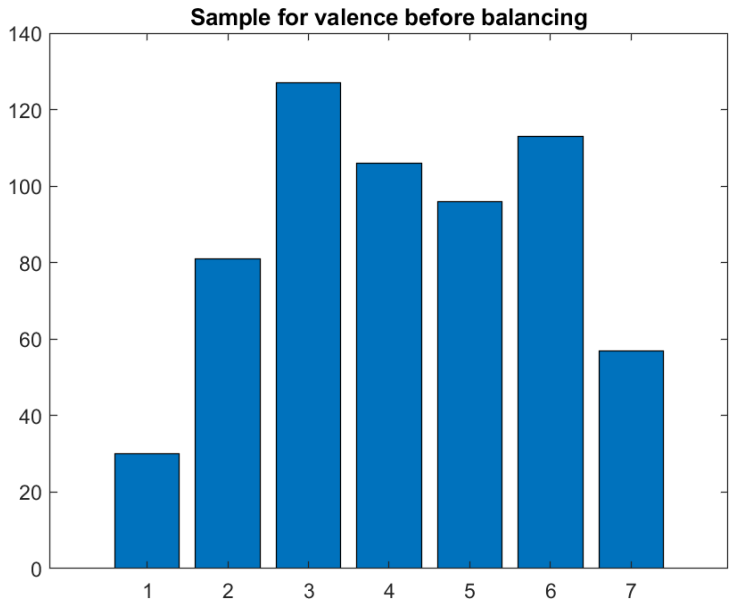

Figure 2.2: Classes distribution for valence before balancing

It is clear that the samples are very unbalanced; so we need to use an algorithm to make them balanced. The latter is based on the concept of data augmentation, obtained by multiplying selected data by a random number between $[0.95; 1.05]$. Data augmentation is applied on samples which belong to the least common class for arousal (valence) and not to the most common class for valence (arousal) and then the samples which belong to the most common class for arousal (valence) and don't belong to the least common class for valence (arousal) are removed. These steps were repeated *rep* times. After performing the balancing, these are the distributions of samples for both the arousal and the valence:

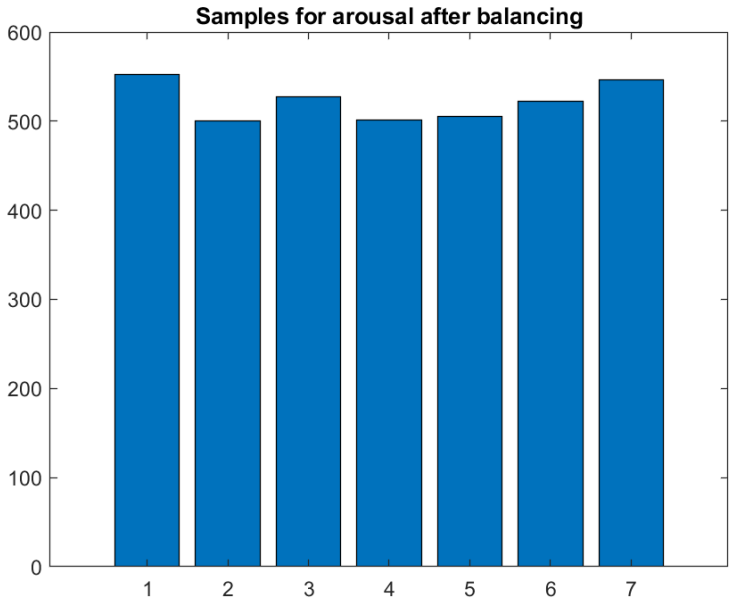

Figure 2.3: Classes distribution for arousal after balancing

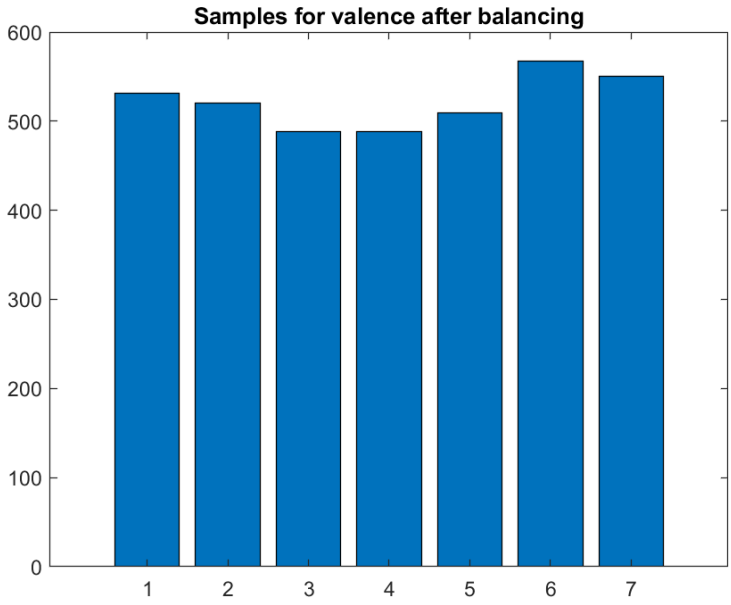

Figure 2.4: Classes distribution for valence after balancing

2.2 Features Selection

Before selecting the features it is necessary to divide the data in different sets: one for training and one for the testing phase. This is very important because if we use all the data to perform features selection we have a bias because the test data have been already seen by the network. In order to select the features, I performed 5 times *sequentialfs* for arousal levels and 5 times for the valence ones. The *sequentialfs* routine should be repeated a statistically relevant number of times, and 5 repetitions are not the case, but it is a good compromise done in order to maintain an acceptable number of features, an acceptable quality of data and, overall, a good execution time for the algorithm execution. Each selected feature has been traced in a counter vector and, at the end, this vector is sorted in descending order to obtain the list of the most important features. Obviously, a sorted vector for arousal and another one for the valence levels are obtained. At the end of this process, the obtained data have been saved into the following *.mat* files:

- *clean_dataset.mat* contains the entire dataset without infinite values, outliers and with balanced classes distributions
- *training_valence.mat* contains a struct with the training input only for the selected features and the relative target output for valence
- *training_arousal.mat* contains a struct with the training input only for the selected features and the relative target output for arousal
- *testing_valence.mat* contains a struct with the test input only for the selected features and the relative target output for valence
- *testing_arousal.mat* contains a struct with the test input only for the selected features and the relative target output for arousal

- *best3.mat* contains a struct with training and testing data for arousal of the 3 best features selected. It will be useful for the task 3.3

2.3 Normalization of data

The normalization of the data has been performed using the MATLAB function *normalize*, specifying only the vector to be normalized, without any other parameters. The function returns the vectorwise z-score of the data in A with center 0 and standard deviation 1.

3 | Multi-layer Perceptron Networks

The first part of the implementation of the task 3.1 is showed in this chapter, by showing the design and development of two MLP networks that have the objective to estimate the arousal and valence levels of a person. The results have been obtained thanks to experimentally derived configurations.

3.1 MLP network for arousal level

The configuration that have been used is:

```
% Optimal Neural Network Architecture found for arousal
hidden_neurons = 25;
mlp_net_arousal = fitnet(hidden_neurons);
mlp_net_arousal.divideParam.trainRatio = 0.7;
mlp_net_arousal.divideParam.testRatio = 0.1;
mlp_net_arousal.divideParam.valRatio = 0.2;
mlp_net_arousal.trainParam.showWindow = 1;
mlp_net_arousal.trainParam.showCommandLine = 1;
mlp_net_arousal.trainParam.lr = 0.1;
mlp_net_arousal.trainParam.epochs = 100;
mlp_net_arousal.trainParam.max_fail = 10;
```

Figure 3.1: Configuration for MLP network for arousal level

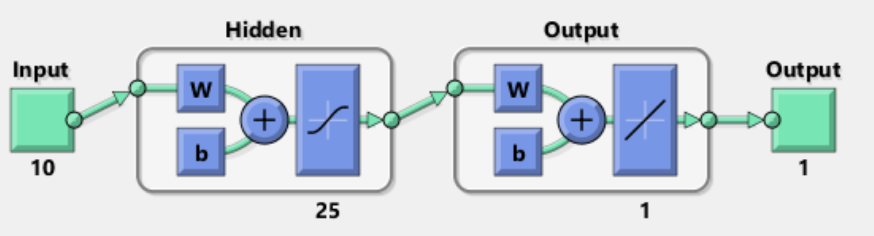

Figure 3.2: Architecture of the arousal MLP network

In order to be able to perform the early stopping technique, the validation set has been used:

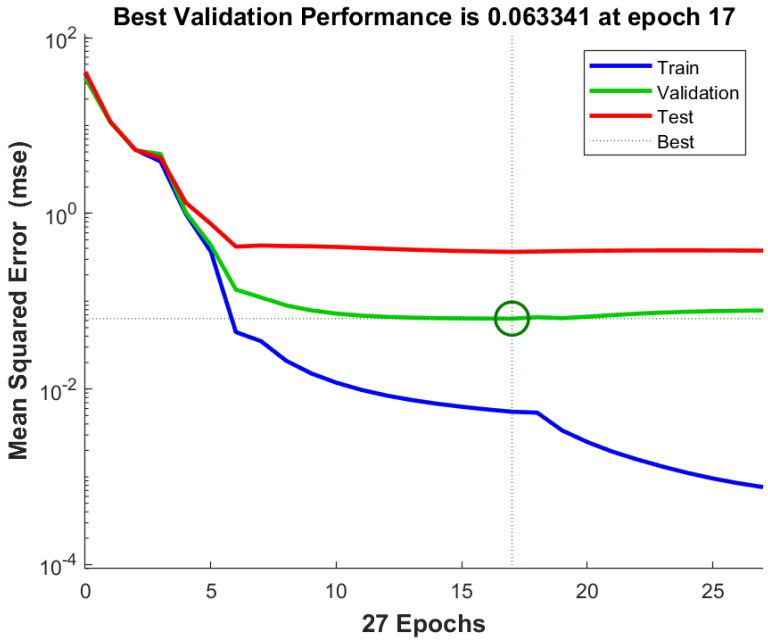

Figure 3.3: Best validation performance for arousal levels

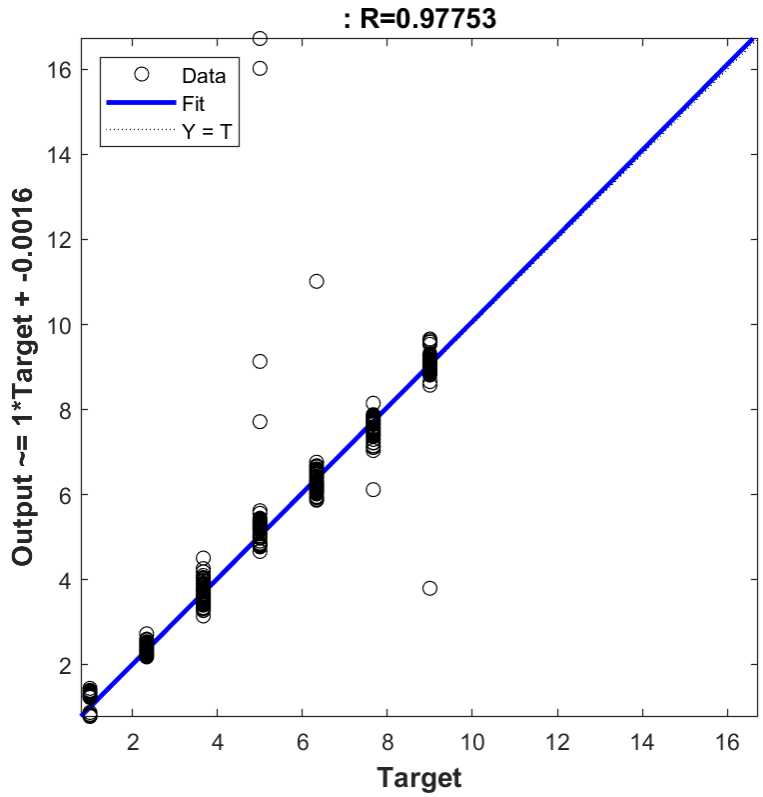

Figure 3.4: Regression for arousal MLP network

3.2 MLP network for valence level

The configuration that have been used is:

```
% Optimal Neural Network Architecture found for valence
hidden_neurons = 60;
mlp_net_valence = fitnet(hidden_neurons);
mlp_net_valence.divideParam.trainRatio = 0.7;
mlp_net_valence.divideParam.testRatio = 0.1;
mlp_net_valence.divideParam.valRatio = 0.2;
mlp_net_valence.trainParam.showWindow = 1;
mlp_net_valence.trainParam.showCommandLine = 1;
mlp_net_valence.trainParam.lr = 0.1;
mlp_net_valence.trainParam.epochs = 100;
mlp_net_valence.trainParam.max_fail = 15;
```

Figure 3.5: Configuration for MLP network for valence level

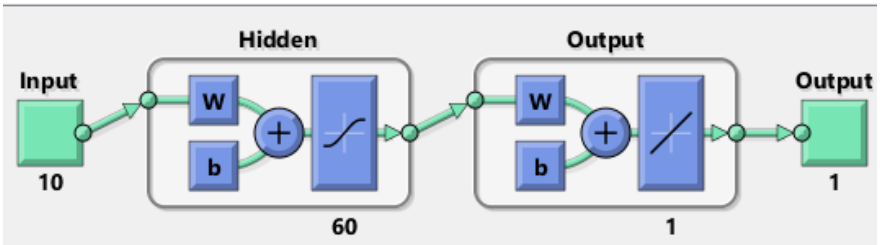

Figure 3.6: Architecture of the valence MLP network

In order to be able to perform the early stopping technique, the validation set has been used:

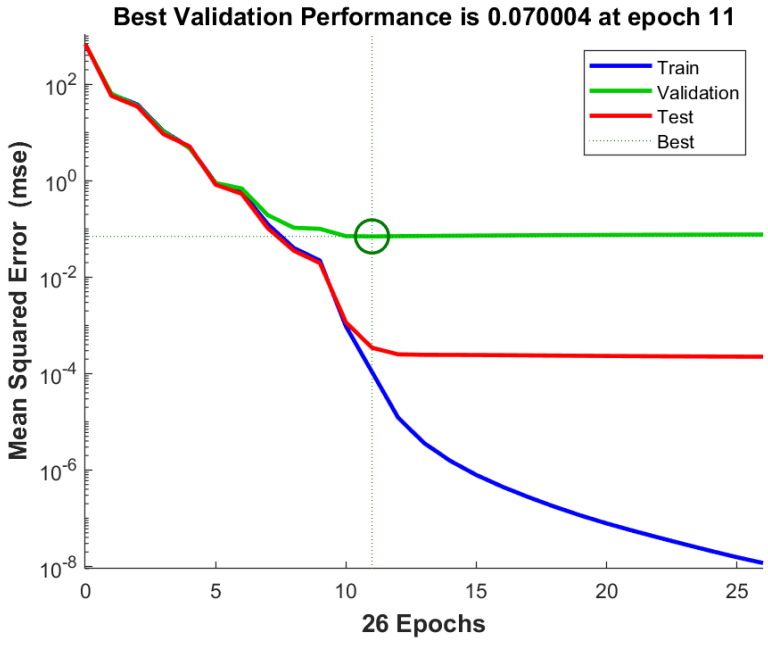

Figure 3.7: Best validation performance for valence levels

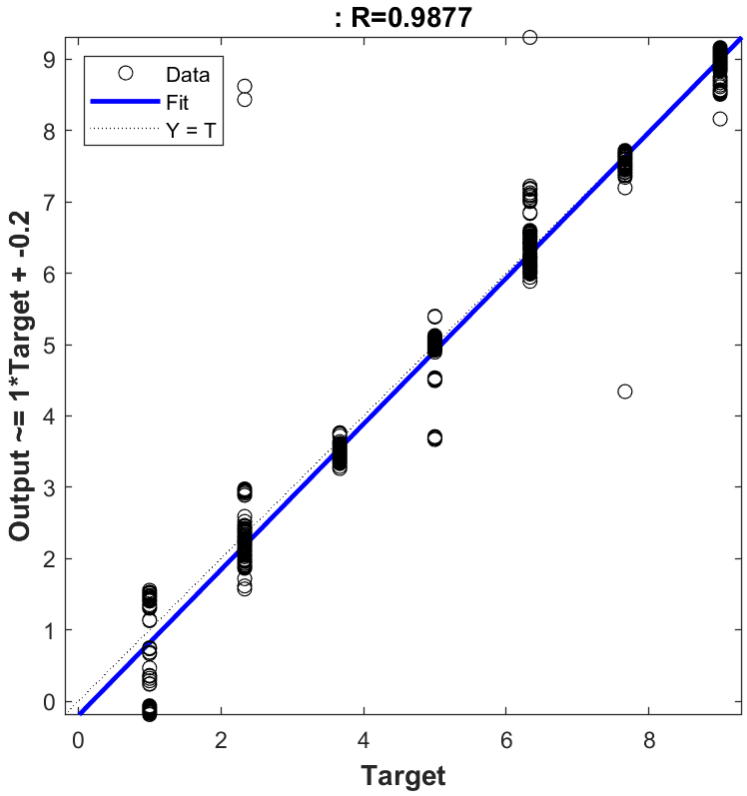

Figure 3.8: Regression for valence MLP network

As it is clear from the images above, the results in terms of error and regression values are very good, but we have also to state that they are highly variable depending on the starting point on the error surface, since the weights are taken randomly at the beginning of the training.

4 | Radial Basis Function Networks

In this section, the design and development of two RBF networks aiming to estimate arousal and valence levels of a person are showed, as asked in the second part of the task 3.1. As before, different experiments have been performed to find the best configuration, by varying the *spread* value. In this way I found that good results are achieved by means of a *spread* constant equal to 1, then the number of neurons will be set in order to achieve the goal (Mean Square Error), that has been set to 0.2 for both the networks. The configurations and the results for both the networks are:

```
%Creation of RBFN
goal_ar = 0.02;
spread_ar = 1;
K_ar = 200; %max neurons
Ki_ar = 20; %neurons to add
```

Figure 4.1: Configuration for RBF network for both arousal and valence levels

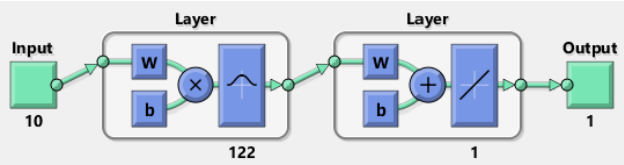

Figure 4.2: Architecture of RBF network for arousal level

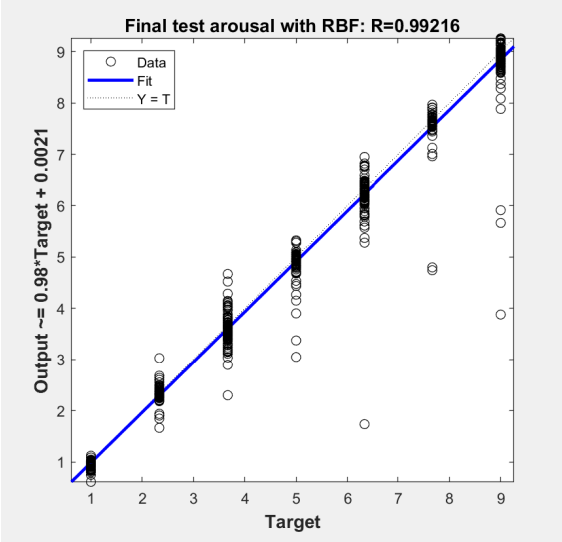

Figure 4.3: Regression for RBF network for arousal level

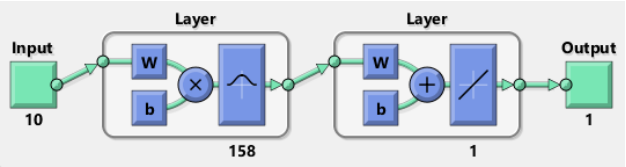

Figure 4.4: Architecture for RBF network for valence level

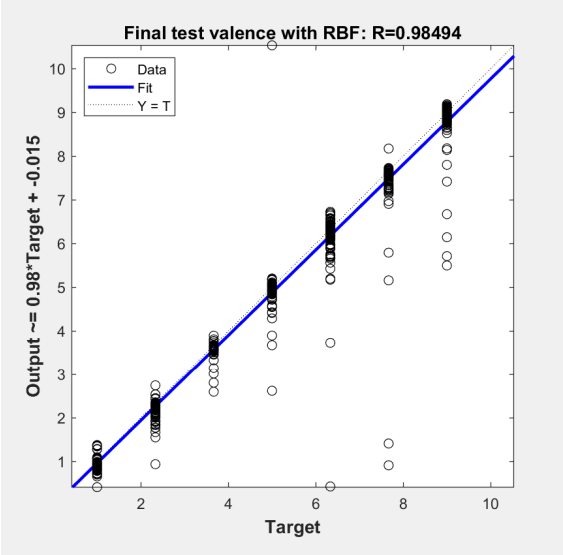

Figure 4.5: Regression for RBF network for valence level

5 | Fuzzy Inference System

After selecting the three best features by means of the *sequentialfs* routine, I studied the empirical distributions of them by means of some histograms, in order to be able to develop the Mamdani Fuzzy Inference System requested by the task 3.3. The selected features are:

- **Feature 24**: $[-2.17; 2.95]$
- **Feature 27**: $[-2.51; 1.81]$
- **Feature 35**: $[-2.8; 2.47]$

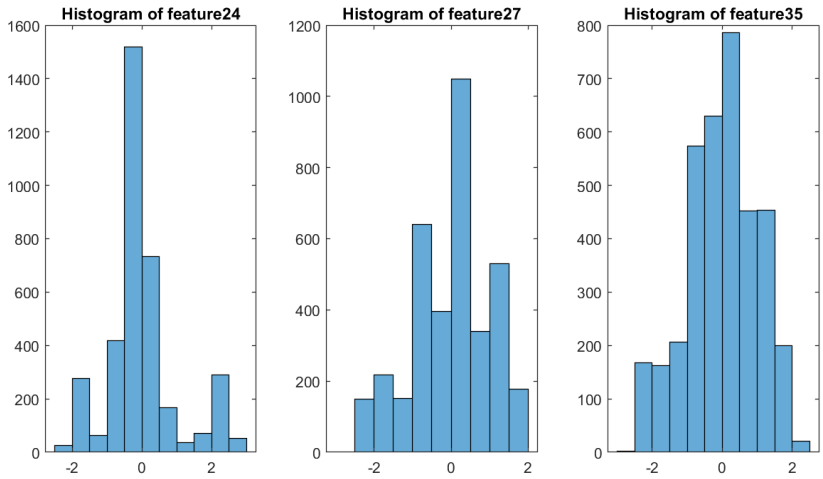

Figure 5.1: Empirical distributions of the three best features

The boundaries of each feature, together with their empirical distributions, have been used to model the linguistic variables of the features. The approach is that, since the empirical distributions are "fuzzy", but also tend to have peaks in some points, those peaks have been considered as central points of particular linguistic labels.

In order to define the rules of the fuzzy system, a correlation analysis with the different combinations of the 3 features was performed, to detect possible correlation between them.

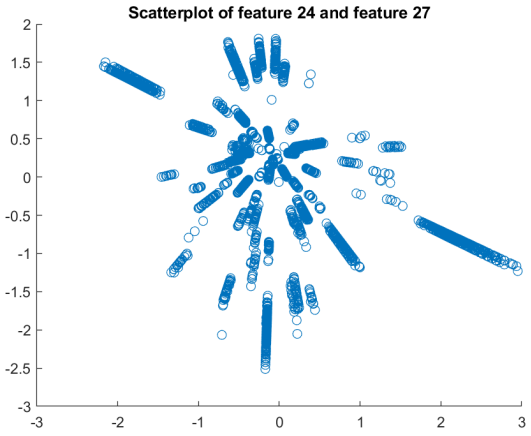

Figure 5.2: Correlation between feature 24 and 27

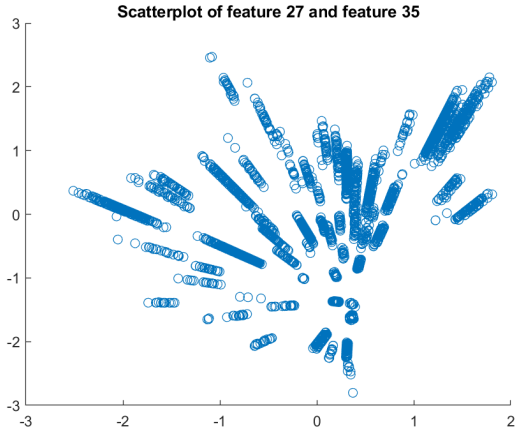

Figure 5.3: Correlation between feature 27 and 35

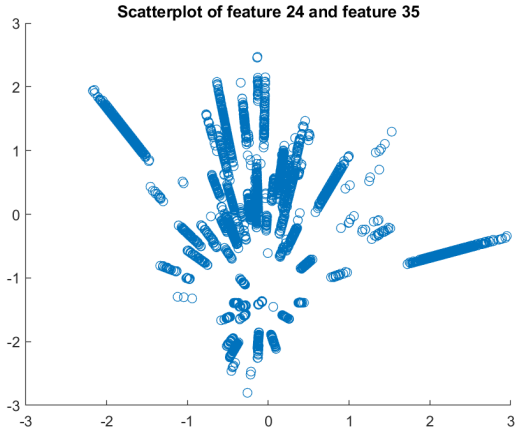

Figure 5.4: Correlation between feature 24 and 35

The features are clearly not correlated. Now we can define the rules by analyzing additional

histograms that show how many samples with low, medium, high output values correspond to the different values of the features. So we obtained 3 histograms for each feature:

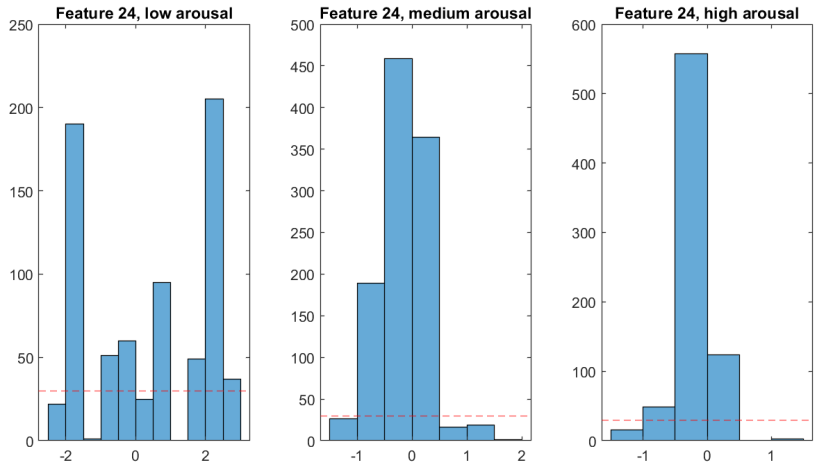

Figure 5.5: Number of samples for each arousal value among the different values of feature 24

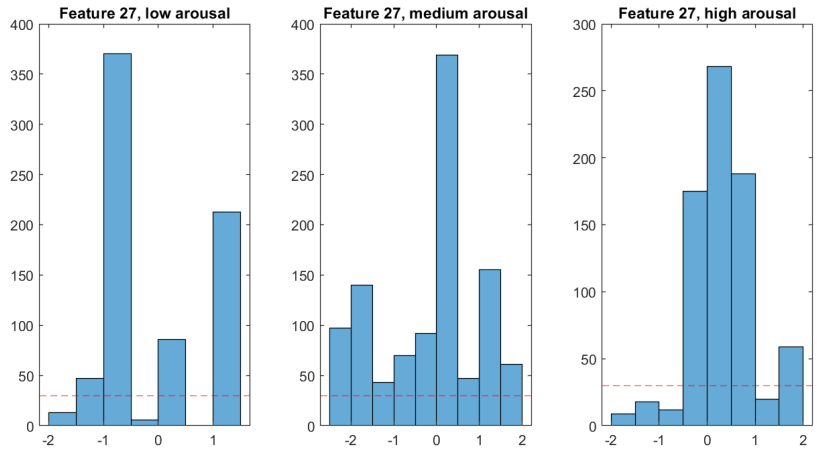

Figure 5.6: Number of samples for each arousal value among the different values of feature 27

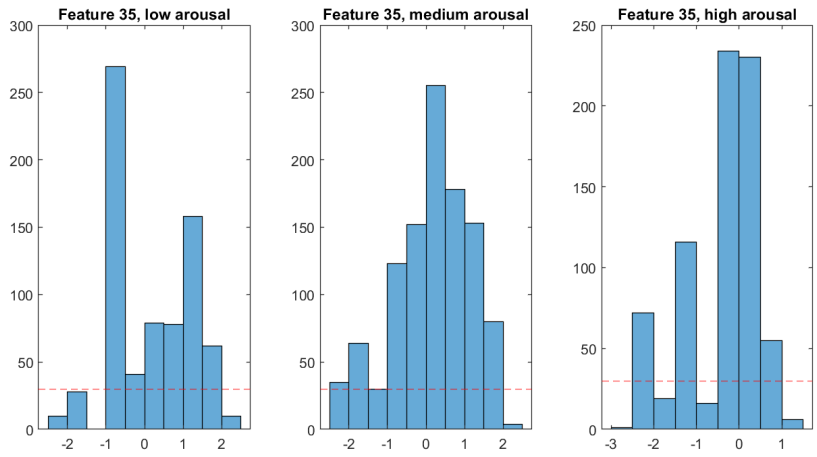

Figure 5.7: Number of samples for each arousal value among the different values of feature 35

Now the fuzzy system can be designed by means of the graphical interface provided by MATLAB *fuzzyLogicDesigner*. For what concerns the linguistic variable, the central membership function is always selected as a triangle and on the sides there are trapezoidal membership functions, in order to cover all the universe of discourse of each feature. The modelling in the input is showed in the following images

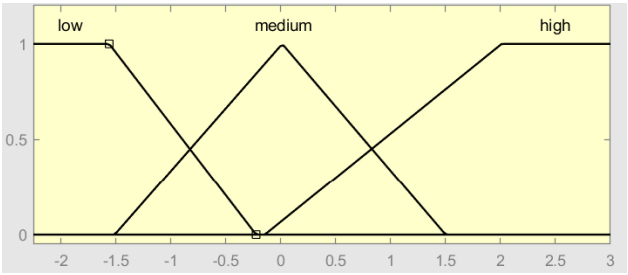

Figure 5.8: Feature 24 linguistic variable modeling

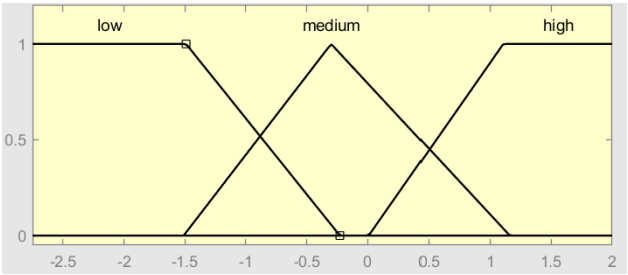

Figure 5.9: Feature 27 linguistic variable modeling

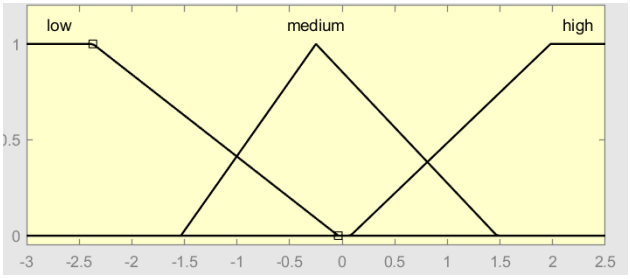

Figure 5.10: Feature 35 linguistic variable modeling

For what concerns the output, there are 7 possible arousal values, so we can divide the dataset in 7 classes. The first two classes were used to implement the low arousal value, while the third, fourth and fifth classes implement the medium level and the last two classes implement the high level.
The output has been modeled as follows:

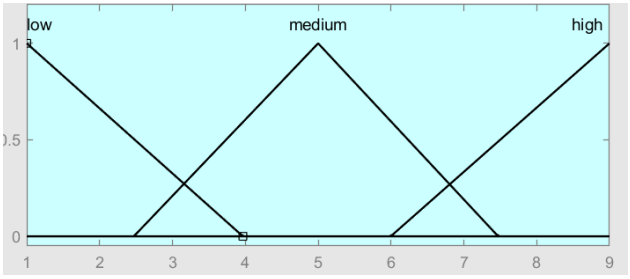

Figure 5.11: Output linguistic variable modeling

The final fuzzy rules of the system are:

1. If (Feature_24 is high) then (arousal is low) (1)
2. If (Feature_24 is medium) then (arousal is not low) (1)
3. If (Feature_24 is low) then (arousal is low) (1)
4. If (Feature_27 is low) then (arousal is medium) (1)
5. If (Feature_35 is medium) then (arousal is not low) (1)
6. If (Feature_35 is high) then (arousal is not high) (1)
7. If (Feature_35 is low) then (arousal is not low) (1)
8. If (Feature_35 is medium) then (arousal is not low) (1)
9. If (Feature_27 is medium) then (arousal is not low) (1)

Figure 5.12: Fuzzy rules

6 | Classification of emotions using facial expressions

This section is about the task 4.2, so the development and training of a CNN based on the network AlexNet. The network will be used to classify emotions by means of facial expressions, the possible classes to be classified are:

- Anger
- Disgust
- Fear
- Happiness

The provided dataset was unbalanced, so it was necessary a selection of the images. I started by selecting at random 300 images for each class. In order to better understand the performance of the system I analyzed at first the case with only two classes and then I performed the same experiment with all of them. In all cases, 70% of the images were devoted to the training set, 20% to the validation set and 10% to the testing set. All the images used have been resized in order to fit the input size required by AlexNet. In addition to it, in order to improve performance and prevent overfitting, a random horizontal and vertical translation of 30 pixels was performed.
AlexNet is trained to classify thousands of categories, while this project aims to perform classification among only 4 (or 2) classes, so the last 3 layers have been subsstituted with other layers. In the following there are the results of the different tests I performed.

6.1 Classification among 2 classes

6.1.1 Anger and Happiness, selected images

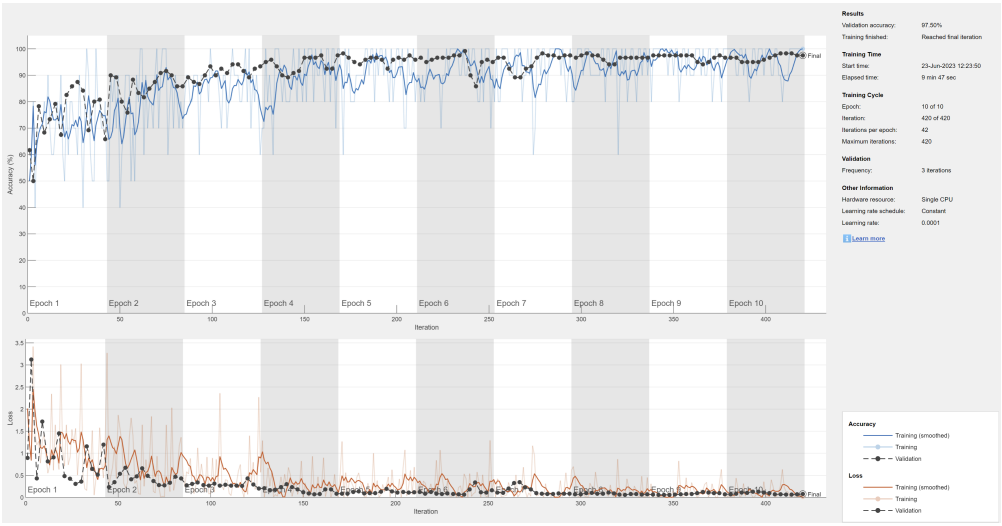

Figure 6.1: Training plot for pre-trained CNN among anger and happiness, 300 images each

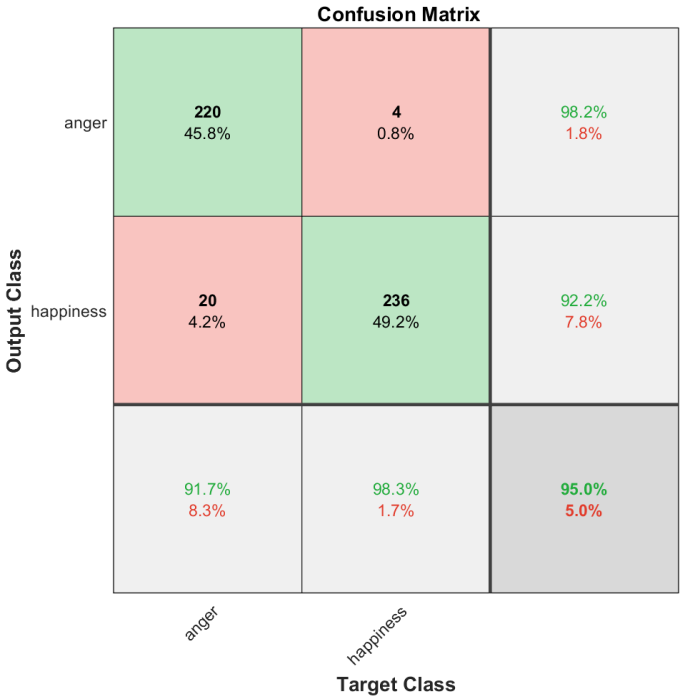

Figure 6.2: Confusion matrix for pre-trained CNN among anger and happiness, 300 images each

As can be seen from the images, the classification of the expressions of anger and happiness turns out to have very good accuracy values. This is also because the images were not chosen totally randomly to perform this test, but the most ambiguous images were excluded. We have also to state that the expressions of anger and happiness are easily distinguishable.

6.1.2 Anger and Happiness without selection of images

In this case we classify among the same 2 classes, but this time the 300 images are selected at random. We can observe a small reduction in performance:

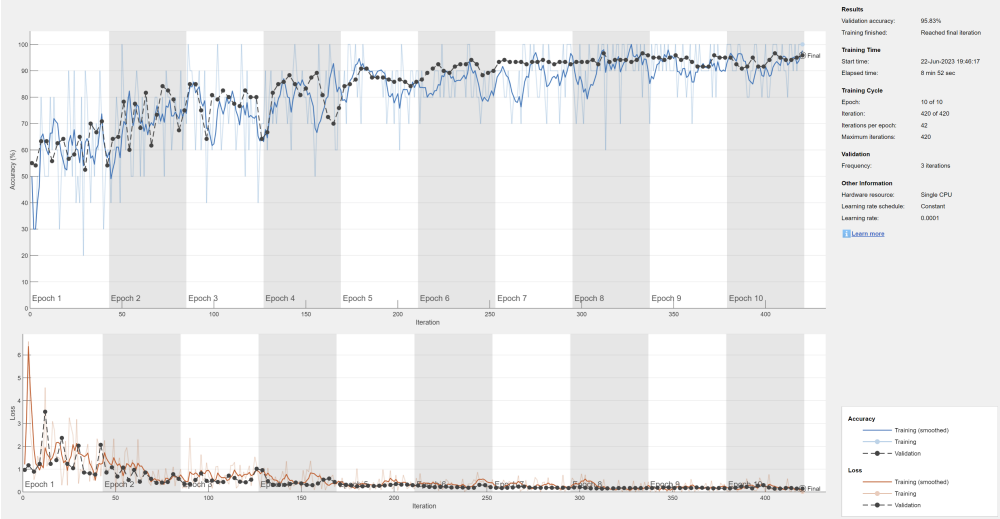

Figure 6.3: Training plot for pre-trained CNN among anger and happiness, 300 images each (at random)

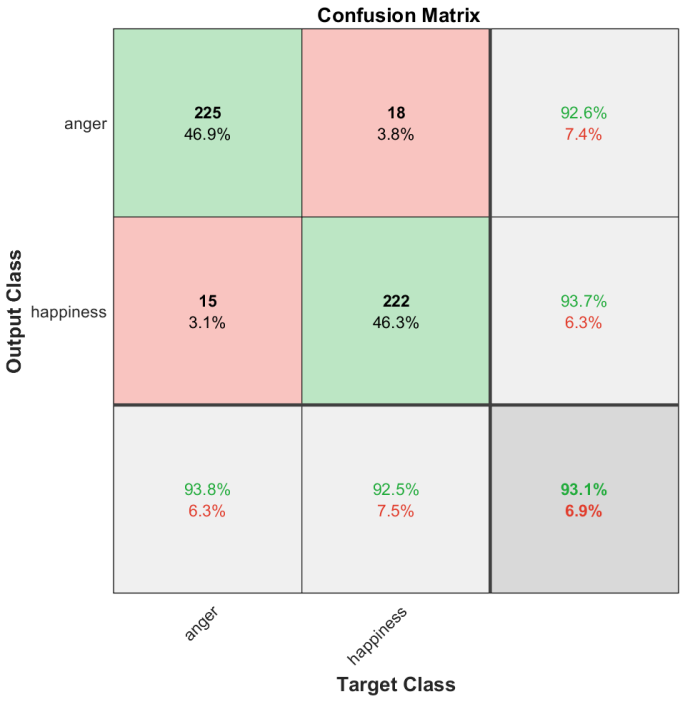

Figure 6.4: Confusion matrix for pre-trained CNN among anger and happiness, 300 images each (at random)

6.1.3 Anger and Disgust

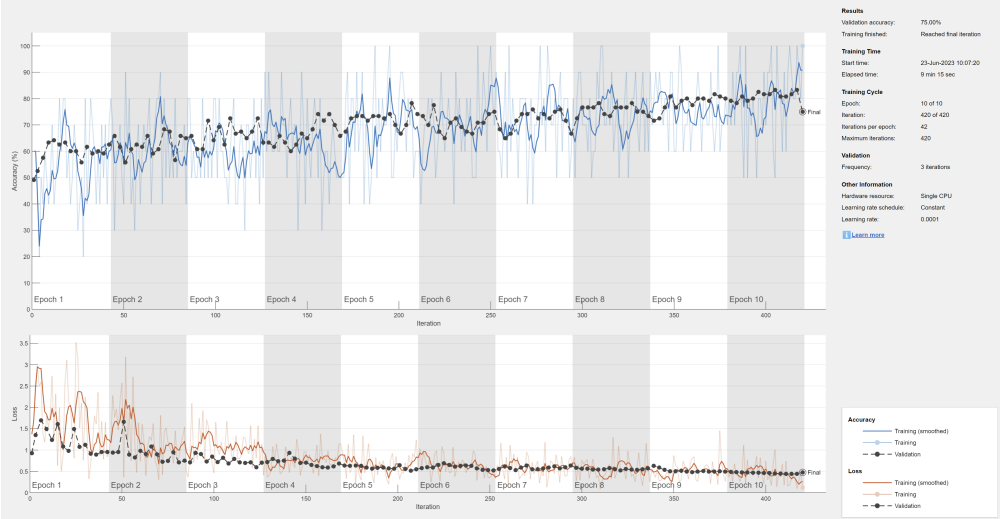

Figure 6.5: Training plot for pre-trained CNN among anger and disgust, 300 images each

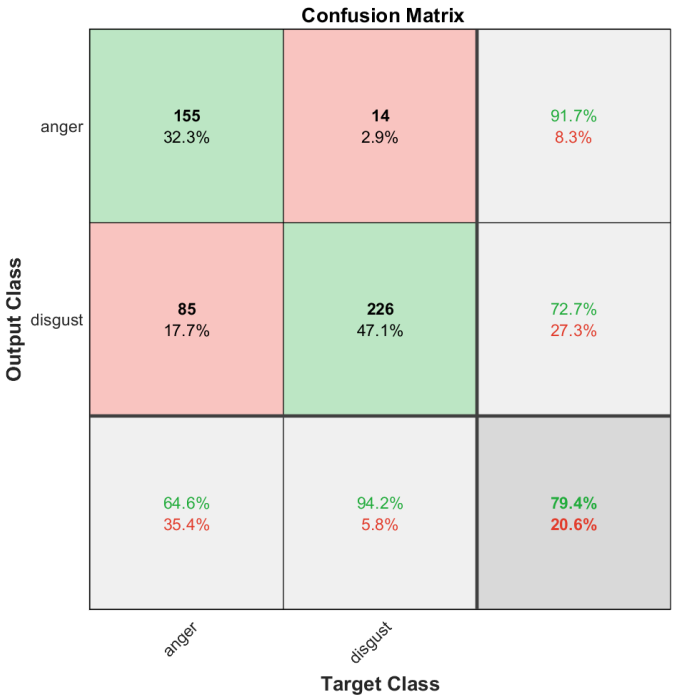

Figure 6.6: Confusion matrix for pre-trained CNN among anger and disgust, 300 images each

As we can see from the images above, the classification among anger and disgust have worse performance, because these classes of expressions are more difficult to distinguish.

6.2 Classification among 4 classes

6.2.1 300 selected images

If the 300 used images are selected by discarding the most ambiguous ones, we obtain the following results

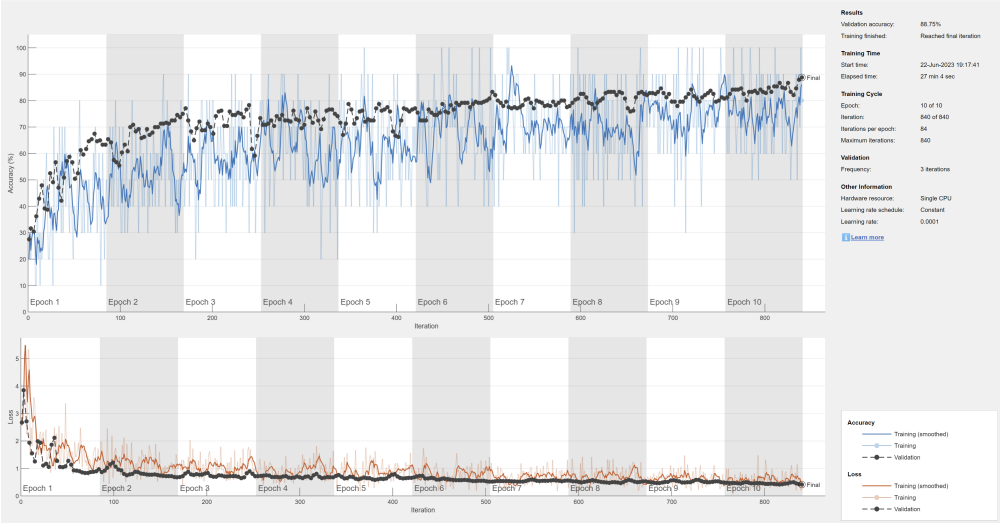

Figure 6.7: Training plot for pre-trained CNN among 4 classes, 300 images each

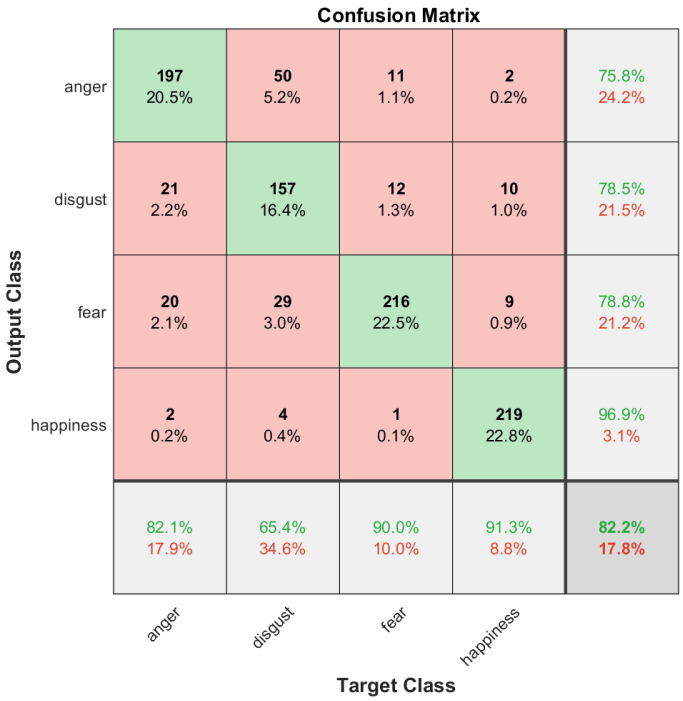

Figure 6.8: Confusion matrix for pre-trained CNN among 4 classes, 300 images each

We instantly notice an execution time that has increased respect to the case in which we were classifying among only 2 classes. Obviously enough there is also a reduction of performance, given the fact that the classification errors increase as the number of classes increase.

6.2.2 300 images randomly selected

If we try to repeat the same experiment with unselected images, we notice a reduction in performance:

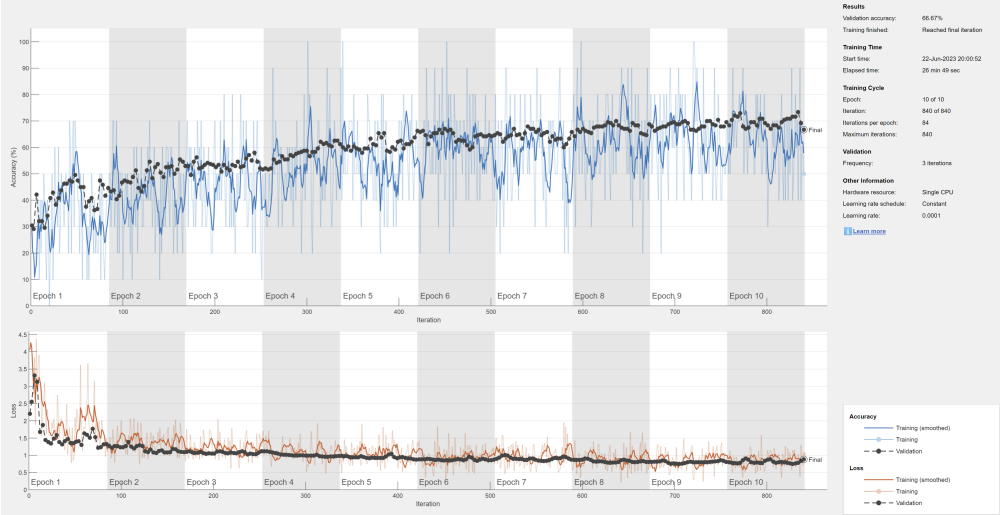

Figure 6.9: Training plot for pre-trained CNN among 4 classes, 300 images each (at random)

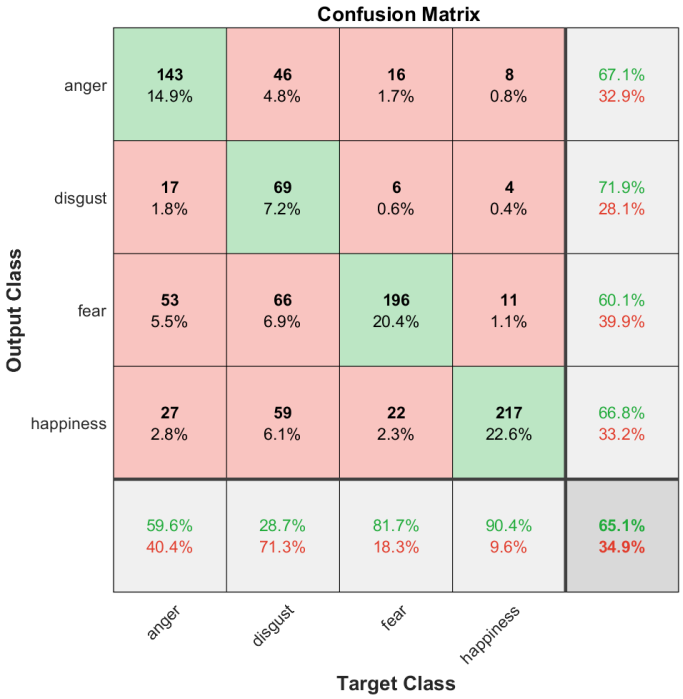

Figure 6.10: Confusion matrix for pre-trained CNN among 4 classes, 300 images each (at random)

6.2.3 500 images randomly selected

I decided to try to increase the number of images selected for each different class, in order to understand what is going to change in the results:

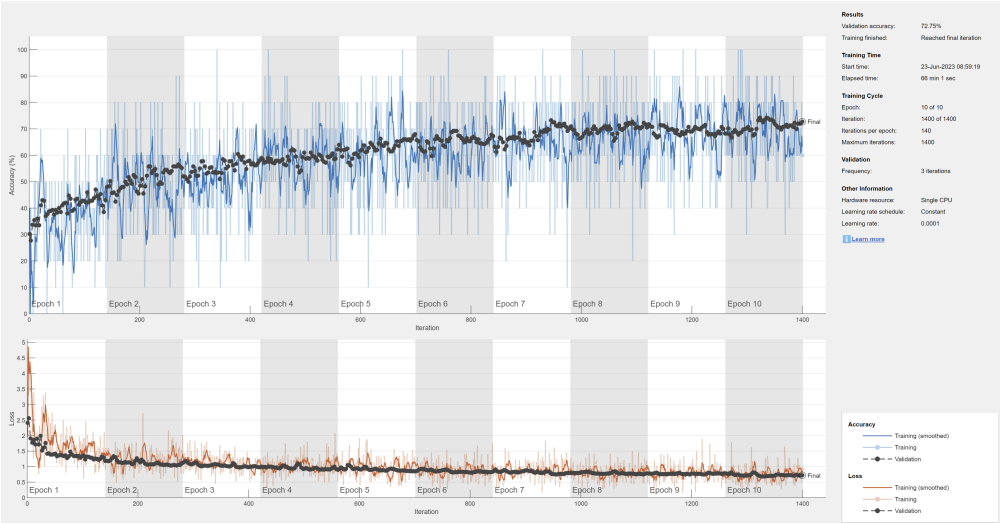

Figure 6.11: Training plot for pre-trained CNN among 4 classes, 500 images each (at random)

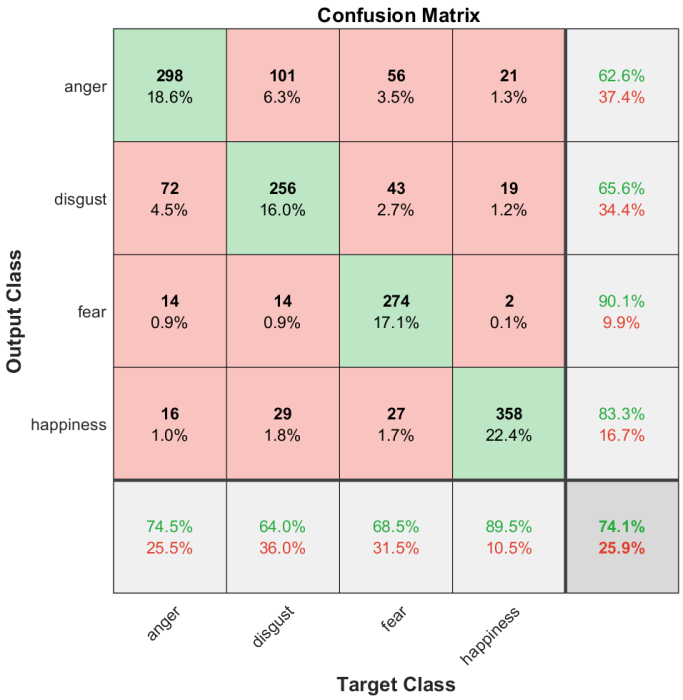

Figure 6.12: Confusion matrix for pre-trained CNN among 4 classes, 500 images each (at random)

We notice how the execution time increases, but also the performance of the network increase. This is due to the fact that increasing the size of the training set allows to the network to generalize more successfully.

6.2.4 1000 images randomly selected

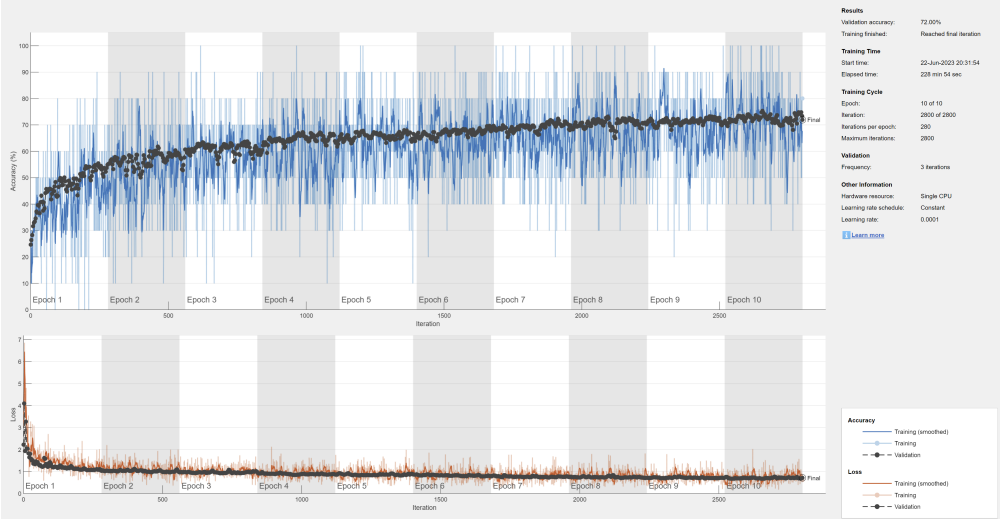

Figure 6.13: Training plot for pre-trained CNN among 4 classes, 1000 images each (at random)

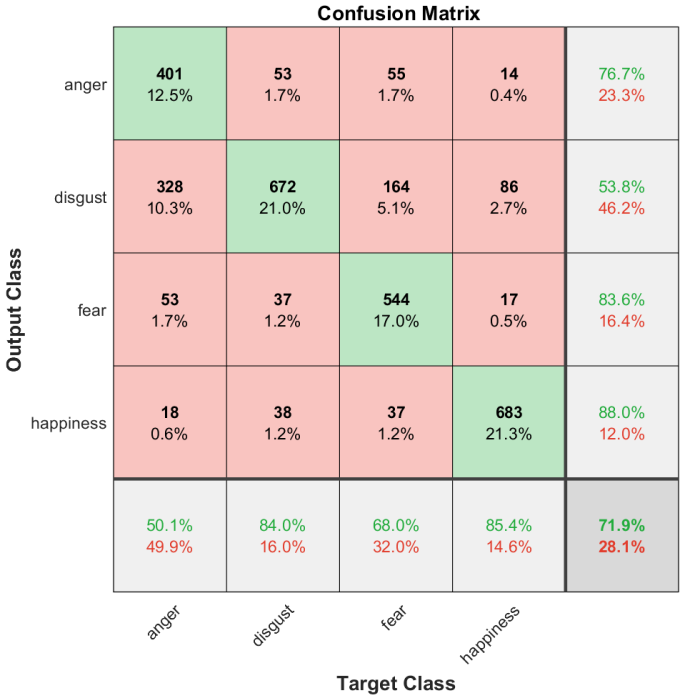

Figure 6.14: Confusion matrix for pre-trained CNN among 4 classes, 1000 images each (at random)

If I keep increasing the number of selected images, I obtain no improvement in performance, while I'm experiencing a significantly larger execution time. This can be due to the fact that a larger number of images can bring benefits from the training point of view but can also go to increase network classification errors, due to ambiguous or wrong images. For this reason, 500 (unselected) images turn out to be a good compromise between speed of execution and classification accuracy.