Intelligent Systems Project Report

Author:

Francesco Iemma

M.Sc. in Computer Engineering

Academic Year 2020/21

Contents

1 Data Cleaning **3**
1.1 Data Balancing 4
1.2 Features Selection 5

2 Neural Networks **7**
2.1 Fitnet 7
2.2 RBF 11

3 Fuzzy Inference System **13**

4 Convolutional Neural Networks **17**
4.1 CNN For 2-Classes Classification Problem 17
4.2 CNN For 4-Classes Classification Problem 18
4.3 Design Choices 19
4.4 Final Results 20

Introduction

The tasks performed in this project are the following:

- *3.1* Design and develop two MLP artificial neural networks that accurately estimate a person's valence and arousal and two RBF networks that do the same thing as the MLPs.
- *3.3* Design and develop a fuzzy inference system to estimate a person's arousal.
- *4.2* Fine-tune a pretrained CNN to obtain a convolutional neural network (CNN) that accurately classifies a person's emotion, based on facial expression.

The dataset at our disposal are two, one for tasks *3.1* and *3.3* and another one for task *4.2*. For what concern the first dataset, i.e. the one with biomedical signals for estimate arousal and valence, it is important to perform a cleaning of the data in order to obtain better performance for the neural networks that will be trained on it. This process, which is performed by the script `/matlab/data.m` is explained in the chapter 1.
Instead for the second dataset the selection modalities are discussed in the chapter 4.
After chapter 1, for each task is dedicated a chapter in which are explained the choices done and the results obtained in terms of performance.

Chapter 1

Data Cleaning

In this chapter we will see the data cleaning procedure performed in order to obtain better performance for the NNs. The steps done are:

- Remove non numeric values
- Remove outliers
- Balance the data among the different values of arousal and valence
- Features selection

All the procedure is contained in the file `/matlab/data.m`. The first two steps are performed thanks two matlab functions:

- `isinf(A)` that given a matrix returns a logic matrix where is indicated if the correspondent element of the input matrix are infinite (1) or not (0)
- `rmoutliers(dataset, method)` that given a dataset remove the outliers found using the method specified in input that is 'median' by default (i.e. "Outliers are defined as elements more than three scaled MAD from the median. The scaled MAD is defined as $c \times median(abs(A - median(A))$"[1]).

Then after the first two steps there are the most interesting part: data balancing and features selection.

[1] From Matlab documentation

1.1 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 (1, $2.\bar{3}$, $3.\bar{6}$, 5, $6.\bar{3}$, $7.\bar{6}$, 9), thus we can divide the dataset according 7 class for arousal and valence. The distribution of the samples among the classes is represented in the histograms in figure 1.1 and 1.2.

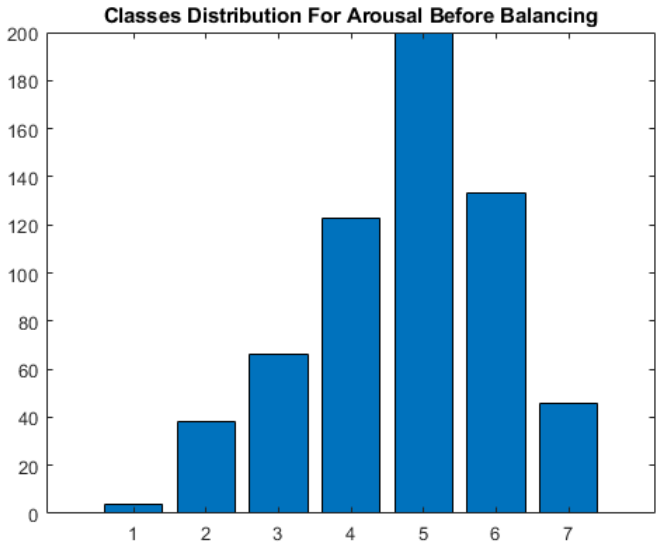

Figure 1.1: Classes Distribution For Arousal Before Balancing

Figure 1.2: Classes Distribution For Valence Before Balancing

As we can see the samples are heavily unbalanced, for this reason an algorithm to balance the data has been used. The algorithm is based on the concepts of undersampling, oversampling and data augmentation.
The steps are the following:

1. I augment the samples that belong to the minority class of arousal and don't belong to the majority class of valence and viceversa (i.e. the samples that belong to the minority class of valence and don't belong to the majority class of arousal).

2. I remove the samples that belong to the majority class of arousal and don't belong to the minority class of valence and viceversa (i.e. the samples that belong to the majority class of valence and don't belong to the minority class of arousal).

3. I repeat steps 1 and 2 for a $n = 80$ (after some experiments is possible to conclude that 80 is the number that gives the best results) times and for each repetition I compute the new majority and minority class both for arousal and valence.

At the end of this procedure the data are balanced as we can see in figure 1.3 and 1.4.

Figure 1.3: Classes Distribution For Arousal After Balancing

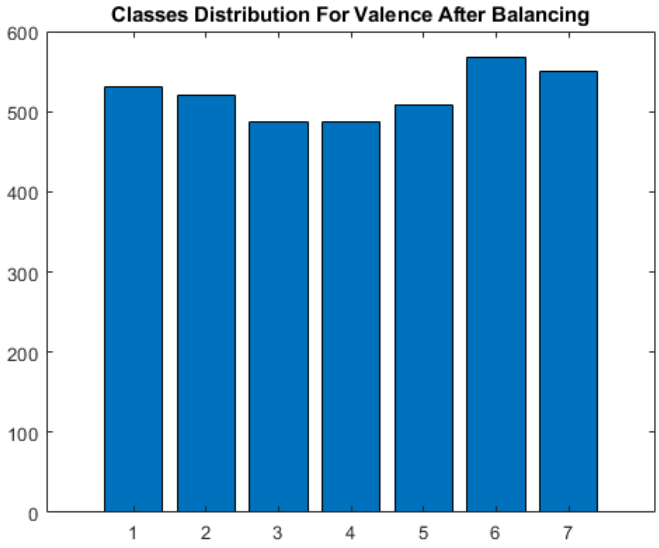

Figure 1.4: Classes Distribution For Valence After Balancing

1.2 Features Selection

Before the features selection is necessary to divide the data in two set: one for training and one for test. 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.

Thus after the extraction of the holdout partition we perform 5 times the `sequentialfs` for arousal and 5 times for valence. Then we select the first `FEATURES_TO_SELECT` (constant set at the beginning of the script that is 8) features that appear most times in the different runs of sequentialfs, this operation is performed separately for arousal and valence.
It's important to underline the fact that `sequentialfs` should be repeated a statistically relevant number of times, thus 5 repetitions are not a statistically relevant number of times, anyway is the compromise done in order to maintain an acceptable number of features, an acceptable quality of data and an acceptable time needed for the algorithm (9 hours for valence and 9 hours for arousal with the hardware at disposal).
At the end the data obtained are saved into five `.mat` files:

- `/matlab/data/biomedical_signals/dataset_cleaned.mat`

  It contains the entire dataset without infinite values, outliers and with balanced classes distributions.

- `/matlab/data/biomedical_signals/training_data_valence.mat`

  It contains a `struct` with the training input (only the selected features) and the correspondent target output for valence.

- `/matlab/data/biomedical_signals/training_data_arousal.mat`

  It contains a `struct` with the training input (only the selected features) and the correspondent target output for arousal.

- `/matlab/data/biomedical_signals/test_data_arousal.mat`

  It contains a `struct` with the test input (only the selected features) and the correspondent expected output for arousal.

- `/matlab/data/biomedical_signals/test_data_valence.mat`

  It contains a `struct` with the test input (only the selected features) and the correspondent expected output for valence.

- `/matlab/data/biomedical_signals/fuzzyData.mat`

  It contains all the data needed by the fuzzy inference system.

The division of training data and test data in two files, one for valence and one for arousal, is done only for practical reasons, in fact the repetitions of sequentialfs has been performed only for arousal and only for valence separately due to the time needed.

Chapter 2

Neural Networks

In this chapter we will see two types of neural networks that resolve the same problem, that is to estimate the values of arousal and valence given a set of biomedical signals. The code is in the script `matlab/neuralNetworks.m`.

2.1 Fitnet

After some experiments it has been obtained that the fitnet achieve good results with the default parameters, then after other tests performed in order to asses the number of neurons, the result was that 35 and 30 are the numbers that ensure good results with a low number of neurons. In figures 2.1, 2.2, 2.3, 2.4 there are the results of the training of the fitnet using to estimate valence.

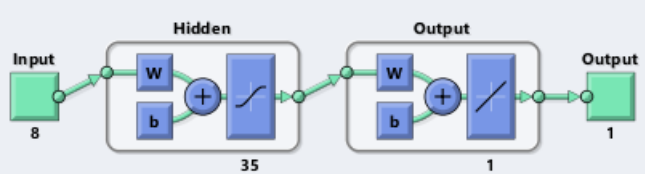

Figure 2.1: Architecture of fitnet for valence

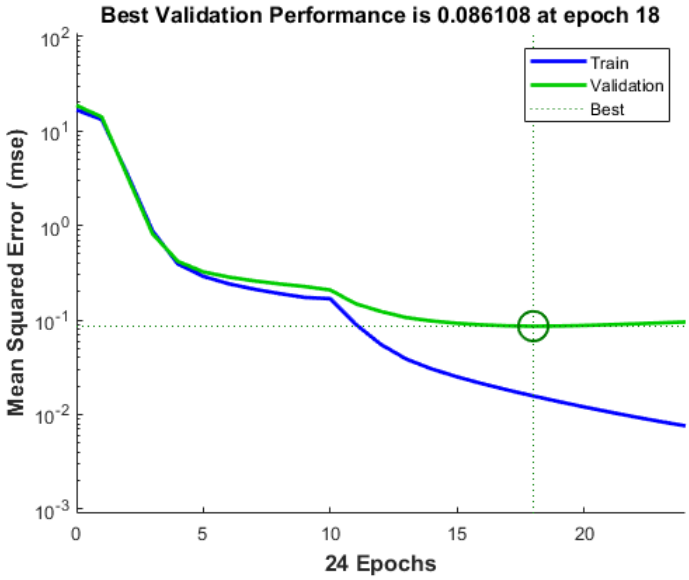

Figure 2.2: Performance progress of fitnet for valence

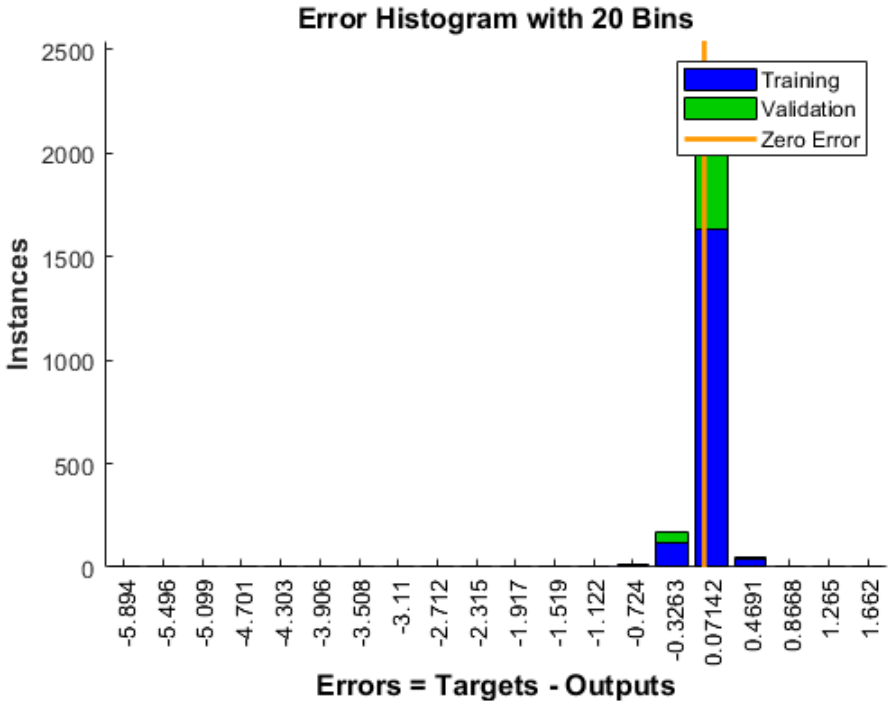

Figure 2.3: Error histogram of fitnet for valence

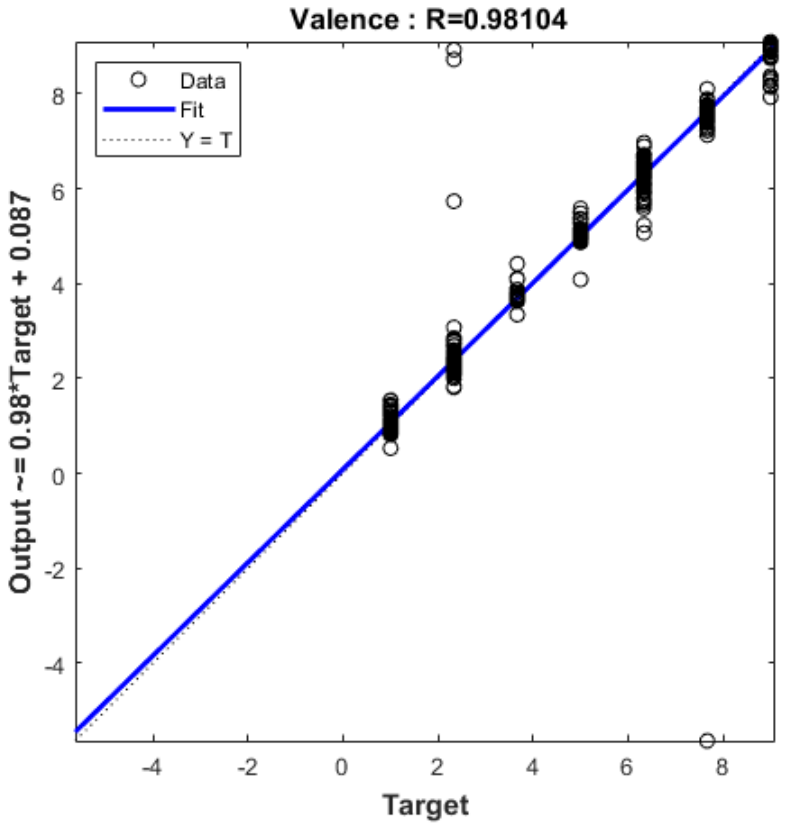

Figure 2.4: Regression of fitnet for valence

In figures 2.5, 2.6, 2.7, 2.8 there are the results of the training of the fitnet using for estimate arousal.

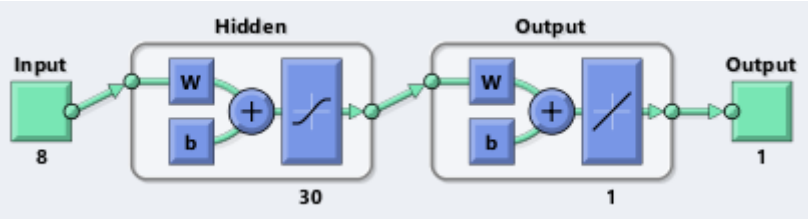

Figure 2.5: Architecture of fitnet for arousal

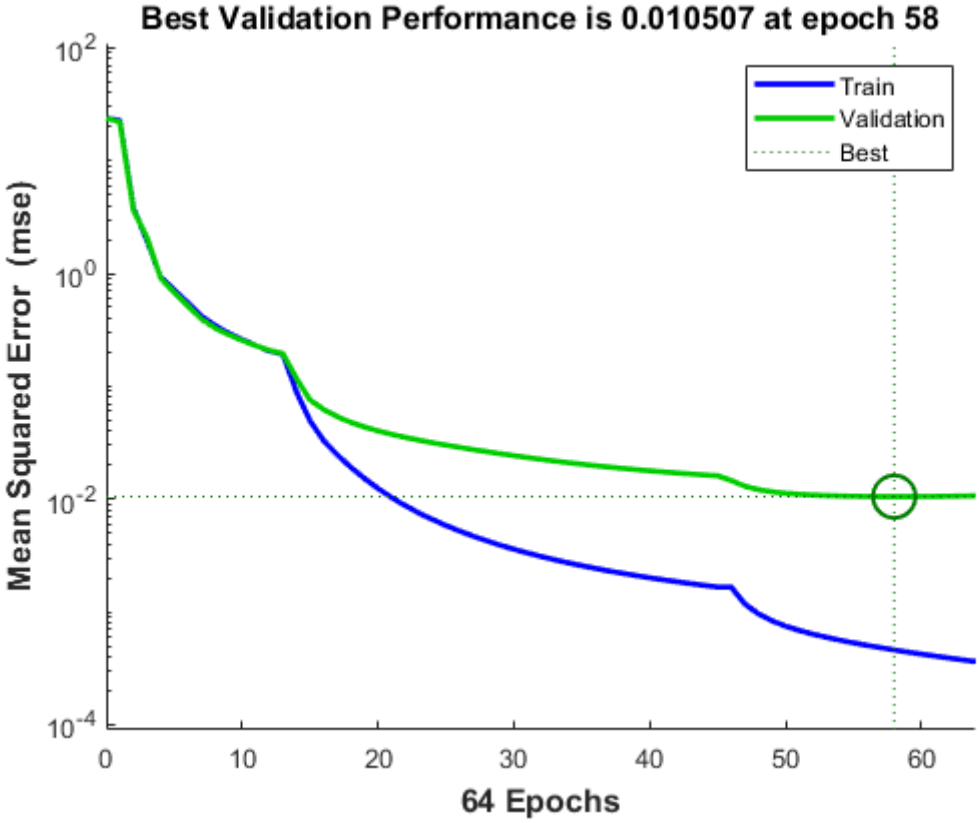

Figure 2.6: Performance progress of fitnet for arousal

Figure 2.7: Error histogram of fitnet for arousal

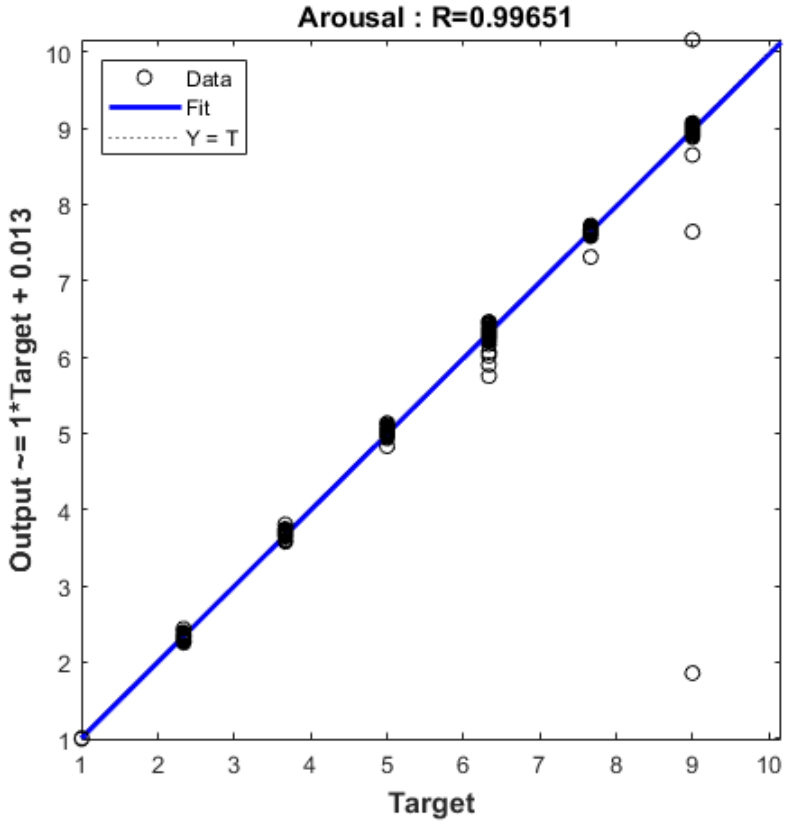

Figure 2.8: Regression of fitnet for arousal

The results obtained are good for both fitnet and RBF (for RBF see next section): in particular the regression value is very high and the error is reasonable. An observation for what concern the fitnet that it is possible to done is that the error is quite variable depending on the training, in fact different training give different error values (all reasonable but some are better than others) depending from the starting point on the error surface (the weights assume random values at the beginning of the training).

2.2 RBF

In this case good results have been achieved with a spread constant equal to 1, then the number of neurons is set in order to achieve the goal that is in both cases of arousal and valence equal to 0.02 (Mean Square Error).
The results of the RBF network for valence are in figures 2.9, 2.10.

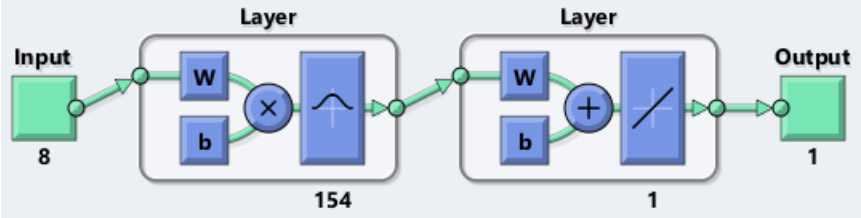

Figure 2.9: Architecture of RBF for valence

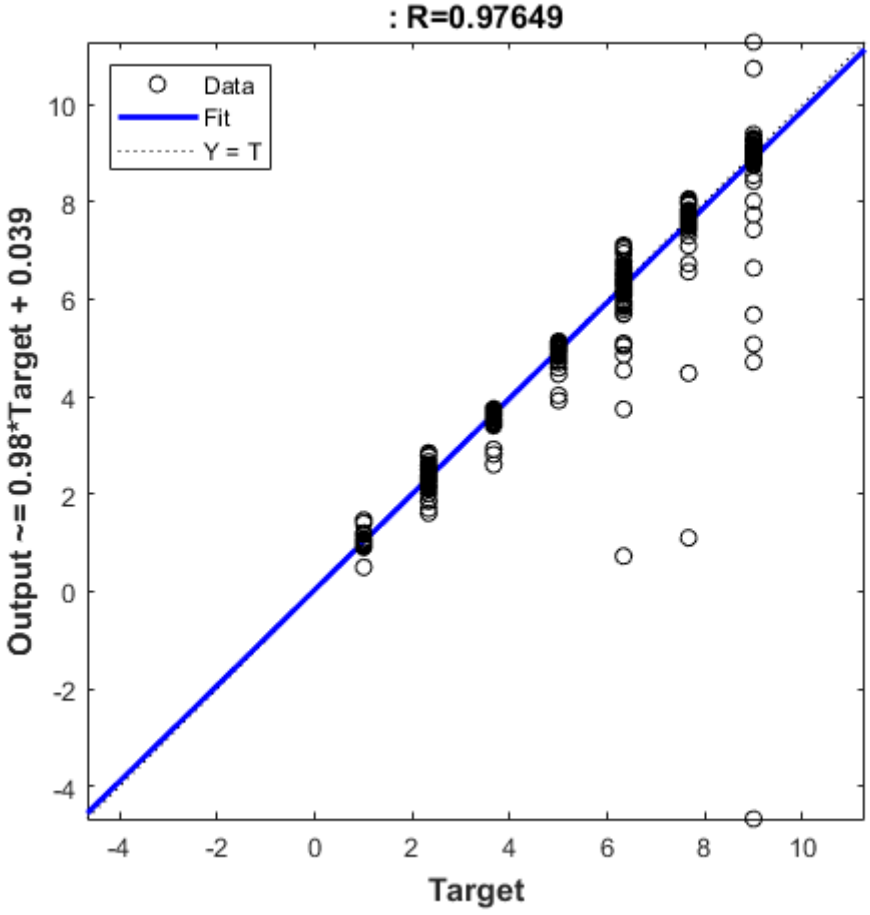

Figure 2.10: Regression of RBF for valence

The results of the RBF network for arousal are in figures 2.11, 2.12.

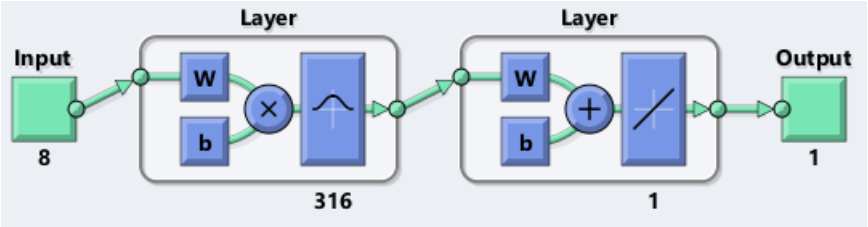

Figure 2.11: Architecture of RBF for arousal

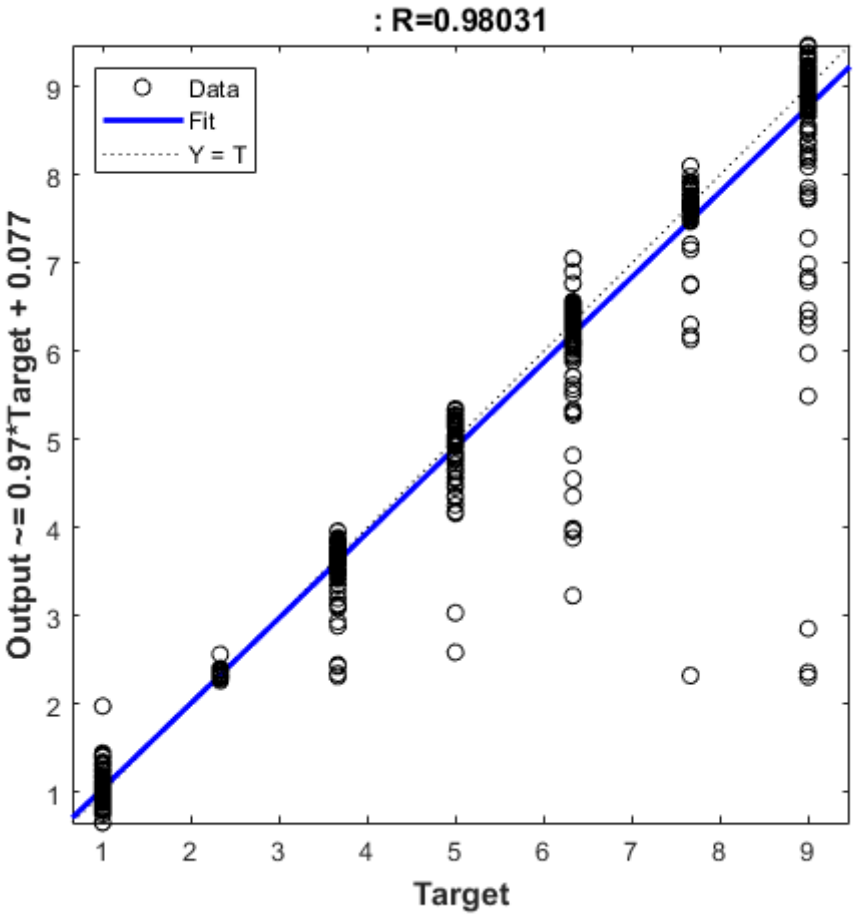

Figure 2.12: Regression of RBF for arousal

Chapter 3

Fuzzy Inference System

In order to develop a fuzzy inference system to estimate the values of arousal an analysis of the three best features selected by running 5 times the sequentialfs algorithm (the reason behind the choice of this number is the same explained in the introduction) must be done. This analysis is performed by the script `matlab/analysis_for_fis.m`. The other files of this task are:

- `matlab/FIS_arousal.fis`

  It contains the Fuzzy Inference System developed through the fuzzy logic designer.

- `matlab/fis_evaluation.m`

  It is the script used to evaluate the system performance through `evalfis`.

As first thing we have to fix the universe of discourse of the three features, in order to do so we look for the maximum and the minimum value for each feature, the results are the following:

- Feature 24: $[-1.276359, 1.849861]$
- Feature 27: $[-3 \times 10^{-5}, 5 \times 10^{-5}]$
- Feature 37: $[-0.064933, 0.034879]$

Then we have to check how many samples that has a low/medium/high output correspond to the different values of a particular feature. In order to do so we have to fix the meaning of low, medium and high output. As we have seen the possible values of the arousal/valence can be divided in 7 class, then values of first and second class are considered low, values of third, fourth and fifth are considered neutral and values of sixth and seventh class are considered high.
From the implementation viewpoint we have a vector with the seven values and the we access this vector giving the class, thanks to this we deal with classes and not with the exact value. This is important in order to understand the following code:

```
y_level_low = find(y_train==y_values(1) | y_train==y_values(2));
y_level_medium = find(
  y_train==y_values(3) | y_train==y_values(4) | y_train==y_values(5));
y_level_high = find(y_train==y_values(6) | y_train==y_values(7));

%Low
if PLOT_HIST_LOW==1
  figure(1);
  histogram(x_train(y_level_low,1));
  figure(2);
  histogram(x_train(y_level_low,2));
  figure(3);
  histogram(x_train(y_level_low,3));
end

%.... mutata mutandis for medium and high
```

Now I am ready to perform the development of the fuzzy system. The approach used is the one that consist in using the graphical user interface and so the command `fuzzyLogicDesigner`. The input variables and the correspondent membership functions are the ones shown in figure 3.1, 3.2 and 3.3. In figure 3.4 is shown the output function with the correspondent membership functions.

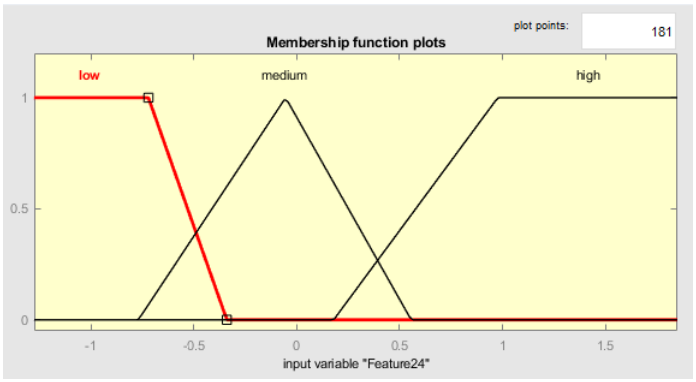

Figure 3.1: Feature 24

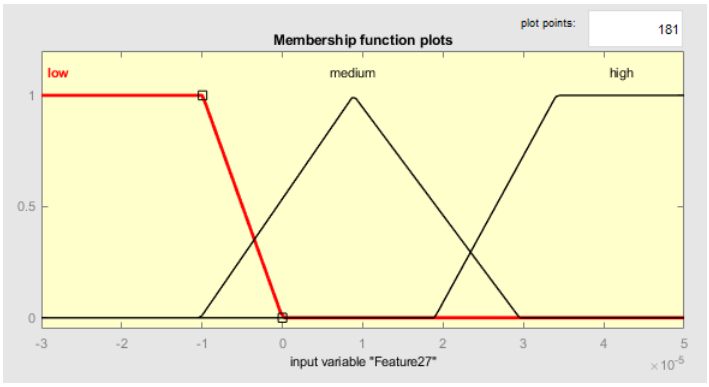

Figure 3.2: Feature 27

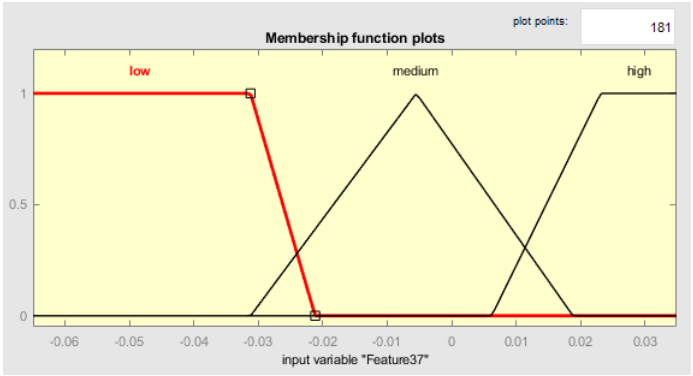

Figure 3.3: Feature 37

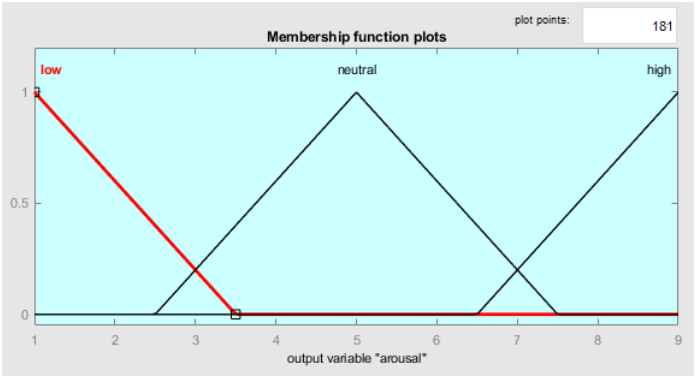

Figure 3.4: Output

The rules and also the membership functions for each features, have been written thanks to the histogram plotted in the analysis phase, a subset of the histograms are the ones in figures 3.5, 3.6 and 3.7 that show how many samples with low, medium or high values of output correspond to the different values of feature 24 (hence we have three plots for each feature). Then in figure 3.8 it is possible to see the different rules.

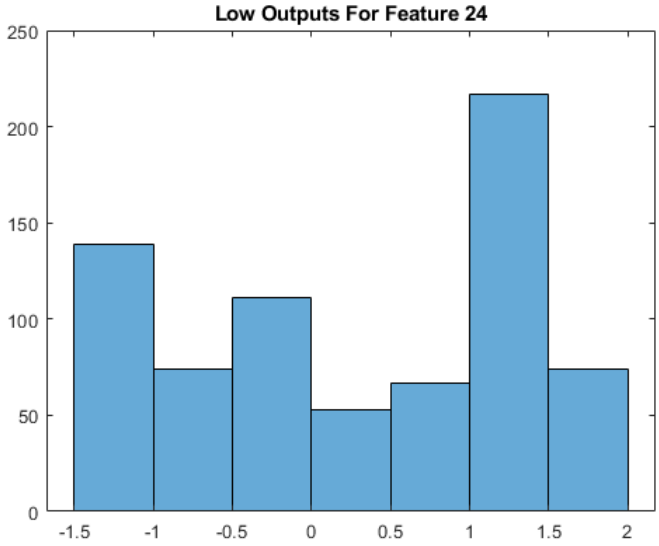

Figure 3.5: Histogram that represents the number of samples that have a low arousal value and their distribution among the different values of feature 24.

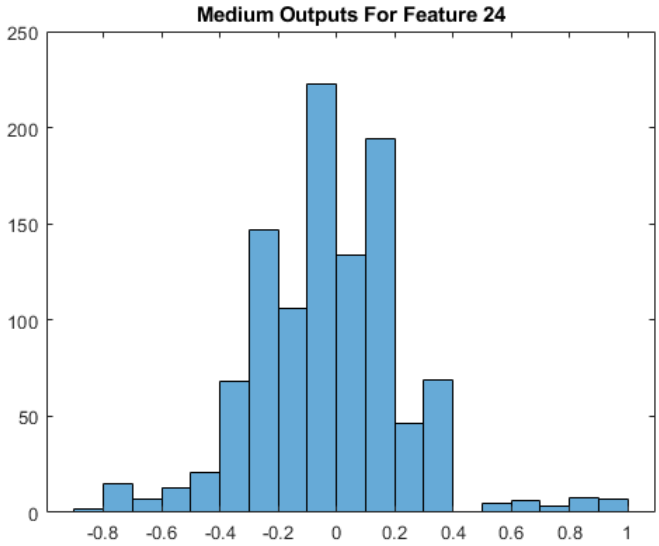

Figure 3.6: Histogram that represents the number of samples that have a medium arousal value and their distribution among the different values of feature 24.

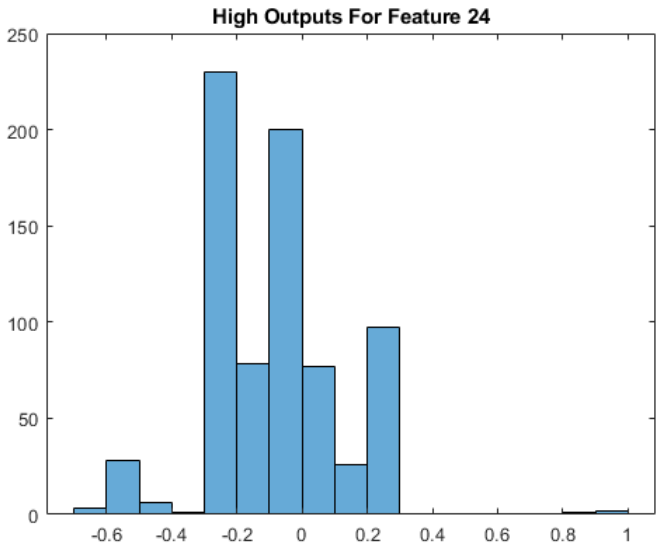

Figure 3.7: Histogram that represents the number of samples that have an high arousal value and their distribution among the different values of feature 24.

1. If (Feature27 is medium) then (arousal is neutral) (1)
2. If (Feature24 is medium) then (arousal is not low) (1)
3. If (Feature24 is high) then (arousal is low) (1)
4. If (Feature37 is low) then (arousal is not high) (1)
5. If (Feature37 is medium) then (arousal is not low) (1)

Figure 3.8: Fuzzy Rules

In order to understand the reasoning behind the rules let make an example. We take into account the rule number 3 i.e. `"If (Feature24 is high) then (arousal is low)"`.
This rule can be inferred analyzing the three graphs corresponding to feature 24 (figures 3.5, 3.6, 3.7), in fact feature 24 has *high* values more or less for values greater than 0.5 (figure 3.1) and for these values there are no samples for what deal with the medium and high arousal outputs (see 3.6 and 3.7) whereas there are values for low arousal outputs (see 3.5). This means that if the feature 24 has values more or less greater or equal than 0.5 (so it assume *high* values) then the arousal is *low*. So we have inferred the fuzzy rule 3.

Chapter 4

Convolutional Neural Networks

In this chapter is discussed the development and the training of a CNN based on the pre-trained AlexNet which has the aim of classifying facial expressions. The possible classes are four:

- Happiness
- Anger
- Disgust
- Fear

Two CNNs have been developed (both are based on AlexNet). The first one is able to classify images in two classes (happiness and anger), instead the second one is able to classify images in all of the four classes. The scripts are `matlab/cnn_2classes.m` and `matlab/cnn_4classes.m`.

4.1 CNN For 2-Classes Classification Problem

The matlab code for this CNN is in the file `/matlab/cnn_2classes.m`. Due to the fact that the images of happiness and the ones for anger are very different it is possible to obtain good result (accuracy equal to 82.67%) without a proper selection of the images. In fact as first experiment 500 images for each class are selected at random from the dataset. This experiment returns the results shown in figure 4.1:

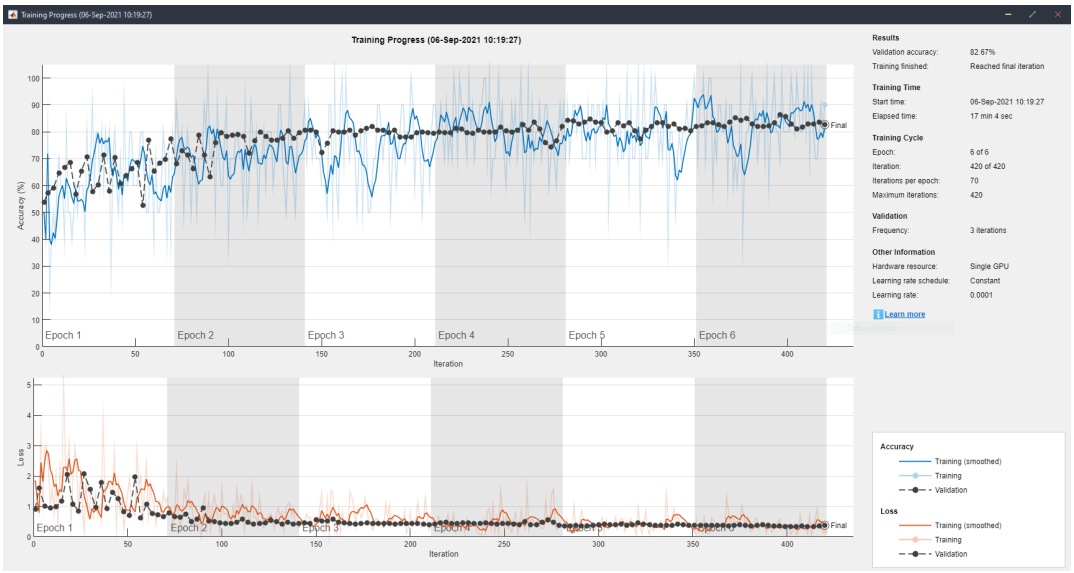

Figure 4.1: First experiment: no images selection

Then a selection of the images has been performed and from the 500 images only 300 have been selected. The selection consist in the removal of the images with ambiguous facial expressions. After this selection the results in figure 4.2 are obtained.

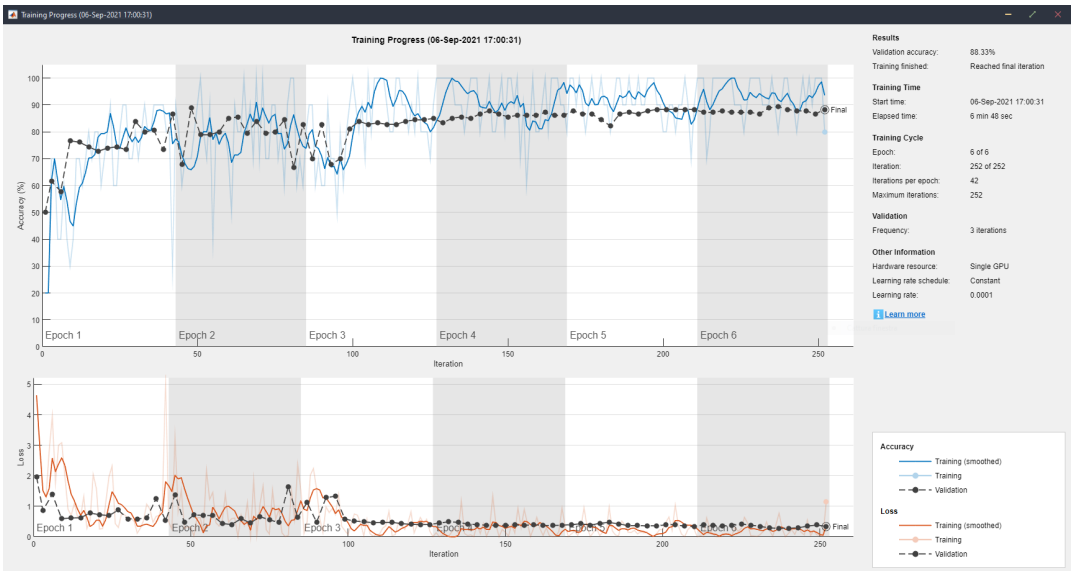

Figure 4.2: Second experiment: after selection

Thanks to the selection of the images the validation accuracy has increased by $\approx 6\%$: from 82.67% to 88.33%.

4.2 CNN For 4-Classes Classification Problem

The matlab code for this CNN is in the file `/matlab/cnn_4classes.m`. As in the previous case two experiments have been performed, the first one with 500 images per class selected at random, the second one with 300 images selected removing the images classified wrong or with ambiguous facial expressions.
The results for the experiments are respectively in the figures 4.3 and 4.4.
Analyzing the two results it is possible to see that, thanks to the selection, we have improved the accuracy from 58.17% to 63.61% that can be considered a reasonable result for a classification problem with 4 classes, in any case this result can be improved, see section 4.4.

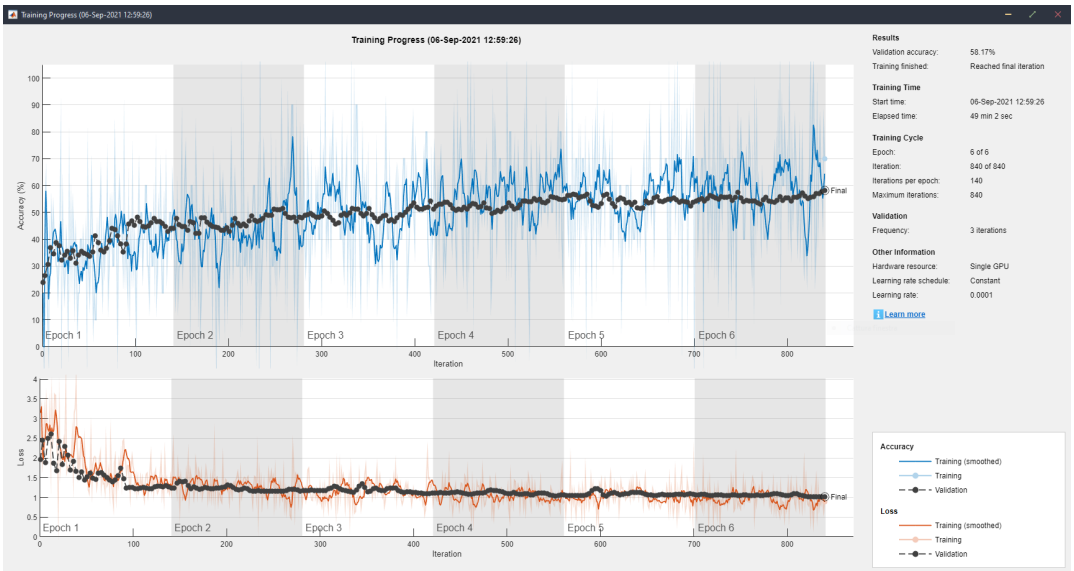

Figure 4.3: First experiment: no images selection

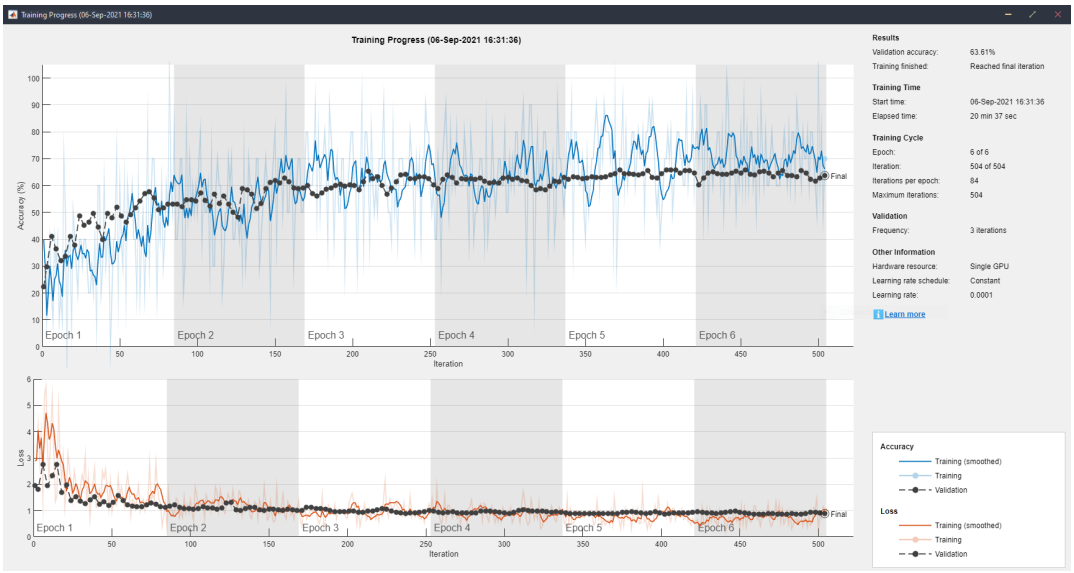

Figure 4.4: Second experiment: after images selection

4.3 Design Choices

The dataset of images is very huge, but in order to maintain a reasonable training time only 500 images have been selected at random for each classes, then in order to improve performance a selection has been performed reducing this number to 300 images per class.
In all experiments the images used in training have been augmented:

```
pixelRange = [-30 30];
imageAugmenter = imageDataAugmenter( ...
'RandXReflection', true, ...
'RandXTranslation', pixelRange, ...
'RandYTranslation', pixelRange);

augmented_image_data_train = augmentedImageDatastore(
input_size(1:2), img_data_train, ...
'DataAugmentation', imageAugmenter);

augmented_image_data_validation = augmentedImageDatastore(
input_size(1:2), img_data_validation);

augmented_image_data_test = augmentedImageDatastore(
input_size(1:2), img_data_test);
```

The reason why also the image for validation and for testing are augmented is because the method for augmentation (that is `augmentedImageDatastore`) automatically resize the image in order to fit the input size requested from AlexNet. In fact you can see that there is no `imageAugmenter` in the `augmentedImageDatastore` for validation and for test because it is used only for resizing, instead in the case of training images the same method perform both resizing and augmentation.
For what concerns the CNN architecture only the last three layers of AlexNet have been removed and substituted with other three layer:

```
net_layers = [
original_layers
fullyConnectedLayer(
   numberOfClasses,'WeightLearnRateFactor',20,'BiasLearnRateFactor',20)
softmaxLayer
classificationLayer];
```

4.4 Final Results

At the end after some experiments the best architectures is the one in the two matlab scripts. Some hyper-parameters have been changed (for instance the initial learning rate, the weight learn rate factor, the bias learn rate factor ecc..), but the parameter that improve the performance is the maximum number of epochs, in fact increasing this parameter the network has more time to be trained and so the the training is more effective. Another observation is that at every epoch the data are shuffled and so the presentation order of the training images is different, hence the CNN training is not dependent from the presentation order. The results obtained are the ones in figure 4.5 and 4.6. For what concern the test accuracy:

- Test Accuracy 2-classes problem: 97.08%
- Test Accuracy 4-classes problem: 80.31%

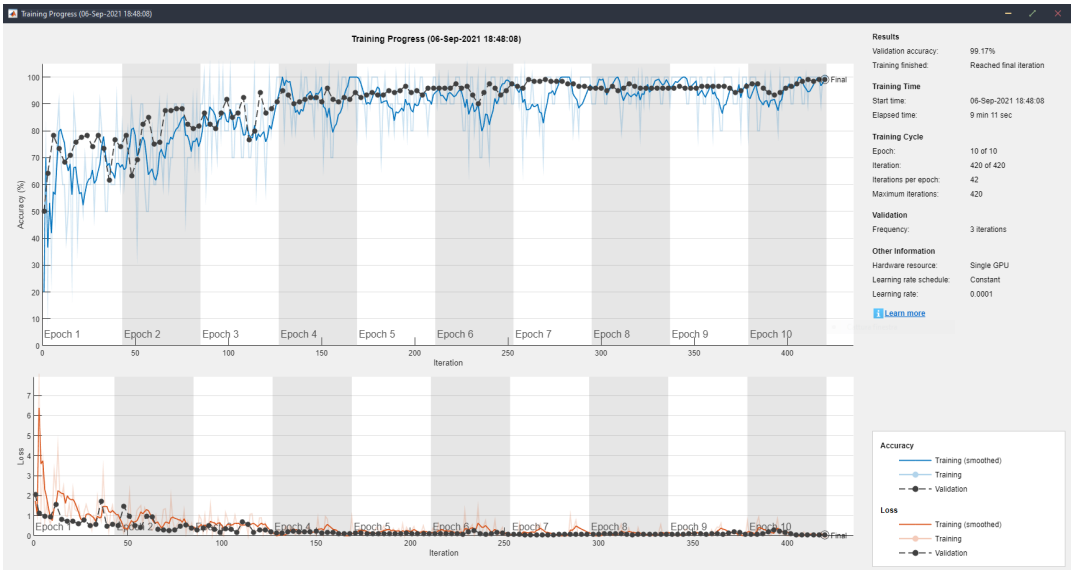

Figure 4.5: Final result for two-classes classification problem

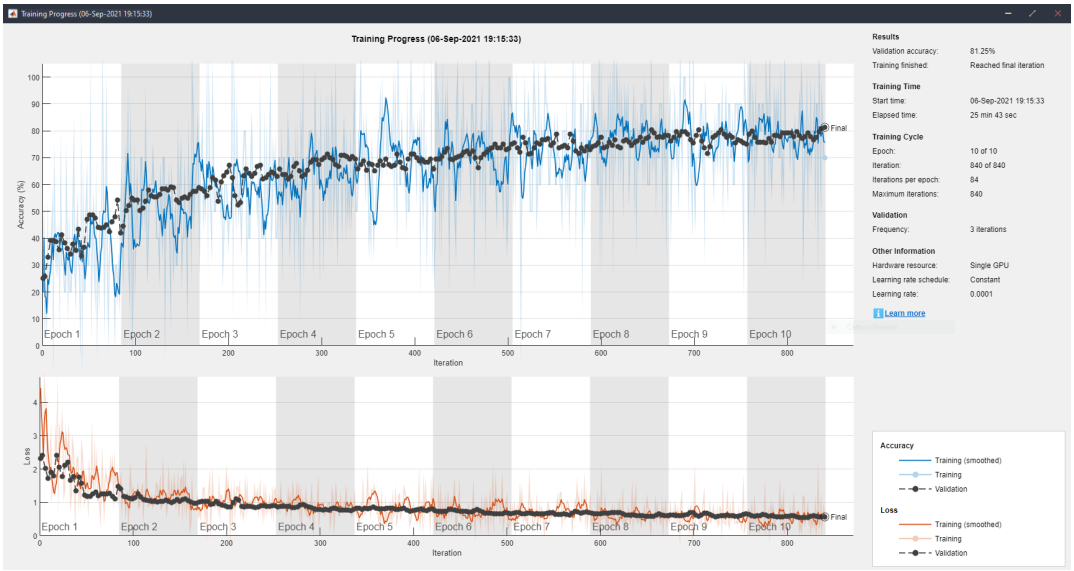

Figure 4.6: Final result for four-classes classification problem

For what concern the confusion matrix the results in figure 4.7 and 4.8 have been obtained.

Figure 4.7: Confusion matrix for test images in 2-classes classification problem

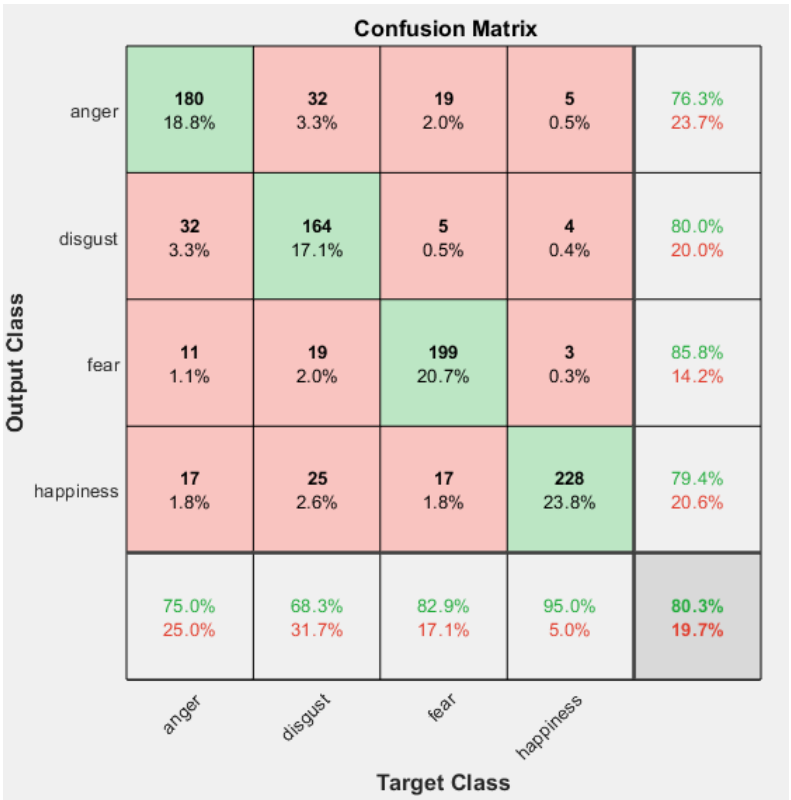

Figure 4.8: Confusion matrix for test images in 4-classes classification problem