UNIVERSITÀ DI PISA

Computer Engineering

Intelligent Systems

Project Report

TEAM MEMBERS:
Olgerti Xhanej

Academic Year: 2020/2021

Contents

1 Introduction **2**
1.1 Tasks and deliverables 2

2 Task 3.1 **3**
2.1 Dataset Preparation 3
2.2 MLP Training 4
2.3 RBF Training 7

3 Task 3.3 **9**

4 Task 4.1 **13**
4.1 Dataset Preparation 13

1 — Introduction

1.1 Tasks and deliverables

For this project has been carried out tasks **3.1**, **3.3** and **4.1**.

For the first two tasks: files `data_preparation_results`, `data_preparation_results_50`, `data_preparation_results_100` represent the results of the feature selection process by repeating the `sequentialfs` function respectively 10, 50 and 100 times. The latter file has been used for obtaining the results of task 3.1 and 3.3.

For the third task: the folders `datasets/images/images_to_use_experiment_2` and `datasets/images/images_to_use_experiment_2_two_classes` contains the images used for the training of the CNN that respectively classify 4 emotions and 2 emotions (fear and disgust).

2 — Task 3.1

2.1 Dataset Preparation

For the first task of the project (also for task 3.3) a preprocessing of the given dataset has been carried out in the MATLAB file `dataset_preparation.m`.

First of all a **cleaning process** has been done in order to remove Non-Numeric data or Infinite numbers in the dataset.
Then a procedure of **outliers removal** has been carried out: to do so, some experiments have been done out with the way to perform outlier removal. Two parallel experiment were made and at the end the *median* method has been selected as optimal by comparing the neural network regression performance.

The next step was **data balancing**, since the outputs can assume only seven different values for both arousal and valence. In order to obtain the latter point, **data augmentation** has been performed by multiplying selected data by a random number between $[0.95, 1.05]$.
The data to augment was obtained as follows: from the indexes of the least common class for arousal(valence) were removed the indexes of the most common class for valence(arousal). Then, the resulting indexes were lowered if by adding them to the least common class, the result would go beyond the most common class of that type. This last operation permitted to obtain a certain degree of "convergence" of the whole process, which was repeated different times.
In the following image are presented the results of data augmentation:

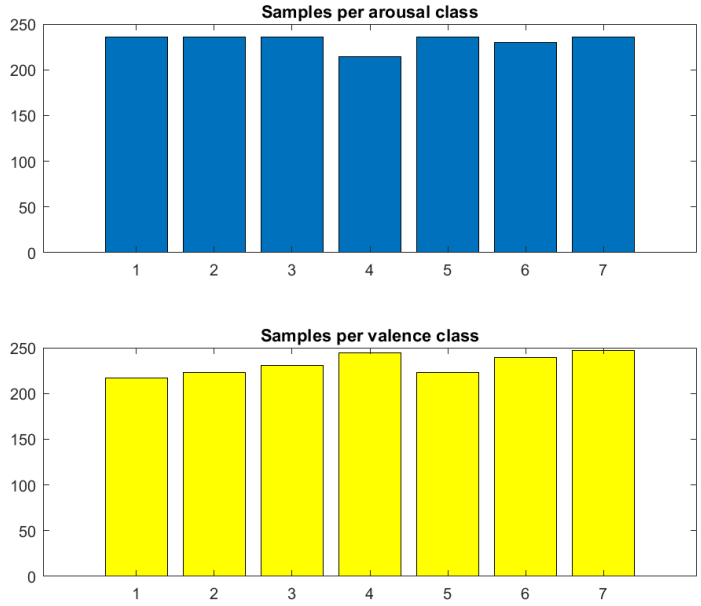

Figure 1: Data Balancing results

The last step of the process was the **feature selection step**: firstly, data has been divided through an **holdout partition** into training and test data (the latter will be used

to assess the final performance of the net); secondly, through the `sequentialfs` function the features have been extracted from the training data by exploiting the following function:

```
function err = fun2(x_train, t_train, x_test, t_test)
  % Around 1000 samples for training
  % 1 input neuron, 1 output neuron
  % Chose 60 neurons for the hidden layer of the fitnet

  net = fitnet(60);
  net.trainParam.showWindow=0;
  xx = x_train';
  tt = t_train';
  net = train(net, xx, tt);
  y=net(x_test');
  err = perform(net,t_test',y);
end
```

For what concerns the **number of hidden neurons**, the following formula has been used in order to prevent over-fitting by determining a upper bound for neurons on hidden layer:

$$N_h = \frac{N_s}{\alpha * (N_i + N_o)} \tag{1}$$

Where N_h represents the upper bound for hidden neuron, N_s the number of samples, N_i the number of input neurons, N_o the number of output neurons and α a scaling factor between 2 and 10.
The method of matching the number of weights as 5 or 10 times the number of samples was not used, since it would imply an higher number of neurons in the fun2 function and the general feature selection process would take much longer computational effort.

The `sequentialfs` has been ran **100 times** for giving statistical meaning to results and at the same time limit the computational effort of the machine. Each feature selected has been counted in a counter vector and at the end this vector has been sorted in descending order in order to obtain the list of most important features. Two different sorted vector were obtained, one for arousal and another for valence.

2.2 MLP Training

After loading the results of the feature selection process, some **experiments for finding the best architecture** for MLPs in terms of pure fitting of arousal and valence has been carried out: by **changing the number of neurons** (i.e for the first hidden layer around the interval [5, 120], trying to minimize this number as soon as good results started to appear), by changing the training function, by modifying the maximum number of **validation fails**, the **number of epochs** and by adding some momentum and so on. In general good performance (in terms of regression) were obtained by using **only one hidden layer** and a similar number of neurons used in the feature selection process while other parameters didn't seem to influence greatly the regression of the MLP.
For what concerns the best architecture for MLPs and the related regression performance (w.r.t. the test set obtained from the holdout partition before the feature selection), in the following are presented those results:

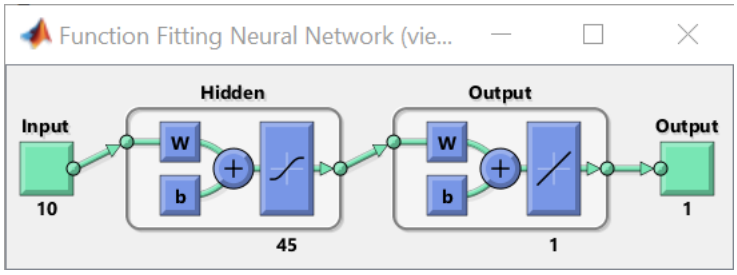

Figure 2: MLP Structure for arousal

```
% Configuration for training of the MLP for arousal
% (non present fields are the default ones)
mlp_net_arousal.trainParam.lr = 0.1;
mlp_net_arousal.trainParam.epochs = 100;
mlp_net_arousal.trainParam.max_fail = 10;
```

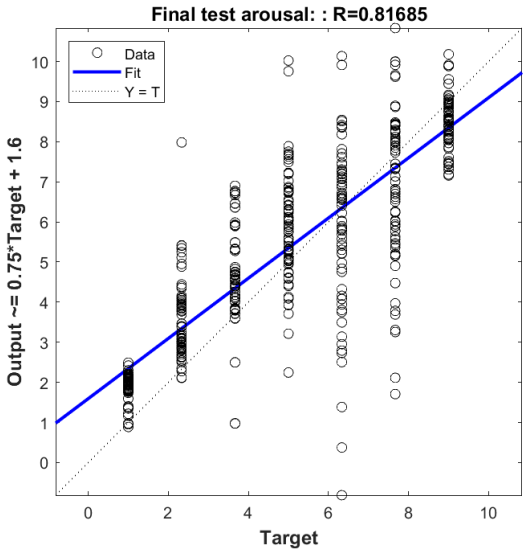

Figure 3: Regression for arousal

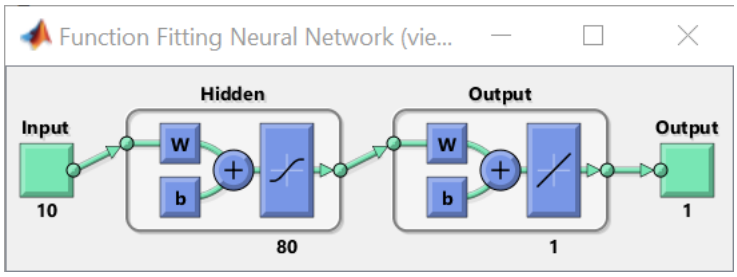

Figure 4: MLP Structure for valence

```
% Configuration for training of the MLP for valence
% (non present fields are the default ones)
mlp_net_valence.trainParam.lr = 0.1;
mlp_net_valence.trainParam.epochs = 100;
mlp_net_valence.trainParam.max_fail = 15;
```

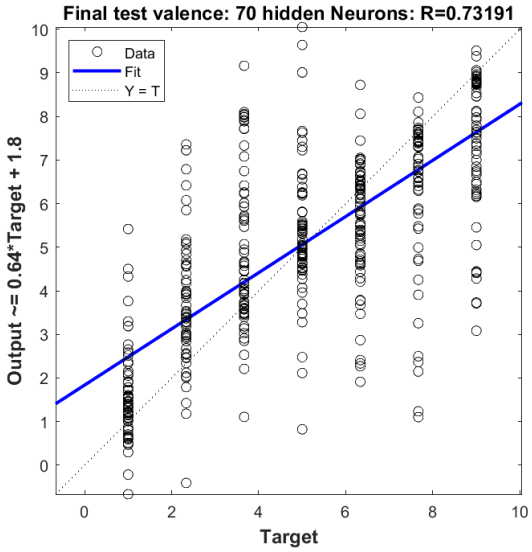

Figure 5: Regression for valence

As we can see better results were obtained with the MLP for the arousal. An interesting observation can be inferred by watching the result of the training process: data was split another time into a training set, a test set and a validation set (for early stopping) with the following results in term of regression:

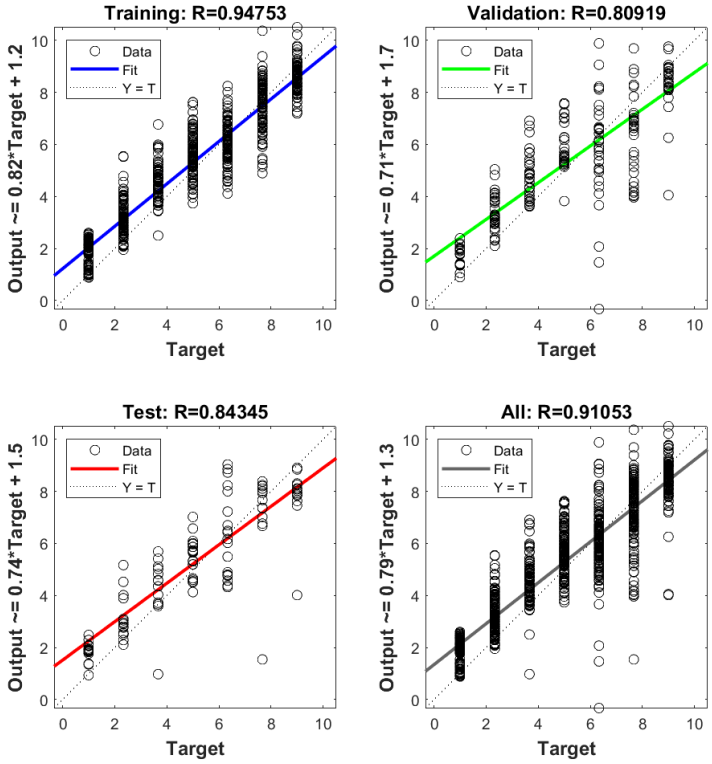

Figure 6: MLP regression results after training

In this case the test set has a slightly higher R-squared w.r.t. the test-set obtained at

the beginning with the holdout partition. In fact those samples has been given in input to the `sequentialfs` giving a positive bias for the regression of the data.

2.3 RBF Training

For what concerns RBF networks, the training process adopted was simpler since there were less degrees of freedom with the `newrb` function. In particular experiments were carried out by changing values of the **spread** in order to obtain the best possible result in terms of regression w.r.t. the test set obtained by the holdout partition. The final parameters were the following:

```
%Parameters for training the RBF for arousal
spread_ar = 1.07;
goal_ar = 0;
K_ar = 1200; %Maximum number of neurons
Ki_ar = 100; %Step (default was 50, to make the process quicker it
 was increased)
```

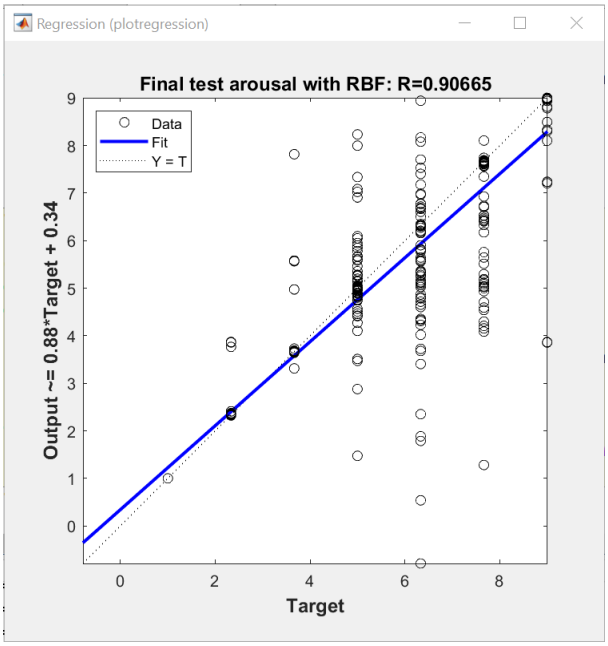

Figure 7: Regression of RBF for arousal with Test Set

```
%Parameters for training the RBF for valence
spread_va = 0.7;
goal_va = 0;
K_va = 1200;
Ki_va = 100;
```

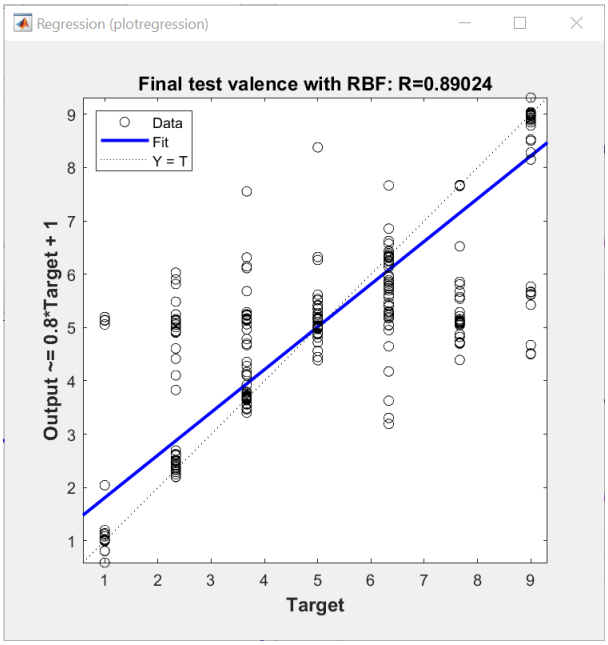

Figure 8: Regression of MLP with Test Set

3 — Task 3.3

In order to project the Mamdani Fuzzy Inference System of this task, an analysis of the three best features of `sequentialfs` has been done. In particular, were studied the **empirical distributions** of the features through some histograms. Then, a **correlation study** has been carried out by plotting scatter-plots with different combinations of pairs of these three features. In the following are presented those results:

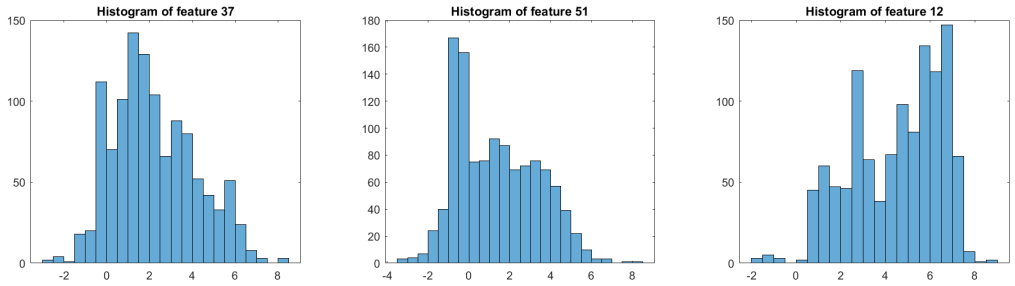

Figure 9: Histograms of three features

The empirical distribution has been used to model the linguistic variables of the three features. The approach adopted was the following: since the empirical distributions are "fuzzy", but also tends to have peaks in some points, those peaks were considered as central point of a particular linguistic label.

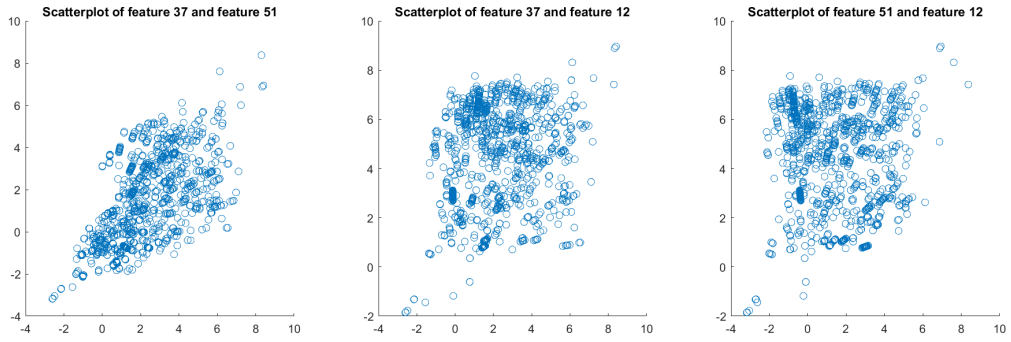

Figure 10: Scatter-plot of pairs of features

By watching the scatter-plot, we can see a certain positive correlation between feature 37 and 51 in the scatter-plot on the left, while other pairs seems uncorrelated due to the fact that the plot looks "square" without a particular tendency like the first one.
As a conclusion,in the antecedent of a fuzzy rule, an high(low) level of feature 37 with low(high) level of feature 51 won't be used due to their positive correlation.

So, for what concerns modeling of the linguistic variables of the **inputs**, when three linguistic labels were applied the central membership function was selected as a **triangle** while on the sides a **trapezoidal** membership function to cover all the universe of discourse of that particular feature.
As stated before the position of the various membership function was done w.r.t. the

peaks of the empirical distribution of features.

They inputs has been modeled as follows:

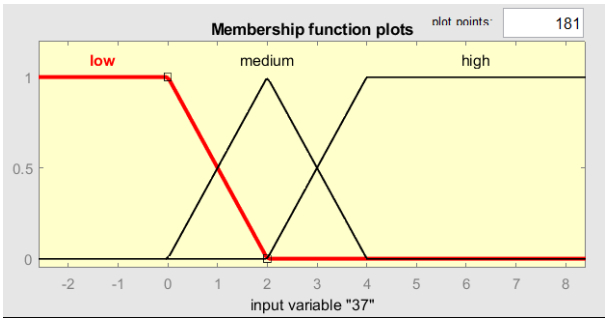

Figure 11: Modeling of linguistic variable of the input for feature 37

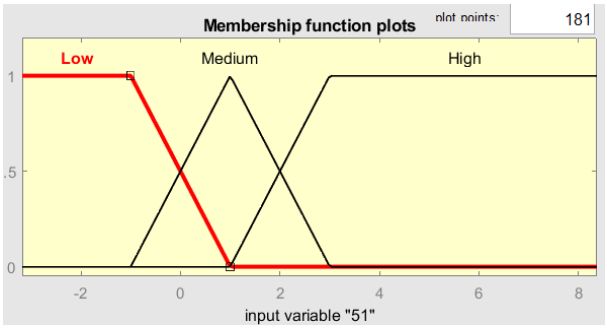

Figure 12: Modeling of linguistic variable of the input for feature 51

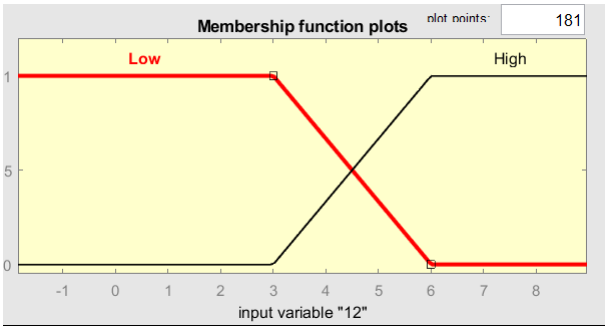

Figure 13: Modeling of linguistic variable of the input for feature 12

For the arousal linguistic variable **output**, since the input data is not particularly "defined" but has some uncertainty in the histograms, the output has been modeled in only three different linguistic labels with triangular membership function otherwise creating some fuzzy rules was very complicate:

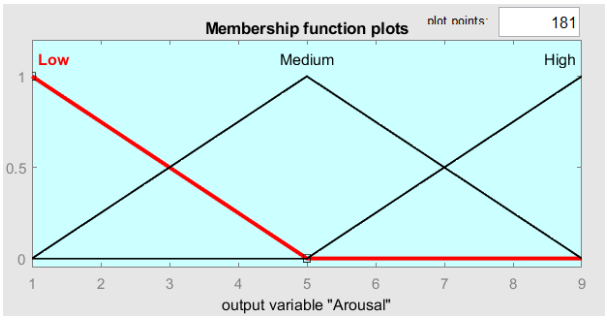

Figure 14: Modeling of linguistic variable of the arousal output

For what concerns the rules, other histograms were plotted, this time by only considering a subset of the samples coinciding to a specific range of arousal. Some samples were not considering in the modeling of the rules (red dotted line is used as a threshold).

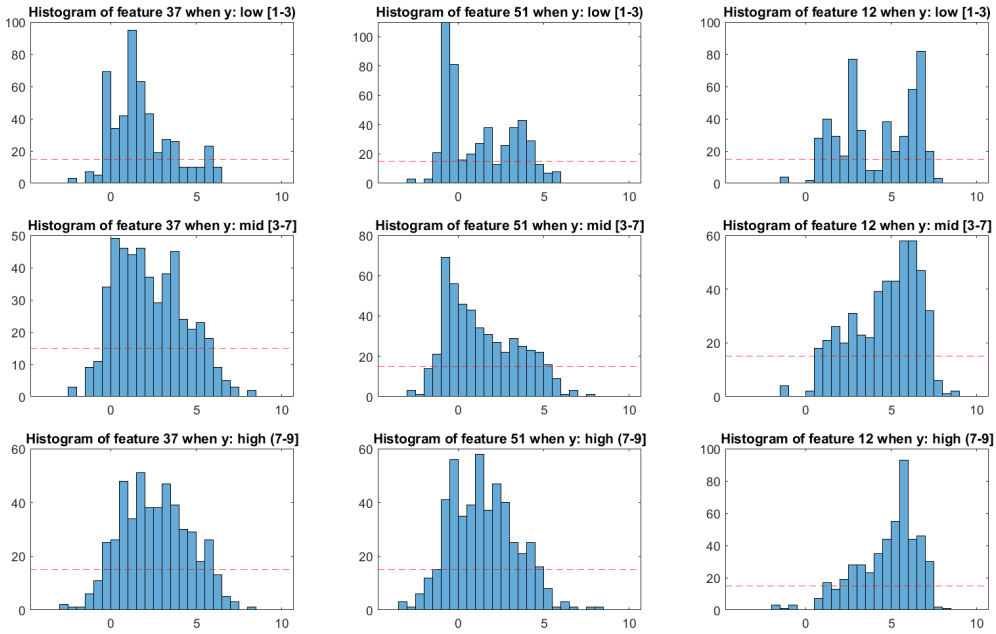

Figure 15: Histograms of three features with ranges of arousal

The rules modeled are the following:

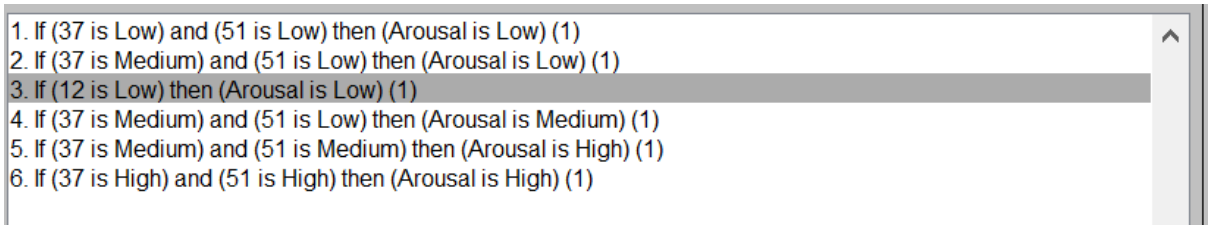

Figure 16: Fuzzy Rules

For example: rule 3 was defined since only in the first histogram of feature 12 "lower" values appear. For features 37 and 31 was not very simple to find patterns so, at the

end, the rules were generated by respecting the statement on the correlation above and comparing the positions of the peaks.

4 — Task 4.1

4.1 Dataset Preparation

This task of the project has been carried out in the following way: two CNNs were modeled, one for classifying all 4 emotions, one for only two emotions (fear and disgust). In order to do so two subsets of the initial dataset were chosen: since random selection of images gave poor performance in terms of accuracy, **250 images per class were hand selected** for both CNN. The hand selection permitted to share a specific characteristic between images of the same classing (like smiling in case of happiness or eyes open widely in case of fear)
Since the training set obtained (80% of the whole set of images for both cases) were not so high, **data augmentation was performed** by reflecting, slightly rotating and translating images in the following way:

```
%Method for image augmentation
imageAugmenter = imageDataAugmenter('RandRotation',[-20,20],...
'RandXReflection', true, ...
'RandXTranslation', [-10 10], ...
'RandYTranslation', [-10 10]);
```

In general both CNN were developed in order to obtain the best validation accuracy and minimizing at the same time the size of both CNN. A thing noticed through some experiments was that leakyReluLayers in combination with `sgdm` algorithm permitted to achieve a speed up of the training process. Going deeper with the layers, Convolutional Layers decreases the stride and the filter dimension, while increasing the number of those was. MaxPooling Layers do the same thing for the stride parameter.
The final architectures of CNNs obtained taking into account all the points expressed were the following:

```
  %CNN layers for two-classes classification
  leaky_layers_two = [
    imageInputLayer(size(base_img))

    convolution2dLayer(16,6,'Stride',4,'Padding','same')
    batchNormalizationLayer
    leakyReluLayer

    maxPooling2dLayer(2,'Stride',2)

    convolution2dLayer(8,12,'Stride',2,'Padding','same')
    batchNormalizationLayer
    leakyReluLayer

    maxPooling2dLayer(2,'Stride',2)

    fullyConnectedLayer(40)
    fullyConnectedLayer(2)

    softmaxLayer

    classificationLayer
  ];
```

```
  %CNN layers for four-classes classification
```

```
leaky_layers_four = [
  imageInputLayer(size(base_img))

  convolution2dLayer(16,8,'Stride',4,'Padding','same')
  batchNormalizationLayer
  leakyReluLayer

  maxPooling2dLayer(2,'Stride',2)

  convolution2dLayer(8,16,'Stride',2,'Padding','same')
  batchNormalizationLayer
  leakyReluLayer

  maxPooling2dLayer(2,'Stride', 1)

  convolution2dLayer(4,32,'Stride',2,'Padding','same')
  batchNormalizationLayer
  leakyReluLayer

  maxPooling2dLayer(2,'Stride',1)

  fullyConnectedLayer(100)
  fullyConnectedLayer(100)
  fullyConnectedLayer(4)

  softmaxLayer

  classificationLayer
];
```

In the following are presented the result of training process for both CNNs:

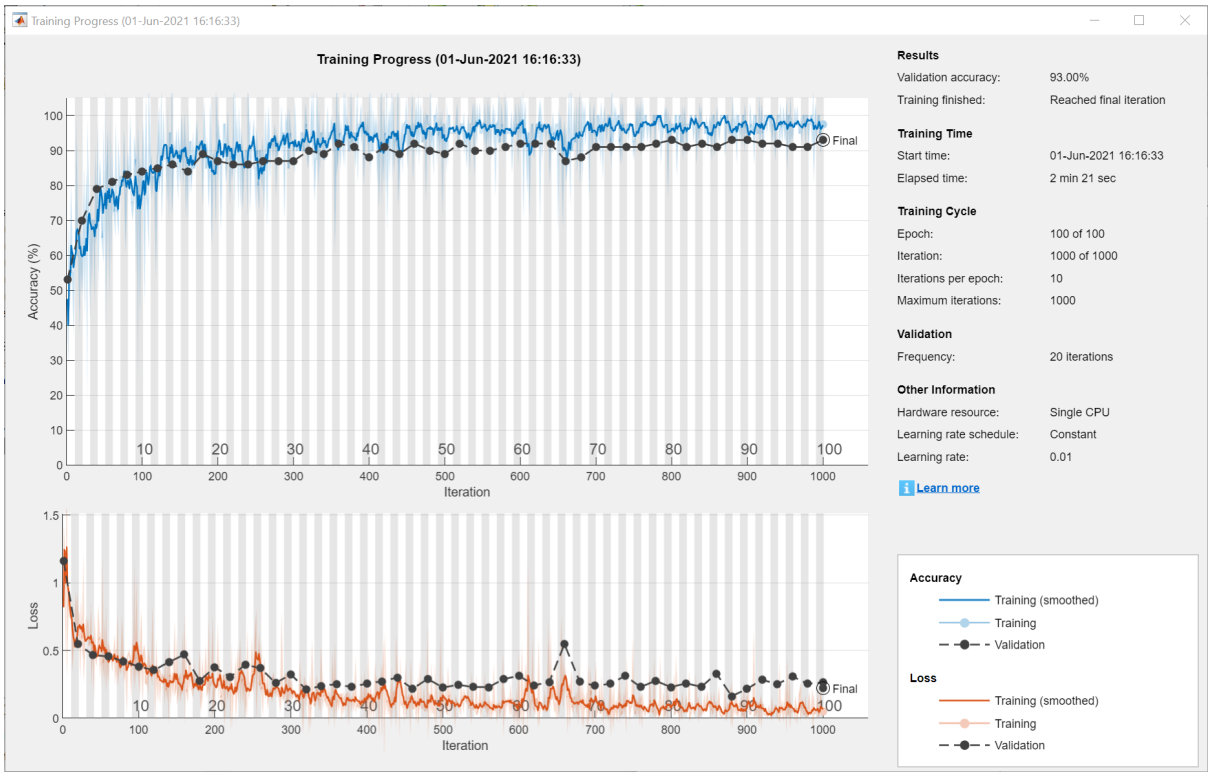

Figure 17: Training process of the CNN with 2 classes (disgust and fear)

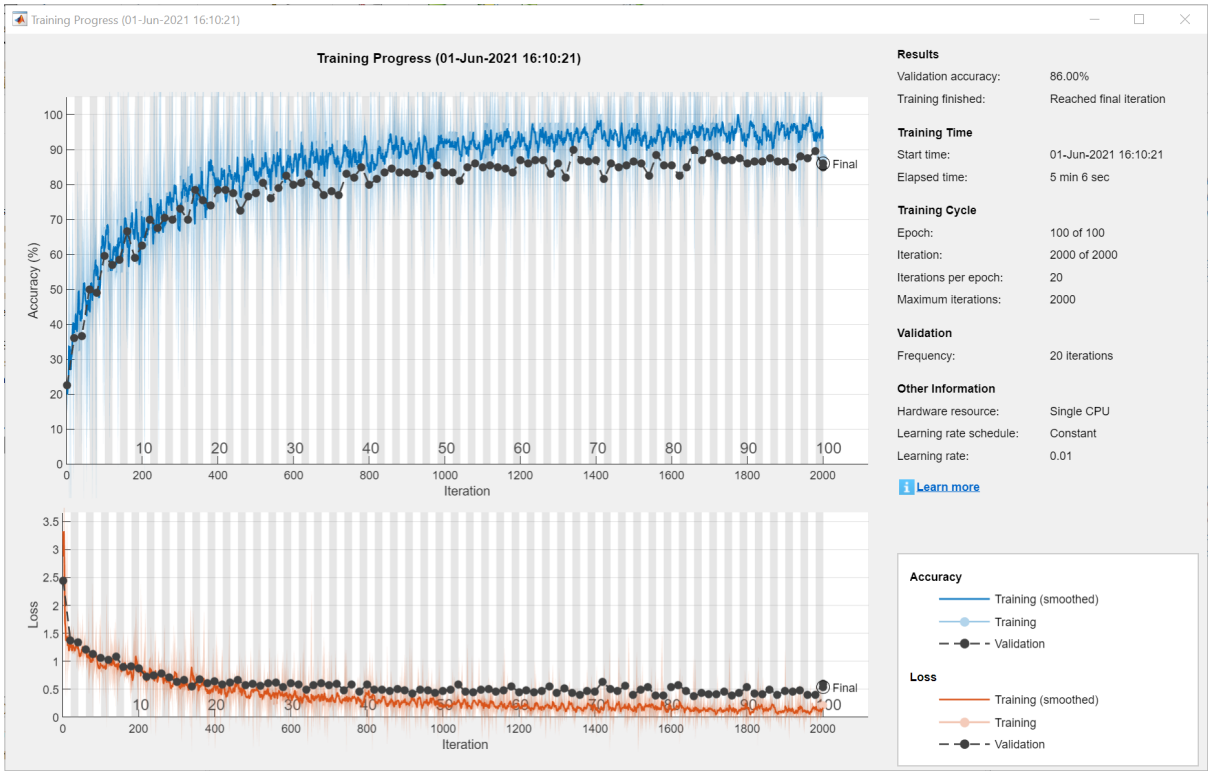

Figure 18: Training process of the CNN with 4 classes

As a last point the validation accuracy was analysed in a deeper by plotting the Confusion matrix for both CNNs. In fact, during the experimentation. by modifying the hand selected images of poor classified classes, the CNNs accuracy improved. The final confusion matrix obtained are the following:

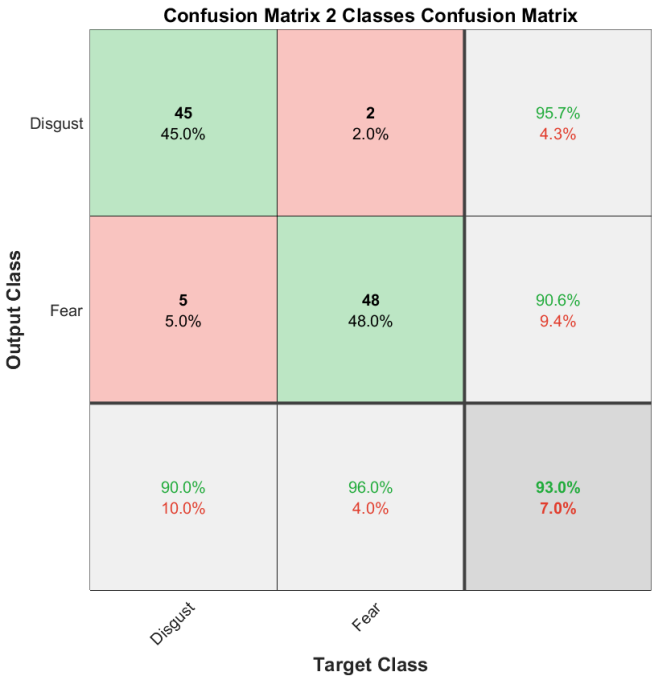

Figure 19: Confusion Matrix of CNN with two classes (confusion and fear)

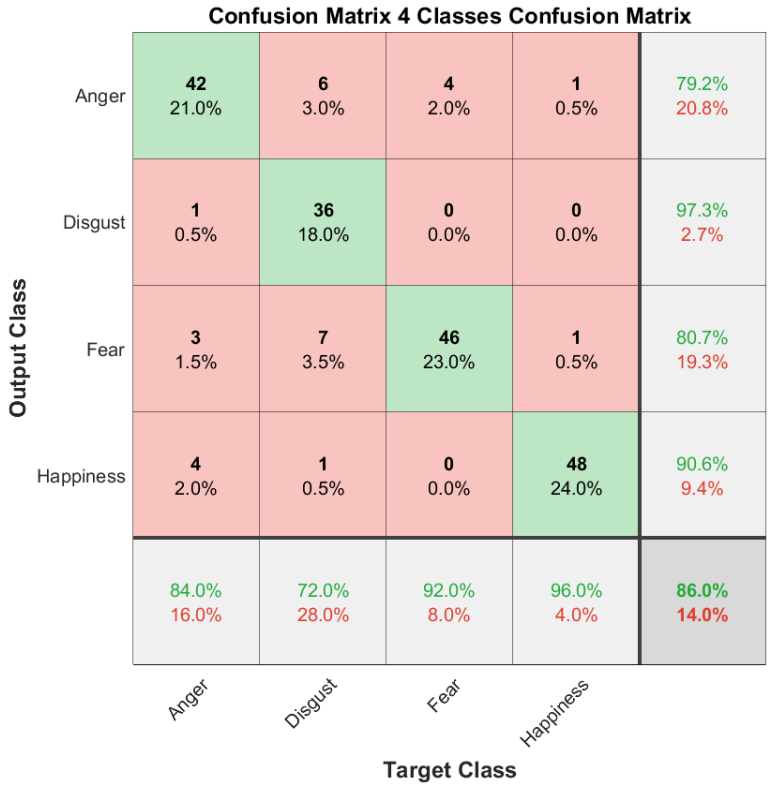

Figure 20: Confusion Matrix of CNN with four classes