Università di Pisa

School of Engineering

Master of Science in Computer Engineering

Intelligent Systems

PROJECT DOCUMENTATION

Clarissa Polidori

Academic Year 2022/2023

Contents

Chapter 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 that have been carried out for this project are:

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

Chapter 2

Dataset pre-processing

Before starting to work with the provided data, it is necessary to perform a pre-processing of the dataset, which was performed as follows:

- Removing non-numeric data and infinite values using the function isinf function of MATLAB.
- Removing outliers using MATLAB's rmoutliers function with the default median method.
- Balancing of the dataset.
- Feature selection.
- Normalization of data.

We now talk about the most relevant steps of data pre-processing.

The first three points are contained in the file *data_preparation.mat*, the last two in the file *feature_selection.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, the median method has been selected.

The last 3 points will be discussed in the following sections.

2.1 Balancing of the dataset

The dataset is composed by 54 signals and for each of them a value for valence and one for arousal are presented. Considering that valence and arousal can take 7 different values, the dataset can be split in 7 classes. The distribution of samples among the classes is the following:

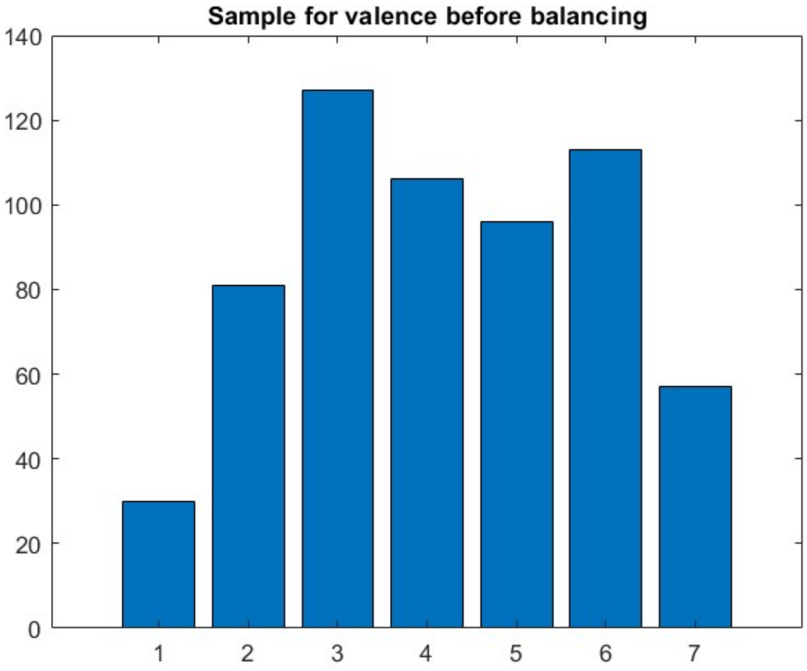


Figure 2.1: Samples distributions for valence before balancing

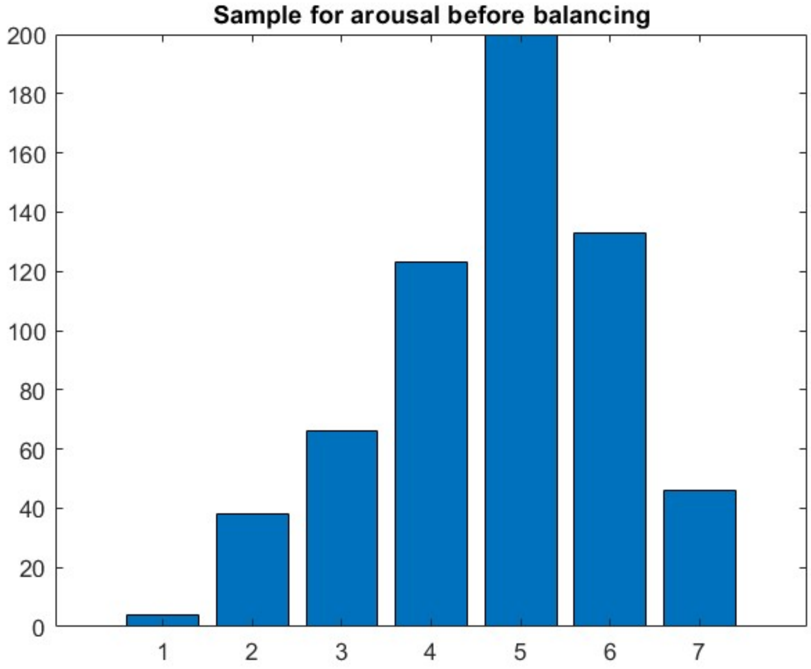


Figure 2.2: Samples distributions for arousal before balancing

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].
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:

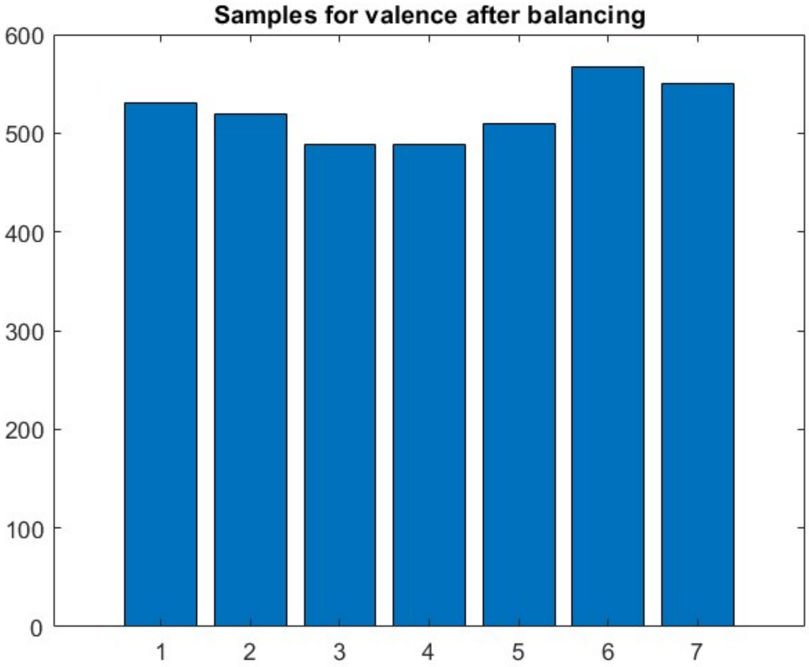


Figure 2.3: Samples distributions for valence after balancing

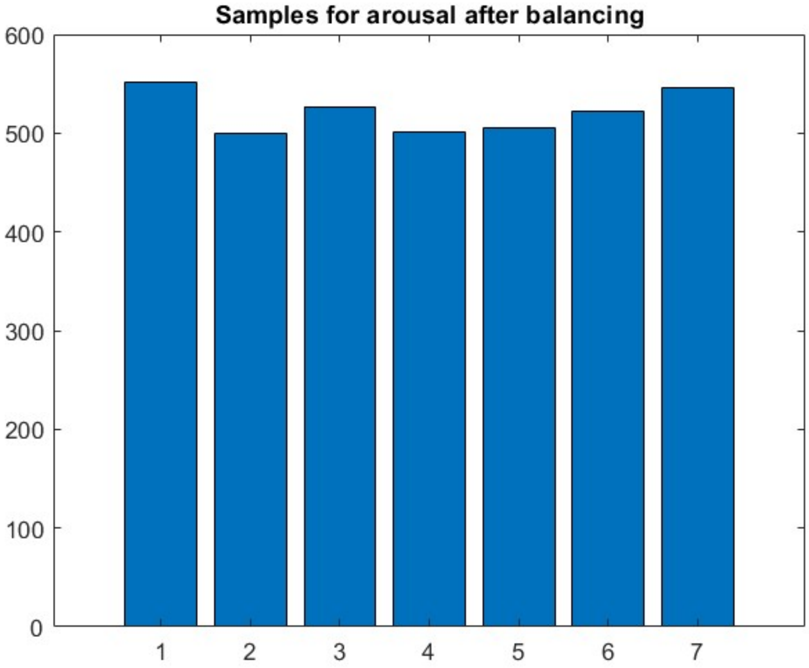
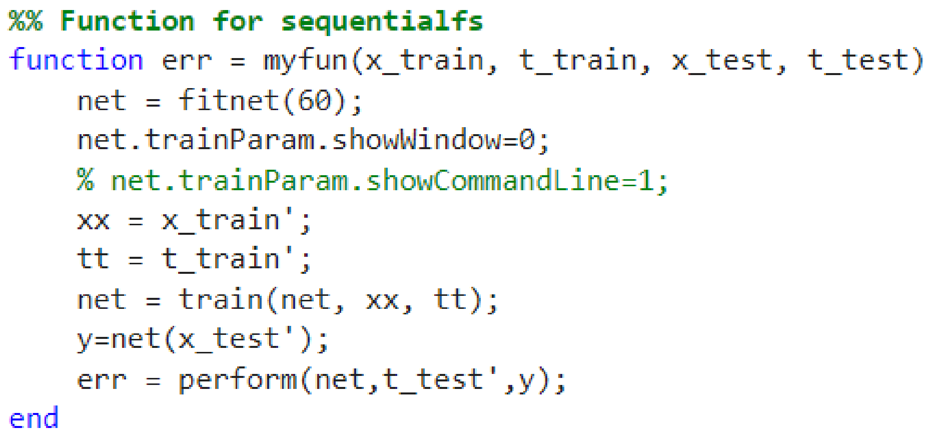


Figure 2.4: Samples distributions for arousal before balancing

2.2 Feature selection

The MATLAB function sequentialfs was used for feature selection, as recommended.

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.



```
%% Function for sequentialfs
function err = myfun(x_train, t_train, x_test, t_test)
    net = fitnet(60);
    net.trainParam.showWindow=0;
    % net.trainParam.showCommandLine=1;
    xx = x_train';
    tt = t_train';
    net = train(net, xx, tt);
    y=net(x_test');
    err = perform(net,t_test',y);
end
```

Figure 2.5: Sequentialfs fucntion used

Thus after the extraction of the holdout partition we perform 5 times the sequentialfs for arousal and 5 times for 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. 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

At the end the data obtained are saved into five **.mat** files:

- *training_arousal.mat:* which contains the training inputs and corresponding target outputs for arousal training

- *testing_arousal.mat:* which contains the testing inputs and corresponding target outputs for arousal testing
- *training_valence.mat:* which contains the training inputs and corresponding target outputs for valence training
- *testing_valence.mat:* which contains the testing inputs and corresponding target outputs for valence testing
- *best3.mat:* which is a structure containing the training data and testing data for arousal of the best 3 features selected. This dataset will be used for the task 3.3

2.3 Normalization of data

Normalization of data was done using the MATLAB function normalize specifying only the vector to be normalized, without any other parameters, the normalize function returns the vectorwise z-score of the data in A with center 0 and standard deviation 1.

Chapter 3

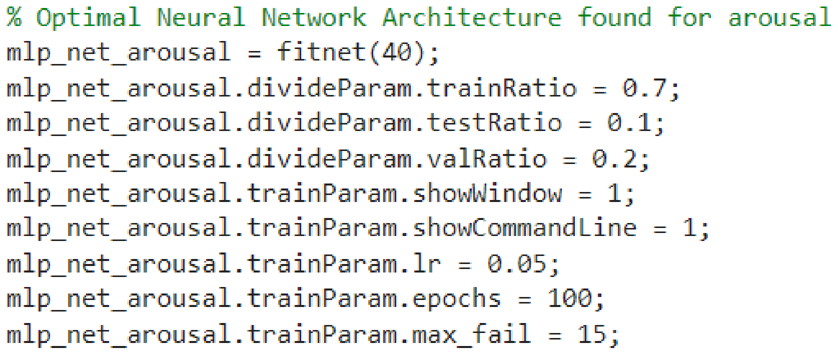
Task 3.1

3.1 Multi-layer Perceptron networks

This section describes the performance of task 3.3 of the project, namely, the design and development of two MLP networks that can accurately estimate a person's valence and arousal levels. What is reported here are the results obtained with experimentally derived configurations.

3.1.1 Arousal MLP network

The configuration used:



```
% Optimal Neural Network Architecture found for arousal
mlp_net_arousal = fitnet(40);
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 = 0.05;
mlp_net_arousal.trainParam.epochs = 100;
mlp_net_arousal.trainParam.max_fail = 15;
```

Figure 3.1: Configuration for MLP network for arousal

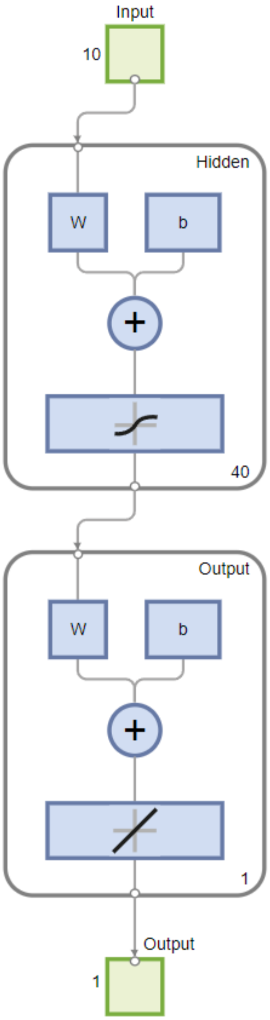


Figure 3.2: Arousal MLP network architecture

The validation set was used for the *early stopping*:

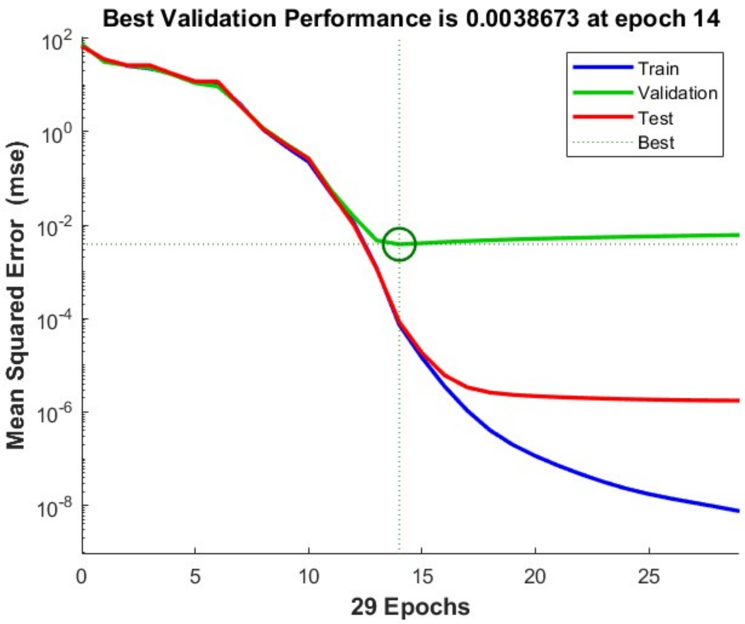


Figure 3.3: Best validation performance for arousal

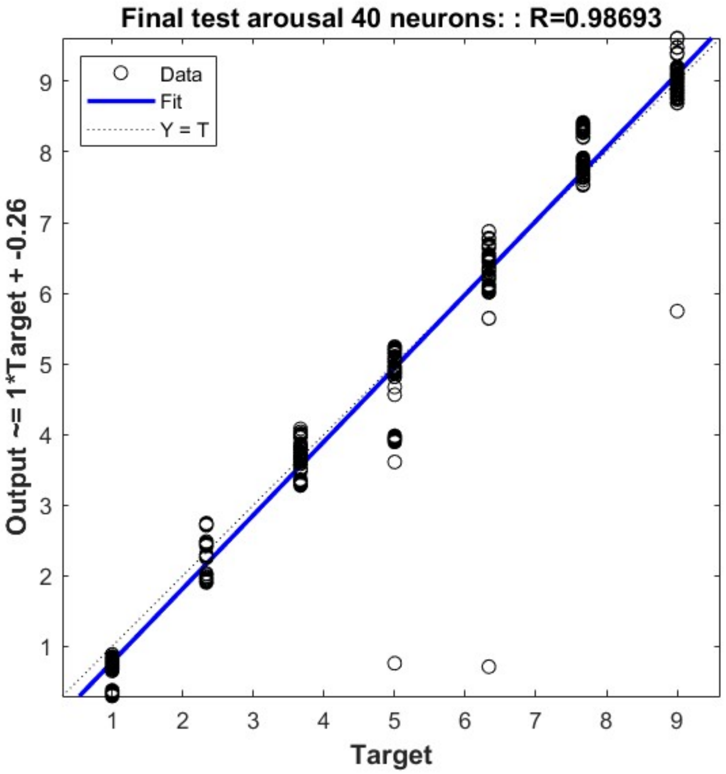
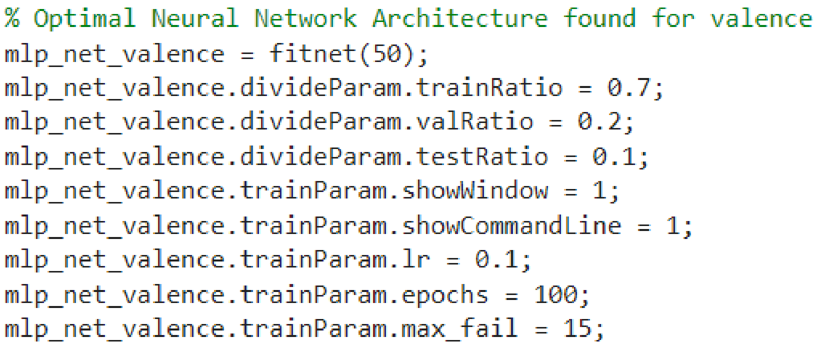


Figure 3.4: Regression for arousal MLP network

3.1.2 Valence MLP network

The configuration used:



```
% Optimal Neural Network Architecture found for valence
mlp_net_valence = fitnet(50);
mlp_net_valence.divideParam.trainRatio = 0.7;
mlp_net_valence.divideParam.valRatio = 0.2;
mlp_net_valence.divideParam.testRatio = 0.1;
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

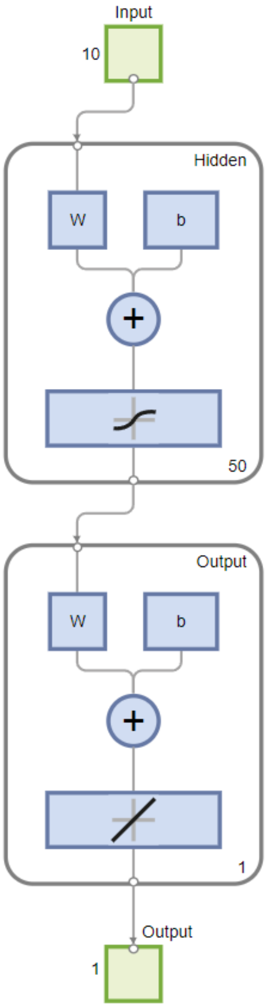


Figure 3.6: Valence MLP network architecture

The validation set was used for the *early stopping*:

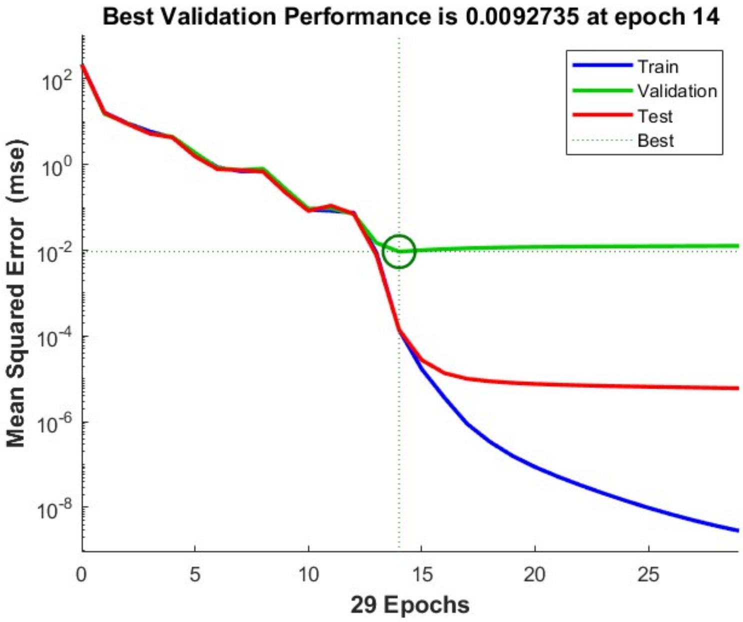


Figure 3.7: Best validation performance for valence

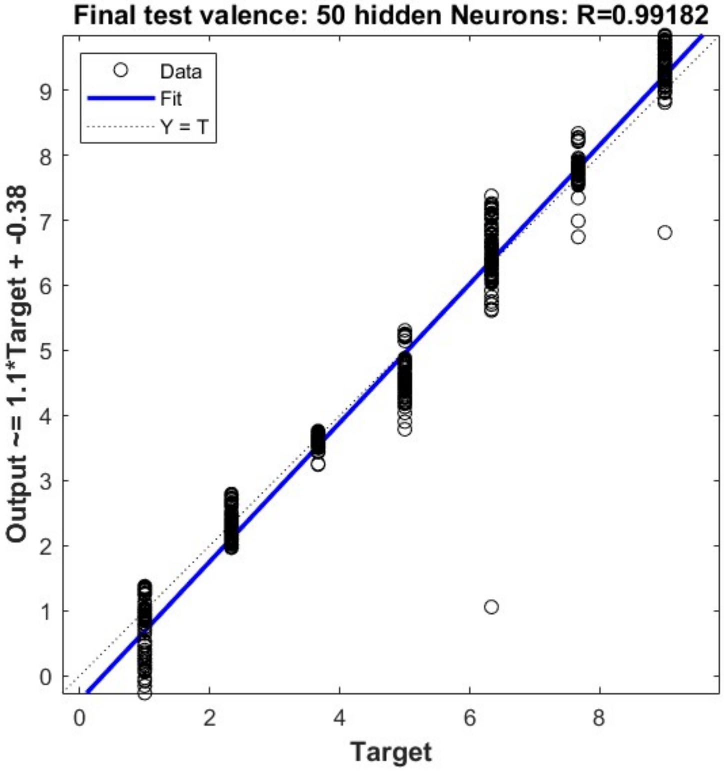
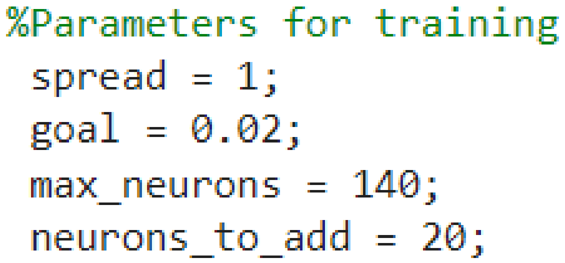


Figure 3.8: Regression for arousal MLP network

Clearly the results obtained for what concerns the error and regression value are in this case very good, but 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.

3.2 Radial Basis Function Networks

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 the following:



```
%Parameters for training
 spread = 1;
 goal = 0.02;
 max_neurons = 140;
 neurons_to_add = 20;
```

Figure 3.9: Arousal RBF network configuration

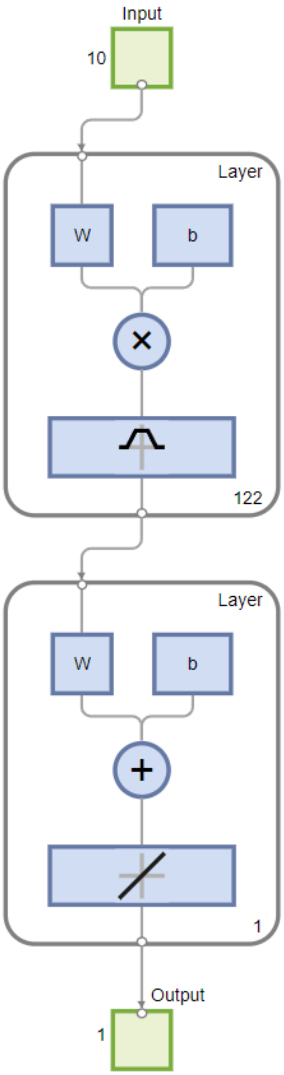


Figure 3.10: Arousal RBF network architecture

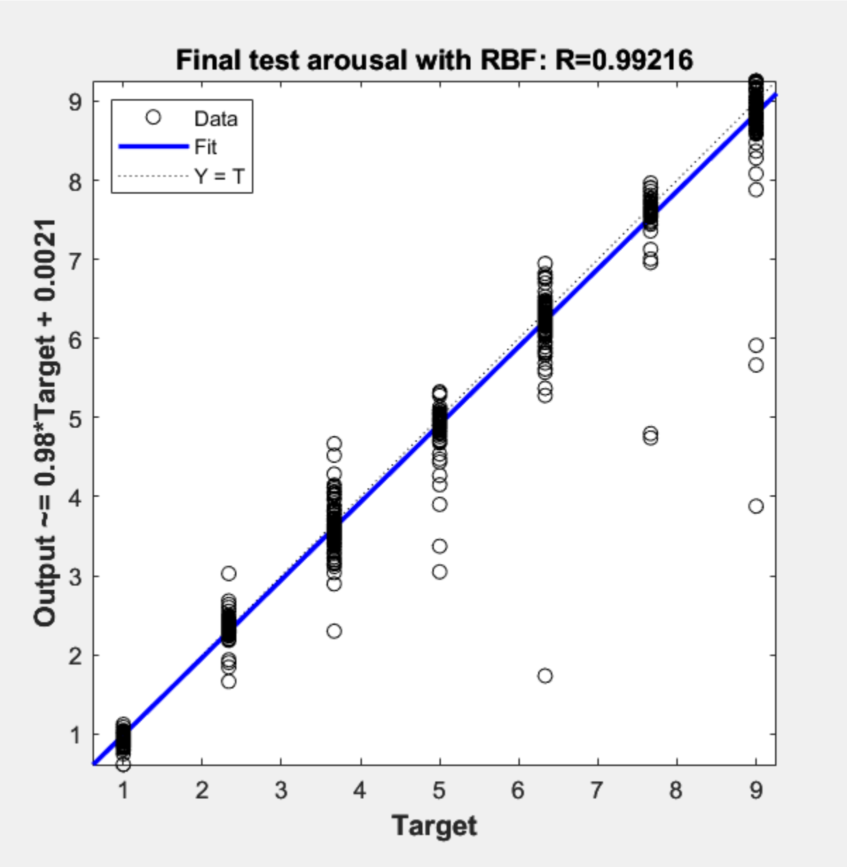
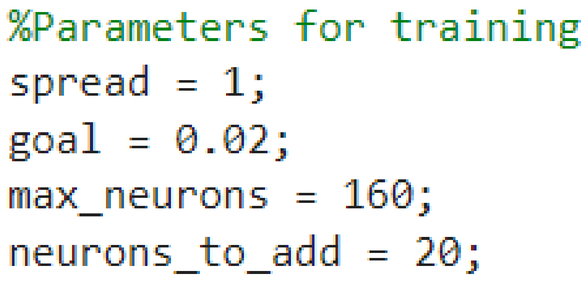


Figure 3.11: Regression for arousal RBF network



```
%Parameters for training
spread = 1;
goal = 0.02;
max_neurons = 160;
neurons_to_add = 20;
```

Figure 3.12: Valence RBF network configuration

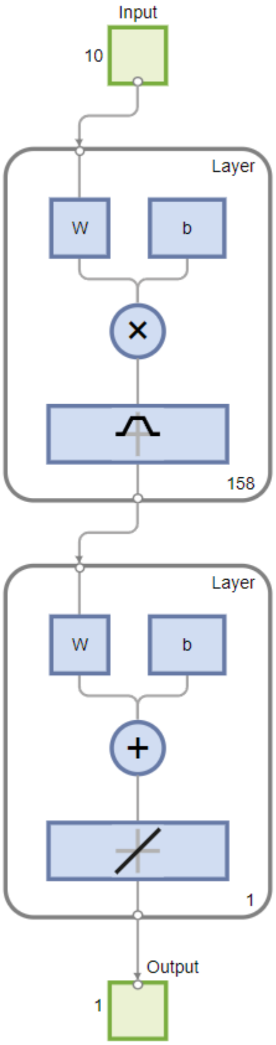


Figure 3.13: Valence RBF network architecture

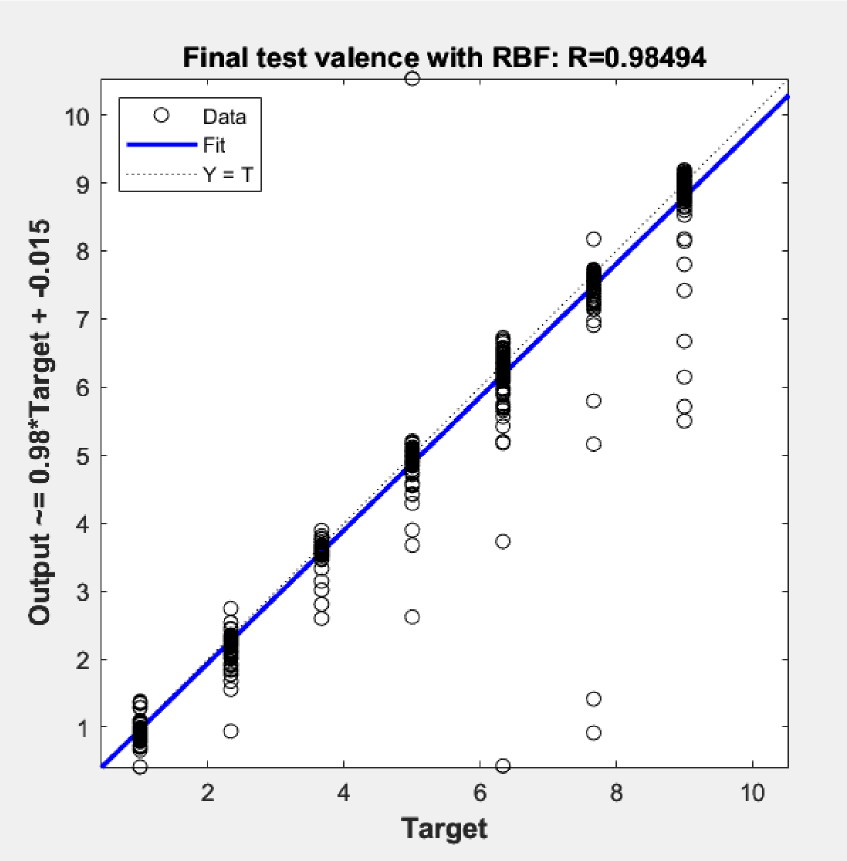


Figure 3.14: Regression for arousal RBF network

Chapter 4

Task 3.3

4.1 Mamdani fuzzy inference system

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:

- Feature 24: [-2.171518, 2.951837]
- Feature 27: [-2.511839, 1.808903]
- Feature 35: [-2.803462, 2.470656]

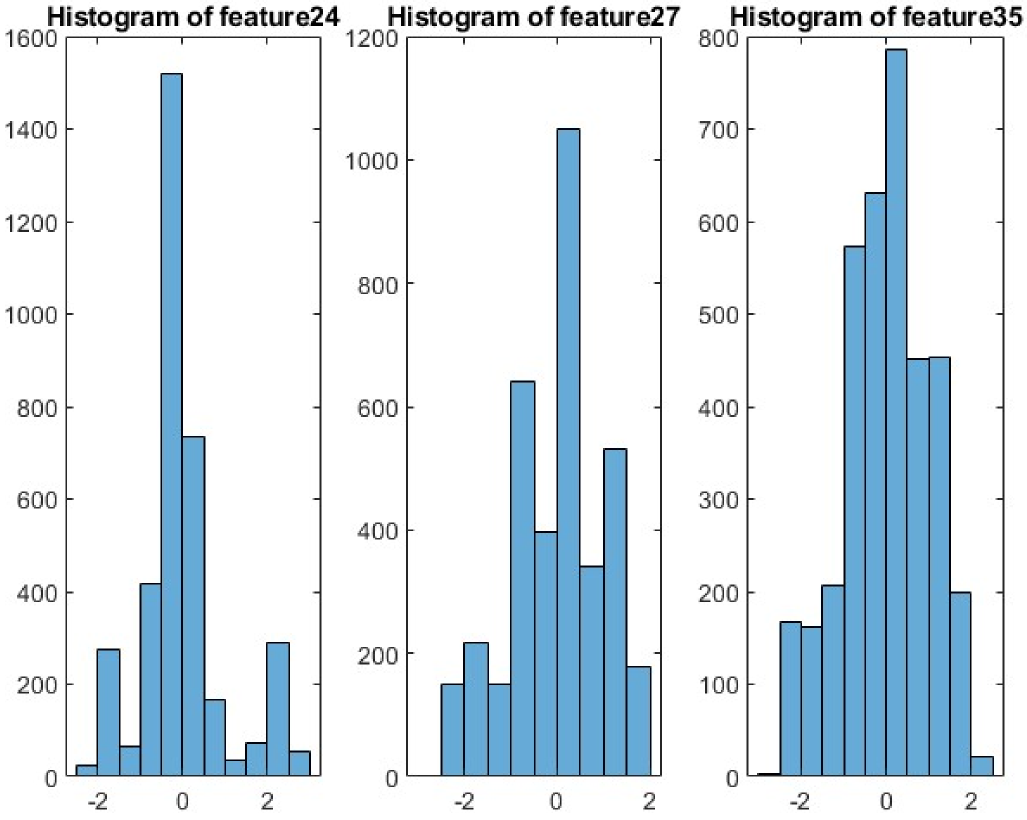


Figure 4.1: Empirical distributions of the three best features

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.

The most delicate part of the design of a Mamdani fuzzy inference system is that of rule definition, for which further analysis was also carried out. In particular, a correlation analysis with the different combinations of input features was performed to check the possible correlation between the selected features.

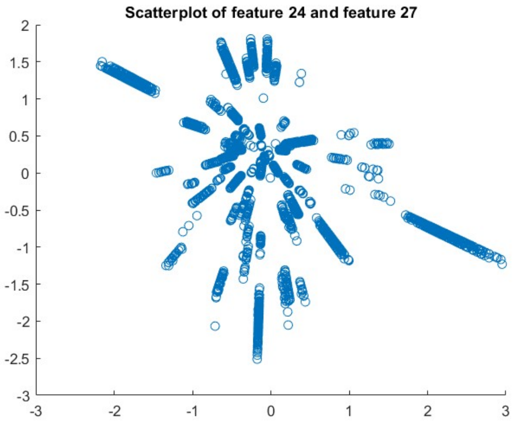


Figure 4.2: Correlation between feature 24 and feature 27

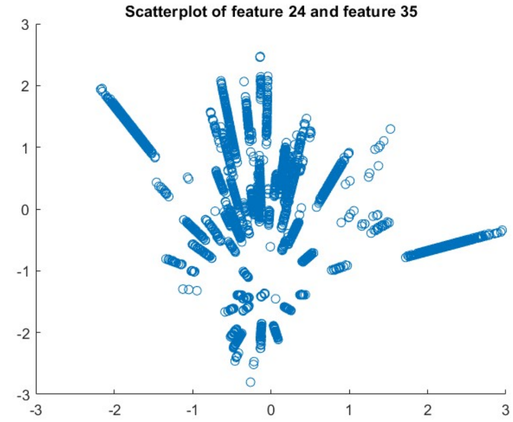


Figure 4.3: Correlation between feature 24 and feature 35

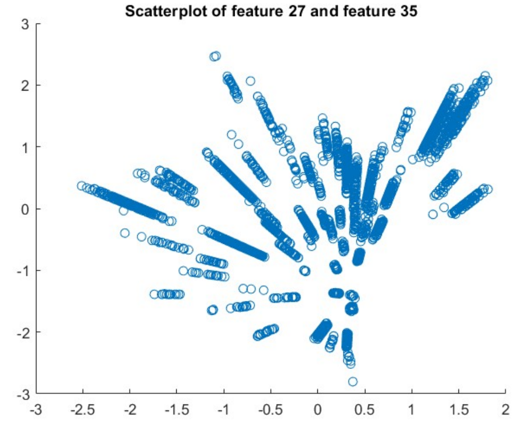


Figure 4.4: Correlation between feature 27 and feature 35

As we can see from the images the features are not correlated and also other histograms obtained considering only a subset of samples that coincide with a specific arousal range are shown.

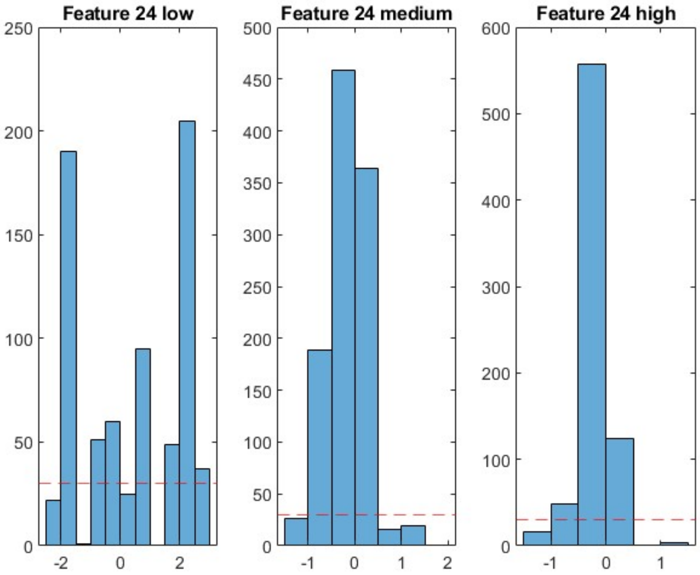


Figure 4.5: Feature 24 with different range of arousal

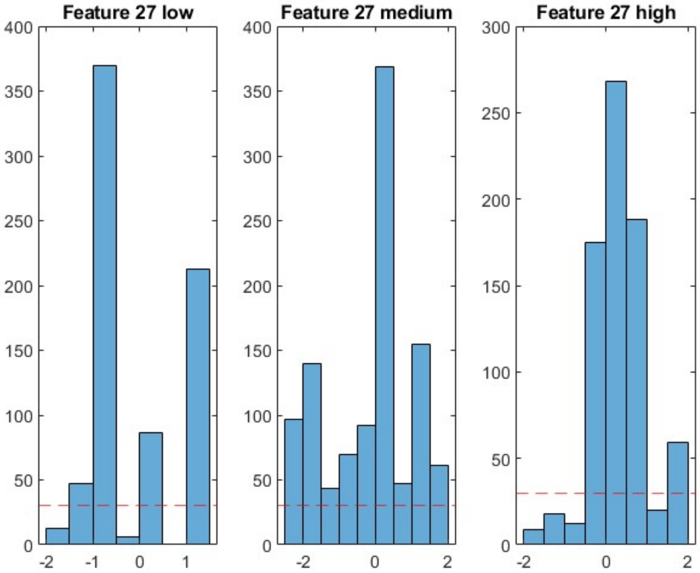


Figure 4.6: Feature 27 with different range of arousal

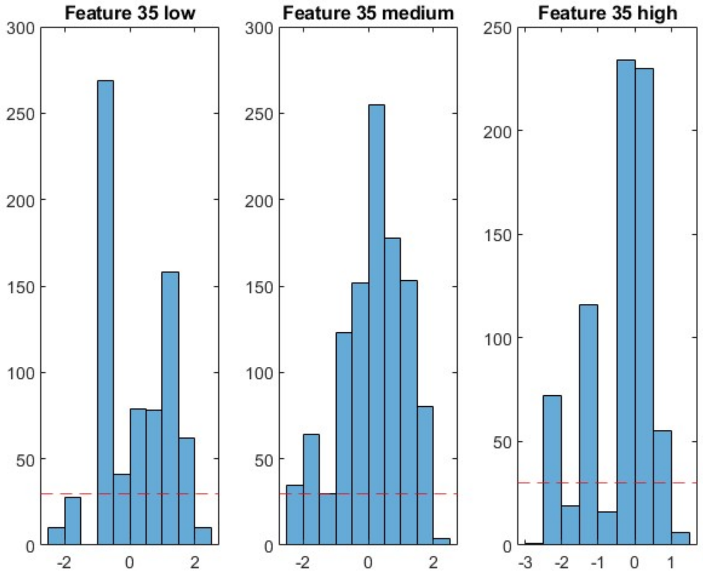


Figure 4.7: Feature 35 with different range of arousal

At this point to develop the fuzzy system we need to use the graphical user interface that is provided by the command *fuzzyLogicDesigner*.

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:

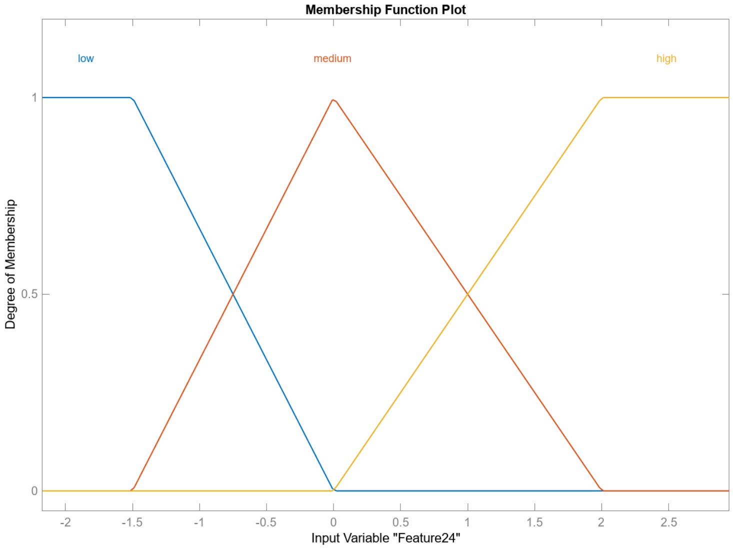


Figure 4.8: Modeling for linguistic variable of feature 24

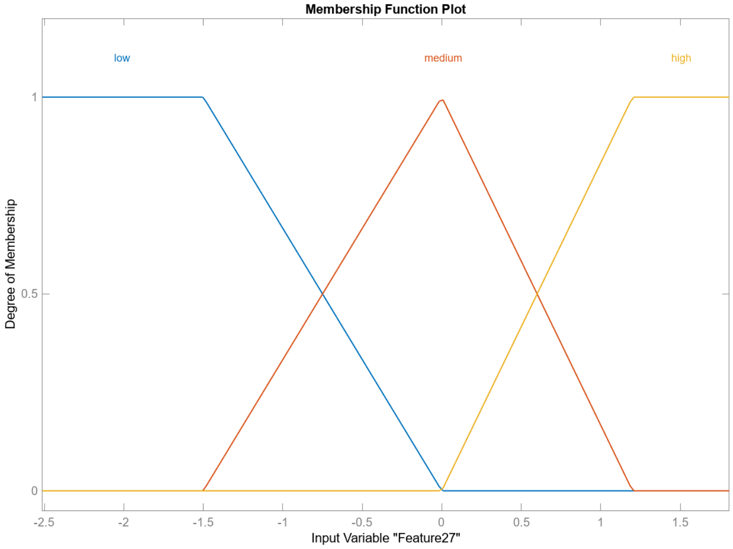


Figure 4.9: Modeling for linguistic variable of feature 27

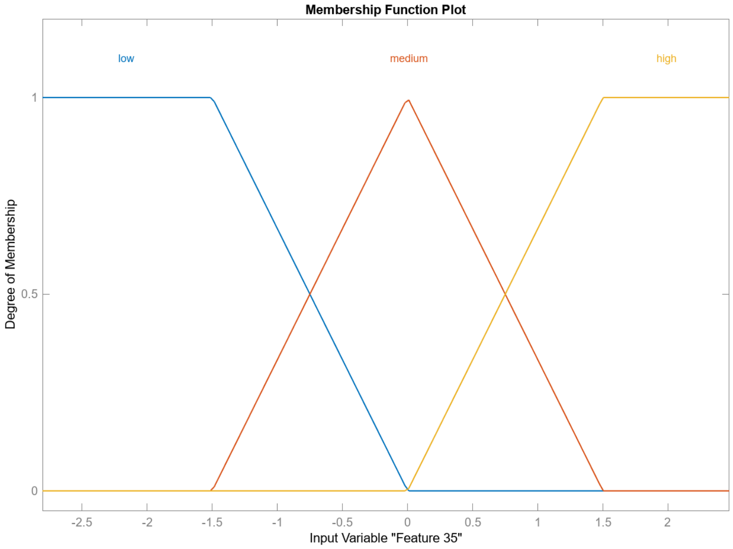


Figure 4.10: Modeling for linguistic variable of feature 35

We have already seen that there are 7 possible arousal values and therefore we can divide the dataset into 7 classes, so the first two classes were used to implement the arousal value *"low"*, the third, fourth and fifth classes implement the value *"medium"* while the last two classes implement the range *"high"*.

They output has been modeled as follows:

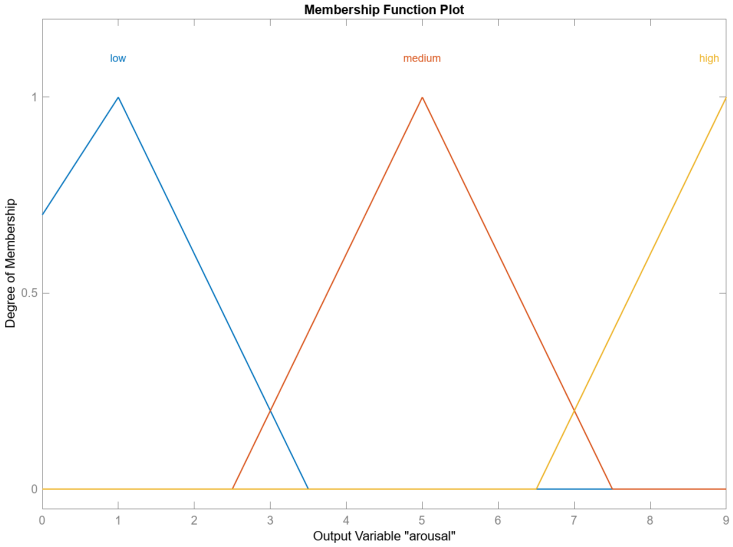
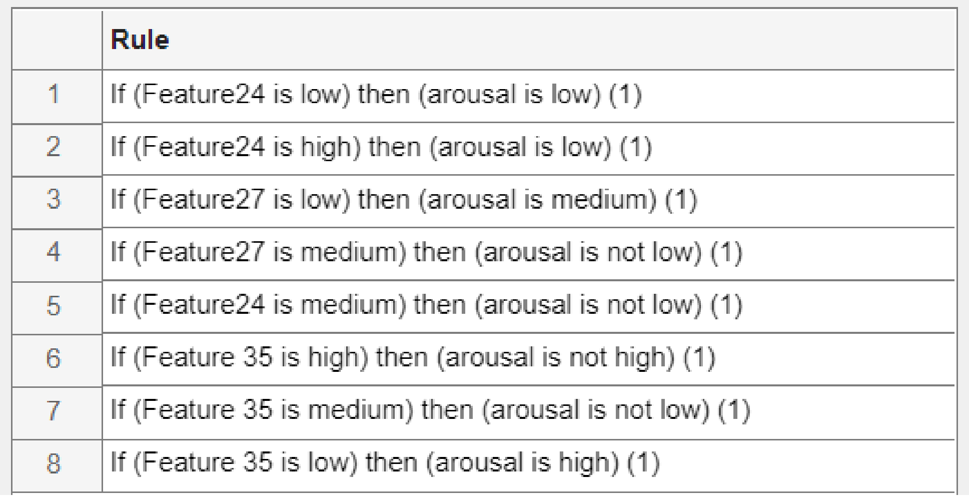


Figure 4.11: Modeling for linguistic variable of output

The following are the modeled rules:



	Rule
1	If (Feature24 is low) then (arousal is low) (1)
2	If (Feature24 is high) then (arousal is low) (1)
3	If (Feature27 is low) then (arousal is medium) (1)
4	If (Feature27 is medium) then (arousal is not low) (1)
5	If (Feature24 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 medium) then (arousal is not low) (1)
8	If (Feature 35 is low) then (arousal is high) (1)

Figure 4.12: Fuzzy rules

Chapter 5

Task 4.2

The following section sets out the development and training of a CNN based on the pretrained AlexNet. The network will be used to classify facial expressions and thus the possible classes to be classified are:

- *Anger*
- *Disgust*
- *Fear*
- *Happiness*

Since the dataset provided was unbalanced it was necessary to select images. For each class, 300 were chosen randomly or not based on the experiment carried out. For performance reasons, the case with only two classes, anger and happiness, was initially considered; later the experiment was repeated with all four classes. In all cases, 70 percent of the images were devoted to the training set, 20 percent to the validation set, and 10 percent to the testing set. All the images used were resized in order to fit the input size required by AlexNet. Moreover, a random translation of 30 pixels in horizontally and vertically way was performed as well on the training set in order to improve performance and prevent overfitting.

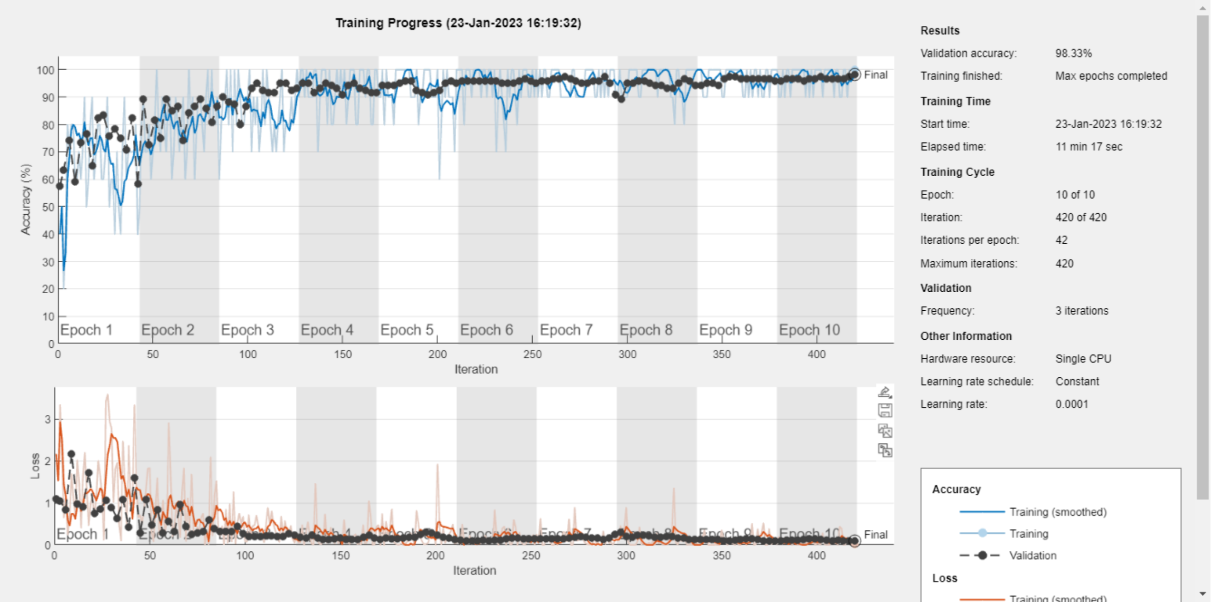
```
pixelRange = [-30 30];
imageAugmenter = imageDataAugmenter( ...
    ’RandXReflection’,true, ...
    ’RandXTranslation’,pixelRange, ...
    ’RandYTranslation’,pixelRange);

augmented_image_data_train = augmentedImageDatastore(input_size(1:2), data_train, ’DataAugmentation’, imageAugmenter);
augmented_image_data_validation = augmentedImageDatastore(input_size(1:2), data_validation);
augmented_image_data_test = augmentedImageDatastore(input_size(1:2), data_test);
```

The last 3 layers of AlexNet were substituted with other layers. This was made because AlexNet is trained to classify thousands of categories, but in this project there is just 2 or 4 classes in which data need to be classified.

```
layers = [
    original_layers
    fullyConnectedLayer(numClasses,’WeightLearnRateFactor’,20,’BiasLearnRateFactor’,20)
    softmaxLayer
    classificationLayer];
```

```
options = trainingOptions(’sgdm’, ...
    ’MiniBatchSize’, 10, ...
    ’MaxEpochs’, 10, ...
    ’InitialLearnRate’, 1e-4, ...
    ’Shuffle’, ’every-epoch’, ...
    ’ValidationData’,augmented_image_data_validation, ...
    ’ValidationFrequency’, 3, ...
    ’Verbose’, false, ...
    ’Plots’, ’training-progress’);
```

5.0.1 Classification with 2 classes



Results

Validation accuracy:	98.33%
Training finished:	Max epochs completed

Training Time

Start time:	23-Jan-2023 16:19:32
Elapsed time:	11 min 17 sec

Training Cycle

Epoch:	10 of 10
Iteration:	420 of 420
Iterations per epoch:	42
Maximum iterations:	420

Validation

Frequency:	3 iterations

Other Information

Hardware resource:	Single CPU
Learning rate schedule:	Constant
Learning rate:	0.0001

Figure 5.1: Training plot for pre-trained CNN with 2 classes and 300 images

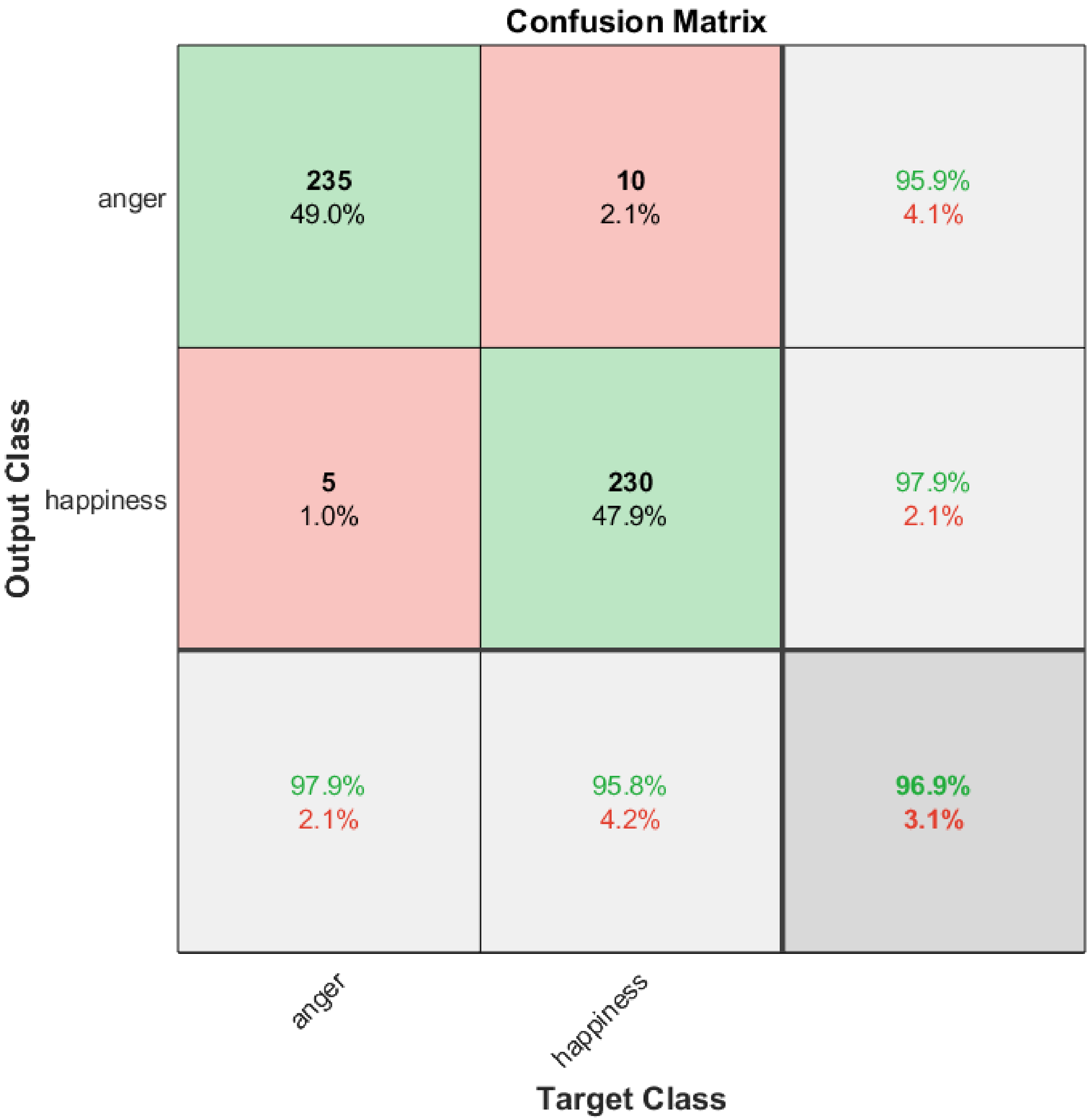
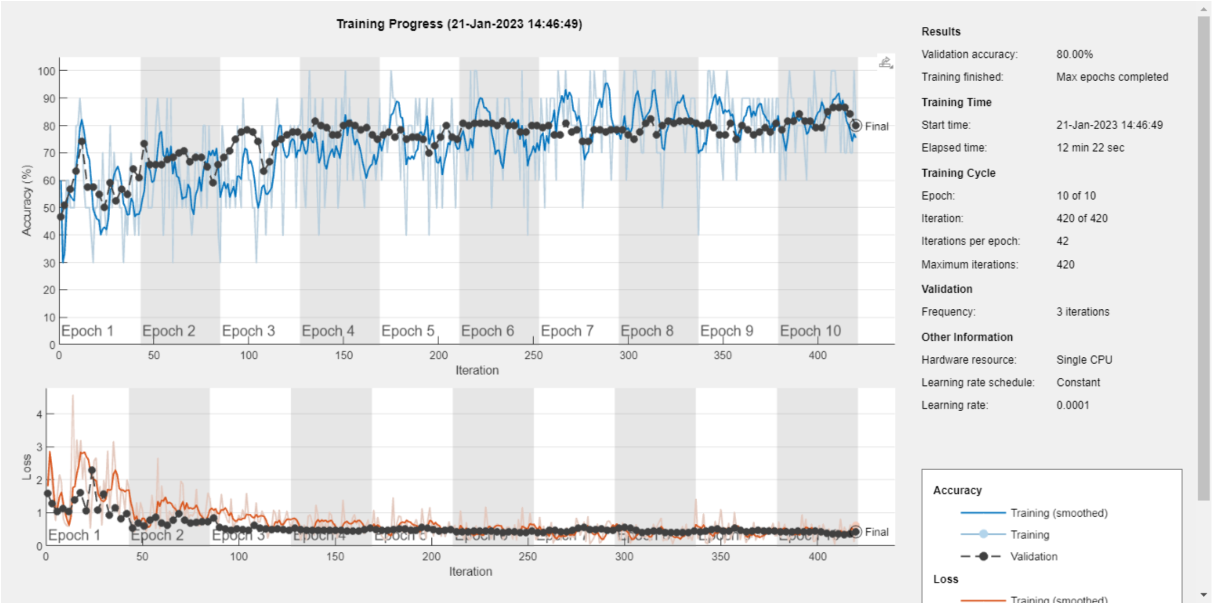


Figure 5.2: Confusion matrix for pretrained CNN with 2 classes and 300 images

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 because the images were not chosen totally randomly to perform this test but the most ambiguous images were excluded and also because the expressions of *anger* and *happiness* are easily distinguishable.

If we try to perform the same experiment with the classes *anger* and *disgust* we notice how the performance drops noticeably because these classes of expressions are more difficult to distinguish:



Results

Validation accuracy: 80.00%

Training finished: Max epochs completed

Training Time

Start time: 21-Jan-2023 14:46:49

Elapsed time: 12 min 22 sec

Training Cycle

Epoch: 10 of 10

Iteration: 420 of 420

Iterations per epoch: 42

Maximum iterations: 420

Validation

Frequency: 3 iterations

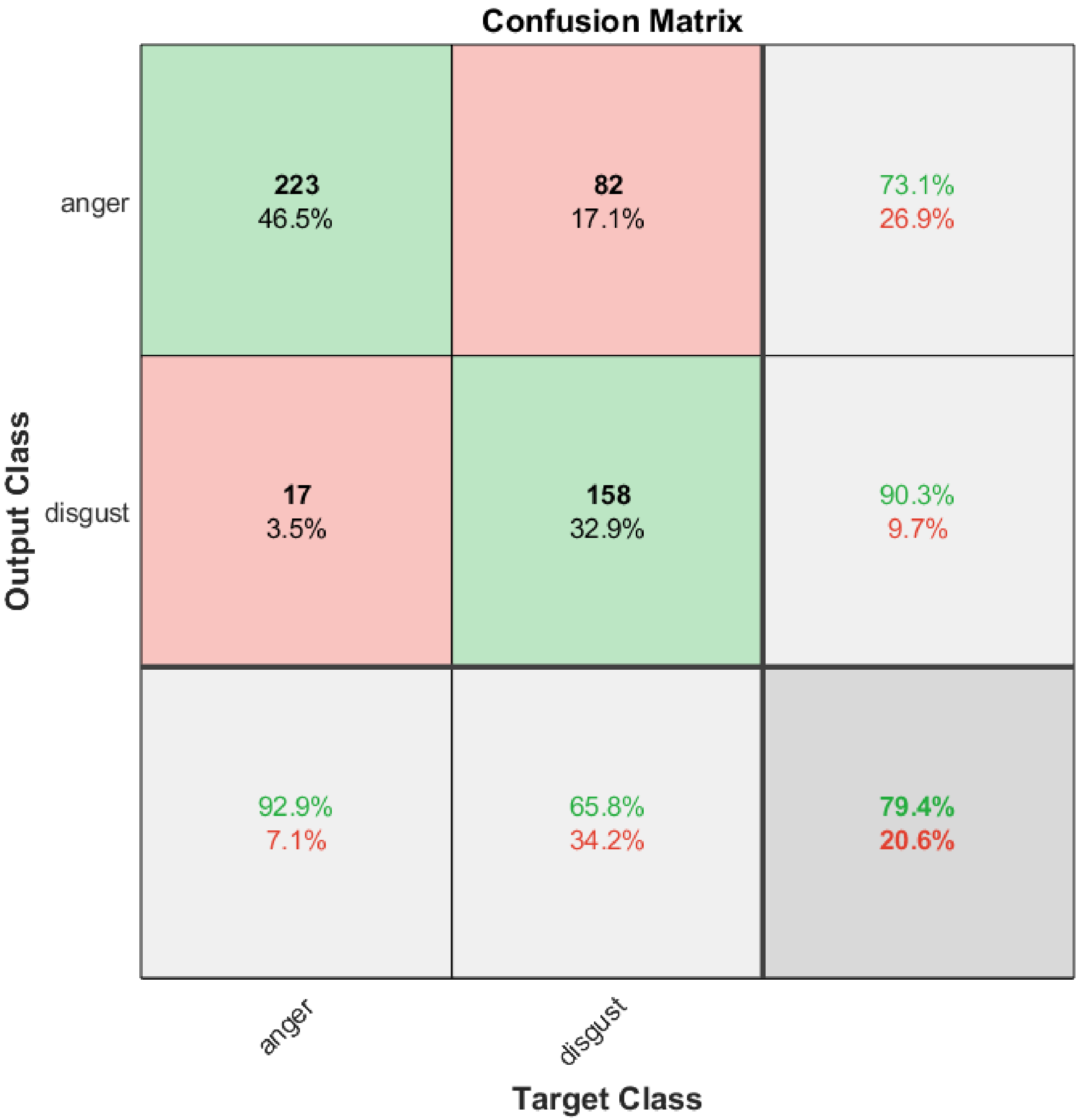
Other Information

Hardware resource: Single CPU

Learning rate schedule: Constant

Learning rate: 0.0001

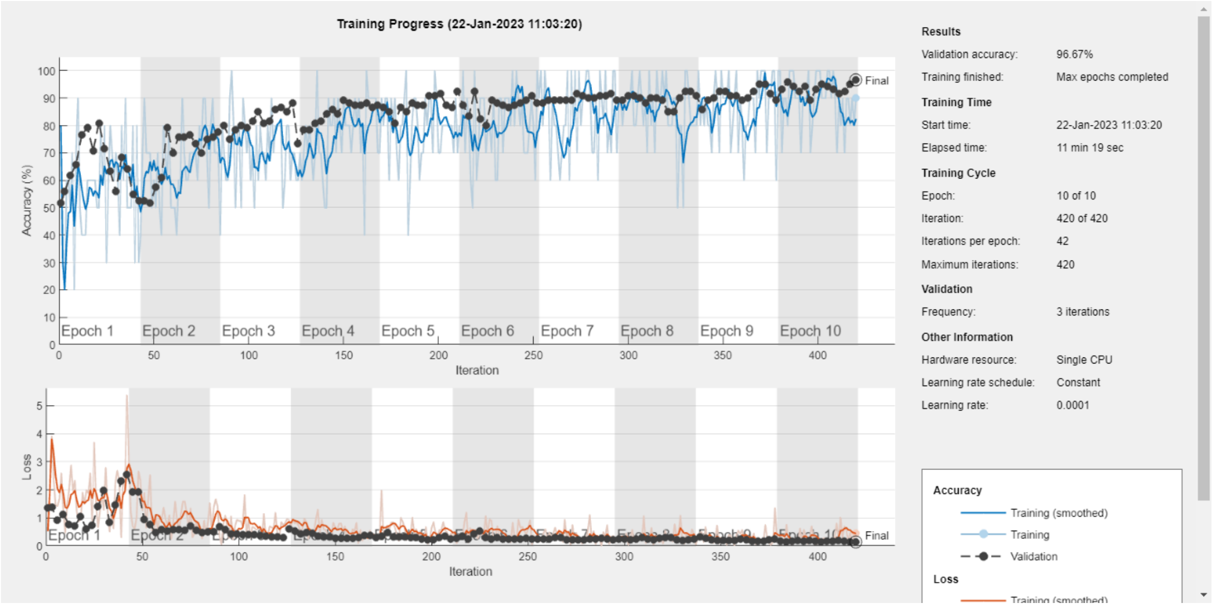
Figure 5.3: Training plot for the same experiment with different classes



	anger	disgust	
anger	223 46.5%	82 17.1%	73.1% 26.9%
disgust	17 3.5%	158 32.9%	90.3% 9.7%
	92.9% 7.1%	65.8% 34.2%	79.4% 20.6%

Figure 5.4: Confusion matrix for the same experiment with different classes

If we try instead to use completely randomly selected images (always considering the classes *anger* and *happiness*) we can see that we have a reduction in performance compared to the previous case.



Results

Validation accuracy: 96.67%

Training finished: Max epochs completed

Training Time

Start time: 22-Jan-2023 11:03:20

Elapsed time: 11 min 19 sec

Training Cycle

Epoch: 10 of 10

Iteration: 420 of 420

Iterations per epoch: 42

Maximum iterations: 420

Validation

Frequency: 3 iterations

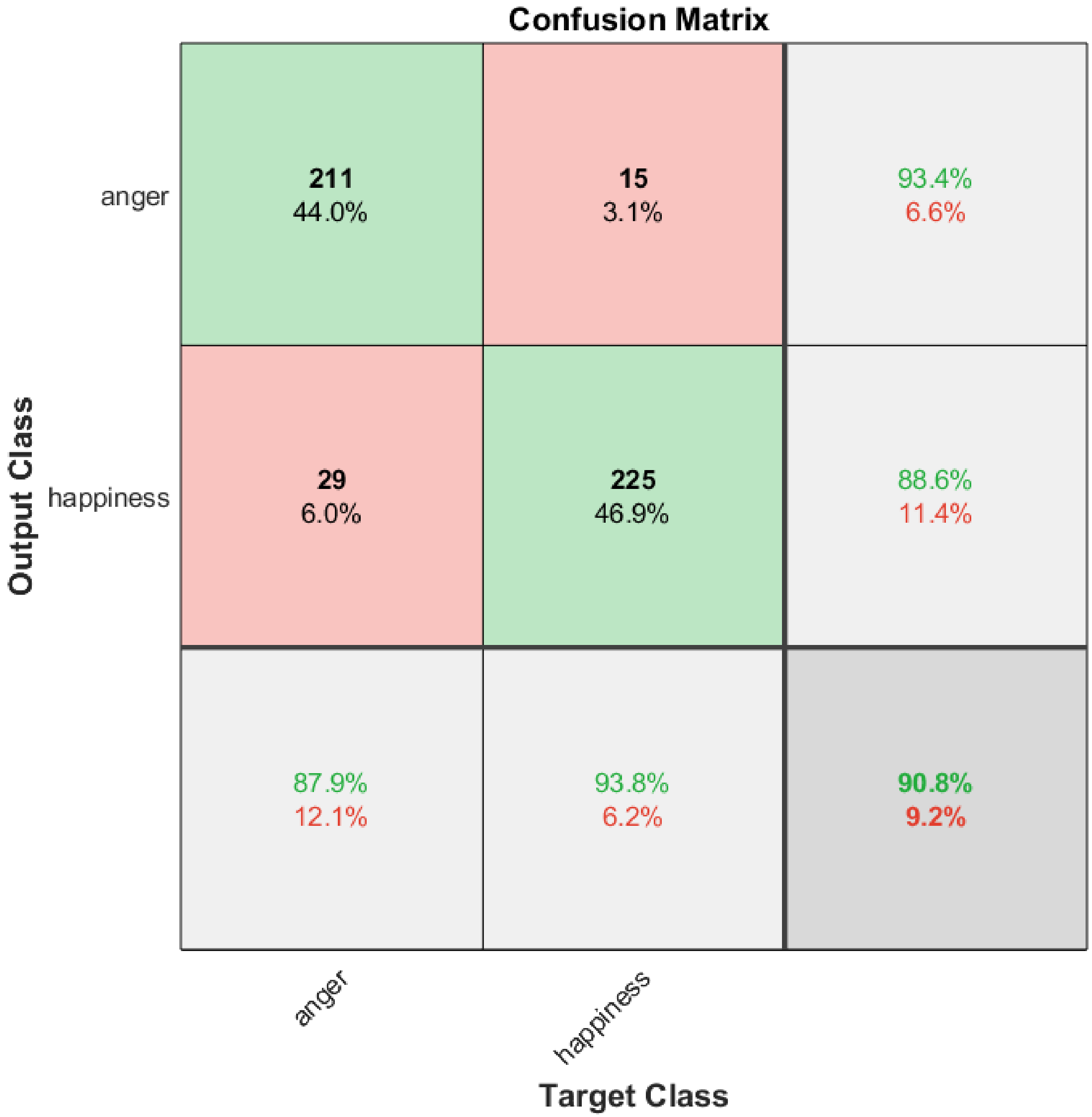
Other Information

Hardware resource: Single CPU

Learning rate schedule: Constant

Learning rate: 0.0001

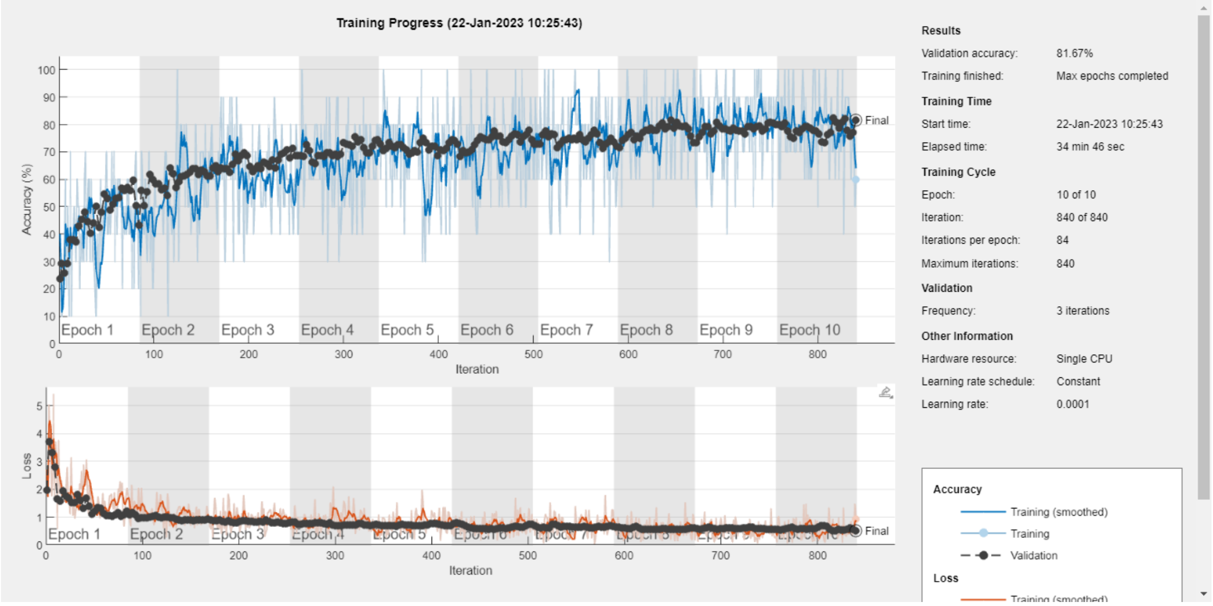
Figure 5.5: Training plot for the same experiment with no selection of images



Output Class \ Target Class	anger	happiness	
anger	211 44.0%	15 3.1%	93.4% 6.6%
happiness	29 6.0%	225 46.9%	88.6% 11.4%
	87.9% 12.1%	93.8% 6.2%	**90.8%** **9.2%**

Figure 5.6: Confusion matrix for the same experiment with no selection of images

5.0.2 Classification with 4 classes



Results

Validation accuracy: 81.67%

Training finished: Max epochs completed

Training Time

Start time: 22-Jan-2023 10:25:43

Elapsed time: 34 min 46 sec

Training Cycle

Epoch: 10 of 10

Iteration: 840 of 840

Iterations per epoch: 84

Maximum iterations: 840

Validation

Frequency: 3 iterations

Other Information

Hardware resource: Single CPU

Learning rate schedule: Constant

Learning rate: 0.0001

Accuracy

Training (smoothed)

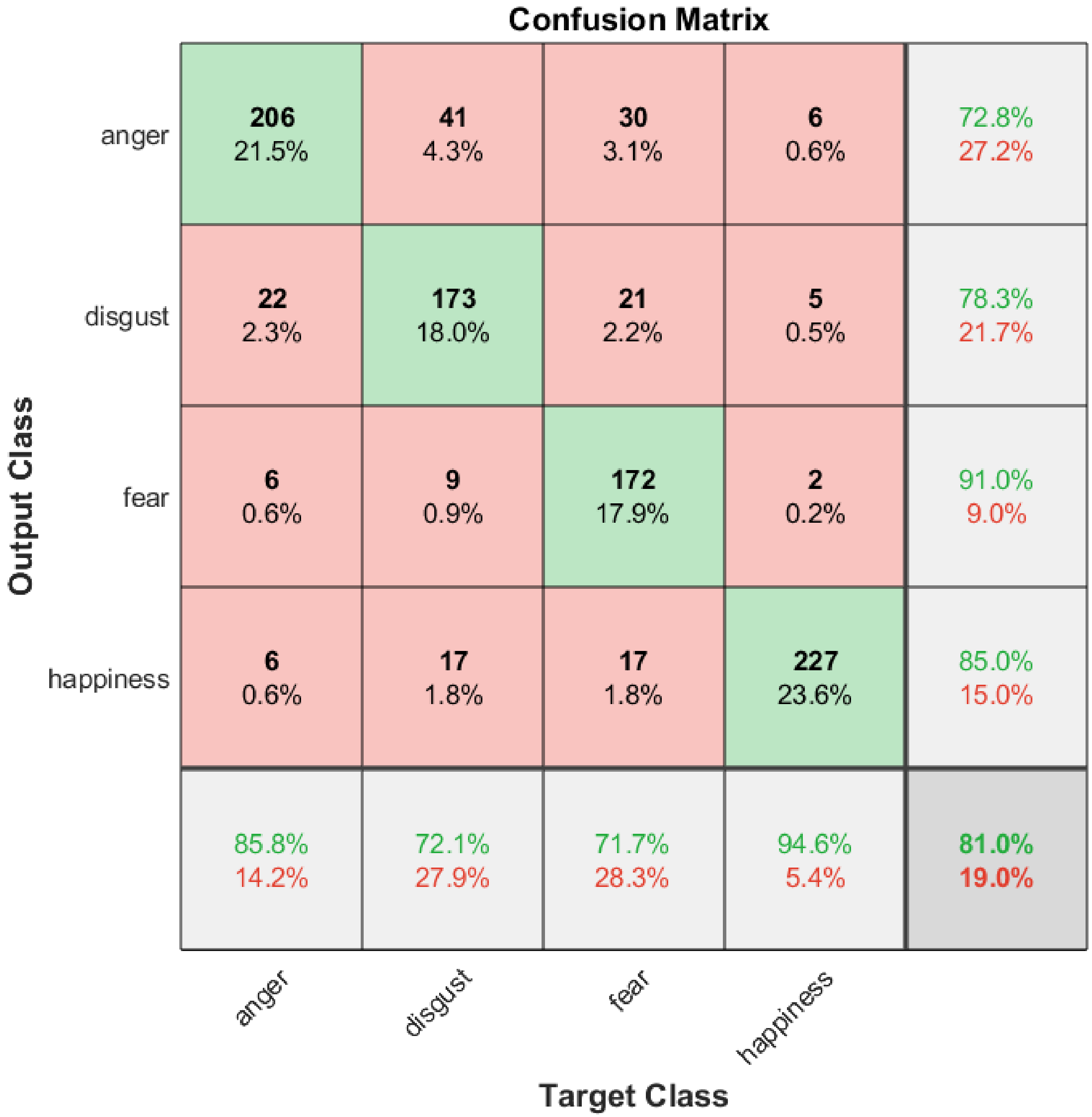
Training

Validation

Loss

Training (smoothed)

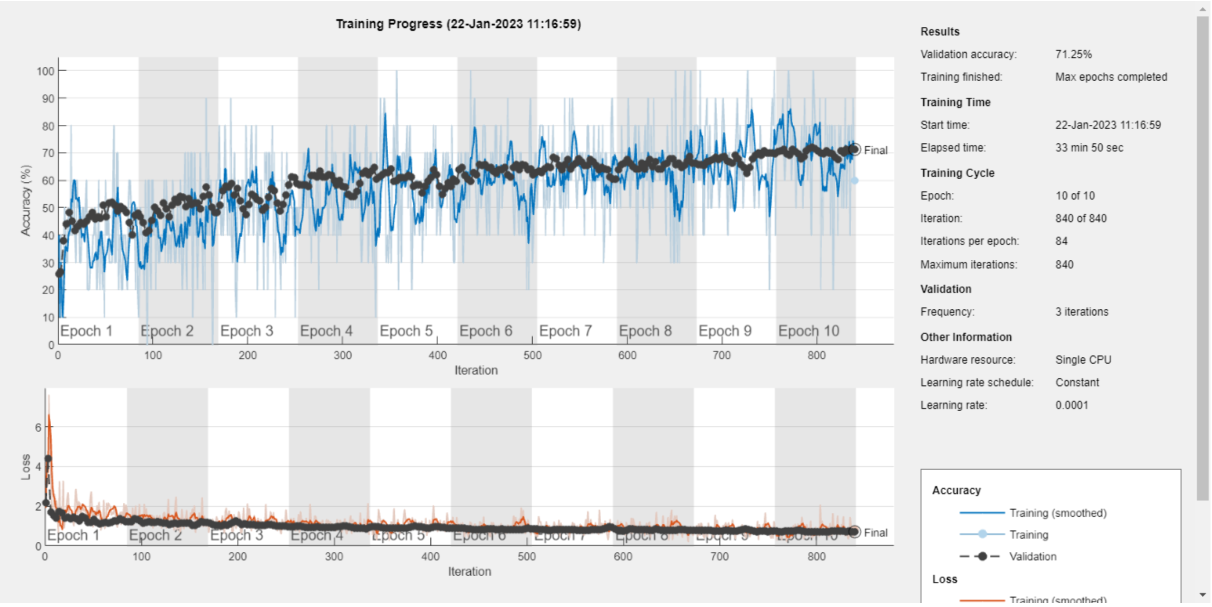
Figure 5.7: Training plot for the same experiment with 4 classes and 300 images



	anger	disgust	fear	happiness	
anger	206 21.5%	41 4.3%	30 3.1%	6 0.6%	72.8% 27.2%
disgust	22 2.3%	173 18.0%	21 2.2%	5 0.5%	78.3% 21.7%
fear	6 0.6%	9 0.9%	172 17.9%	2 0.2%	91.0% 9.0%
happiness	6 0.6%	17 1.8%	17 1.8%	227 23.6%	85.0% 15.0%
	85.8% 14.2%	72.1% 27.9%	71.7% 28.3%	94.6% 5.4%	81.0% 19.0%

Figure 5.8: Confusion matrix for the same experiment with 4 classes and 300 images

If we try to run the experiment with all 4 classes, first we can observe a significant increase in the execution time and also we see that the performance clearly decreases and this is due to the fact that the classification errors increase as the number of classes increase.



Results

Validation accuracy:	71.25%
Training finished:	Max epochs completed

Training Time

Start time:	22-Jan-2023 11:16:59
Elapsed time:	33 min 50 sec

Training Cycle

Epoch:	10 of 10
Iteration:	840 of 840
Iterations per epoch:	84
Maximum iterations:	840

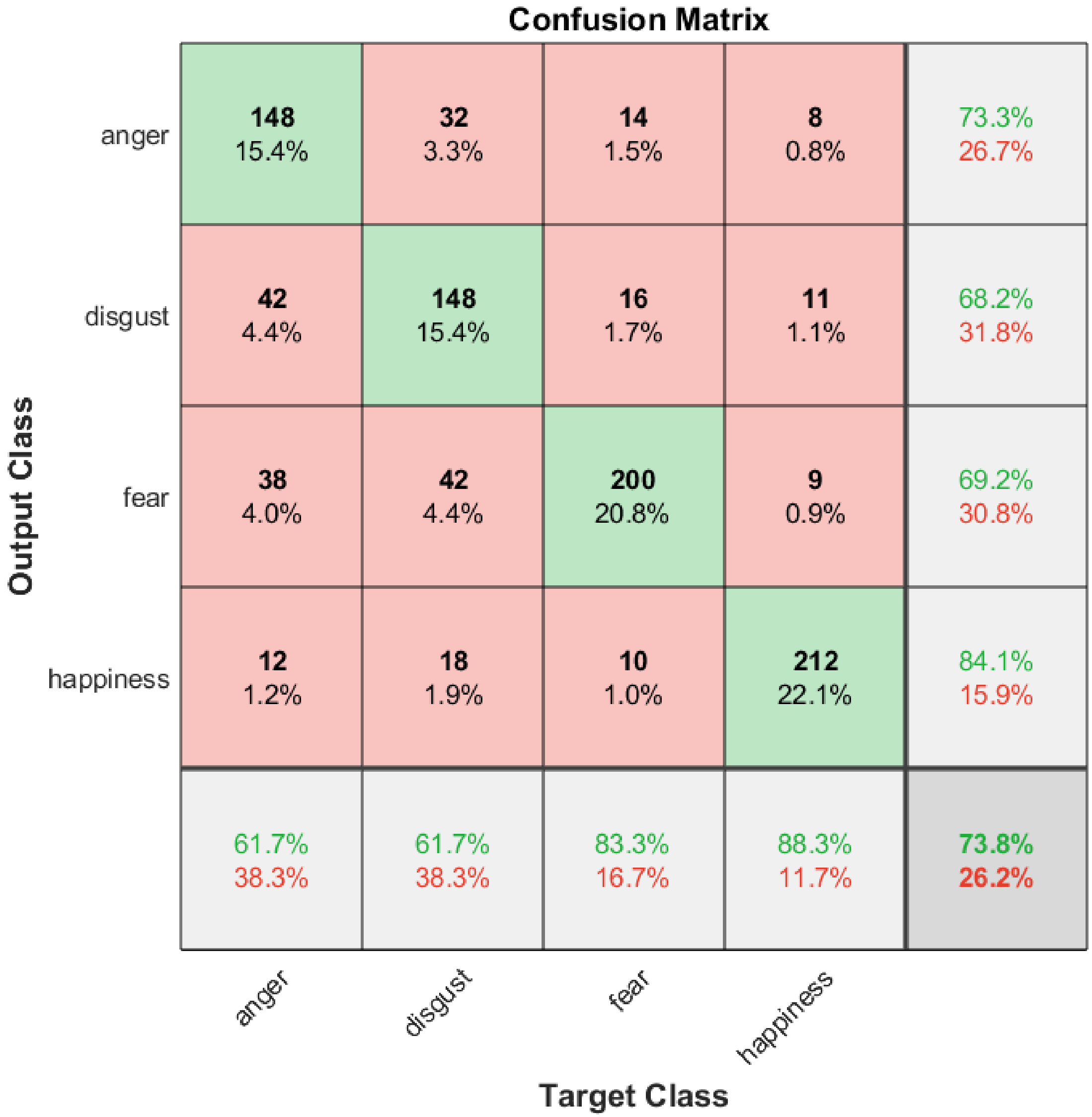
Validation

Frequency:	3 iterations

Other Information

Hardware resource:	Single CPU
Learning rate schedule:	Constant
Learning rate:	0.0001

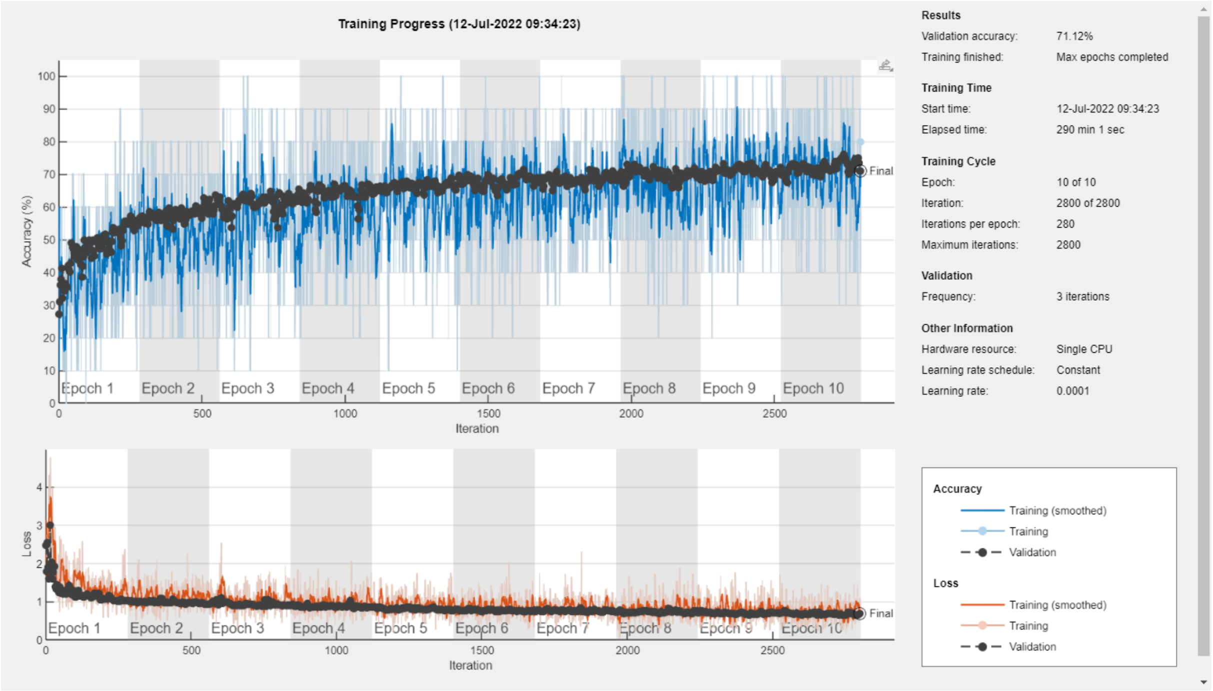
Figure 5.9: Training plot for the same experiment with 4 classes and 300 images not selected


Confusion Matrix
Output Class
Target Class

	anger	disgust	fear	happiness	
anger	**148** 15.4%	**32** 3.3%	**14** 1.5%	**8** 0.8%	73.3% 26.7%
disgust	**42** 4.4%	**148** 15.4%	**16** 1.7%	**11** 1.1%	68.2% 31.8%
fear	**38** 4.0%	**42** 4.4%	**200** 20.8%	**9** 0.9%	69.2% 30.8%
happiness	**12** 1.2%	**18** 1.9%	**10** 1.0%	**212** 22.1%	84.1% 15.9%
	61.7% 38.3%	61.7% 38.3%	83.3% 16.7%	88.3% 11.7%	**73.8%** **26.2%**

Figure 5.10: Confusion matrix for the same experiment with 4 classes and 300 images not selected

If we try to repeat the experiment with unselected images we notice a reduction in performance in the previous case.



Results
Validation accuracy: 71.12%
Training finished: Max epochs completed

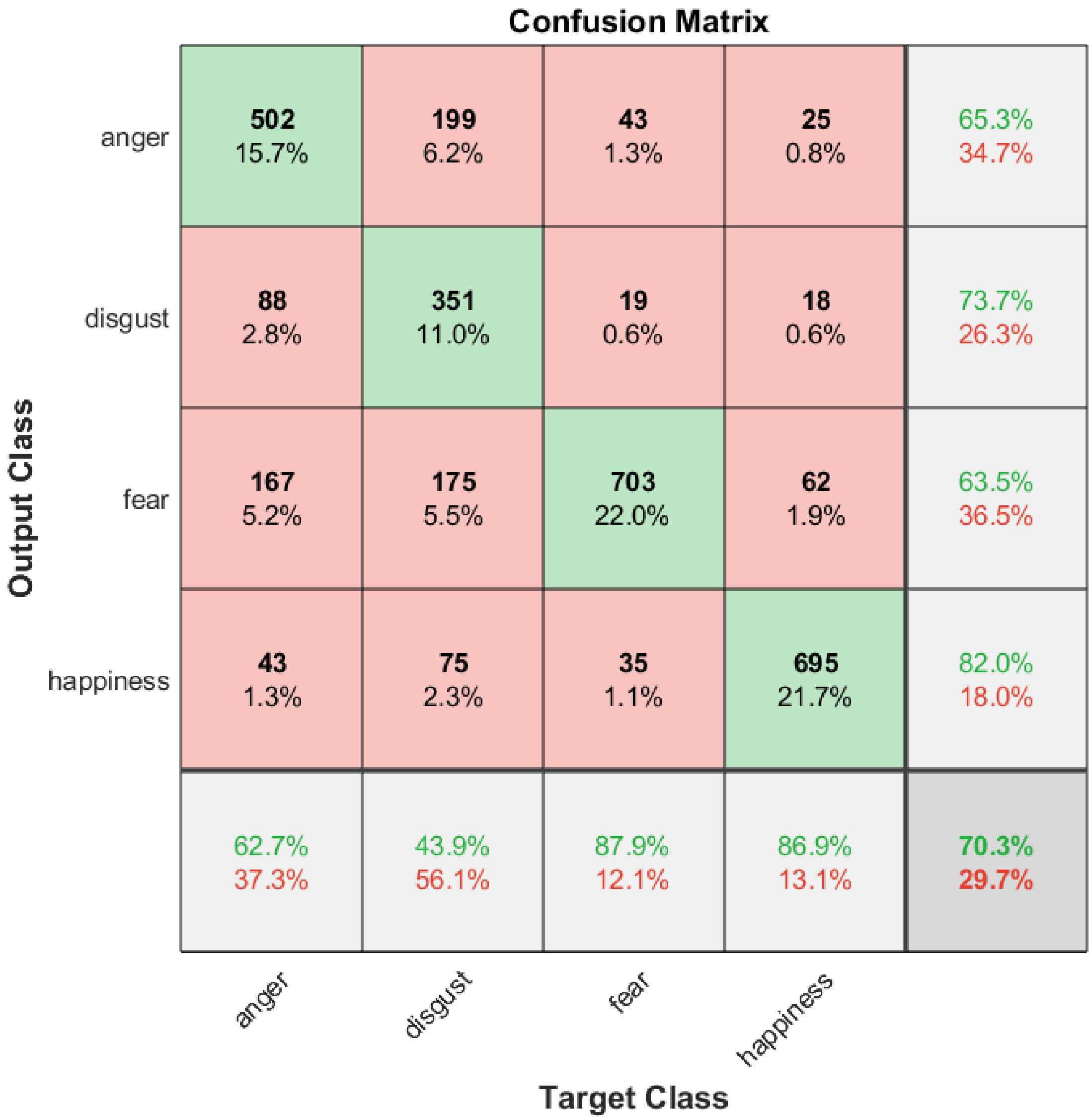
Training Time
Start time: 12-Jul-2022 09:34:23
Elapsed time: 290 min 1 sec

Training Cycle
Epoch: 10 of 10
Iteration: 2800 of 2800
Iterations per epoch: 280
Maximum iterations: 2800

Validation
Frequency: 3 iterations

Other Information
Hardware resource: Single CPU
Learning rate schedule: Constant
Learning rate: 0.0001

Figure 5.11: Training plot for the same experiment with 4 classes and 1000 images not selected



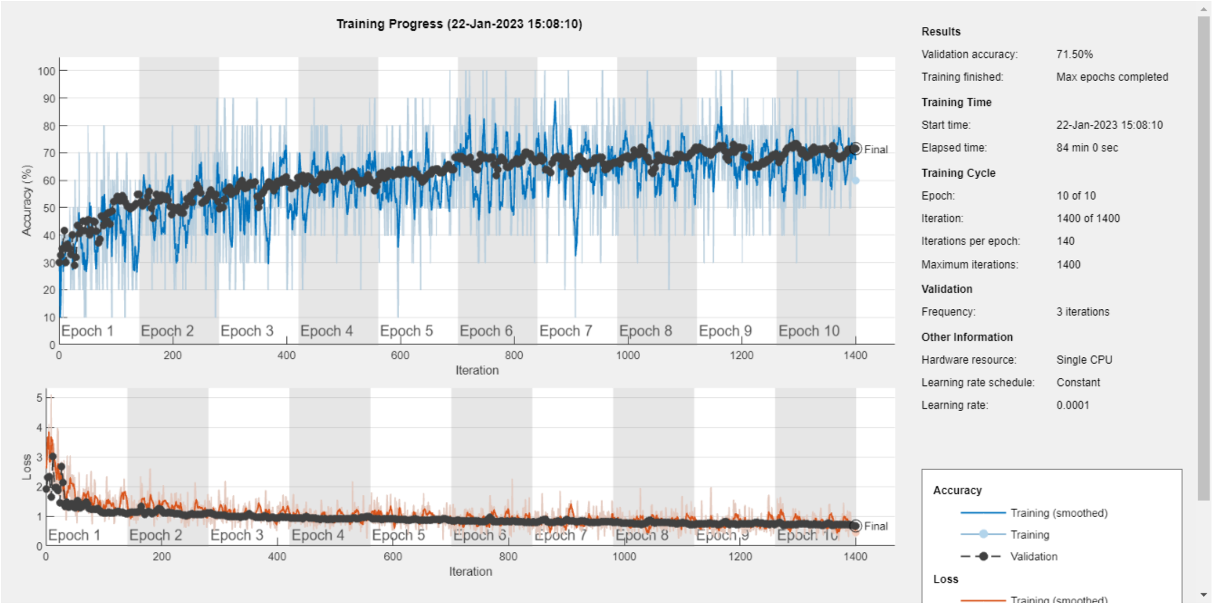
Confusion Matrix

Output Class	anger	disgust	fear	happiness	
anger	**502** 15.7%	**199** 6.2%	**43** 1.3%	**25** 0.8%	65.3% 34.7%
disgust	**88** 2.8%	**351** 11.0%	**19** 0.6%	**18** 0.6%	73.7% 26.3%
fear	**167** 5.2%	**175** 5.5%	**703** 22.0%	**62** 1.9%	63.5% 36.5%
happiness	**43** 1.3%	**75** 2.3%	**35** 1.1%	**695** 21.7%	82.0% 18.0%
	62.7% 37.3%	43.9% 56.1%	87.9% 12.1%	86.9% 13.1%	**70.3%** **29.7%**

Target Class

Figure 5.12: Confusion matrix for the same experiment with 4 classes and 1000 images not selected

If we repeat the experiment with a larger number of images (1000) we will obviously have a significant increase in run time (from about 34 minutes to 290), but also greater accuracy because the size of the training set is larger and allows us to generalize more successfully.



Training Progress (22-Jan-2023 15:08:10)

Results

Validation accuracy: 71.50%

Training finished: Max epochs completed

Training Time

Start time: 22-Jan-2023 15:08:10

Elapsed time: 84 min 0 sec

Training Cycle

Epoch: 10 of 10

Iteration: 1400 of 1400

Iterations per epoch: 140

Maximum iterations: 1400

Validation

Frequency: 3 iterations

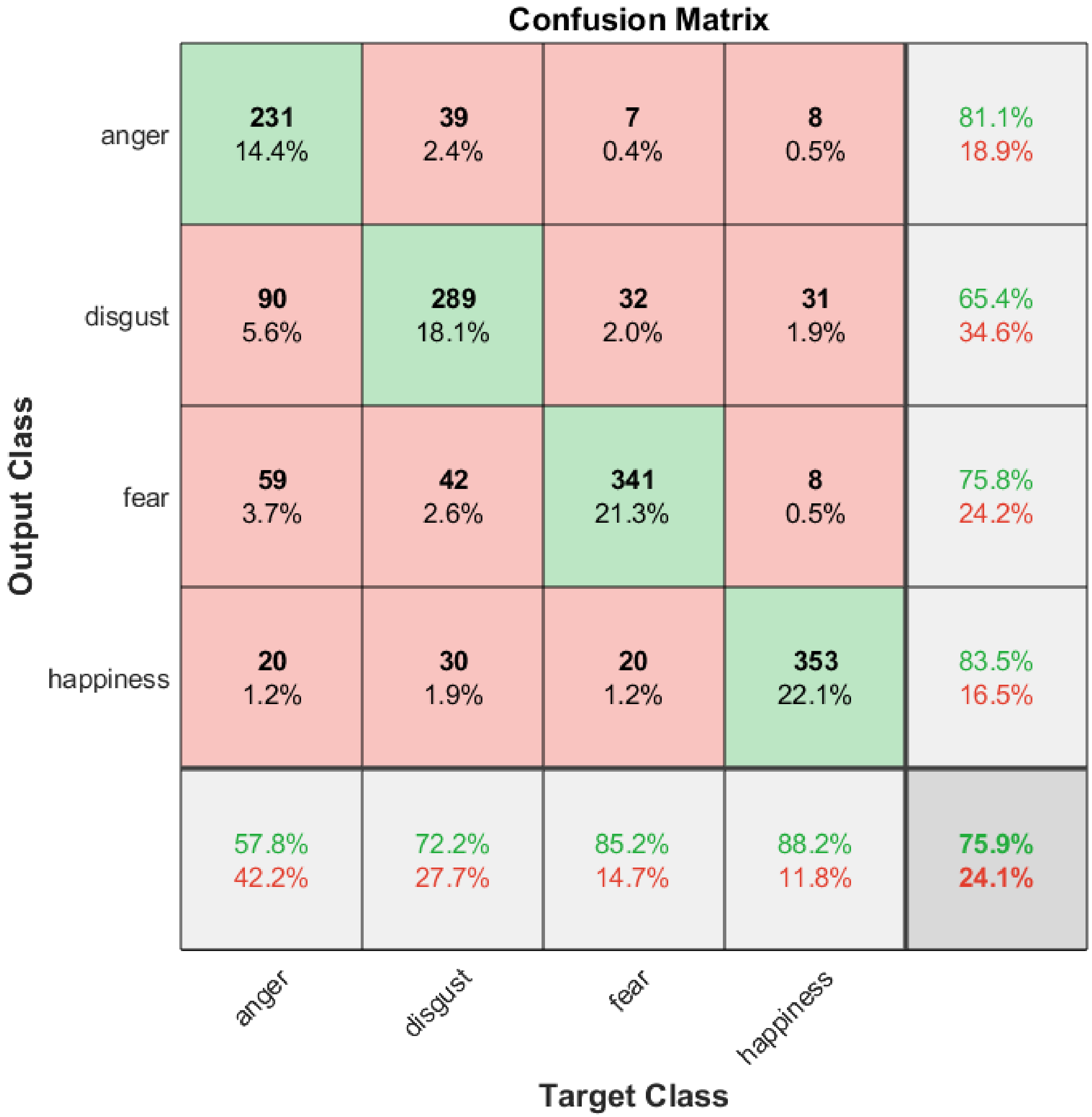
Other Information

Hardware resource: Single CPU

Learning rate schedule: Constant

Learning rate: 0.0001

Figure 5.13: Training plot for the same experiment with 4 classes and 500 images not selected



	anger	disgust	fear	happiness	
anger	**231** 14.4%	**39** 2.4%	**7** 0.4%	**8** 0.5%	81.1% 18.9%
disgust	**90** 5.6%	**289** 18.1%	**32** 2.0%	**31** 1.9%	65.4% 34.6%
fear	**59** 3.7%	**42** 2.6%	**341** 21.3%	**8** 0.5%	75.8% 24.2%
happiness	**20** 1.2%	**30** 1.9%	**20** 1.2%	**353** 22.1%	83.5% 16.5%
	57.8% 42.2%	72.2% 27.7%	85.2% 14.7%	88.2% 11.8%	**75.9%** **24.1%**

Figure 5.14: Confusion matrix for the same experiment with 4 classes and 500 images not selected

Finally, we note how, by repeating the experiment with 500 images instead of 1000, we have no deterioration in performance and, above all, we have significantly less execution time. This can be explained by 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.