UNIVERSITÀ DI PISA

Computer Engineering

Intelligent systems project report

Intelligent systems

Leonardo Poggiani

Academic Year 2021/2022

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 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.

2 Design and develop of an intelligent system

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 the dataset.
- Feature selection
- Normalizing the data.

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

2.1 Balancing of the dataset

.

The dataset consists of 54 signals, and for each of these there is a value for valence and a value for arousal. We can divide the dataset into seven classes considering that there are seven possible values that valence and arousal can take and check the distribution of the classes:

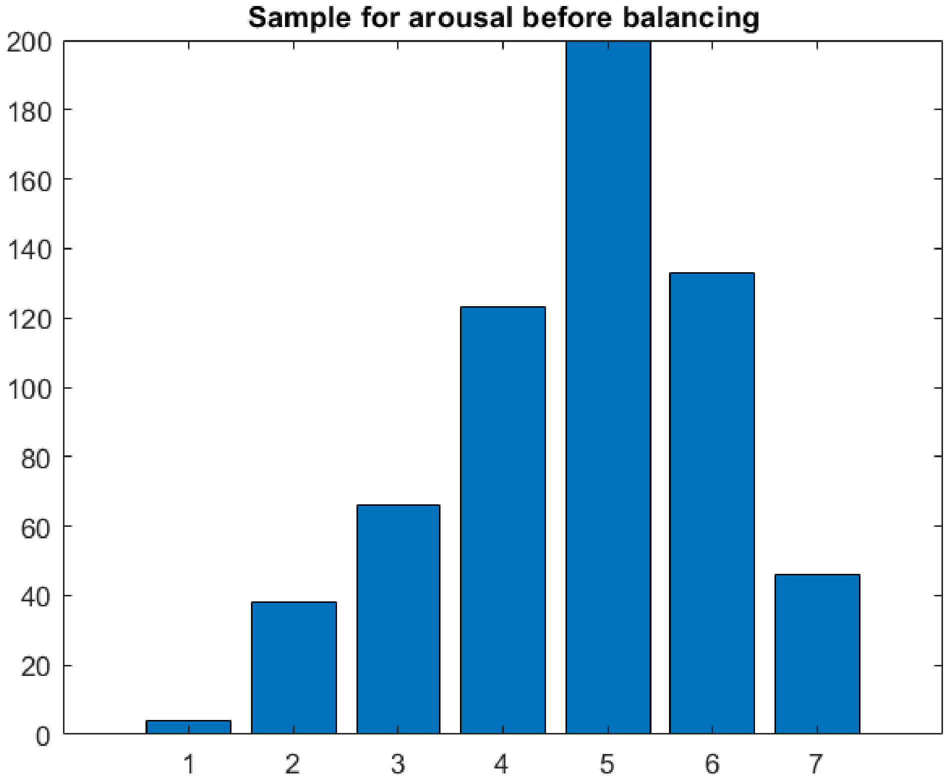


Figure 1: Samples distributions for arousal before balancing

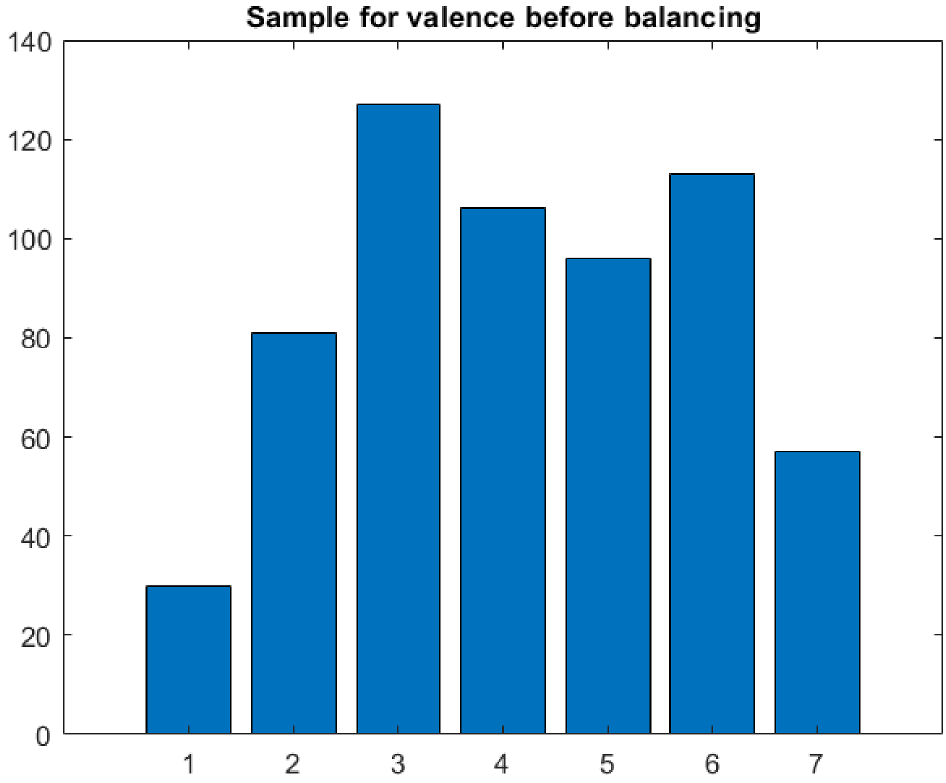


Figure 2: Samples distributions for valence before balancing

As we can see from the images the samples are unbalanced so it is necessary to use an algorithm involving undersampling, oversampling and data augmentation to balance the dataset. Data augmentation is applied to those samples that belong to the common *least* class for arousal (valence) but not to the common *most* class for valence (arousal) after which the samples that belong to the common *most* class for arousal (valence) and do not belong to the common *least* class for valence (arousal) are removed. These steps are repeated for a number of times that has been derived experimentally. As we can see from the images, after applying the algorithm the samples turn out to be balanced.

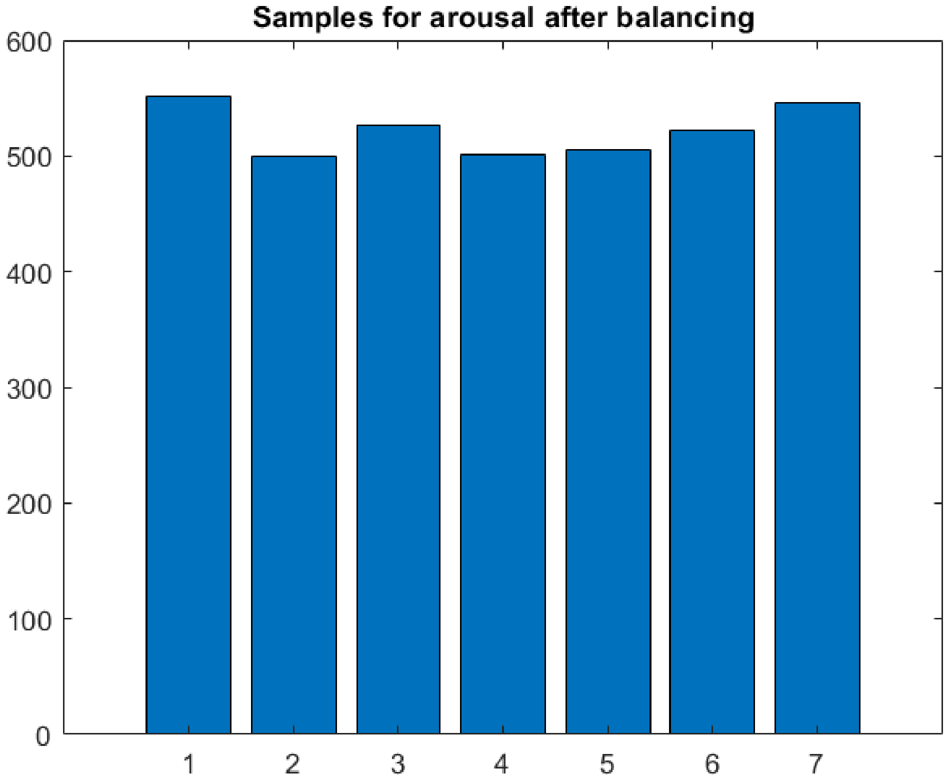


Figure 3: Samples distributions for arousal after balancing

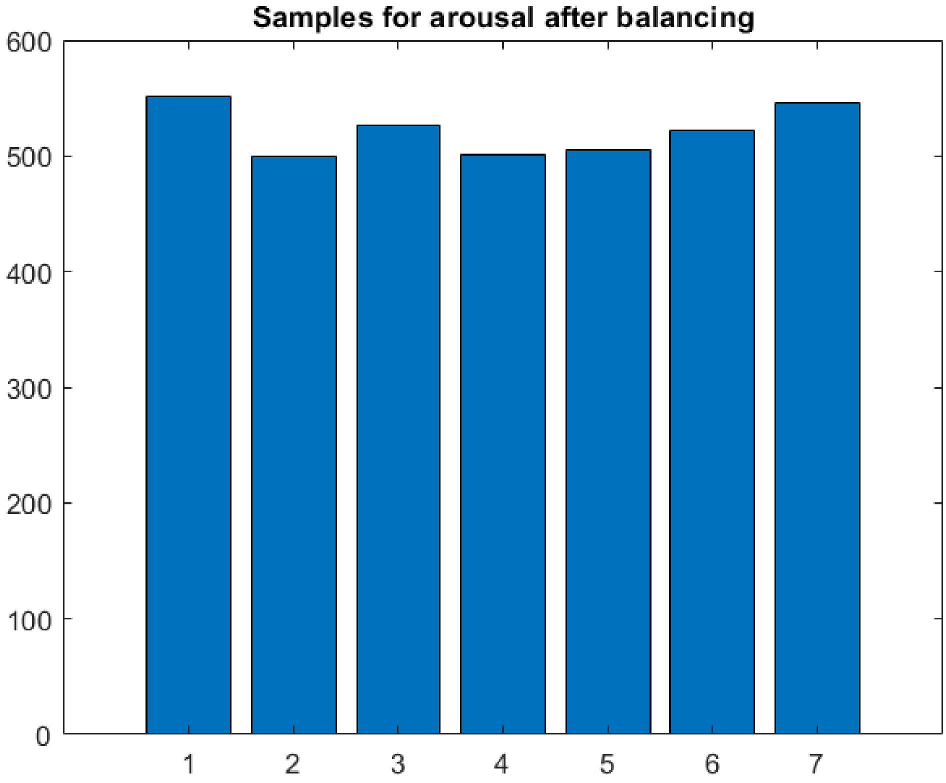
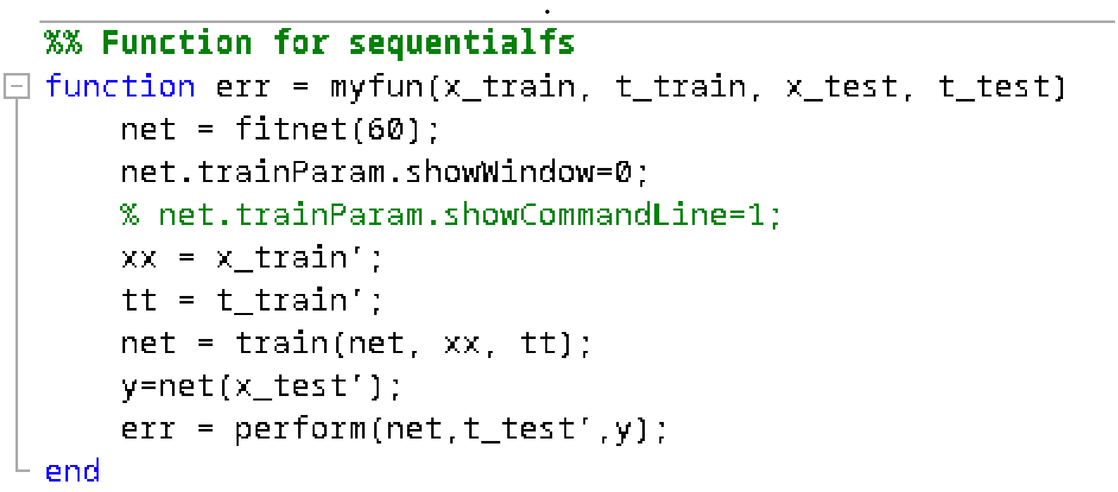


Figure 4: Samples distributions for arousal after balancing

2.2 Feature selection

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



```
%% 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 5: Sequentialfs fucntion used

The function *sequentialfs* is repeated for a number of times that was obtained through a trade-off between the execution time (hours) and the statistical significance of the extracted features, so it was chosen to repeat the selection for 30 times. At the end of the execution of the file *data_cleaning.m*, the folder *data/* is populated with 5 files:

- *training_arousal.mat:* which contains the training inputs and corresponding target outputs for arousal training
- *testing_arousal.mat:* which contains the inputs of testing and the corresponding target outputs for arousal testing
- *training_valence.mat:* which contains the training inputs and the corresponding target outputs for valence training
- *training_valence.mat:* which contains the inputs of testing and the corresponding target outputs for valence training
- *best_features_selected.mat:* which is a structure that contains the training and testing data for arousal of the three best features selected to be used for task 3.3.

2.3 Normalization of data

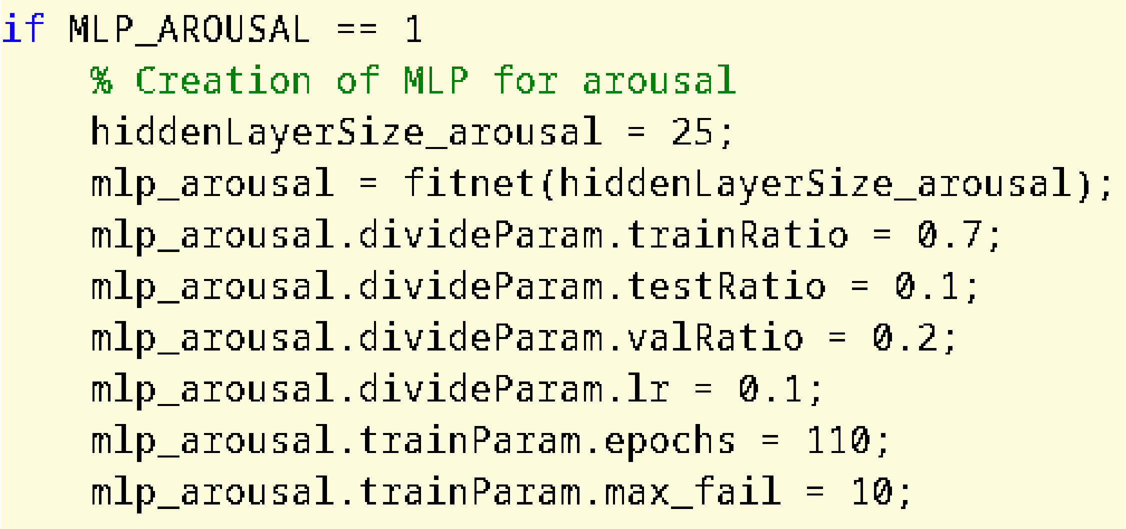
Normalization of data was done using the MATLAB function *normalize* which takes as an argument the data to be normalized. No changes were made to the other parameters.

3 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 Arousal MLP network

The configuration used:



```
if MLP_AROUSAL == 1
    % Creation of MLP for arousal
    hiddenLayerSize_arousal = 25;
    mlp_arousal = fitnet(hiddenLayerSize_arousal);
    mlp_arousal.divideParam.trainRatio = 0.7;
    mlp_arousal.divideParam.testRatio = 0.1;
    mlp_arousal.divideParam.valRatio = 0.2;
    mlp_arousal.divideParam.lr = 0.1;
    mlp_arousal.trainParam.epochs = 110;
    mlp_arousal.trainParam.max_fail = 10;
```

Figure 6: Configuration for MLP network for arousal

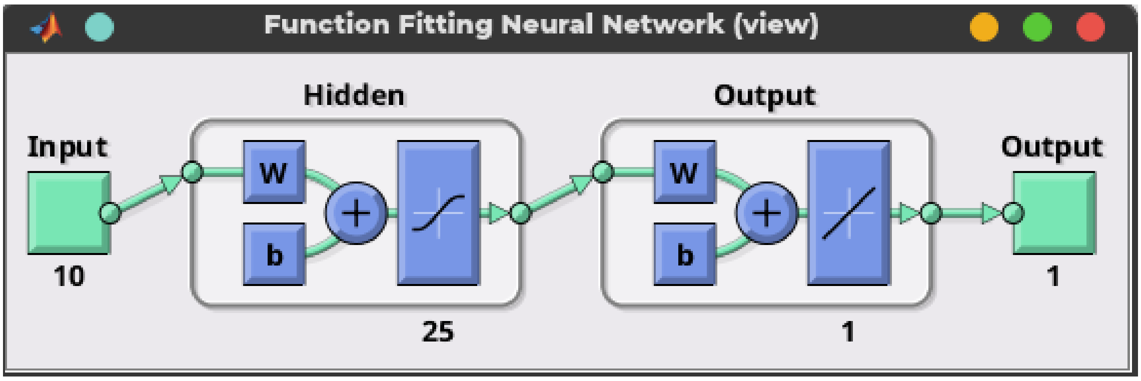


Figure 7: Arousal MLP network architecture

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

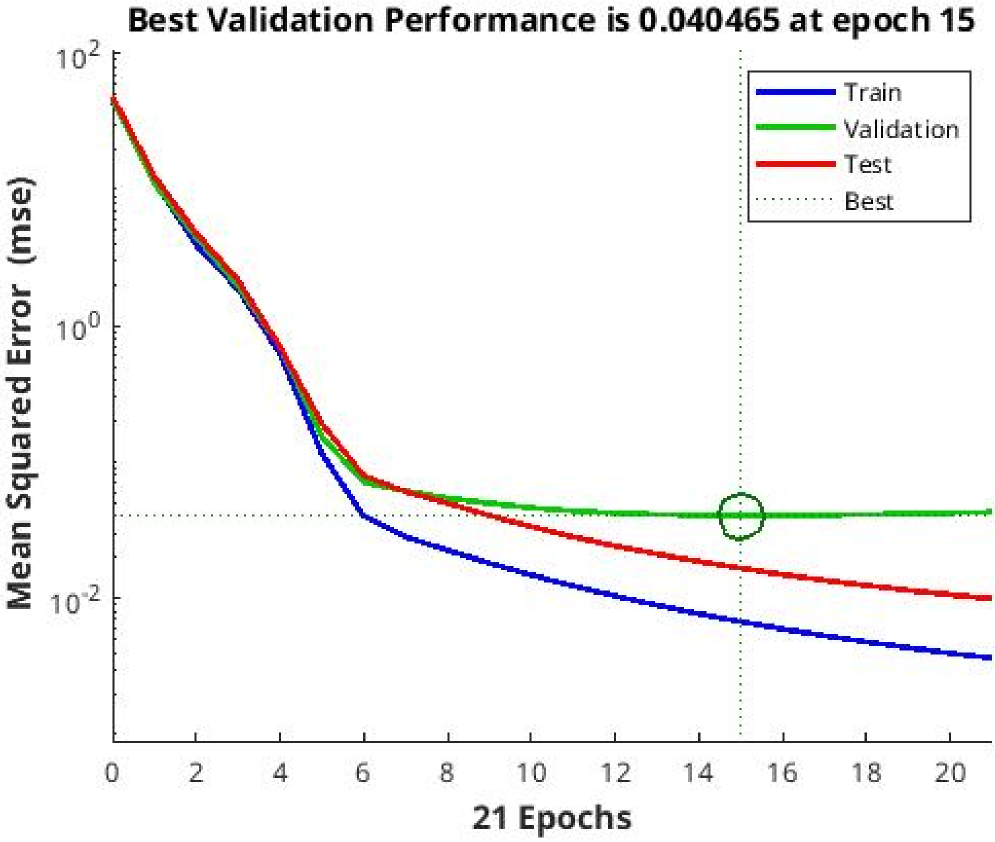


Figure 8: Best validation performance for arousal

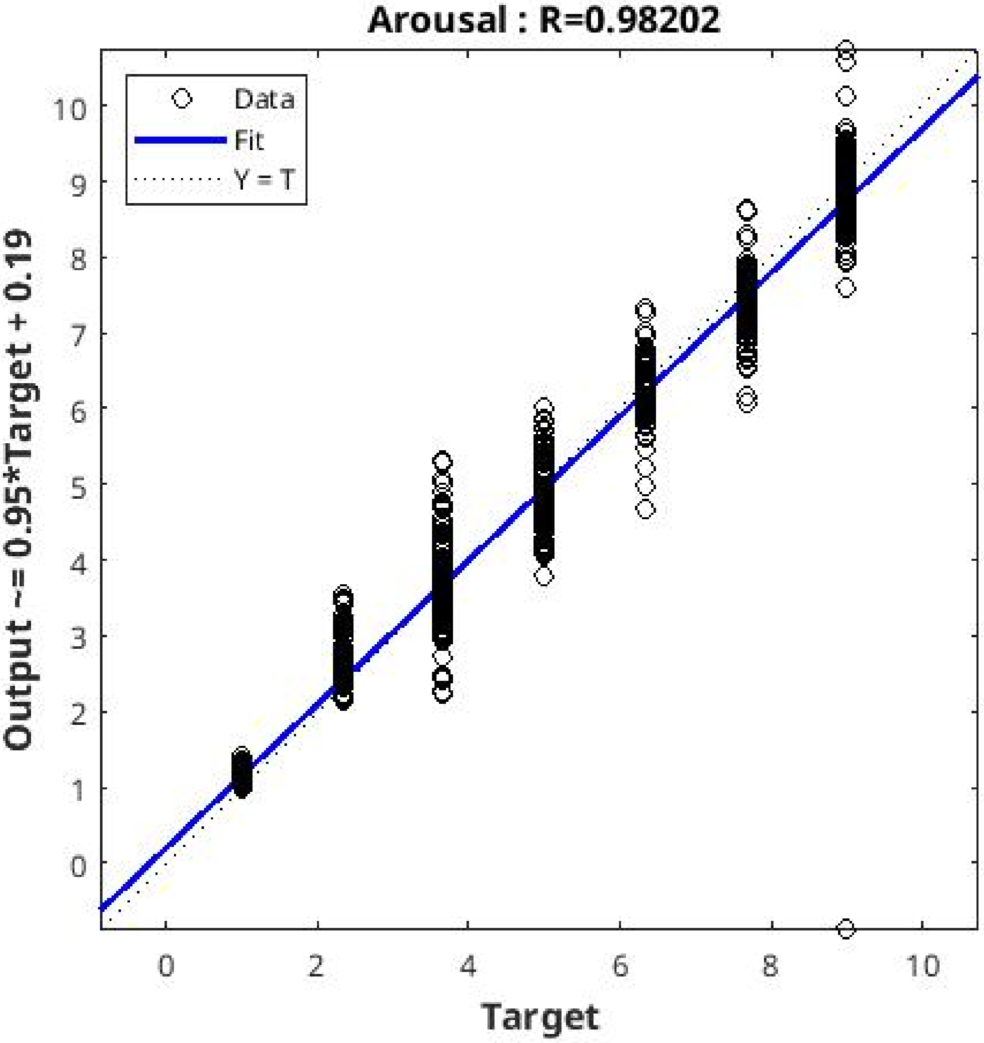
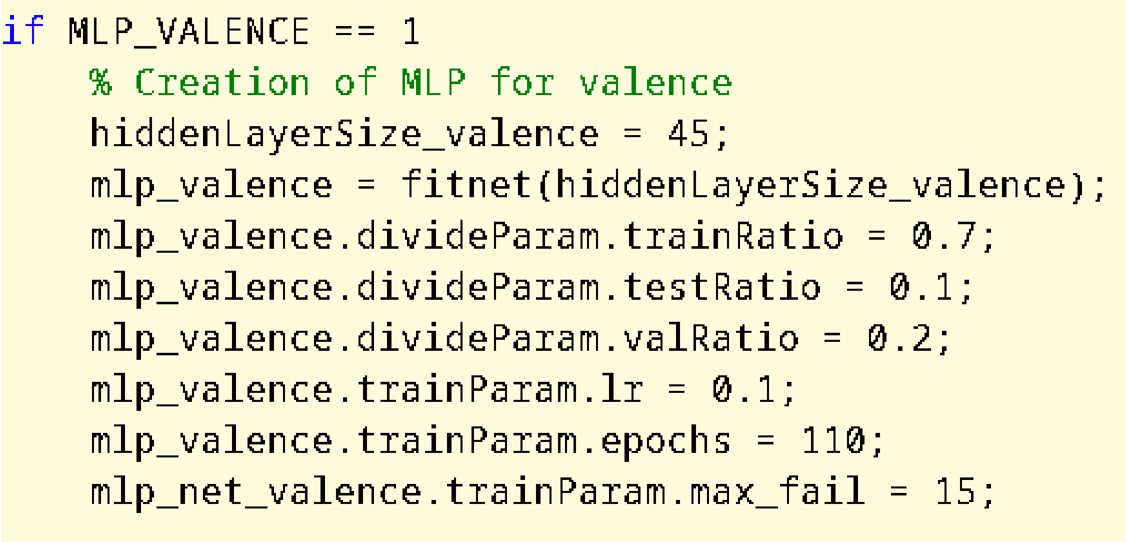


Figure 9: Regression for arousal MLP network

3.2 Valence MLP network

The configuration used:



```
if MLP_VALENCE == 1
    % Creation of MLP for valence
    hiddenLayerSize_valence = 45;
    mlp_valence = fitnet(hiddenLayerSize_valence);
    mlp_valence.divideParam.trainRatio = 0.7;
    mlp_valence.divideParam.testRatio = 0.1;
    mlp_valence.divideParam.valRatio = 0.2;
    mlp_valence.trainParam.lr = 0.1;
    mlp_valence.trainParam.epochs = 110;
    mlp_net_valence.trainParam.max_fail = 15;
```

Figure 10: Configuration for MLP network for valence

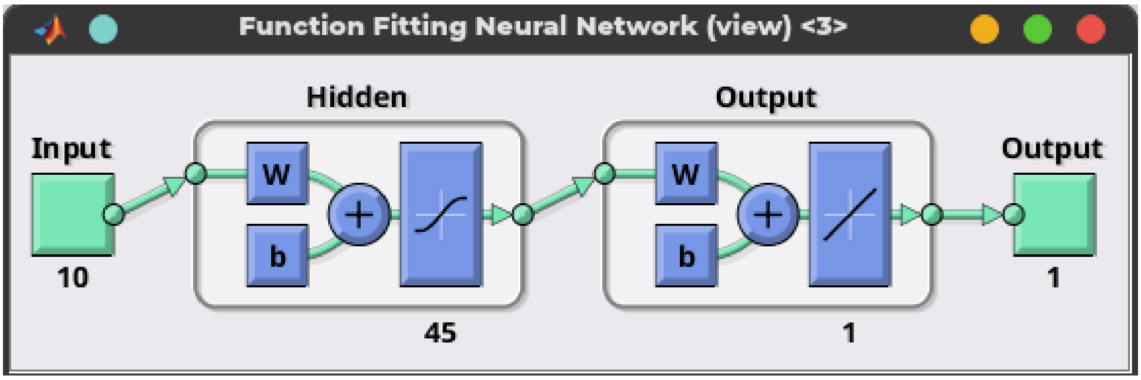


Figure 11: Valence MLP network architecture

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

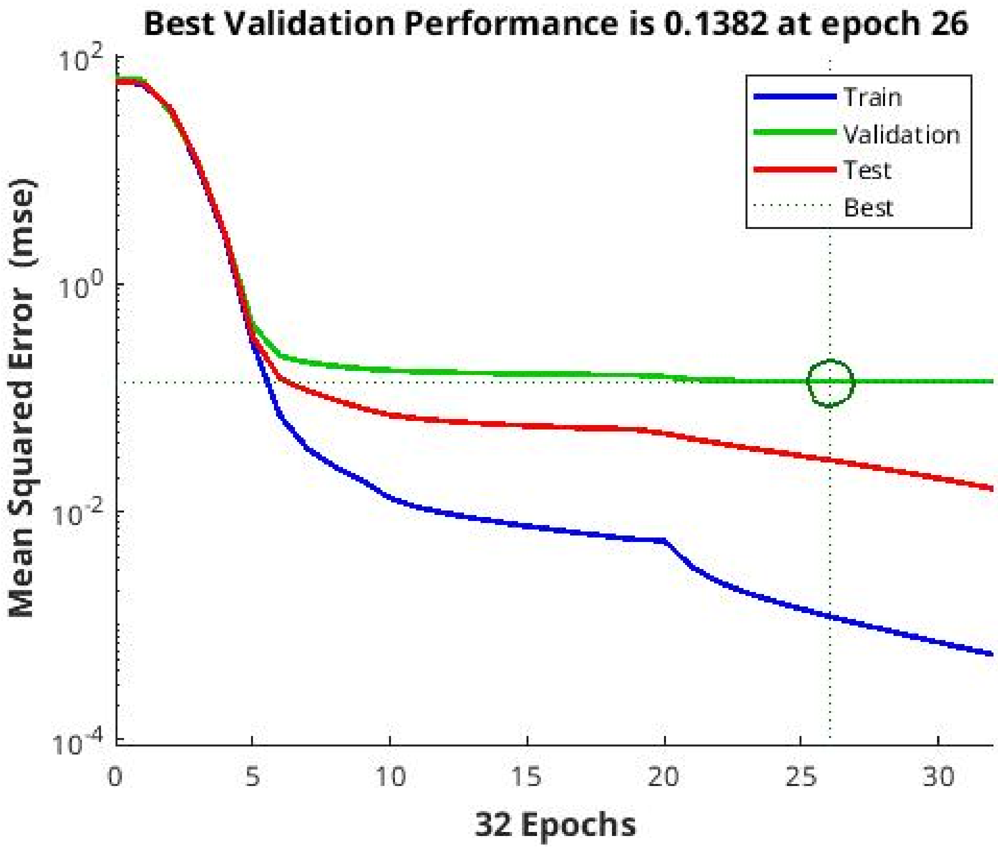


Figure 12: Best validation performance for valence

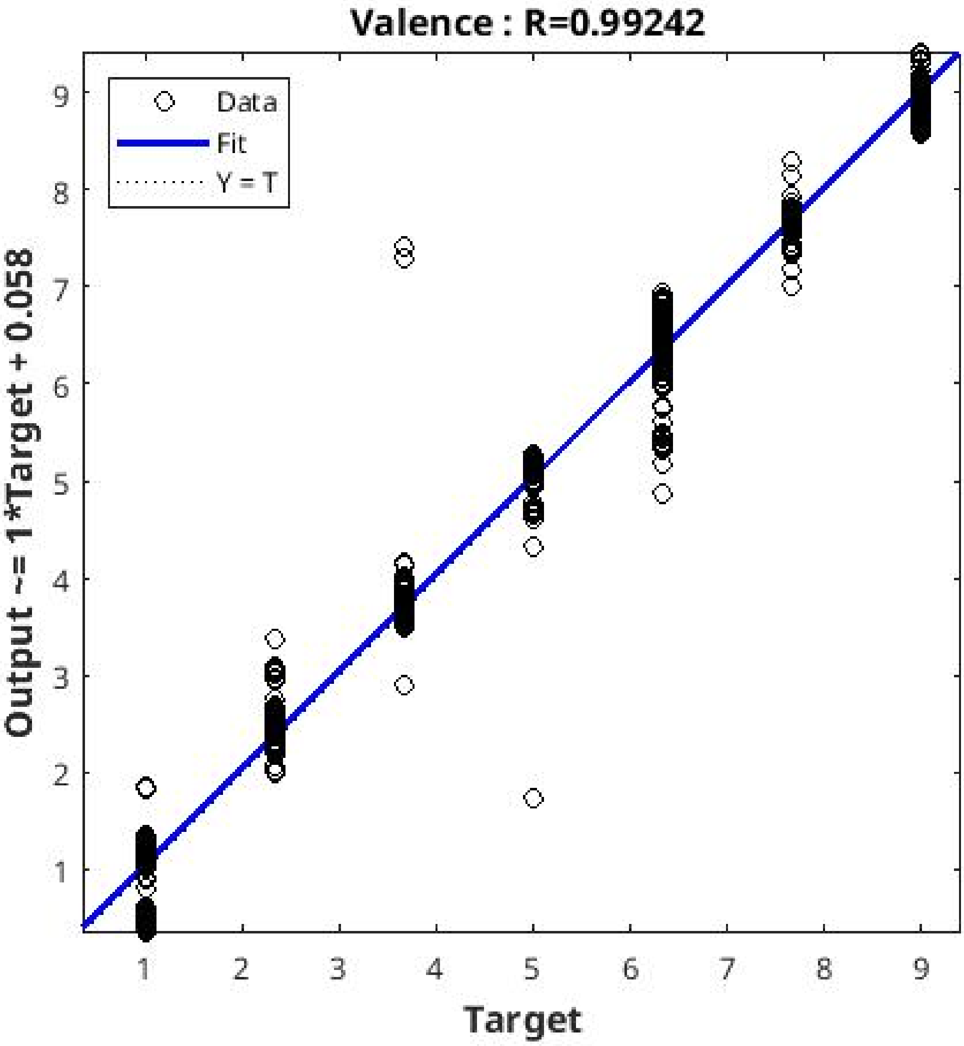


Figure 13: Regression for arousal MLP network

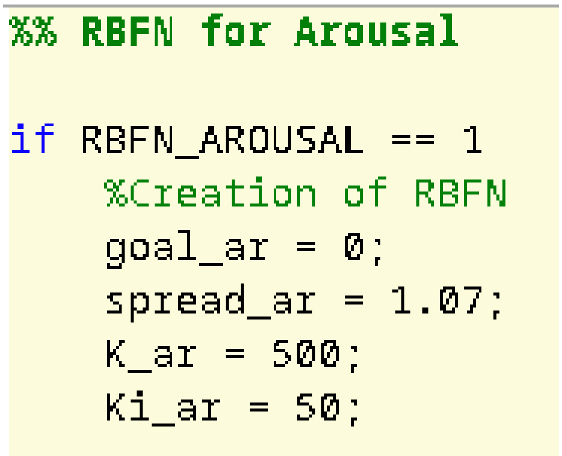
Clearly the results obtained for what concerns the error and regression value are in this case very good, but they are highly variable depending on the starting point on the error surface since the weights are taken randomly at the beginning of the training.

4 Radial Basis Function networks

This section describes the design and development of the two RBF networks on which again experiments were done to find the best configuration by going to vary the values of the *spread* to get the best results from the regression point of view.

4.1 Arousal RBF network

The configuration used:



```
%% RBFN for Arousal

if RBFN_AROUSAL == 1
    %Creation of RBFN
    goal_ar = 0;
    spread_ar = 1.07;
    K_ar = 500;
    Ki_ar = 50;
```

Figure 14: Arousal RBF network configuration

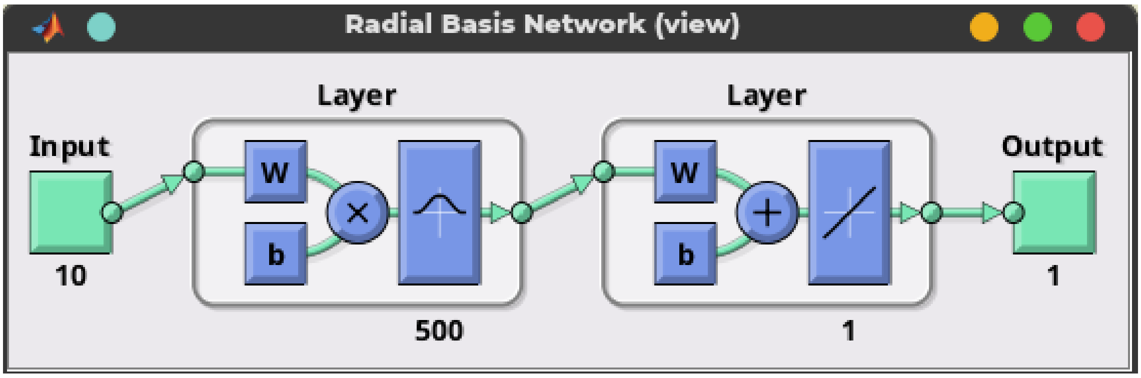


Figure 15: Arousal RBF network architecture

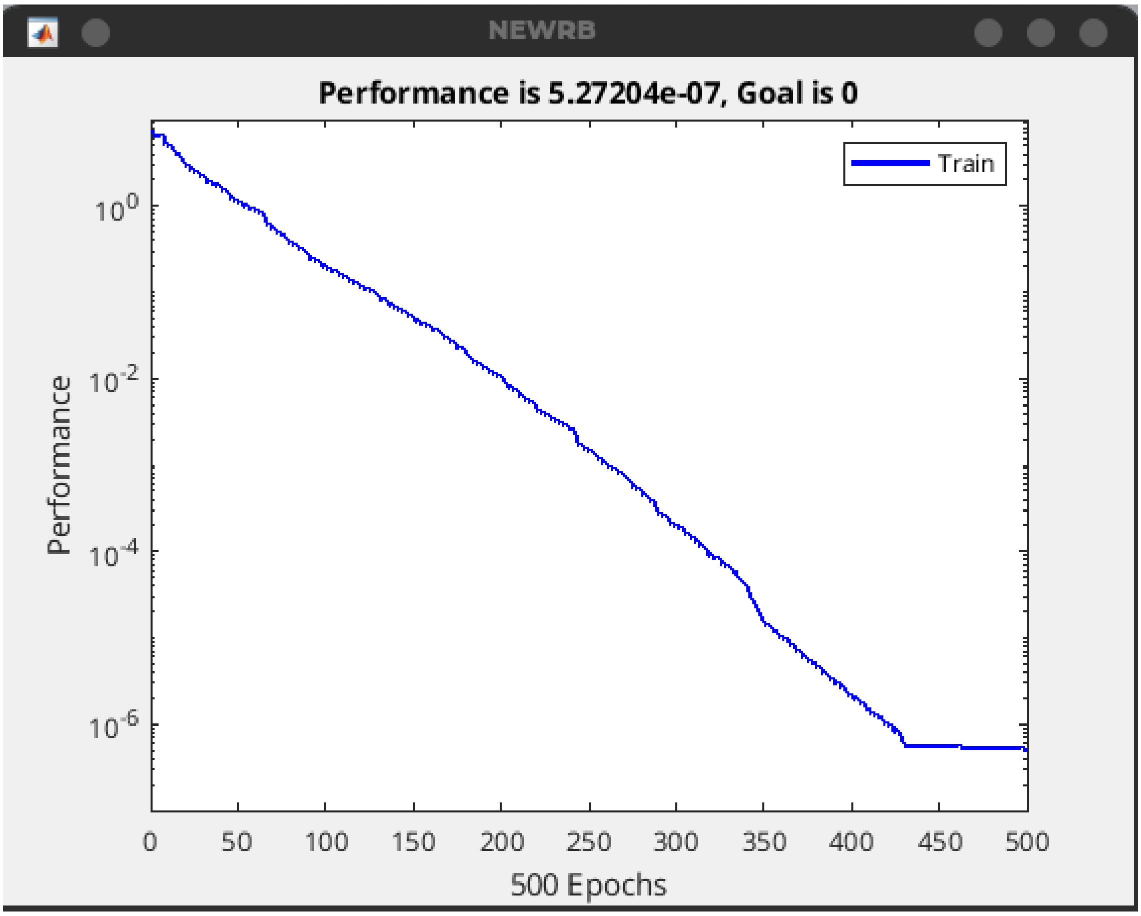


Figure 16: Performance for arousal RBF network

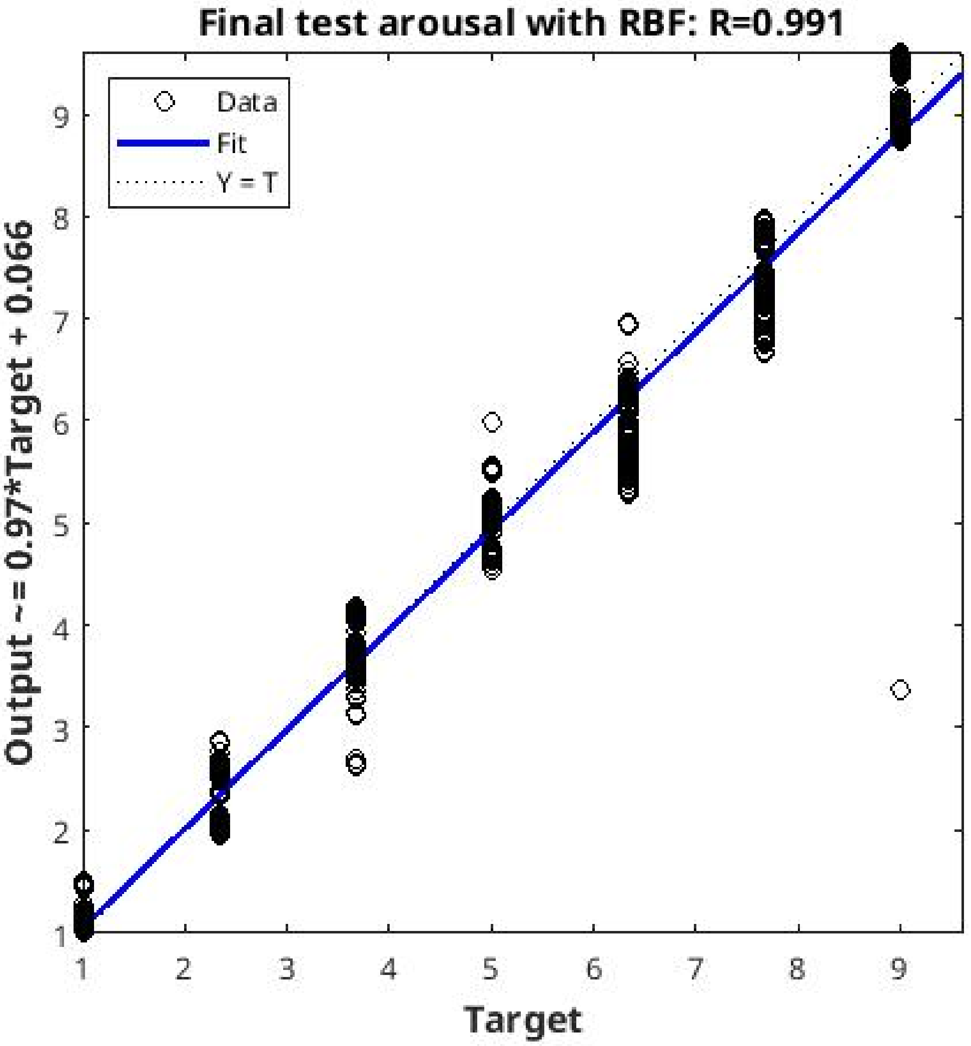
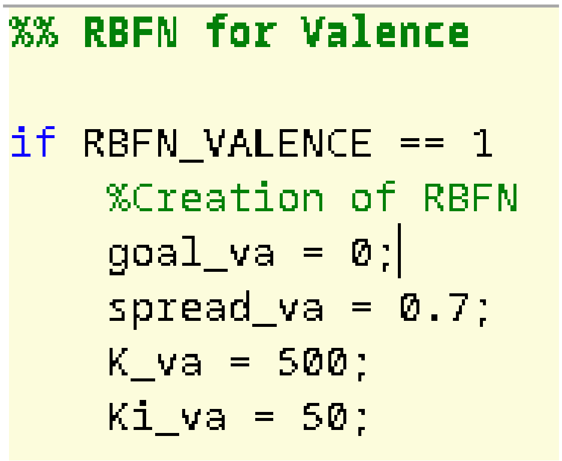


Figure 17: Regression for arousal RBF network

4.2 Valence RBF network

The configuration used:



```
%% RBFN for Valence

if RBFN_VALENCE == 1
    %Creation of RBFN
    goal_va = 0;
    spread_va = 0.7;
    K_va = 500;
    Ki_va = 50;
```

Figure 18: Valence RBF network configuration

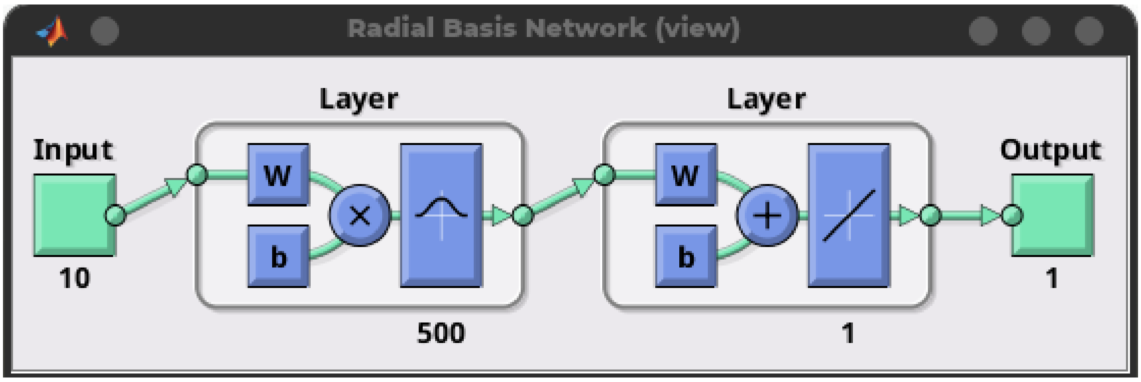


Figure 19: Valence RBF network architecture

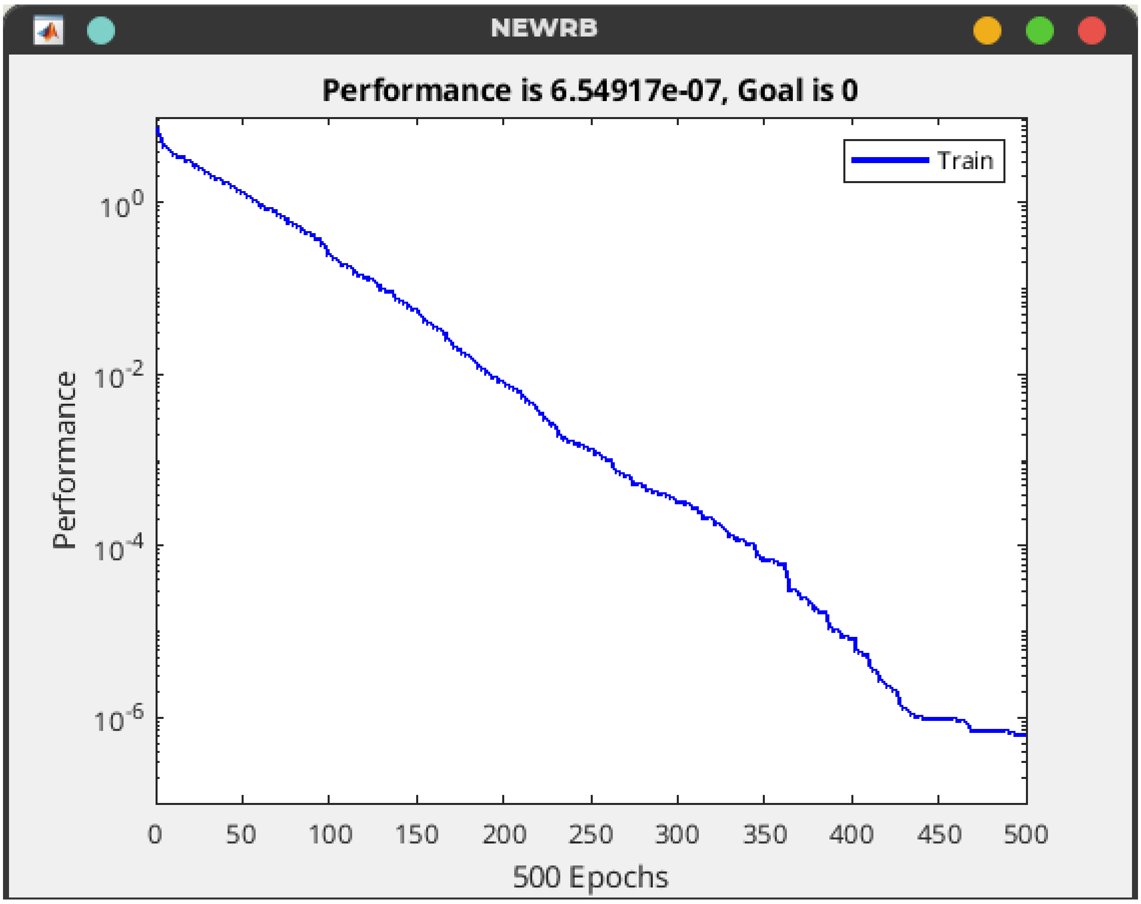


Figure 20: Performance for valence RBF network

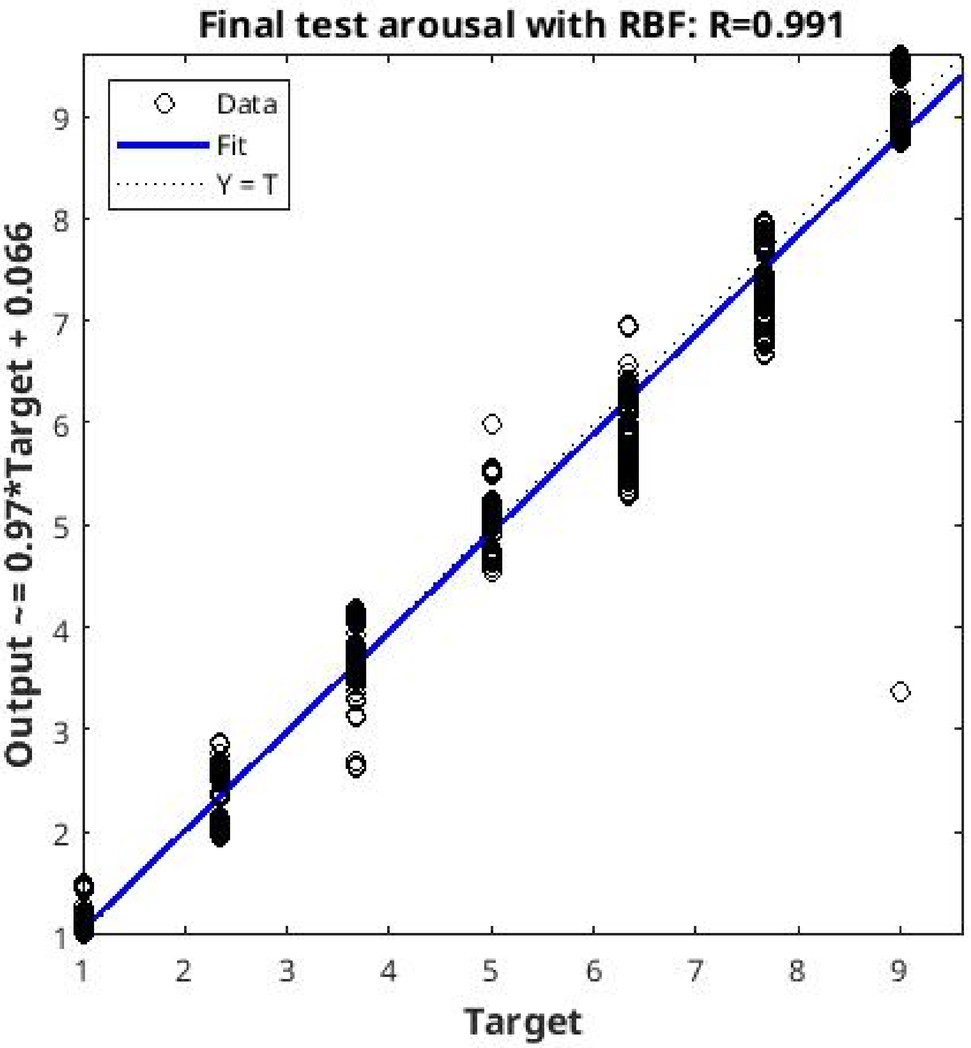


Figure 21: Regression for arousal RBF network

5 Mandami fuzzy inference system

Per prima dobbiamo conoscere l'universo del discorso per le prime tre feature (che sono state ottenute come risultato dall'esecuzione di *sequentialfs*):

In this section, we expose how a *fuzzy inference system* was designed and developed to correct the deficiencies related to the arousal dimension. First we need to know the universe of discourse for the first three features (which were obtained as a result of running *sequentialfs*):

- Feature 27: [-2.349128, 1.940738]
- Feature 11: [-2.284724, 1.979107]

- Feature 13: [-1.907918, 2.367136]

The empirical distributions of the features used were initially studied that were useful in adequately modeling linguistic feature variables.

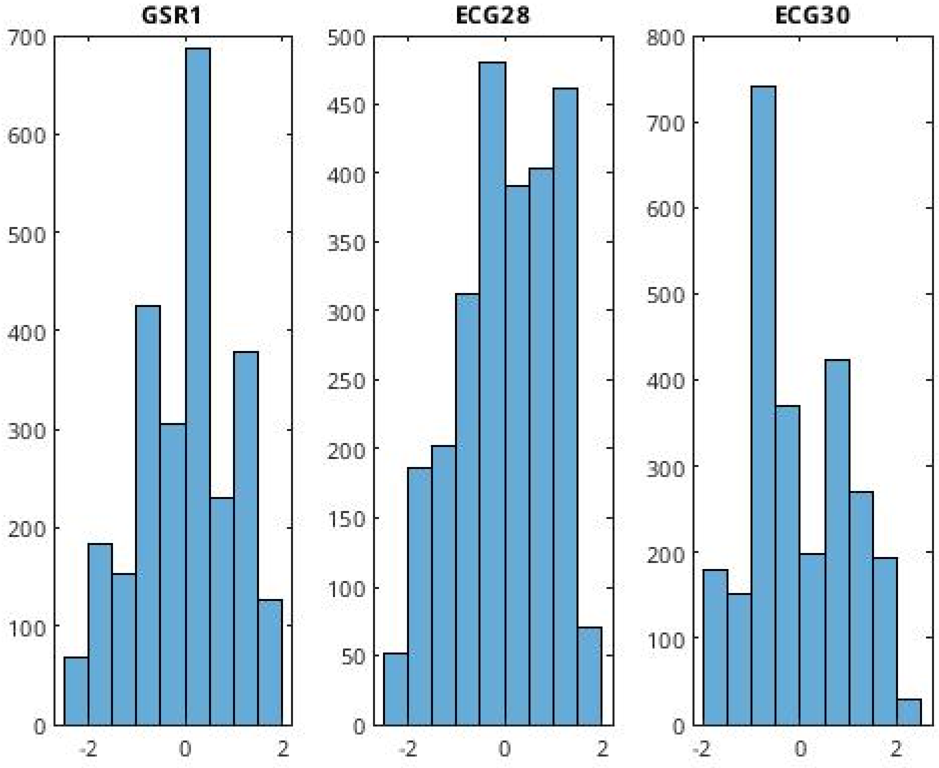


Figure 22: Configuration for MLP network for arousal

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

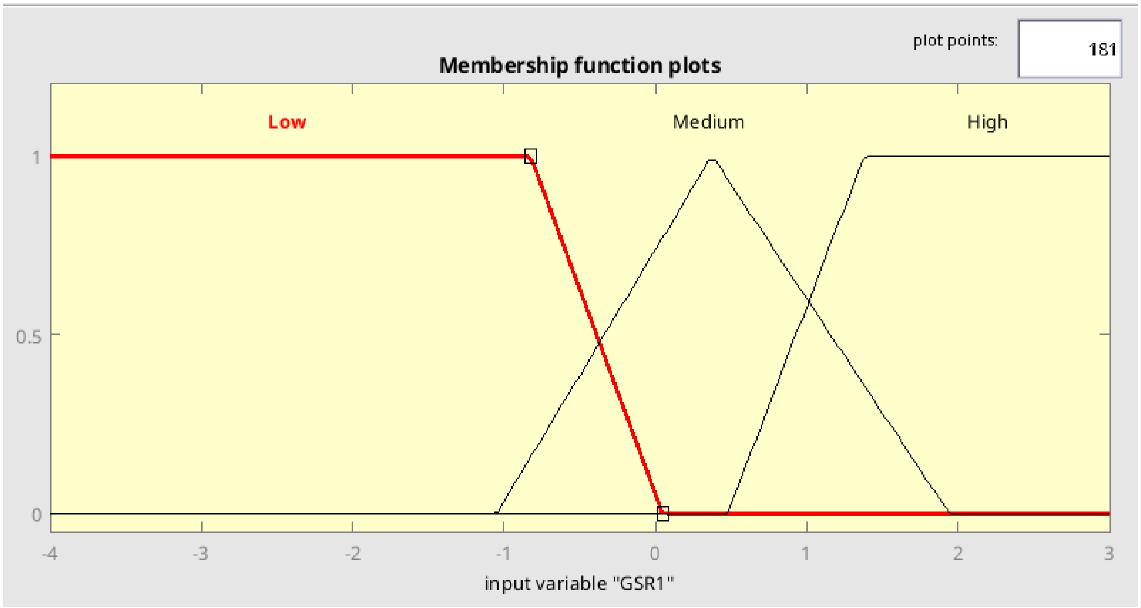


Figure 23: Modeling for linguistic variable of feature 27: GSR1

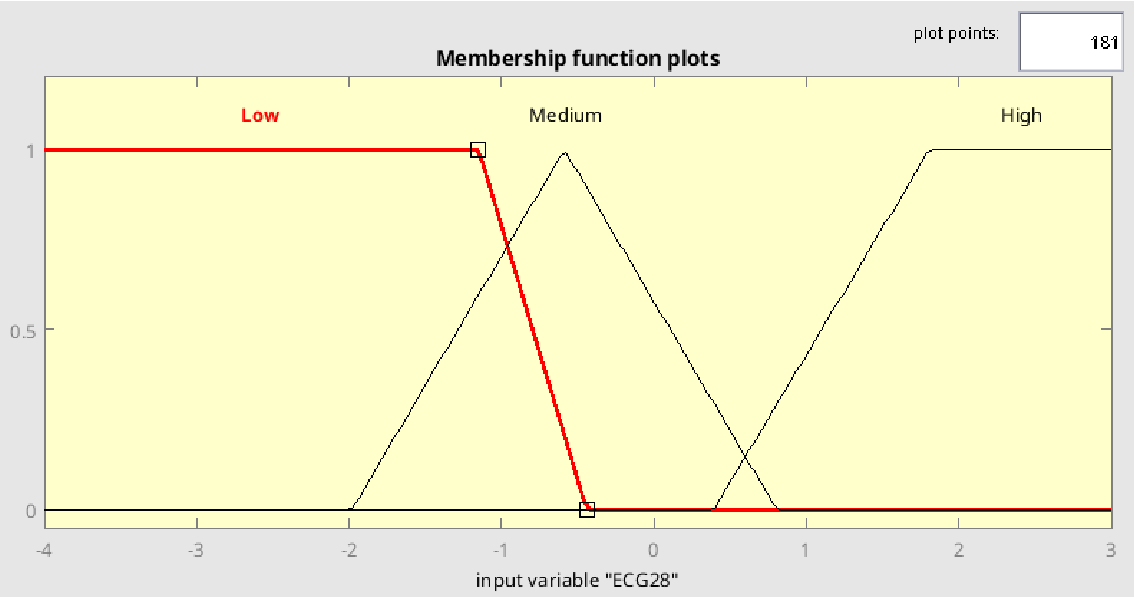


Figure 24: Modeling for linguistic variable of feature 11: ECG28

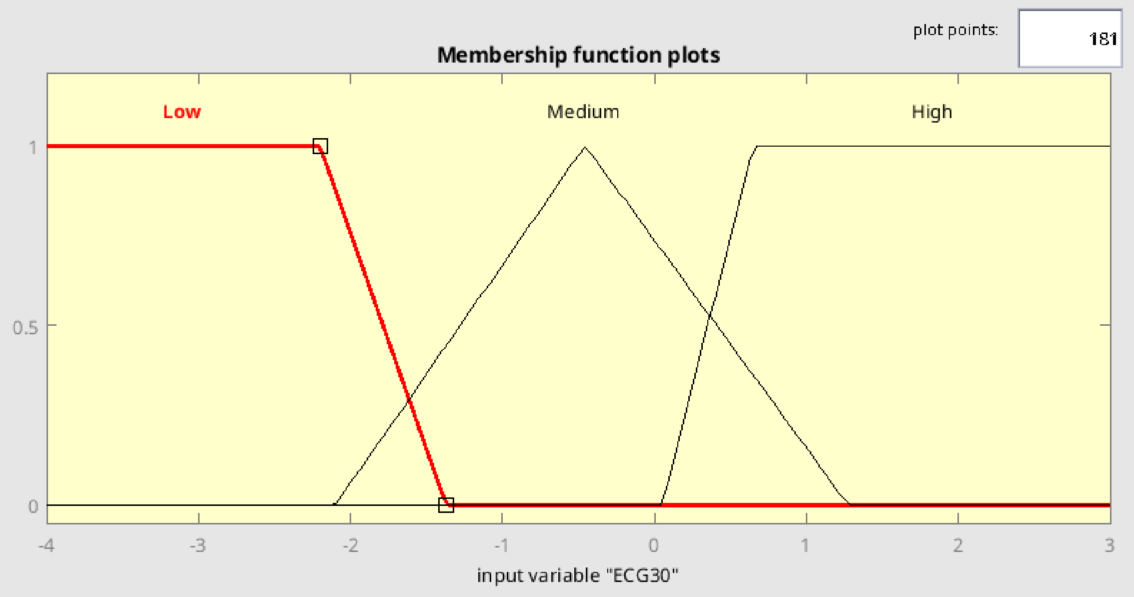


Figure 25: Modeling for linguistic variable of feature 13: ECG30

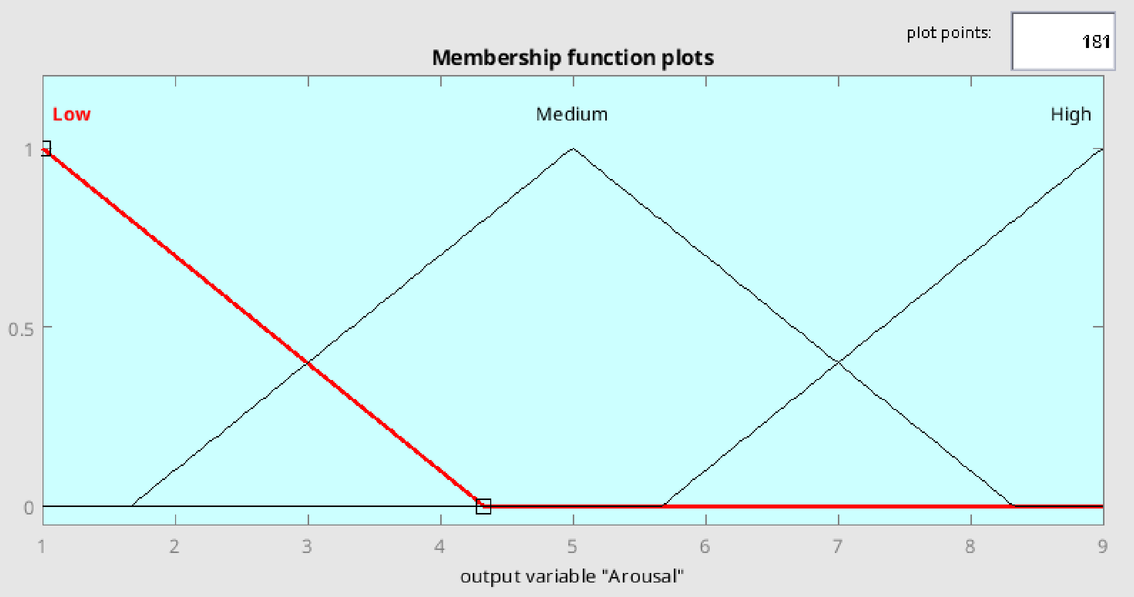


Figure 26: Modeling for linguistic variable of output

The most delicate part of the design of a Mandami 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. 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. 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"*.

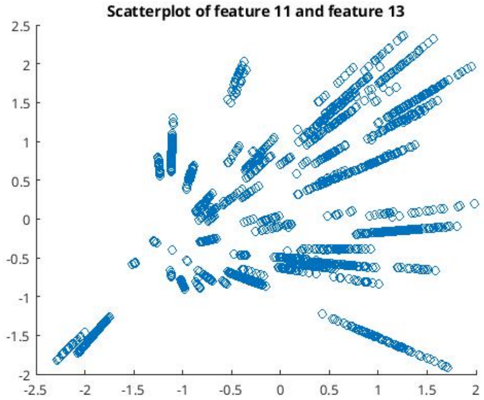


Figure 27: Correlation between feature 11 and feature 13

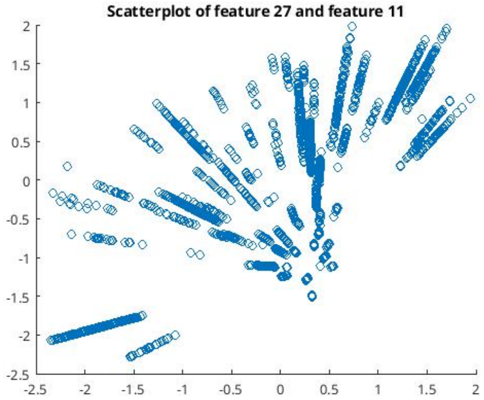


Figure 28: Correlation between feature 27 and feature 11

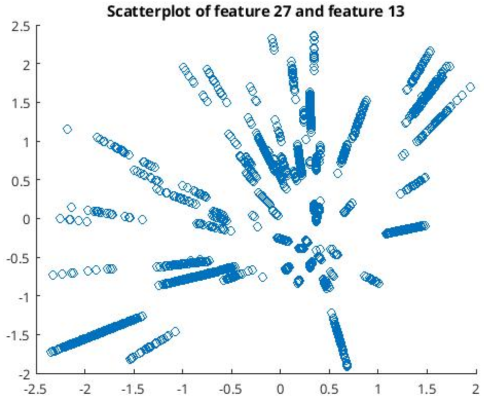


Figure 29: Correlation between feature 27 and feature 13

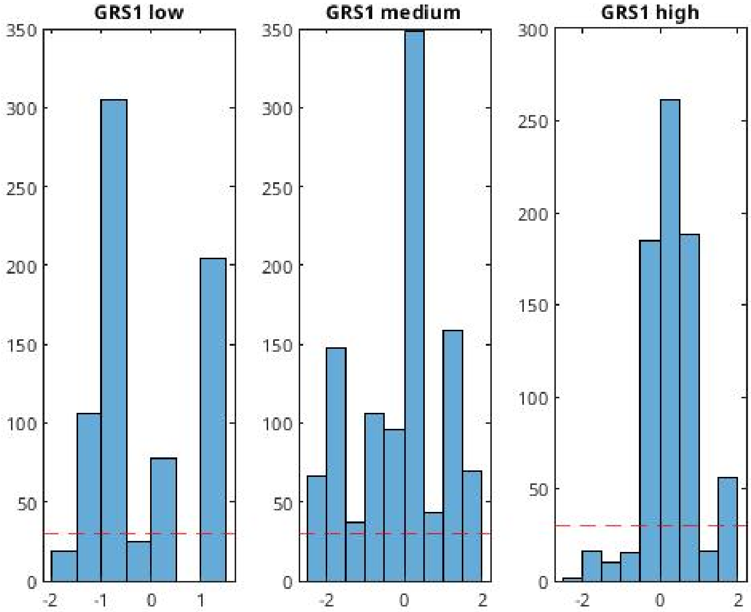


Figure 30: Feature GSR1 with different range of arousal

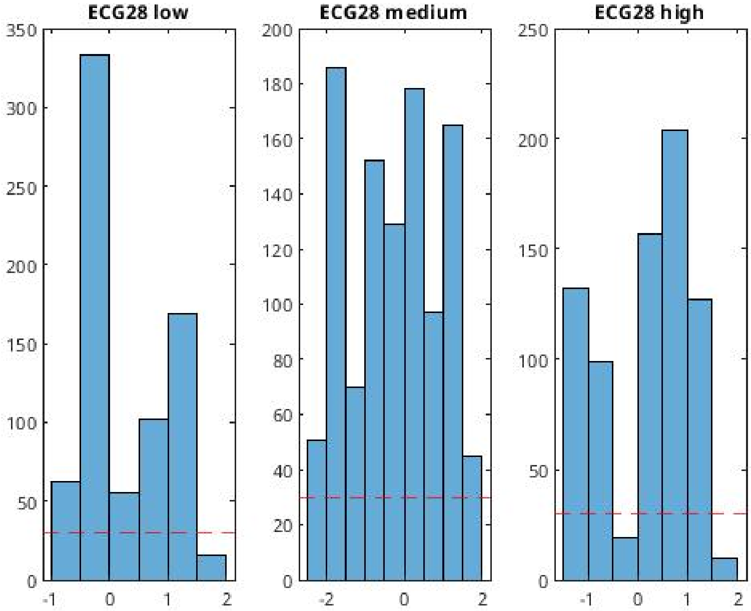


Figure 31: Feature ECG28 with different range of arousal

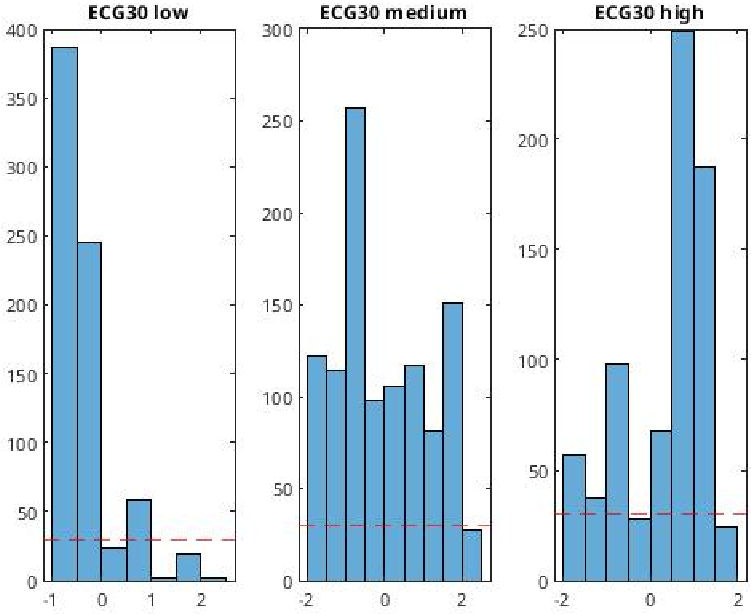
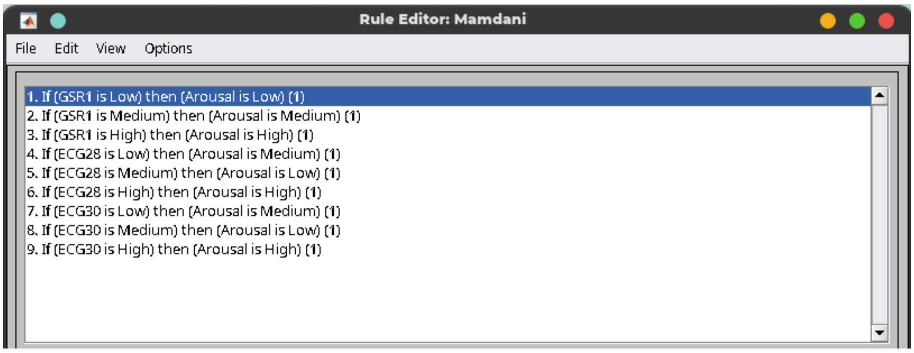


Figure 32: Feature ECG30 with different range of arousal

The following are the modeled rules:



1. If (GSR1 is Low) then (Arousal is Low) (1)
2. If (GSR1 is Medium) then (Arousal is Medium) (1)
3. If (GSR1 is High) then (Arousal is High) (1)
4. If (ECG28 is Low) then (Arousal is Medium) (1)
5. If (ECG28 is Medium) then (Arousal is Low) (1)
6. If (ECG28 is High) then (Arousal is High) (1)
7. If (ECG30 is Low) then (Arousal is Medium) (1)
8. If (ECG30 is Medium) then (Arousal is Low) (1)
9. If (ECG30 is High) then (Arousal is High) (1)

Figure 33: Fuzzy rules

6 Classification of emotions using facial expressions

The following section sets out the development and training of a CNN based on the pre-trained 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, 500 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.

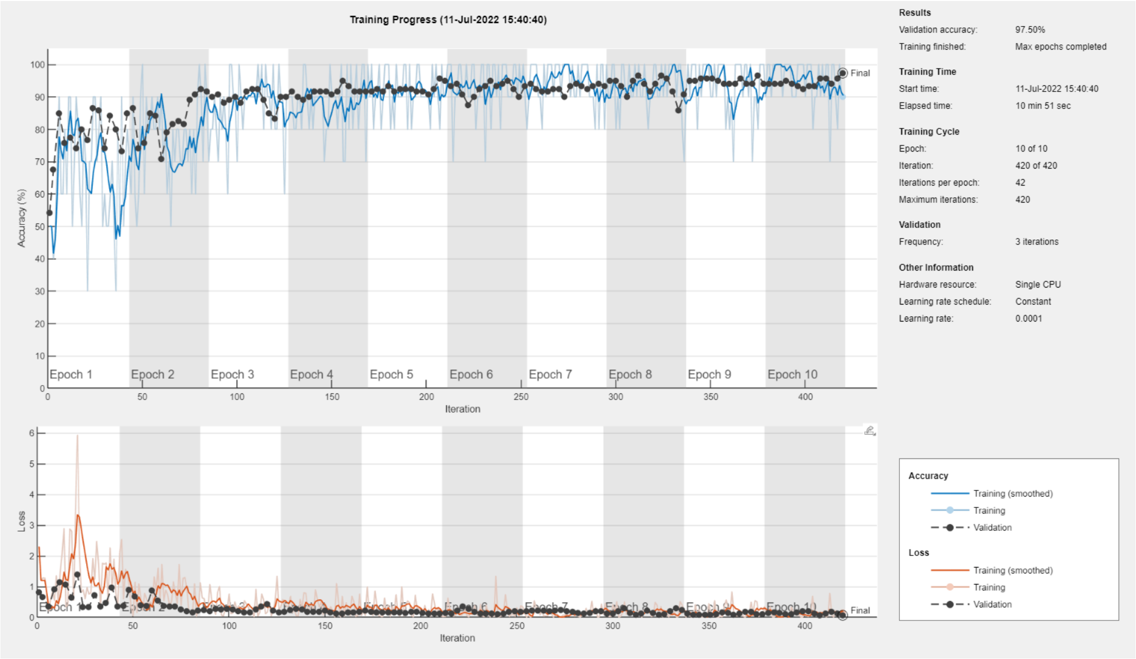
```
pixelRange = [-30 30];
imageAugmenter = imageDataAugmenter( ...
    'RandXReflection',true, ...
    'RandXTranslation',pixelRange, ...
    'RandYTranslation',pixelRange);

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');
```

6.1 Classification with 2 classes



Results

Validation accuracy: 97.50%

Training finished: Max epochs completed

Training Time

Start time: 11-Jul-2022 15:40:40

Elapsed time: 10 min 51 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 34: Training plot for pre-trained CNN with 2 classes and 300 images

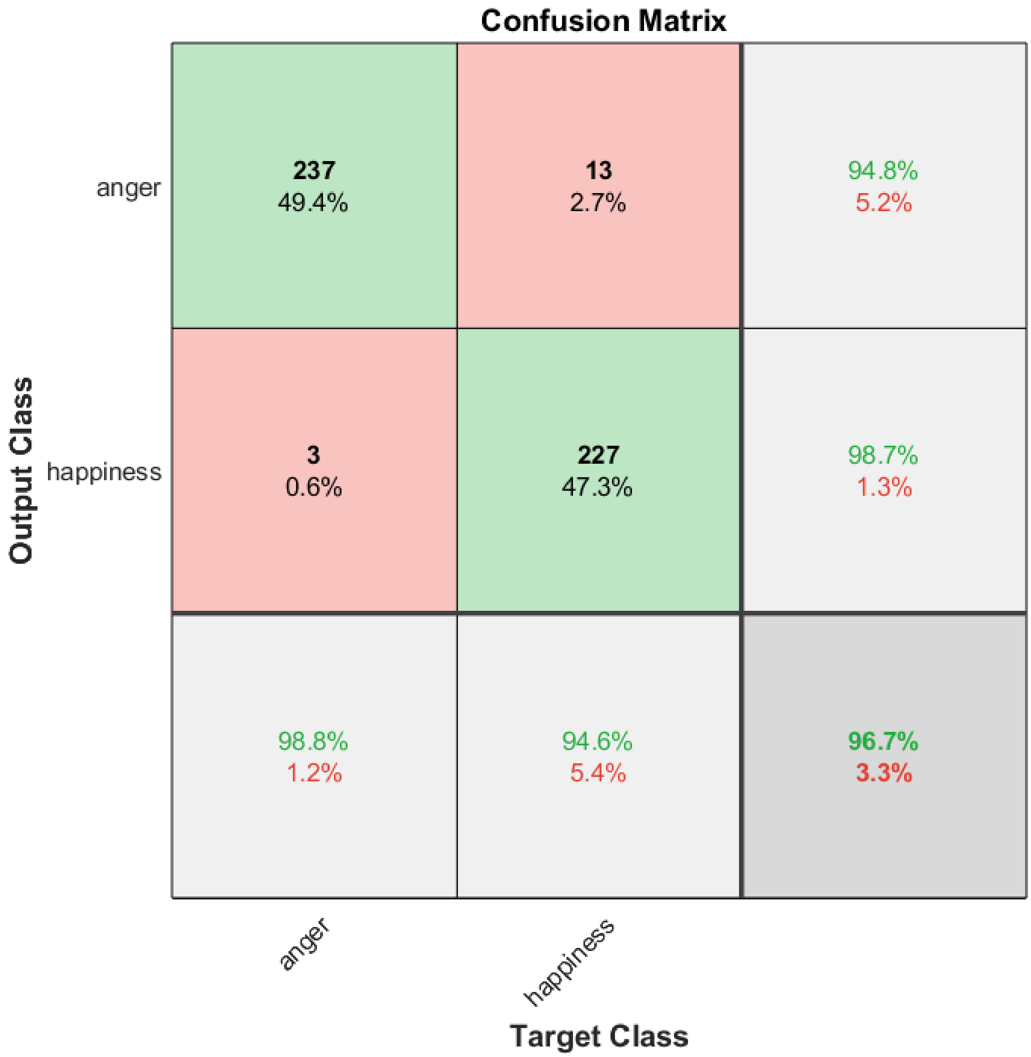
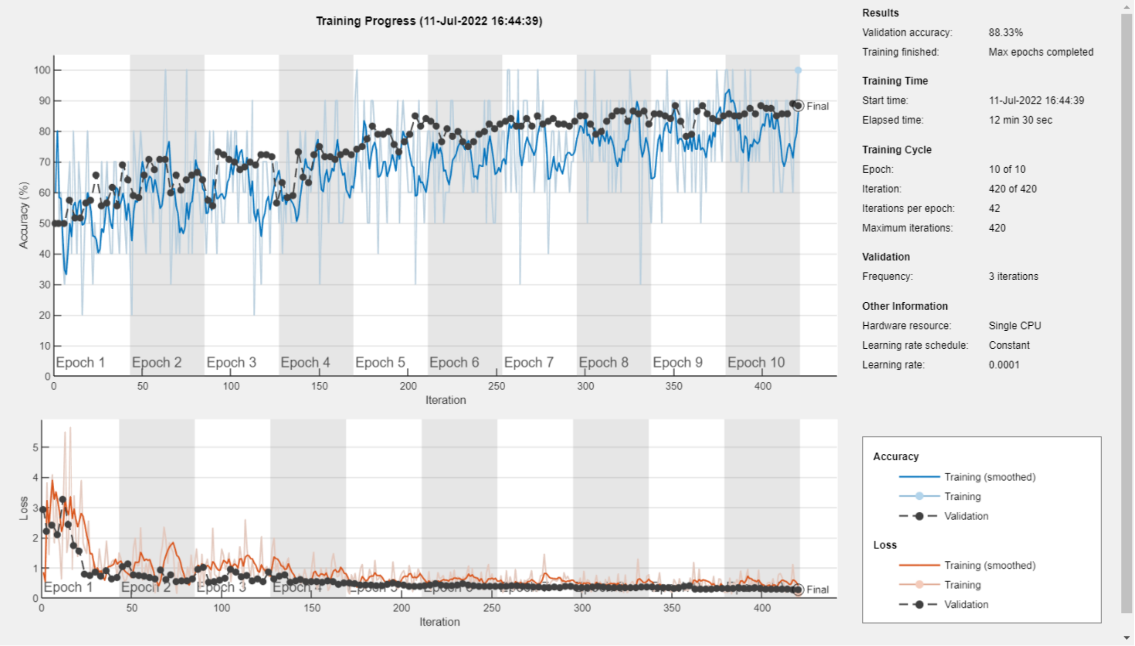


Figure 35: Confusion matrix for pre-trained CNN with 2 classes and 300 images

As can be seen from the images, the classification of the expressions of *fear* 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 *fear* and *happiness* are easily distinguishable.

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



Results

Validation accuracy: 88.33%

Training finished: Max epochs completed

Training Time

Start time: 11-Jul-2022 16:44:39

Elapsed time: 12 min 30 sec

Training Cycle

Epoch: 10 of 10

Iteration: 420 of 420

Iterations per epoch: 42

Maximum iterations: 420

Validation

Frequency: 3 iterations

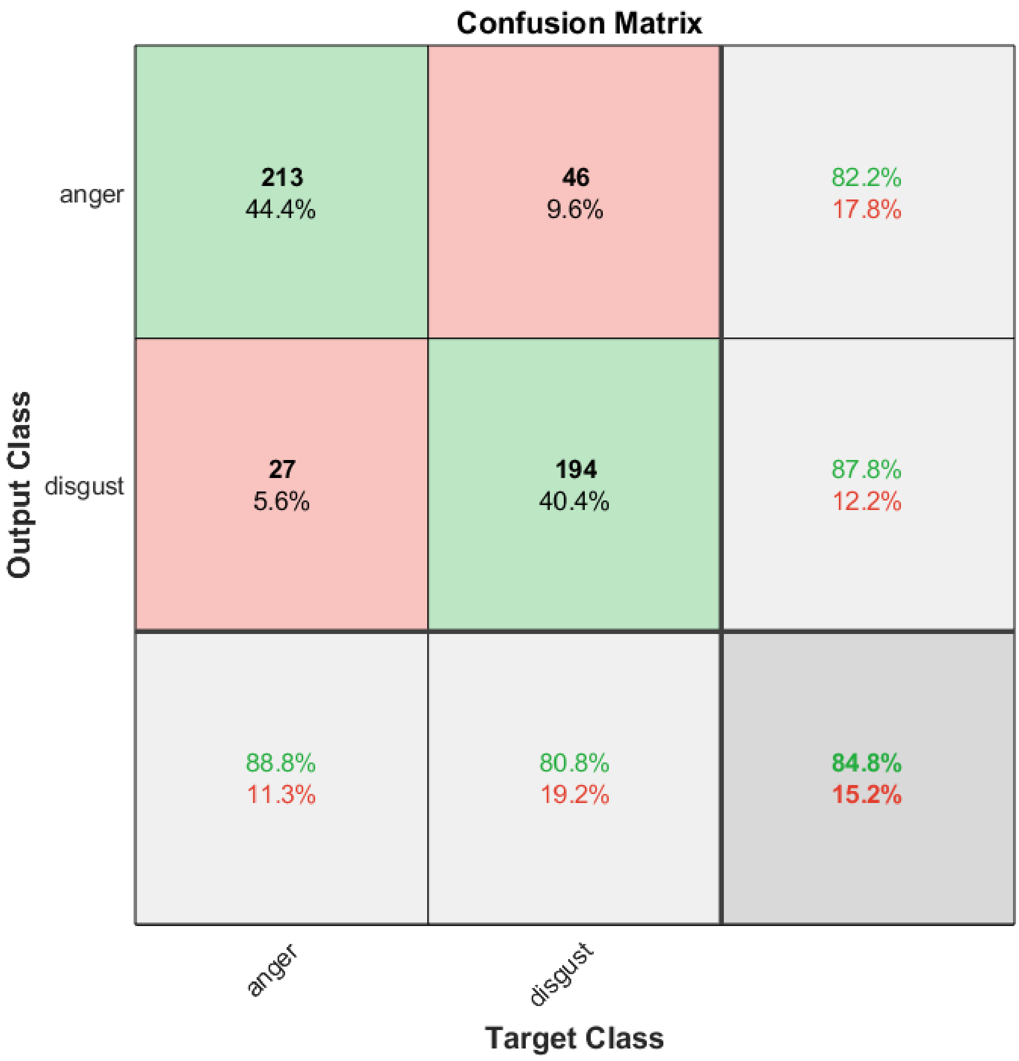
Other Information

Hardware resource: Single CPU

Learning rate schedule: Constant

Learning rate: 0.0001

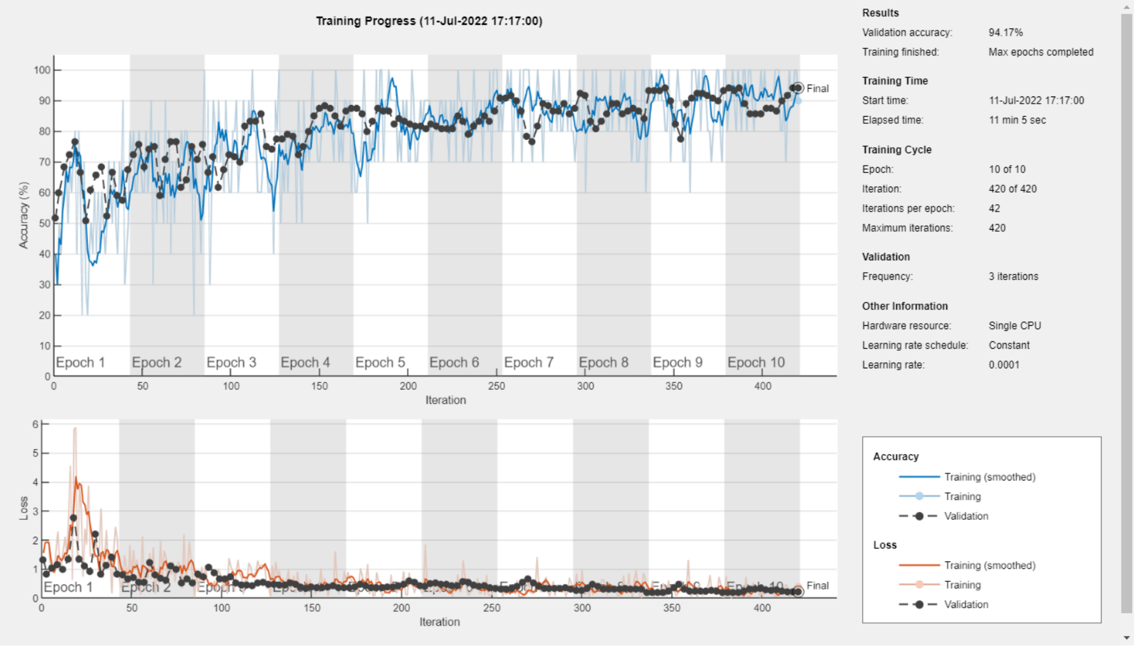
Figure 36: Training plot for the same experiment with different classes



	anger	disgust	
anger	213 44.4%	46 9.6%	82.2% 17.8%
disgust	27 5.6%	194 40.4%	87.8% 12.2%
	88.8% 11.3%	80.8% 19.2%	**84.8%** **15.2%**

Figure 37: Confusion matrix for the same experiment with different classes

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



Results

Validation accuracy: 94.17%

Training finished: Max epochs completed

Training Time

Start time: 11-Jul-2022 17:17:00

Elapsed time: 11 min 5 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 38: Training plot for the same experiment with no selection of images

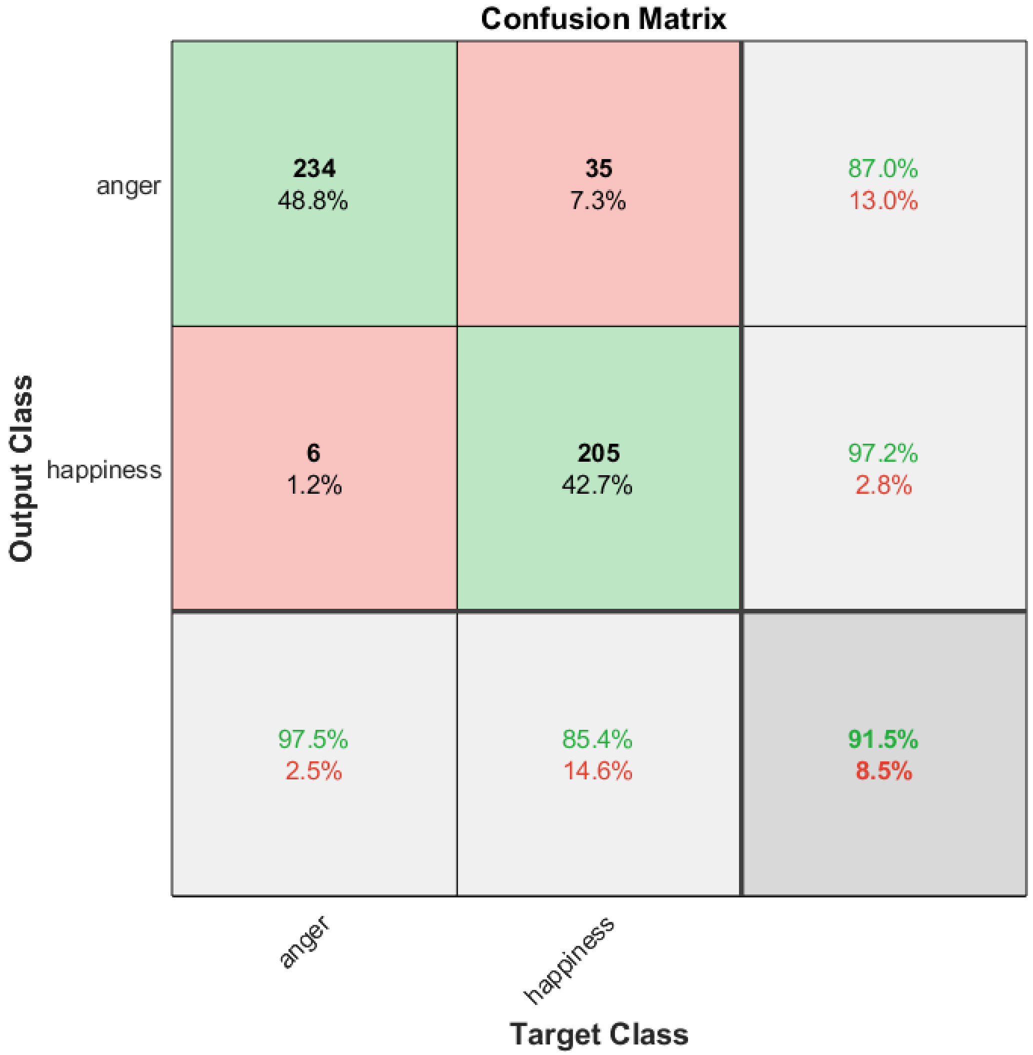
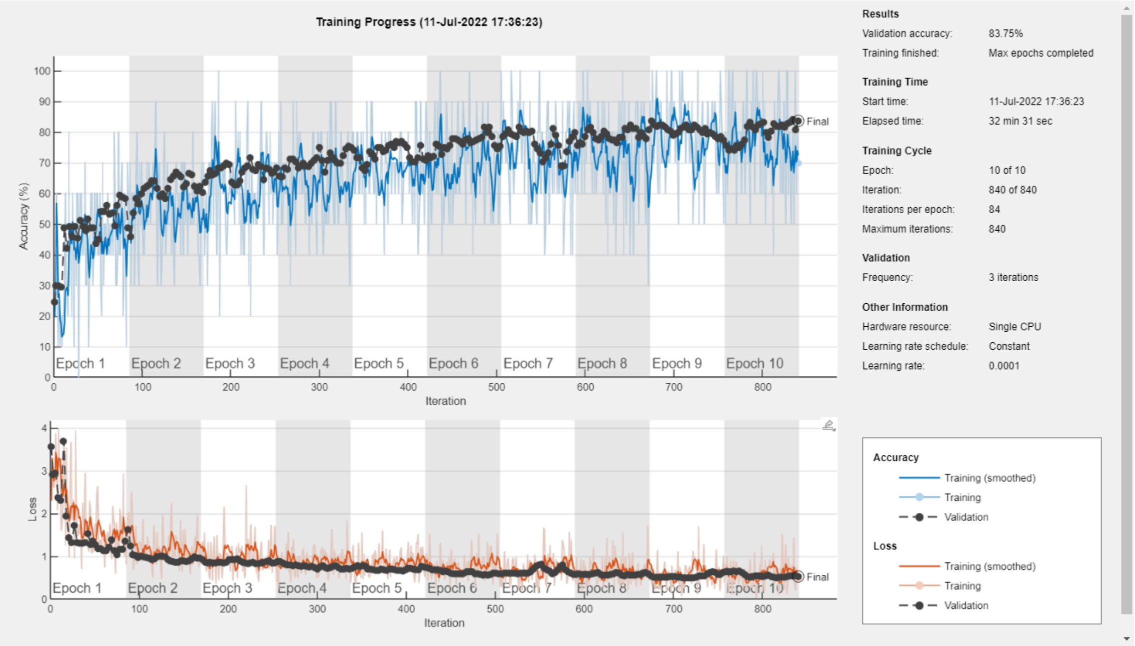


Figure 39: Confusion matrix for the same experiment with no selection of images

6.2 Classification with 4 classes



Results

Validation accuracy: 83.75%

Training finished: Max epochs completed

Training Time

Start time: 11-Jul-2022 17:36:23

Elapsed time: 32 min 31 sec

Training Cycle

Epoch: 10 of 10

Iteration: 840 of 840

Iterations per epoch: 84

Maximum iterations: 840

Validation

Frequency: 3 iterations

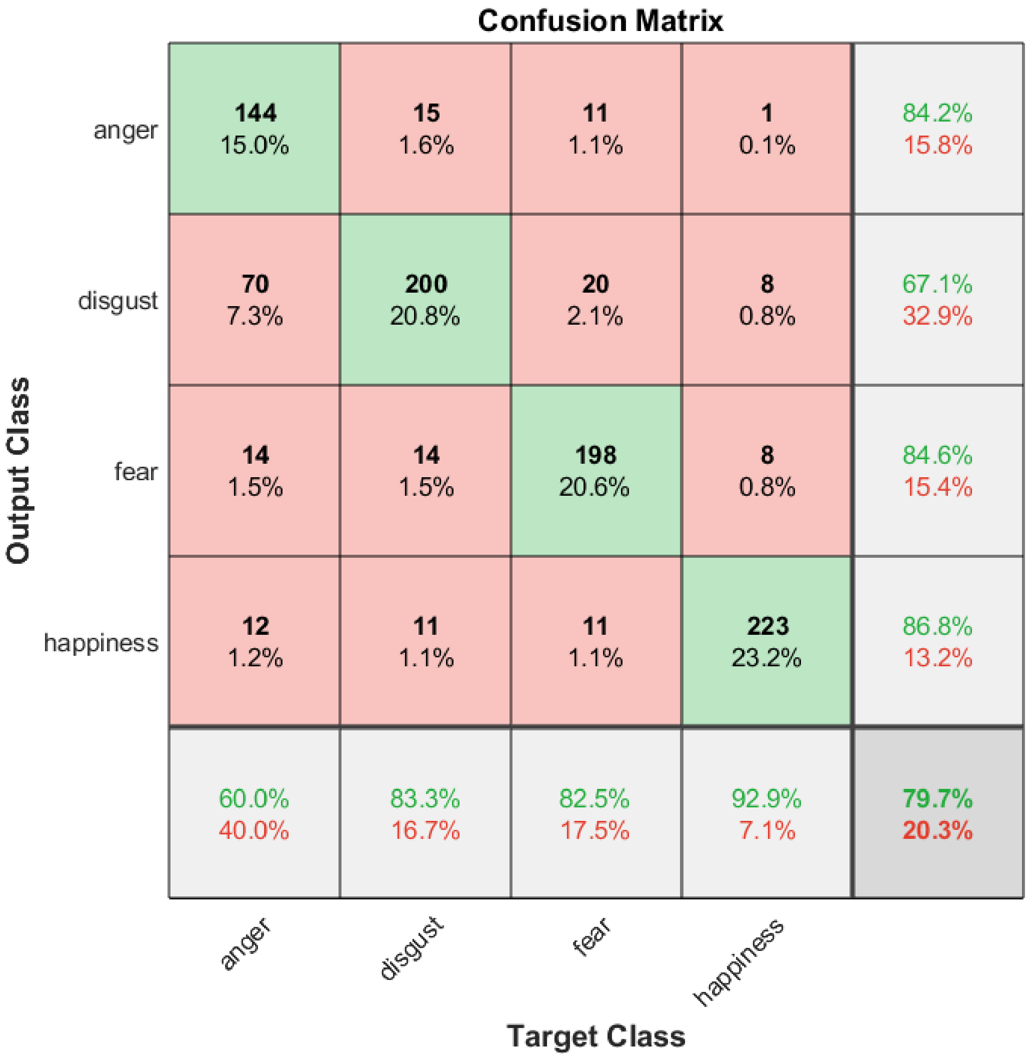
Other Information

Hardware resource: Single CPU

Learning rate schedule: Constant

Learning rate: 0.0001

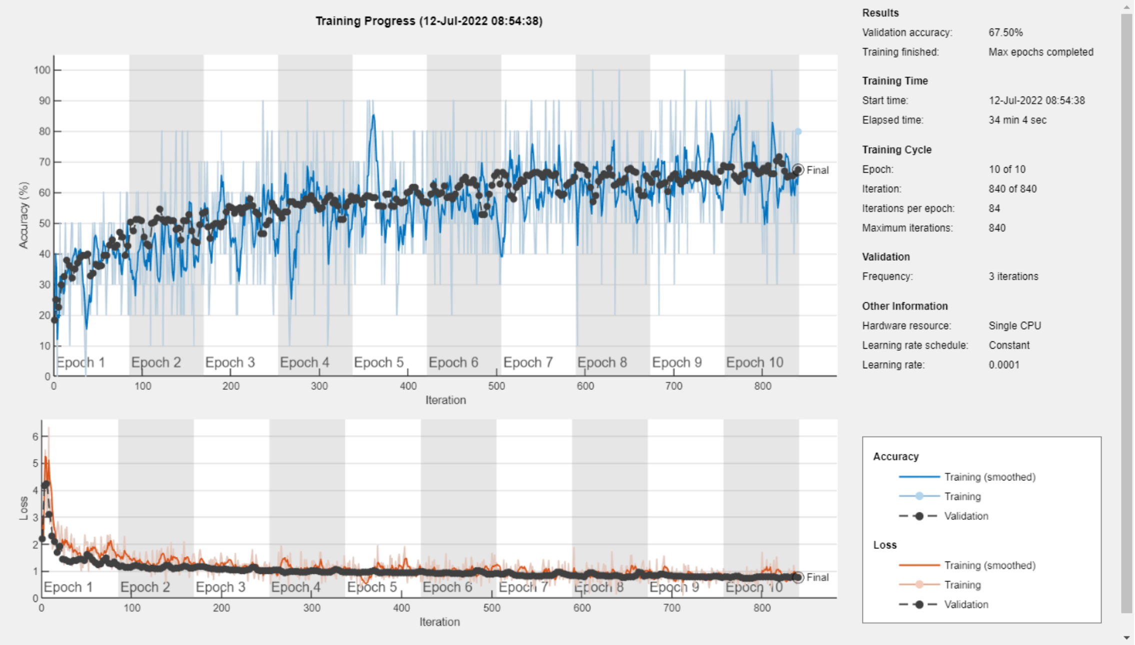
Figure 40: Training plot for the same experiment with 4 classes and 300 images


Confusion Matrix
Output Class
Target Class

	anger	disgust	fear	happiness	
anger	144 15.0%	15 1.6%	11 1.1%	1 0.1%	84.2% 15.8%
disgust	70 7.3%	200 20.8%	20 2.1%	8 0.8%	67.1% 32.9%
fear	14 1.5%	14 1.5%	198 20.6%	8 0.8%	84.6% 15.4%
happiness	12 1.2%	11 1.1%	11 1.1%	223 23.2%	86.8% 13.2%
	60.0% 40.0%	83.3% 16.7%	82.5% 17.5%	92.9% 7.1%	**79.7%** **20.3%**

Figure 41: 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: 67.50%

Training finished: Max epochs completed

Training Time

Start time: 12-Jul-2022 08:54:38

Elapsed time: 34 min 4 sec

Training Cycle

Epoch: 10 of 10

Iteration: 840 of 840

Iterations per epoch: 84

Maximum iterations: 840

Validation

Frequency: 3 iterations

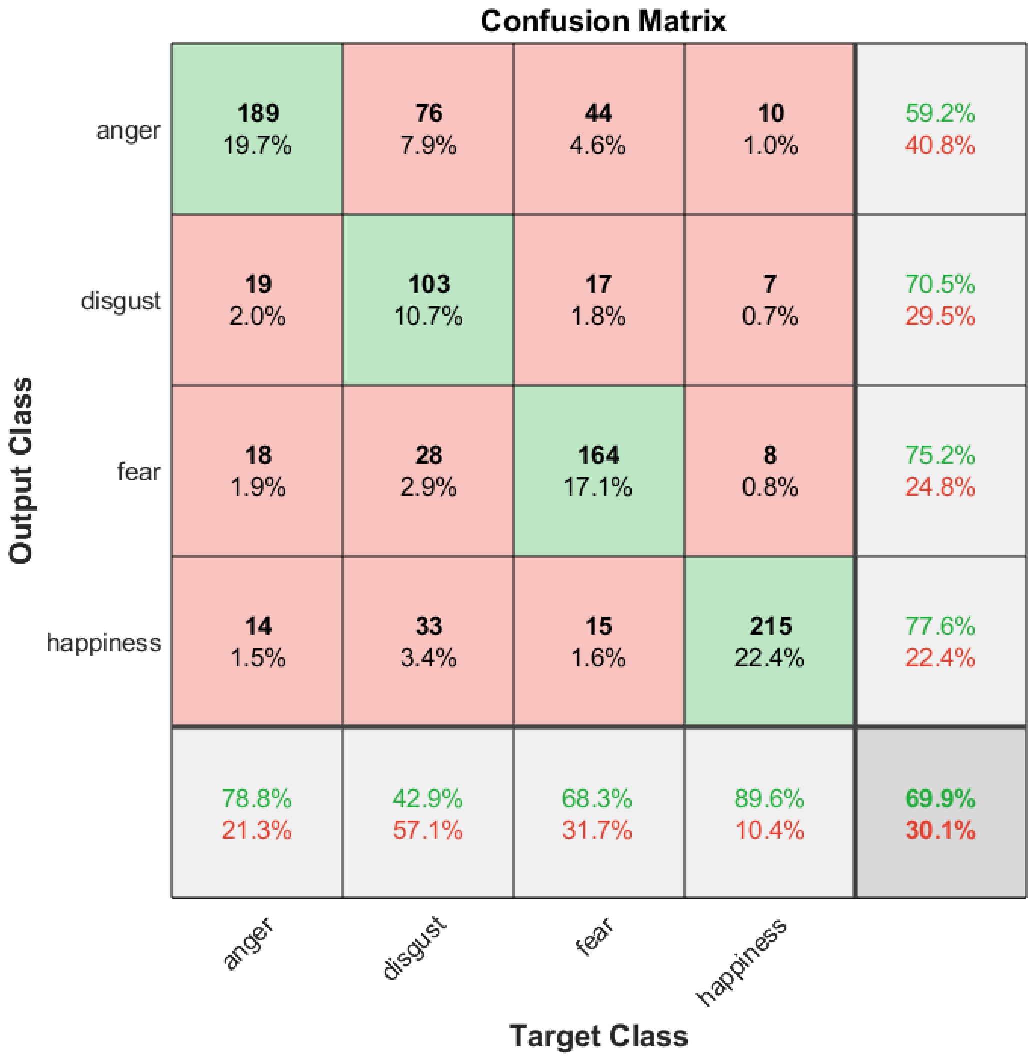
Other Information

Hardware resource: Single CPU

Learning rate schedule: Constant

Learning rate: 0.0001

Figure 42: Training plot for the same experiment with 4 classes and 300 images not selected

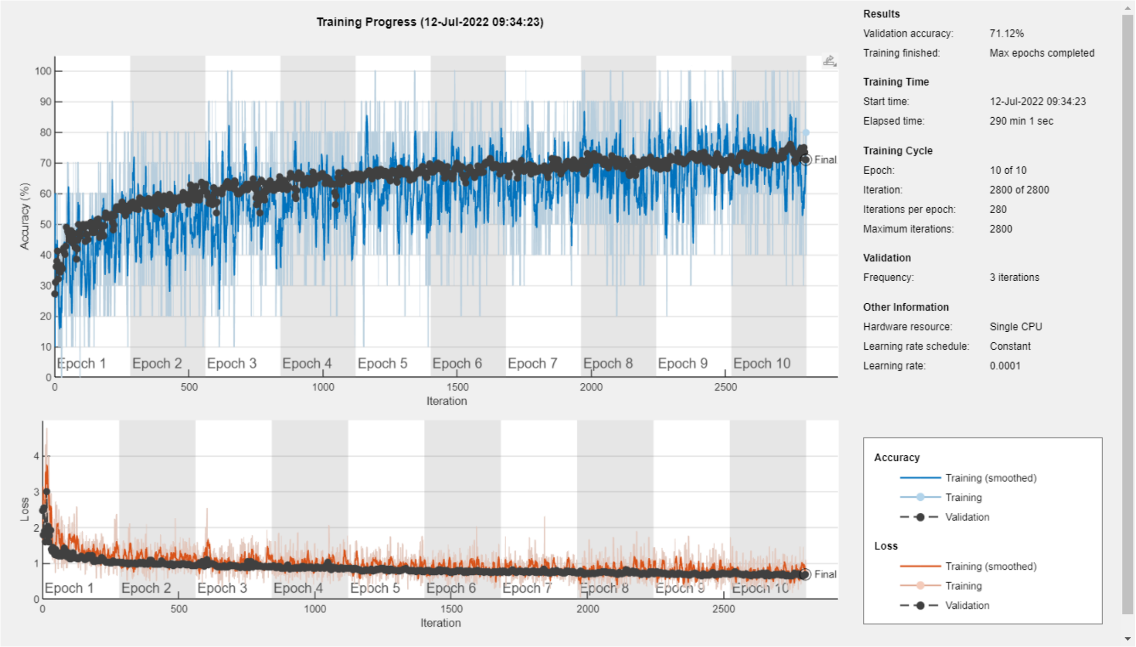


Confusion Matrix

Output Class \ Target Class	anger	disgust	fear	happiness	
anger	**189** 19.7%	**76** 7.9%	**44** 4.6%	**10** 1.0%	59.2% 40.8%
disgust	**19** 2.0%	**103** 10.7%	**17** 1.8%	**7** 0.7%	70.5% 29.5%
fear	**18** 1.9%	**28** 2.9%	**164** 17.1%	**8** 0.8%	75.2% 24.8%
happiness	**14** 1.5%	**33** 3.4%	**15** 1.6%	**215** 22.4%	77.6% 22.4%
	78.8% 21.3%	42.9% 57.1%	68.3% 31.7%	89.6% 10.4%	**69.9%** **30.1%**

Figure 43: 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

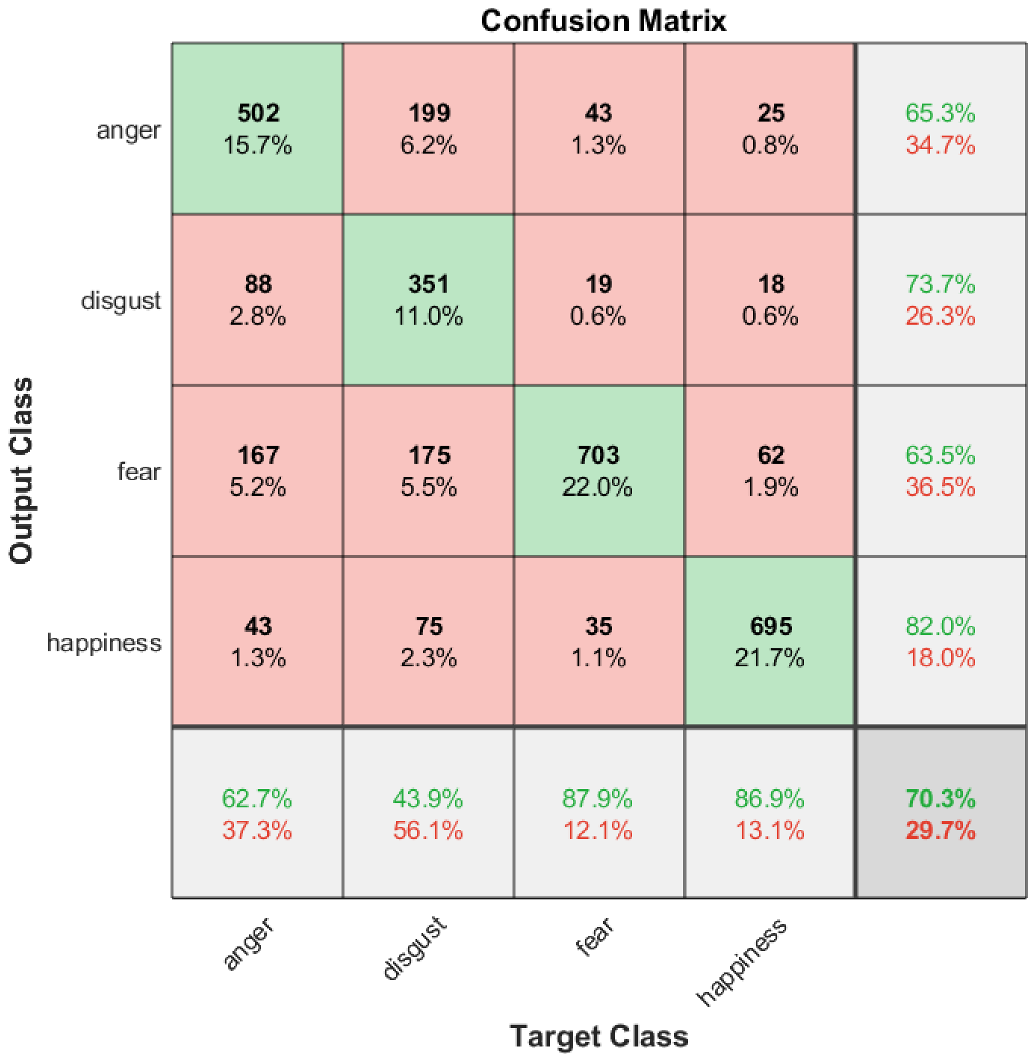
Training Time
Start time: 12-Jul-2022 09:34:23
Elapsed time: 290 min 1 sec

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

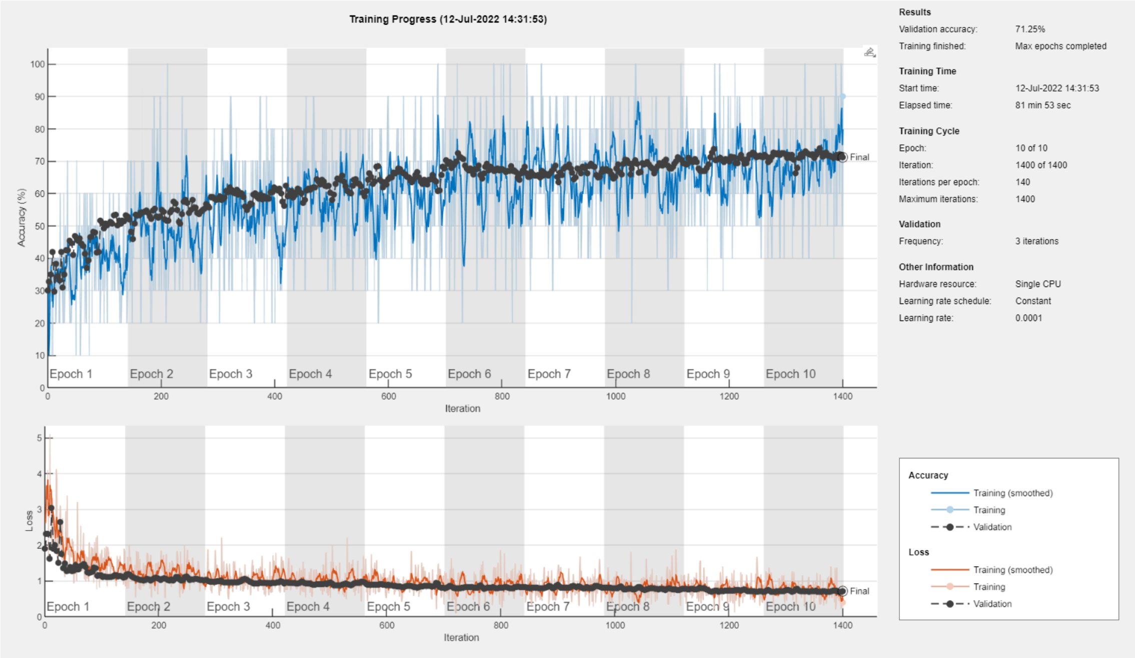


Confusion Matrix

Output Class \ Target 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%**

Figure 45: 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 34 minutes to 290!), but also greater accuracy because the size of the training set is larger and allows us to generalize more successfully.



Results
Validation accuracy: 71.25%
Training finished: Max epochs completed

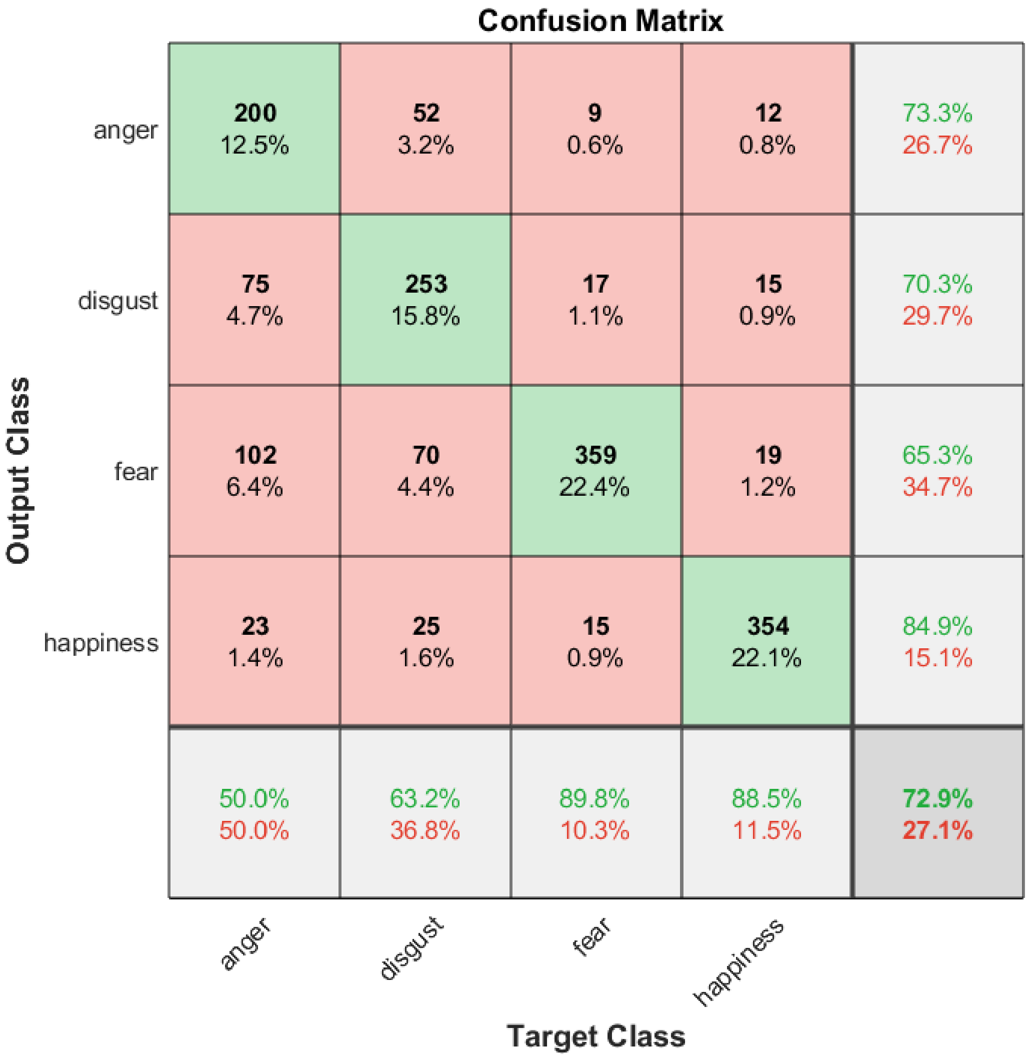
Training Time
Start time: 12-Jul-2022 14:31:53
Elapsed time: 81 min 53 sec

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



	anger	disgust	fear	happiness	
anger	**200** 12.5%	**52** 3.2%	**9** 0.6%	**12** 0.8%	73.3% 26.7%
disgust	**75** 4.7%	**253** 15.8%	**17** 1.1%	**15** 0.9%	70.3% 29.7%
fear	**102** 6.4%	**70** 4.4%	**359** 22.4%	**19** 1.2%	65.3% 34.7%
happiness	**23** 1.4%	**25** 1.6%	**15** 0.9%	**354** 22.1%	84.9% 15.1%
	50.0% 50.0%	63.2% 36.8%	89.8% 10.3%	88.5% 11.5%	**72.9%** **27.1%**

Figure 47: 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.