UNIVERSITÀ DI PISA

Computer Engineering

Intelligent Systems

Project Report

Team Members: Kamran Mehravar

Contents

Introduction

One of the milestones of 21th century is development of the Artificial Neuronal Network to design and develop an intelligent system that measures a person's affective state based on various biomedical signals that are recorded by sensors.
In order to do that fist we used a set of the dataset's, also for make the calculation more easier we normalized the data then balanced them. Now the data is ready to be implemented for balancing and noise-removal (outlier-removal) in MATLAB. After that we compared the valence and arousal of the data to be used in the best way in our code.
As far as the size of the data sometimes can be high and the calculation would take much time we decided to decrease the size of data using sequentialfs MATLAB function and GA algorithm by the magnitude of 10. At this moment we have to first choose the best architecture for out ANN system and then Test the best architecture for MLP for valance and Arousal. Hence, we must Design and Train Radial Basis Function Neural Network.
As a final step we decided to Test the best architecture for obtaining the best RBF.

For the next task we use Mamdani-type fuzzy inference system to Computing the histogram of each three features for each large class.

For improving the MLP and reduce the error , we Implement the fuzzy interface system that we already designed .

Moreover for the Classifying facial expressions we have to employ the convolutional neural networks.
as a last task we Classified facial expressions with a pretrained CNN. In other words, The aim of this part of the project is to fine-tune a pretrained CNN to perform the task described in the previous section.

At this time we use a trained network known as alexnet for our CNN.

For addendum , as far as the amount of data and calculation is too high and we don't have a hardware which can handle this project we have to use external system such as Amazon AWS stack machine and MATLAB cloud infrastructure.

1 First part of project:

The aim of the first part of this project is to design and develop an intelligent system that measures a person's affective state based on various biomedical signals that are recorded by sensors.

1.1 Datasets

A total of 58 participants took part in an experiment in which their biomedical signals were continuously recorded while they were watching a series of 36 excerpts of movie scenes. At the end of each video, each participant rated the emotion felt throughout the video, in terms of valence and arousal levels. A nine-point scale was used with a scale from 1 (very negative) to 9 (very positive) for valence, and from 1 (very boring) to 9 (very exciting) for arousal.

1.1.1 The types of signal recorded were:

• **Electrocardiogram (ECG):** records the electrical activity of the heart using electrodes placed on the participant's chest.

- ***ECG Features***

Column name	Electrocardiogram (ECG)
ECG_18	% of times the feature value is above mean + std (IBI)*
ECG_19	% of times the feature value is below mean – std (IBI)*
ECG_26 - ECG_31	Statistical measurements over heart rate variability (HRV)*

** IBI = Interbeat interval, HRV = Heart rate variability*

• **Electroencephalography (EEG):** records the electrical activity of the brain using electrodes placed on the skin of the participant's head.

- ***EEG Features***

Column name	Electroencephalography (EEG)
EEG _0	Average of first derivative
EEG _1	Proportion of negative differential samples
EEG _2	Mean number of peaks
EEG _3	Mean derivative of the inverse channel signal
EEG _4	Average number of peaks in the inverse signal
EEG _5 - EEG _10	Statistical measurements over the EEG channel

• **Galvanic skin response (GSR):** records the electrical conductivity of the skin using two electrodes placed on the middle and index finger phalanges (conductivity increases in proportion to a person's sweating level).

- ***GSR Features***

Column name	Galvanic Skin Response (GSR)
GSR_0	Mean skin resistance
GSR_1	Mean of first derivatives of skin resistance
GSR_2	Mean of absolute values of first derivatives of skin resistance
GSR_3	Mean first derivative for negative values only
GSR_4	Percentage of time with negative first derivative
GSR_5	Standard deviation of skin resistance
GSR_8 – GSR_11	Log power density estimates; 4 sub-bands in the [0-0.4] Hz band
GSR_12	Standard deviation of skin conductance
GSR_13	Mean of first derivatives of skin conductance
GSR_14	Mean of absolute values of first derivatives of skin conductance
GSR_15	Mean of absolute values of second derivatives of skin conductance
GSR_16	Average number of local minima in the skin resistance signal
GSR_17 – GSR_26	Log power density estimates; 10 sub-bands in the [0-2.4] Hz band
GSR_27	Zero crossing rate of skin conductance low response ([0-0.2] Hz)
GSR_28	Mean skin conductance low response peak magnitude
GSR_29	Zero crossing rate of skin conductance very slow response ([0-0.08] Hz)
GSR_30	Mean skin conductance very low response peak magnitude

Dataset *dataset.mat* contains 1591 samples. Each sample is represented by a set of 54 features including 14 features extracted from the ECG, 11 features extracted from the EEG, and 29 features extracted from the GSR. Each row also contains the arousal and valence levels in the third and fourth columns, respectively.

Some of the extracted features are statistical measurements. Each row that mentions *statistical measurements* refers to the following features, in order:

- ***Statistical Features***

Columns	Statistical Measurements
1	Mean
2	Standard deviation (std)
3	Skewness
4	Kurtosis of the raw feature over time
5	% of times the feature value is above mean + std
6	% of times the feature value is below mean - std

1.2 Preparing data for ANN

This step incudes, cleaning process (removing Non-Numeric data, removing infinite numbers, handling missed data), outlier removal, and data balancing.
For the present dataset we do not need the cleaning process, and outlier removal.
In the following chart we analyze the balance of the data for each class.

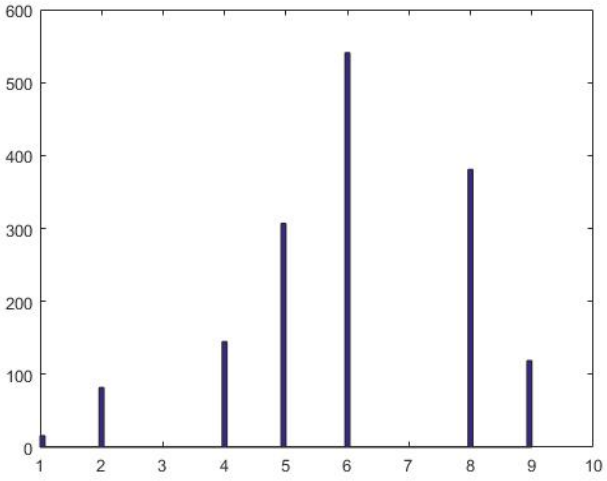


Figure 1-1: Histogram of arousal class distribution

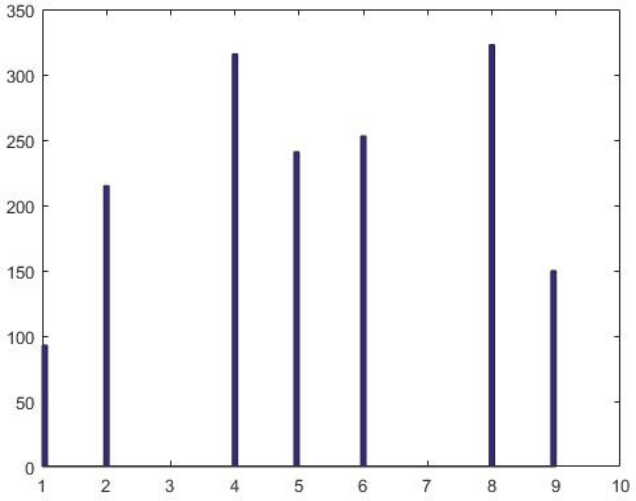
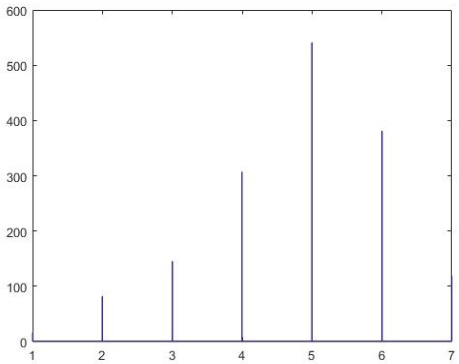
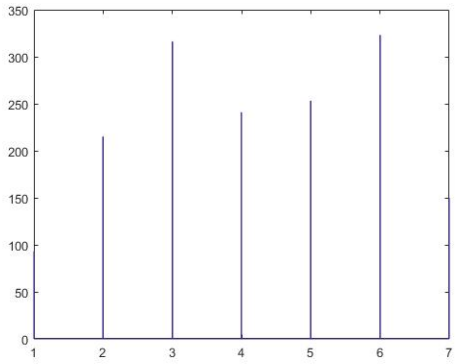


Figure 1-2: Histogram of valance class distribution

Figures 1 and 2 shows the class distribution of the dataset. However we have 7 classes, so we have change the class labels from 1 to 7 in order to reduce computations for acquiring class labels.





As it is shown in fig.1 and fig.2 , the class distribution for both arousal and valance are not balance. We do our simulations in two strategy. The first one is without balancing and the second is with balancing of the class datasets.

1.2.1 Normalizing of the input features

Since the scales of the input variables are not the same, the input features of the dataset is normalizes. Therefore, the impact of each feature on the output of the ANN will be the same. For example one feature range is [0.0001, 00007], and the other feature range is [10 20], therefore the impact of the second feature on the final result is more than the first one. By normalization the impact of each feature on the output of the ANN will be the same.

1.2.2 Data balancing:

There are some ways which can be used to balance the dataset before fitting to the classifier to get the better result. These methods are as follows:

- Under Sampling- Removing the unwanted or repeated data from the majority class and keep only a part of these useful points. In this way, there can be some balance in the data.
- Over Sampling- Try to get more data points for the minority class. Or try to replicate some of the data points of the minority class in order to increase cardinality.
- Generate Data- You can decide to generate synthetic data for the minority class for balancing the data.

1.2.3 Implementation of data-balancing and noise-removal (outlier-removal) in MATLAB

For data-balancing, we have used both under-sampling and generating-data methods. In order to do under-sampling, we have eliminated outliers from each classes. In order to eliminate outliers, the center of each classes has been calculated then the distances of data point belonging to each class to the center point has been calculated. After that, the distances is sorted and the *p-percent* of the farthest data to center are eliminated. For our analysis, *p-percent* = 30%. This process help us to under-sampling the classes with higher number of members. For example consider classes with 10 and 500 members. The algorithm eliminates 3 and 150 data from classes respectively.

Since my MATLAB-version is 2015 the rmoutliers function is not defined in this version. Therefor the new method is developed and outliers were removed depending on the distance to the center of classes.

Another way in order to remove outliers was counting the neighbors of each data point in a predefined radius. Then eliminating the datapoints with the least number of neighbors. However; this method is not implemented in this project.

1.2.4 Implementation of data-balancing in MATLAB

In order to generate data for data-balancing purpose, we randomly select the class which has fewer member compared with the largest class. Roulette wheel is used for this selection, the probability of selecting each class is related to the difference between the members of largest class and each class. We also implemented the function of the Roulette wheel.

After selecting the class by roulette wheel, in order to generate a data point we select a data from the selected class randomly and add some random noise.

Fig.3 and fig.4 show the results of data-balancing for arousal classes and valance classes respectively.

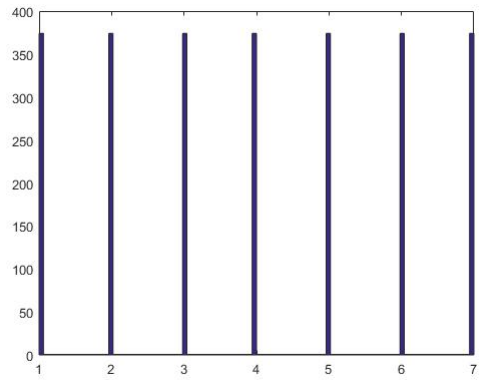


Figure 1-3: the results of data-balancing for arousal classes

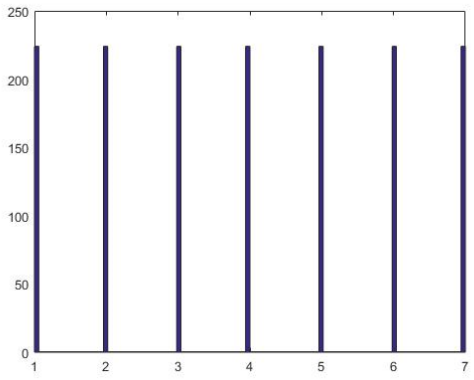


Figure 1-4: the results of data-balancing for valance classes

1.3 Estimating valence and arousal with neural networks

The aim of this part of the project is to design and develop two multi-layer perceptron (MLP) artificial neural networks that accurately estimate a person's valence and arousal levels, respectively. The MLPs take as input a set of features and return the corresponding valence and arousal levels, respectively. The set of extracted features should be reduced by selecting the most significant features to predict the output.

1.3.1 Reducing the feature size to 10 by sequentialfs MATLAB function:

In order to reduce the feature size, one way is to use the sequential feature selection (implemented by the sequentialfs MATLAB function), with a neural network as a criterion function that assesses the accuracy of each subset of features in estimating the valence (or arousal) level. The suggested maximum number of features to select is 10 for each network.
MATLAB help for sequentialfs is shown in Figure 1-1.
The results of finding the best 10 features for the arousal class by means of sequentialfs is:
24 29 14 9 10 21 3 11 23 5
The results of finding the best 10 features for the valance class by means of sequentialfs is:
12 21 6 32 1 41 5 7 39 8
We also develop a genetic algorithm in order to select the best features for the ANN.

1.3.2 Reducing the feature size to 10 by the genetic algorithm

Genetic algorithms (GAs) are search and optimization methods that mimic natural evolution. Progress in natural evolution is based on four fundamental processes:

sequentialfs from MATLAB HELP

Sequential feature selection

Syntax

```
inmodel = sequentialfs(fun,X,y)

inmodel = sequentialfs(fun,X,Y,Z,...)
```

Description

`inmodel = sequentialfs(fun,X,y)` selects a subset of features from the data matrix `X` that best predict the data in `y` by sequentially selecting features until there is no improvement in prediction. Rows of `X` correspond to observations; columns correspond to variables or features. `y` is a column vector of response values or class labels for each observation in `X`. `X` and `y` must have the same number of rows. `fun` is a function handle to a function that defines the criterion used to select features and to determine when to stop. The output `inmodel` is a logical vector indicating which features are finally chosen.

Starting from an empty feature set, `sequentialfs` creates candidate feature subsets by sequentially adding each of the features not yet selected. For each candidate feature subset, `sequentialfs` performs 10-fold cross-validation by repeatedly calling `fun` with different training subsets of `X` and `y`, `XTRAIN` and `ytrain`, and test subsets of `X` and `y`, `XTEST` and `ytest`, as follows:

```
criterion = fun(XTRAIN,ytrain,XTEST,ytest)
```

`XTRAIN` and `ytrain` contain the same subset of rows of `X` and `Y`, while `XTEST` and `ytest` contain the complementary subset of rows. `XTRAIN` and `XTEST`contain the data taken from the columns of `X` that correspond to the current candidate feature set.

Each time it is called, `fun` must return a scalar value `criterion`. Typically, `fun` uses `XTRAIN` and `ytrain` to train or fit a model, then predicts values for`XTEST` using that model, and finally returns some measure of distance, or *loss*, of those predicted values from `ytest`. In the cross-validation calculation for a given candidate feature set, `sequentialfs` sums the values returned by `fun` and divides that sum by the total number of test observations. It then uses that mean value to evaluate each candidate feature subset.

Typical loss measures include sum of squared errors for regression models (`sequentialfs` computes the mean-squared error in this case), and the number of misclassified observations for classification models (`sequentialfs` computes the misclassification rate in this case).

Note: `sequentialfs` divides the sum of the values returned by `fun` across all test sets by the total number Accordingly, `fun` should not divide its output value by the number of test observations.

After computing the mean `criterion` values for each candidate feature subset, `sequentialfs` chooses the candidate feature subset that minimizes the mean criterion value. This process continues until adding more features does not decrease the criterion.

Figure 1-5: MATLAB Help for sequentialfs function.

• selection picks the individuals that will produce offspring,
• recombination (or crossover) combines two different individuals to produce offspring,
• mutation is the random variation of the existing genetic material.
• elitism

In order to select the best 10 features, we also develop a genetic algorithm and select the best features where minimize the classifier (MLP) error.
The chromosomes are consisting of 10 genes. The genes are the same as features. The objective function is the number of errors (or the error) of the classifier algorithm. We have used a 3 layer MLP (2 hidden layer) because it has produces better results for the dataset.
The initial population is 50 and the maximum number of generations is 10.
Parents are selected by the Roulette wheel and the probability of selecting the parents is related to the inverse of error.
The crossover operator is one point crossover.
The mutation is generating all genes randomly, in order to better explore the space.
The percentage of each operator for generating population of the next generation is as follows:

40% crossover
40% elitism
20% mutation
We also use 4-fold cross validation for evaluating the classifier.
The results of finding the best 10 features for the arousal class by means of genetic algorithm is:

Best chromosome of the 1st generation	24	19	37	40	25	12	53	45	33	23
Best chromosome of the 2nd generation	9	50	6	17	41	28	1	7	29	25
	2	37	19	29	24	6	34	7	46	53
	8	11	4	2	10	26	3	22	24	42
	14	1	51	45	30	23	42	4	20	12
	11	14	39	22	20	40	6	5	52	10
	49	17	44	32	46	24	12	27	25	23
	24	9	10	37	30	23	42	4	20	12
	11	29	7	12	27	39	52	9	14	20
	11	29	7	12	27	39	52	9	14	20
	11	29	7	12	27	39	52	9	14	20
	10	38	17	25	44	3	26	22	24	42
	11	7	31	6	10	17	28	37	24	35
	17	23	19	40	49	9	12	27	25	3
	2	11	12	18	25	1	26	10	14	31
	8	24	18	48	3	26	1	52	25	33
	29	31	2	40	10	44	52	48	13	28
	29	10	8	48	14	4	11	27	3	22
	8	24	18	48	3	26	1	52	25	33
	17	23	19	40	49	9	12	27	25	3
	28	4	23	31	10	47	41	46	11	26
	28	4	23	31	10	47	40	32	13	2
	24	10	30	22	32	12	34	28	16	25
	29	28	31	44	4	19	33	47	10	50
	11	27	26	40	9	20	41	46	2	5
	45	34	14	40	49	9	12	27	25	3
	34	11	14	38	2	26	24	25	48	6
	11	26	15	5	47	4	28	10	23	27
	24	10	30	40	7	53	49	11	2	12
	28	4	23	31	10	47	41	46	11	26
	22	24	42	7	36	10	28	31	19	26
	45	34	14	40	49	9	12	27	25	3
	22	46	40	50	53	12	28	10	23	27
	31	24	42	7	36	10	28	35	19	26
	34	11	14	38	2	26	24	25	48	6
	23	40	14	6	25	30	11	53	24	8
	2	32	10	26	12	23	43	41	7	15
	43	23	47	11	8	17	6	32	12	28
	26	24	40	32	19	34	11	42	1	3
	34	11	25	12	27	26	23	21	20	40
	45	34	14	40	49	9	12	27	25	3
	31	24	12	25	1	11	48	22	23	34
	18	25	11	29	15	28	23	10	46	41
	45	34	14	40	49	9	12	27	25	3
	34	11	25	12	27	26	23	21	20	40
	26	30	53	9	41	19	6	14	13	36
	12	10	25	31	24	39	40	1	28	29
	31	24	12	25	1	11	48	22	23	34
	46	20	32	25	6	26	24	39	41	12
Best chromosome of the 50th generation	35	9	3	43	24	14	33	1	25	8

The mostly repeated features in the best chromosomes is as follows

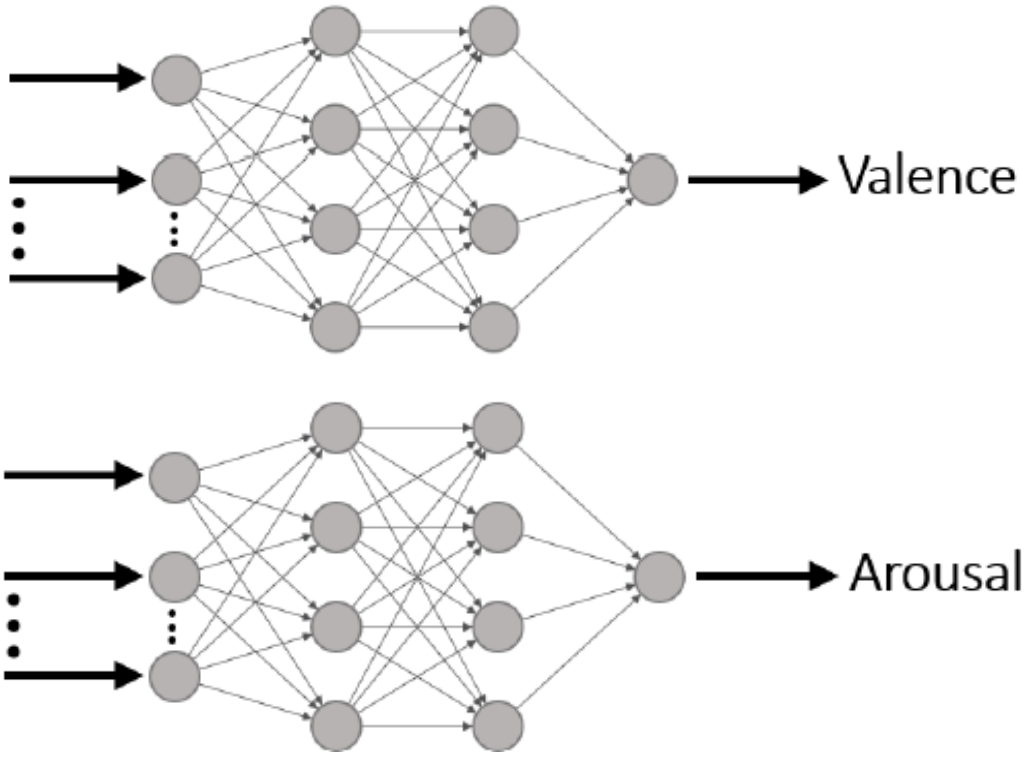
Num rep	25	25	23	22	21	19	19	18	16	16	15	14	13	13	12	10	10	10	10	10	10
Feature	12	25	11	24	10	23	26	40	14	27	28	9	3	34	31	1	2	6	7	20	29

The results of finding the best 10 features for the valance class by means of genetic algorithm, with 20 populations and 10 generations is:

34	11	19	9	22	48	33	36	32	29
47	8	50	4	28	43	21	11	23	19
29	37	25	1	52	10	21	50	32	9
19	52	31	34	50	53	4	49	39	21
19	52	31	34	50	53	4	49	39	21
19	52	31	34	50	53	4	49	39	21
47	19	8	9	12	34	31	29	16	23
30	29	22	14	15	21	49	8	50	35
12	52	10	46	20	48	4	6	19	37
29	37	53	27	52	10	21	50	19	49

1.3.3 Selecting the best architecture for ANNs:

We have two Artificial Neural Network for this problem; one ANN is for detecting the valence level and the other is for detecting the arousal level.



Once the search for the best set of features is completed in the previous step, the next step is to find the best architecture for both ANNs.

In order to find the best architecture for the neural network, we can change the number of hidden layers, and also the number of neurons in each hidden layer. Changing the activation functions and altering the stop condition of the training procedure, for example maximum number of validation fails which is used for avoiding overtraining.

We run the algorithm with, one, two and three hidden layers and observe that the best results is obtained with two hidden layers. Then, we have modified the ANN by changing number of neurons in hidden layers with a heuristic method. We have tested the following vectors as the input for feedforwardnet()function of MATLAB, denoting the number of hidden neurons in each layer. Then select the best architecture according to accuracy percentage.

[10 10], [20 10], [30 10], [40 10], [50 10], [60 10]
[10 20], [20 20], [30 20], [40 20], [50 20], [60 20]
[10 30], [20 30], [30 30], [40 30], [50 30], [60 30]

The error values for each architecture:

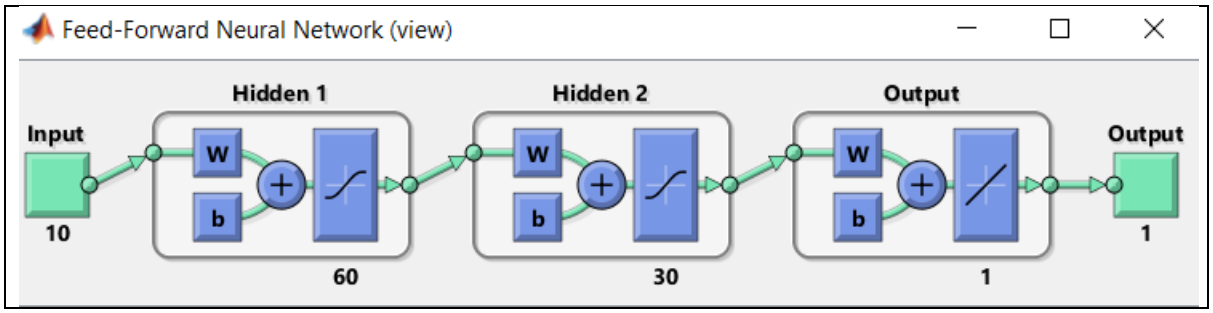
0.573	0.6566	0.5382	0.5924	0.6241	0.5122	0.5909	0.5034	0.5065
0.5932	0.5733	0.5328	0.5646	0.5443	0.5512	0.5474	0.5722	0.4549

Therefore the best architecture is [60 30]

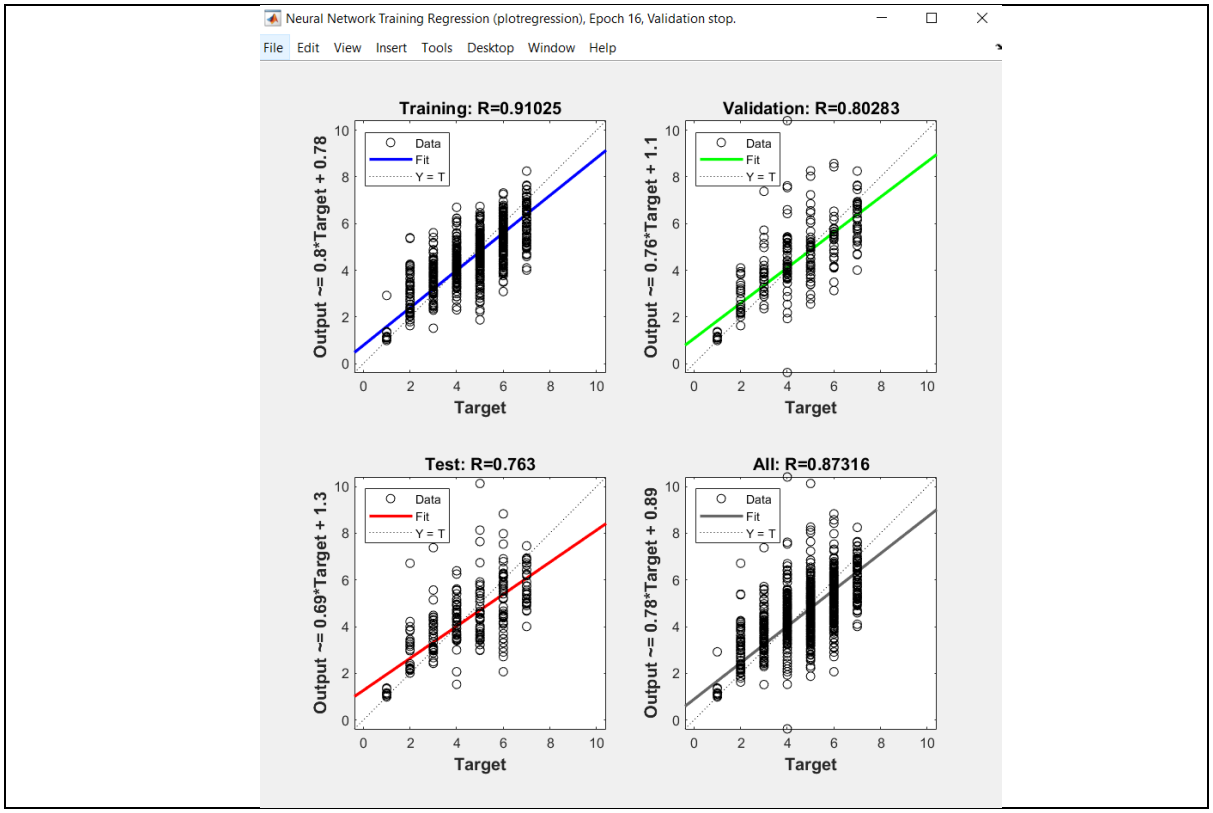
We do the same procedure for valance data set and the best architecture is [60 20]

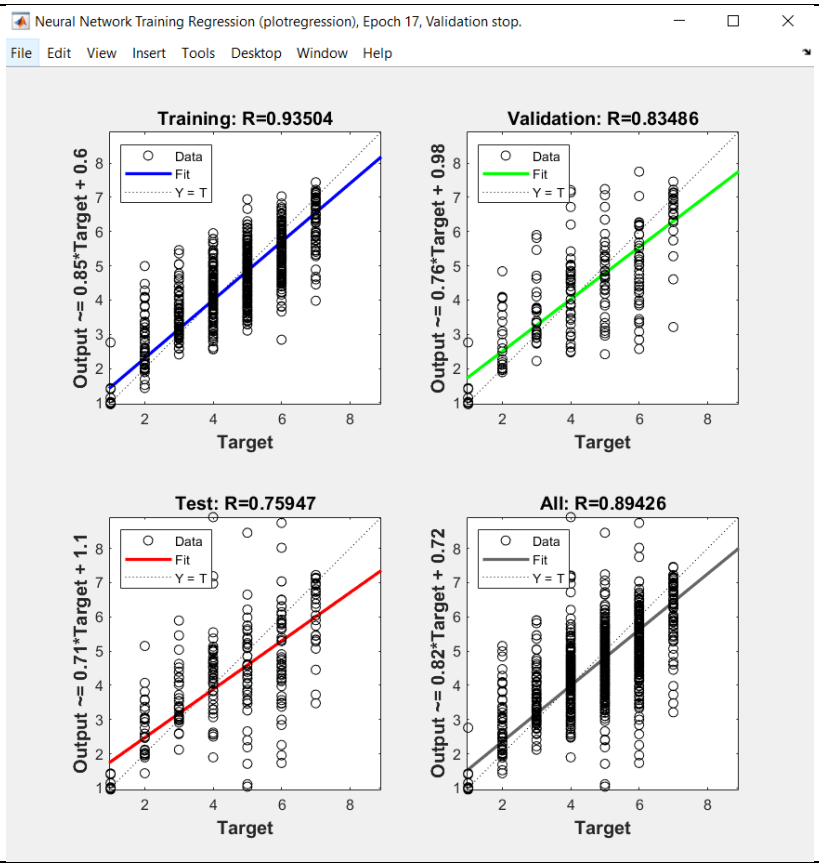
1.3.4 Test the best architecture for MLP for Arousal

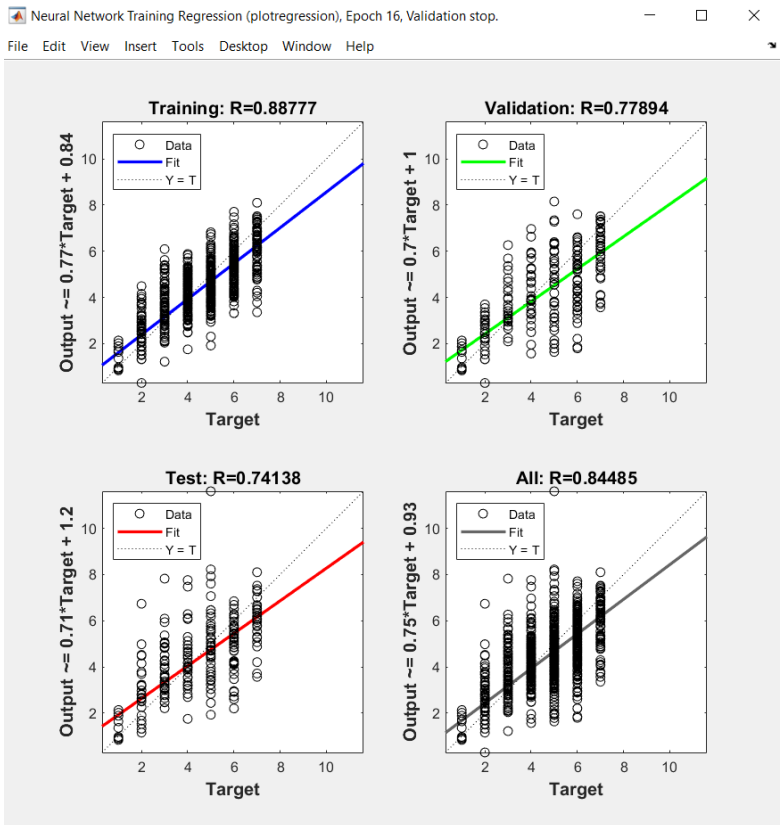
The network structure is as shown in fallowing figure:

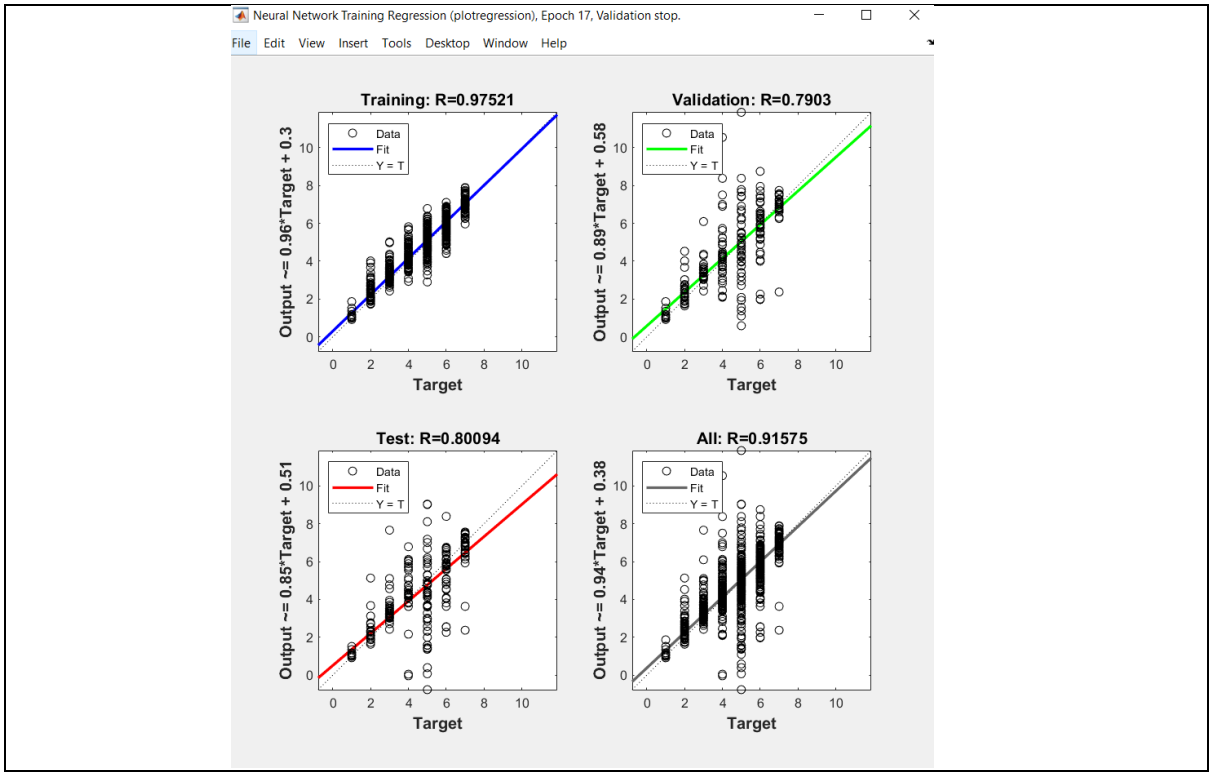


The regression plot for training and test data for each fold of 4-fold cross validation is shown in fallowing figures:



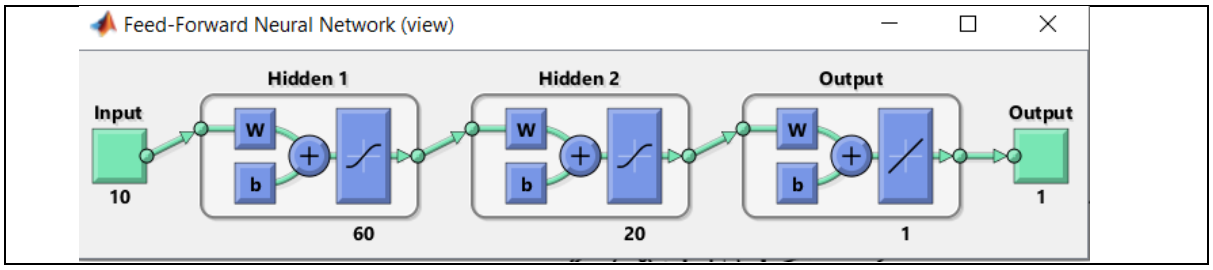

Neural Network Training Regression (plotregression), Epoch 17, Validation stop.
File Edit View Insert Tools Desktop Window Help
Training: R=0.93504
Output ~= 0.85*Target + 0.6
Validation: R=0.83486
Output ~= 0.76*Target + 0.98
Test: R=0.75947
Output ~= 0.71*Target + 1.1
All: R=0.89426
Output ~= 0.82*Target + 0.72
Data
Fit
Y = T
Target


Neural Network Training Regression (plotregression), Epoch 16, Validation stop.
File Edit View Insert Tools Desktop Window Help
Training: R=0.88777
Output ~= 0.77*Target + 0.84
Validation: R=0.77894
Output ~= 0.7*Target + 1
Test: R=0.74138
Output ~= 0.71*Target + 1.2
All: R=0.84485
Output ~= 0.75*Target + 0.93
Data
Fit
Y = T
Target

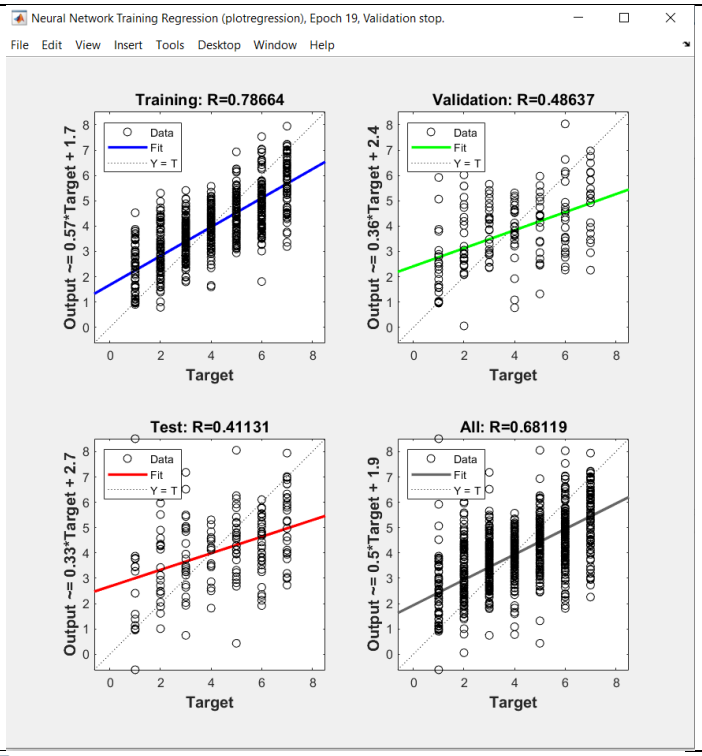


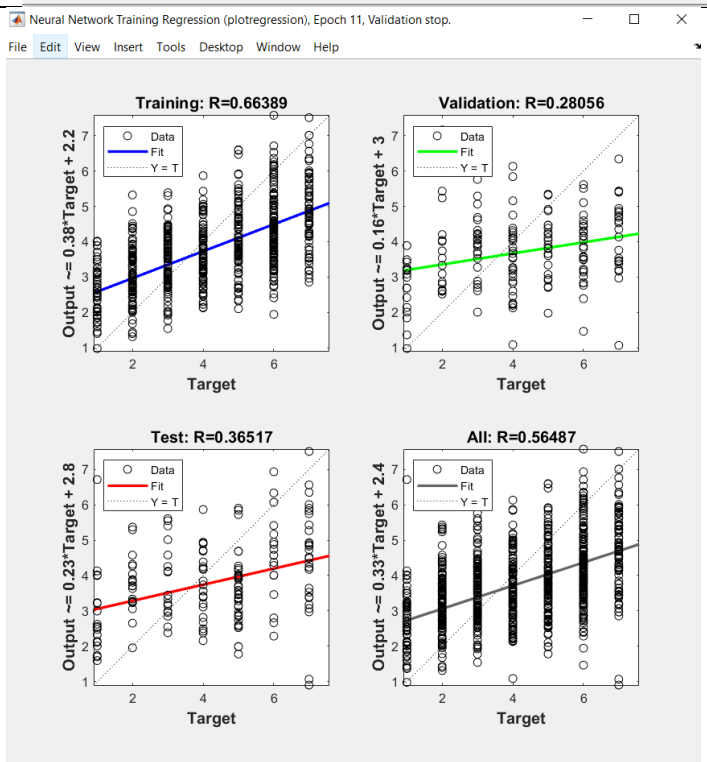
1.3.5 Test the best architecture for MLP for valance

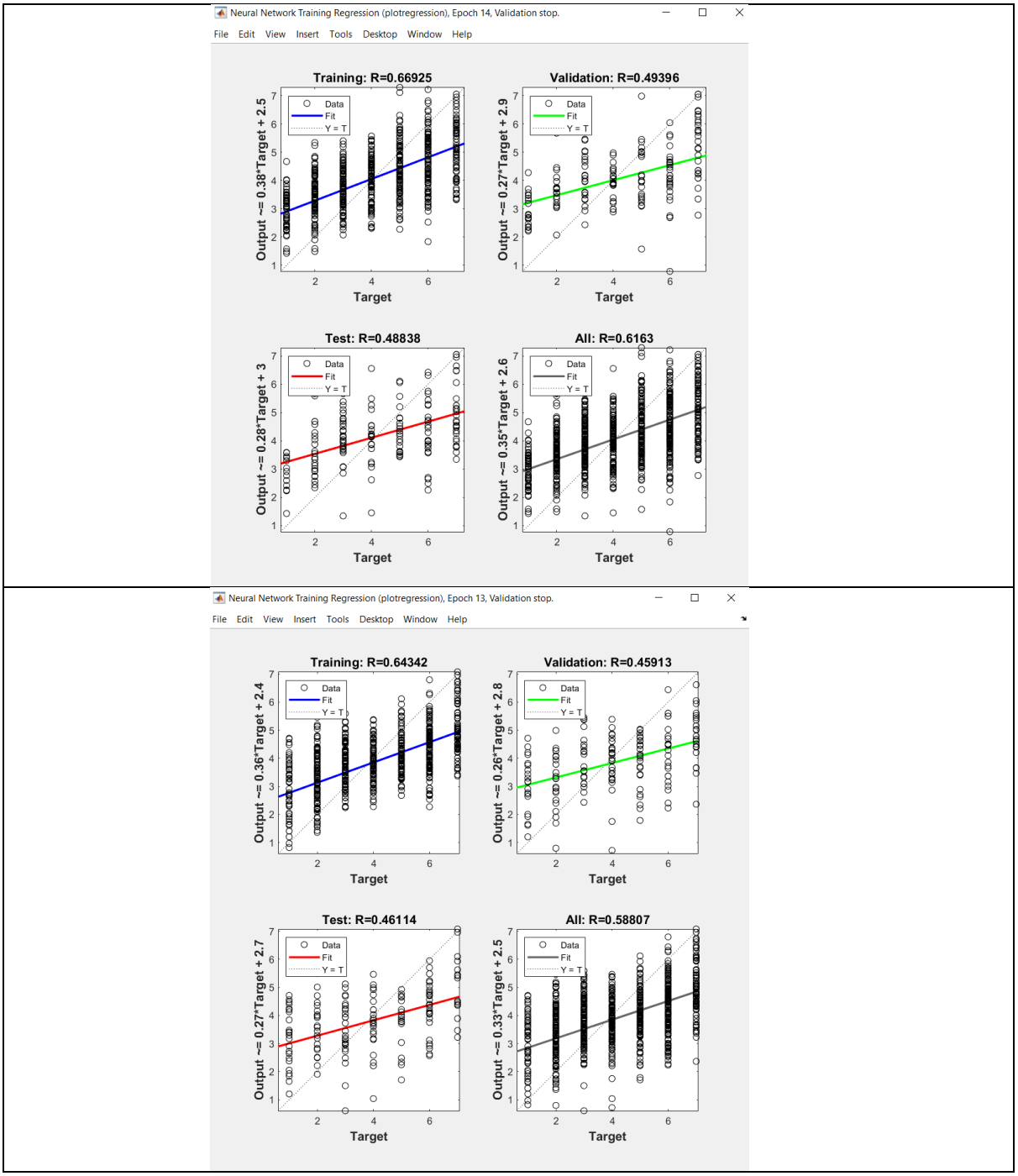
The network structure is as shown in fallowing figure:



The regression plot for training and test data for each fold of 4-fold cross validation is shown in fallowing figures:


Neural Network Training Regression (plotregression), Epoch 19, Validation stop.
File Edit View Insert Tools Desktop Window Help
Training: R=0.78664
Validation: R=0.48637
Test: R=0.41131
All: R=0.68119
Data
Fit
Y = T
Output ~= 0.57*Target + 1.7
Output ~= 0.36*Target + 2.4
Output ~= 0.33*Target + 2.7
Output ~= 0.5*Target + 1.9
Target


Neural Network Training Regression (plotregression), Epoch 11, Validation stop.
File Edit View Insert Tools Desktop Window Help
Training: R=0.66389
Validation: R=0.28056
Test: R=0.36517
All: R=0.56487
Data
Fit
Y = T
Output ~= 0.38*Target + 2.2
Output ~= 0.16*Target + 3
Output ~= 0.23*Target + 2.8
Output ~= 0.33*Target + 2.4
Target



1.3.6 Design and Train Radial Basis Function Neural Network for the Problem:

The last step of this part is to design and train two radial basis function (RBF) networks that do the same thing as the previously developed MLPs.

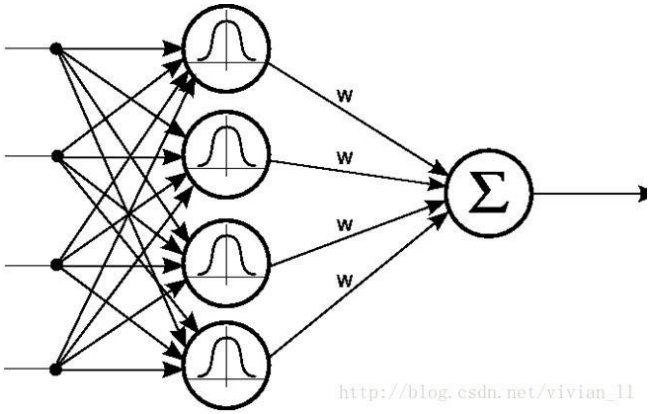


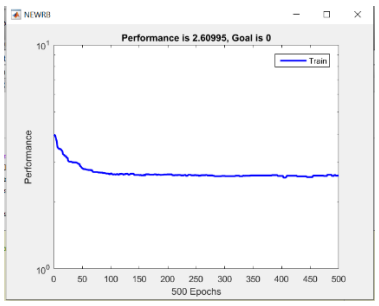
Figure 1-6:RBF ANN

Ref: https://www.programmersought.com/article/28352160688/

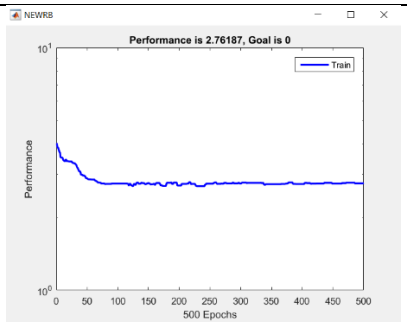
I have used MATLAB function newrb for RBF ANN. i have changed the number of neurons and the spread in order to obtain the best architecture for this ANN. Then report the best architecture accuracy.

1.3.7 Test the best architecture for RBF

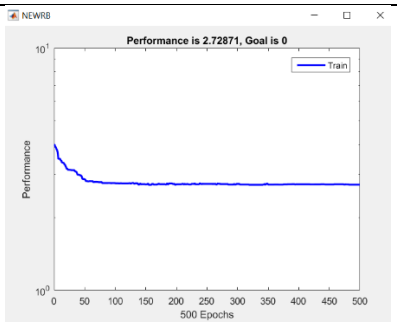
We have tested different values for Spread and report the performance in Following figures:



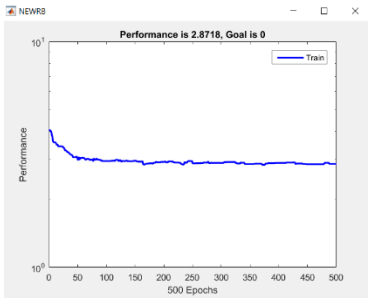
Spread = 0.01



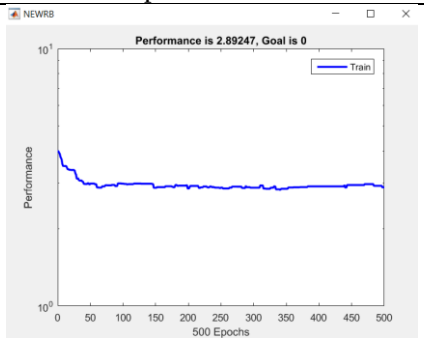
Spread = 0.02



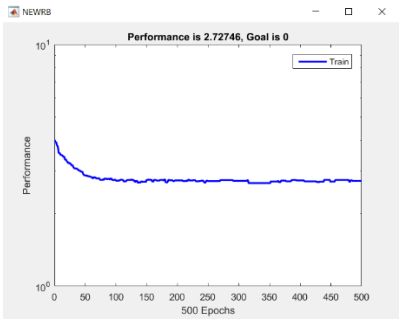
Spread = 0.03



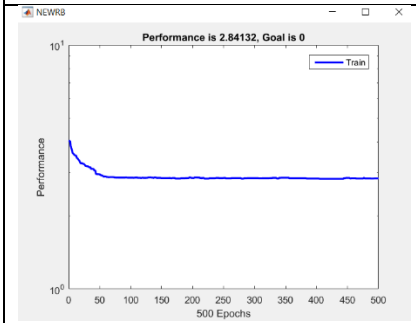
Spread = 0.04



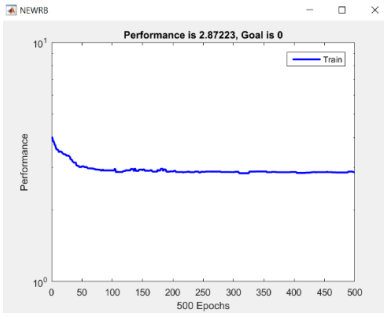
Spread = 0.05



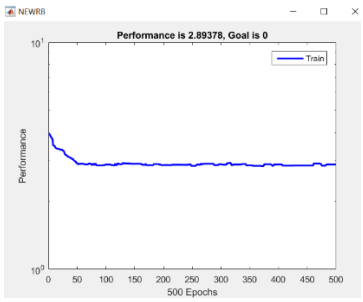
Spread = 0.15



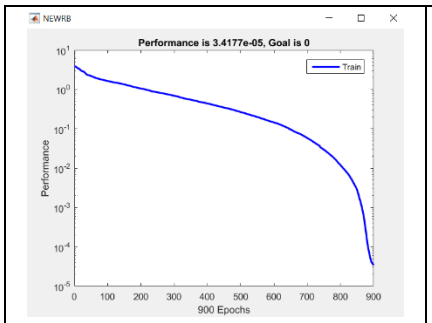
Spread = 0.25



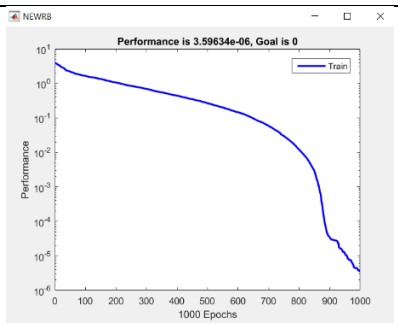
Spread = 0.35



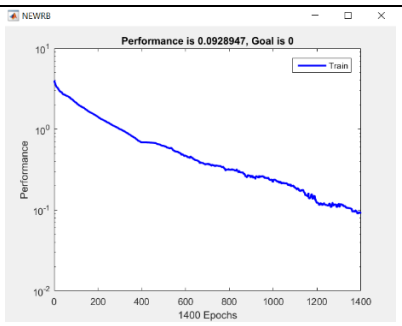
Spread = 0.45



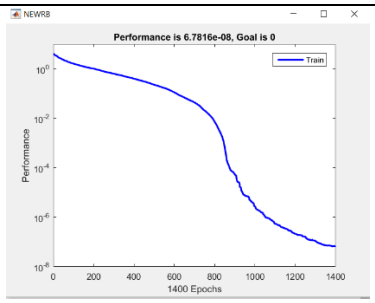
Spread = 0.001



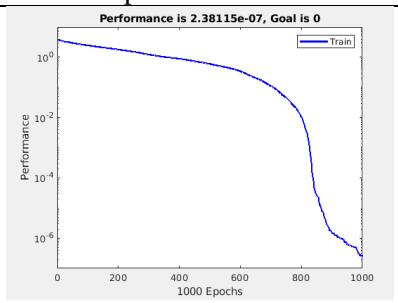
Spread = 0.003



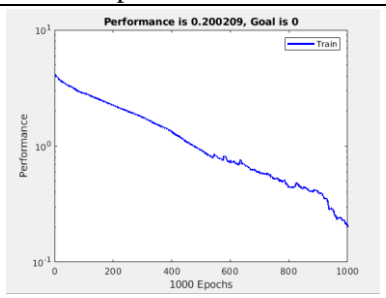
Spread = 0.004



Spread = 0.001
For Valance



Spread = 0.003
For Valance



Spread = 0.004
For Valance

The regression plot for Arousal and valance Values are depicted in the Following Figures:

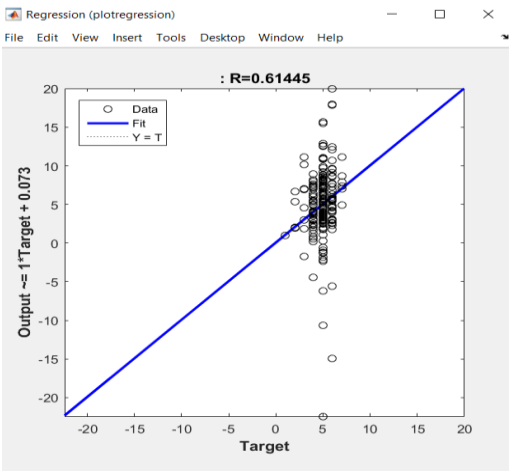


Figure 1-7 regression 0.001 spread for Arousal data in Best Architecture For RBF

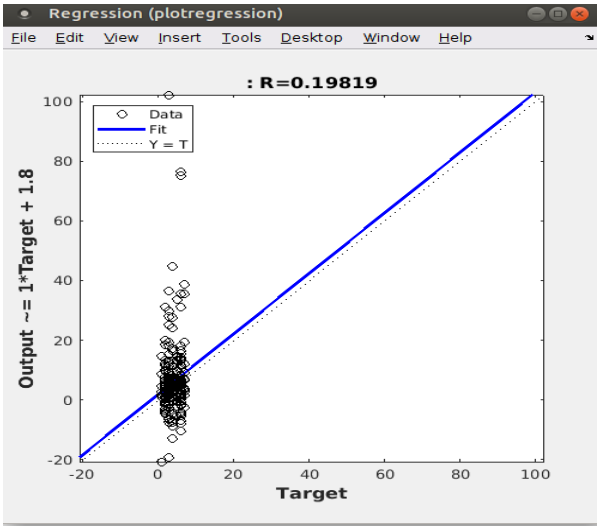


Figure 1-8 regression 0.001 spread for Valance data in Best Architecture for RBF

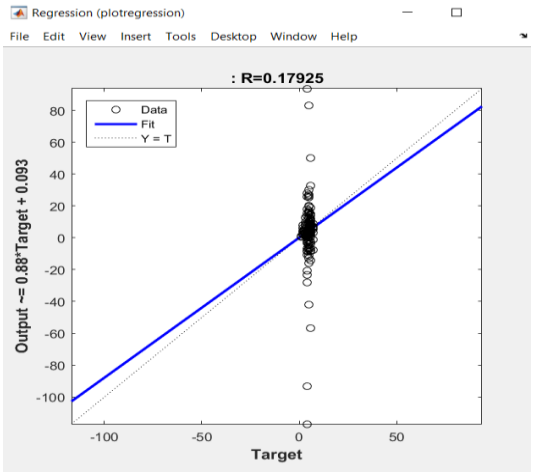


Figure 1-9 regression 0.004 spread for Arousal data in Best Architecture for RBF

1.3.8 MATLAB Help on newrb

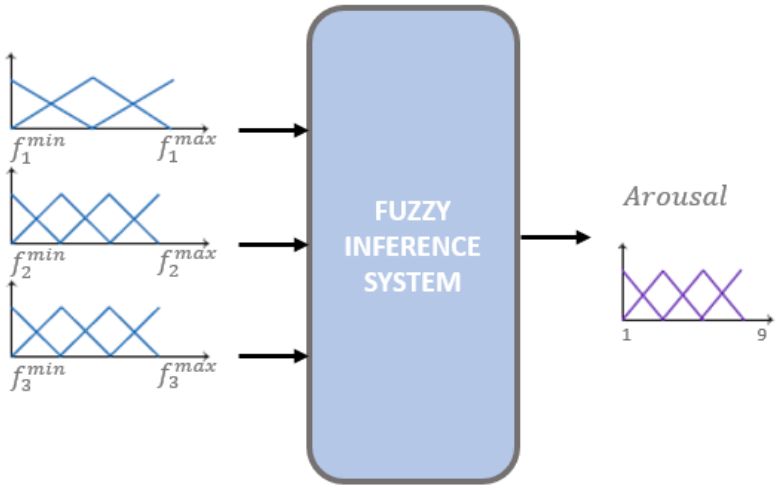
Radial basis networks can be used to approximate functions. newrb adds neurons to the hidden layer of a radial basis network until it meets the specified mean squared error goal.
net = newrb(P,T,goal,spread,MN,DF) takes two of these arguments,

P	R-by-Q matrix of Q input vectors
T	S-by-Q matrix of Q target class vectors
goal	Mean squared error goal (default = 0.0)
spread	Spread of radial basis functions (default = 1.0)
MN	Maximum number of neurons (default is Q)
DF	Number of neurons to add between displays (default = 25)

and returns a new radial basis network.
The larger spread is, the smoother the function approximation. Too large a spread means a lot of neurons are required to fit a fast-changing function. Too small a spread means many neurons are required to fit a smooth function, and the network might not generalize well. Call newrb with different spreads to find the best value for a given problem.

1.4 Task 3.3: Mamdani-type fuzzy inference system

One of the major problems of the model described in Section 1 is that users are not familiar with the concepts of valence and arousal. So it is very common for people to feel a certain emotion but they are unable to correctly evaluate it using valence and arousal levels. For example, feeling fear and attributing valence and arousal levels belonging to the first quadrant of the plane means poor understanding of the model.
The aim of this part is to design and develop a fuzzy inference system to fix the deficiencies in the arousal dimension.



The three inputs of the fuzzy system to design stem from an appropriate linguistic modeling of three features. The three features to model are the most relevant for estimating the arousal (i.e., the three most relevant features selected in Section 3.1). Linguistic modeling of inputs must be performed using appropriate sets of membership functions. Pay particular attention when designing linguistic modeling of features by analyzing the samples that are misclassified by the MLP, trying to fix the problem. The output of the system is a linguistic variable that expresses the arousal based

on the values of the three input features. The linguistic modeling of the output must be performed by means of appropriate membership functions.

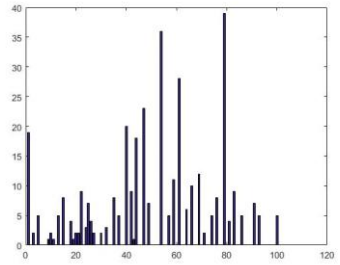
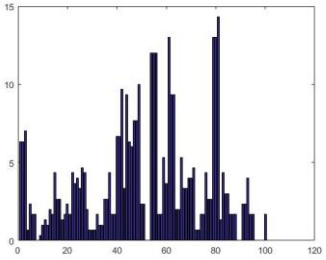
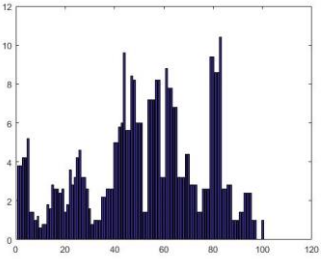
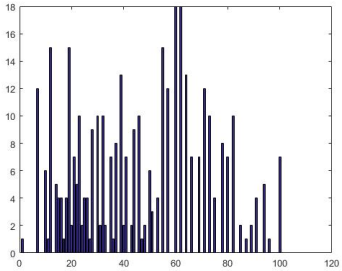
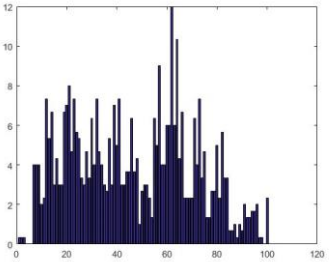
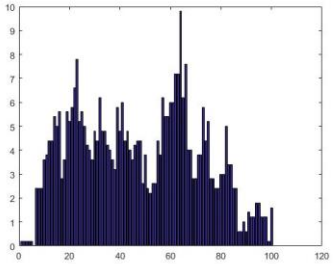
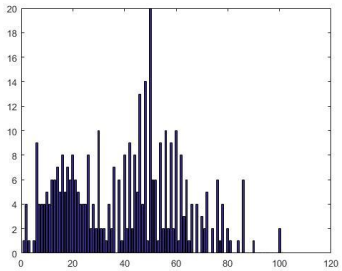
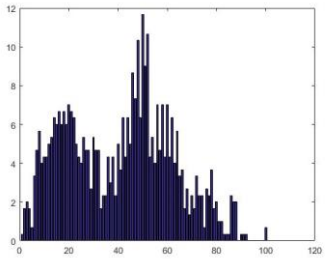
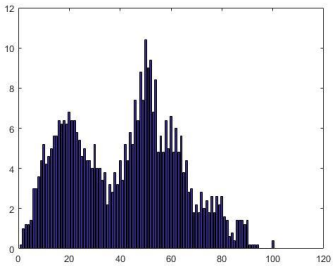
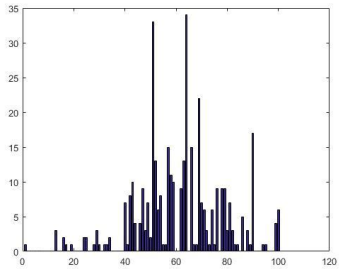
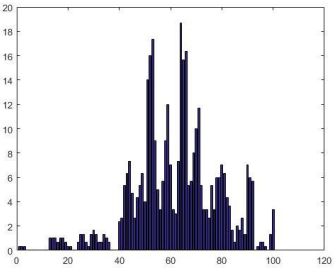
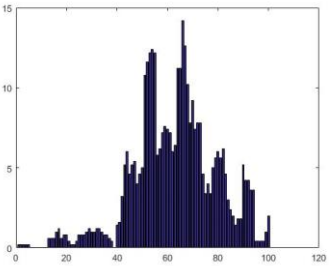
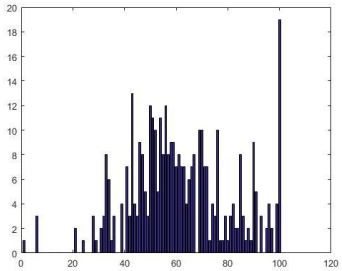
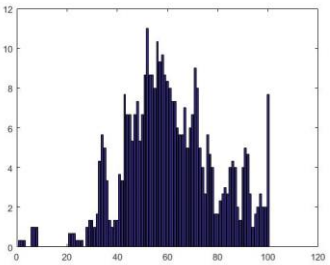
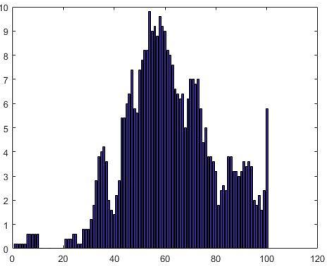
1.4.1 Computing the histogram of each three features for each large class

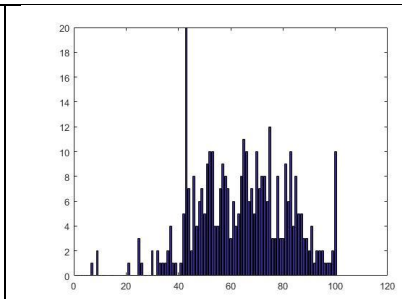
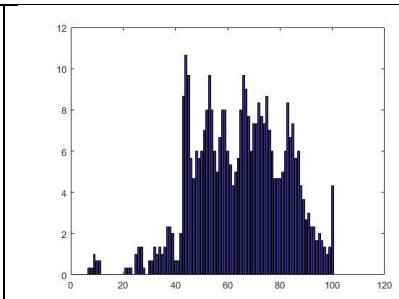
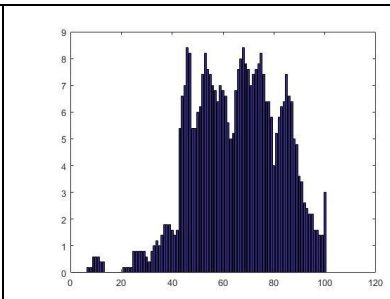
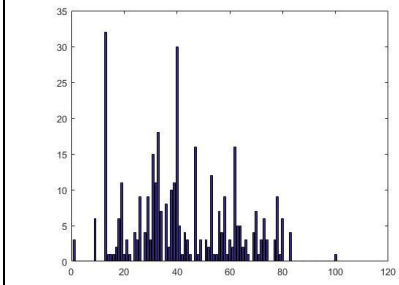
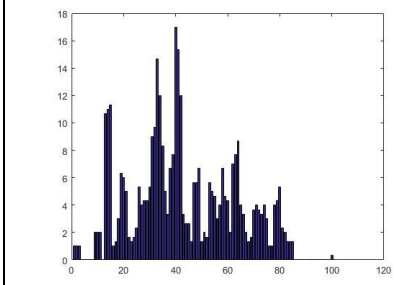
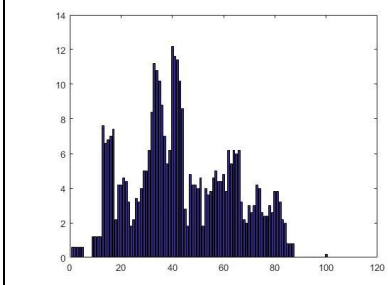
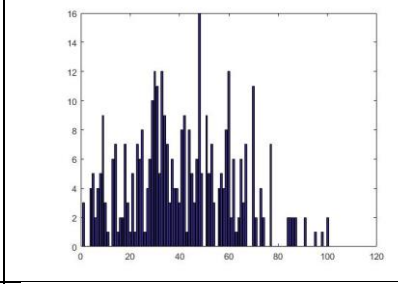
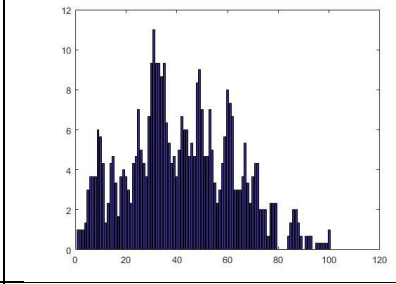
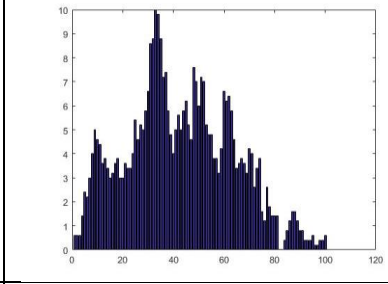
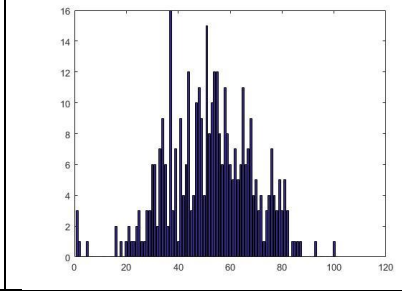
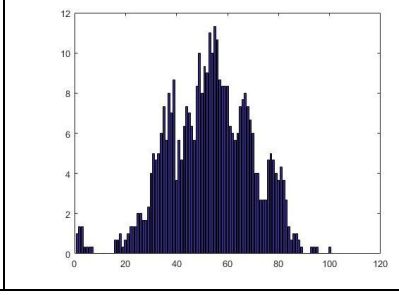
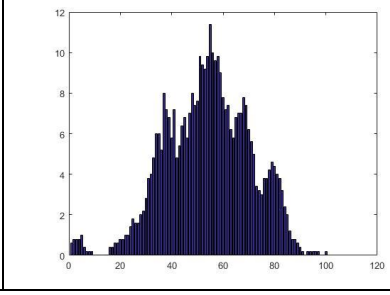
In order to simplify the problem and also reducing the fuzzy rules, we divided the arousal space into the three classes: positive-arousal, negative-arousal and zero-arousal. Then we obtain the membership functions and define rules.

The best features obtained from the previous simulations was 24, 29, 14.

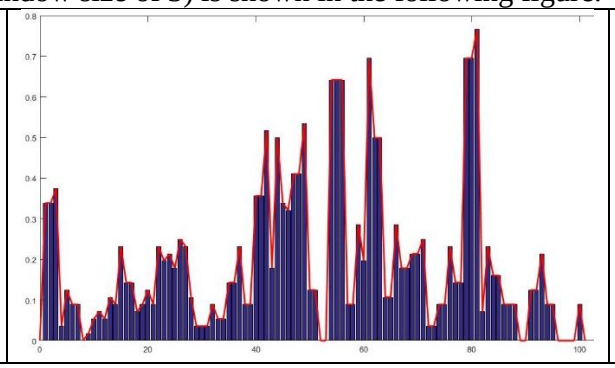
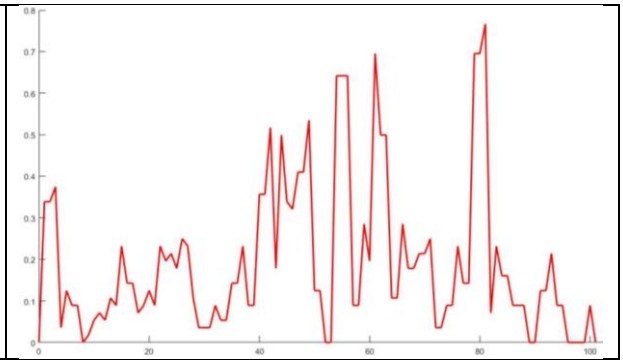
We plot the histograms of each of the three features for each class label. Therefore, we obtain 9 histograms, where are shown in the column 1 of the following figure.

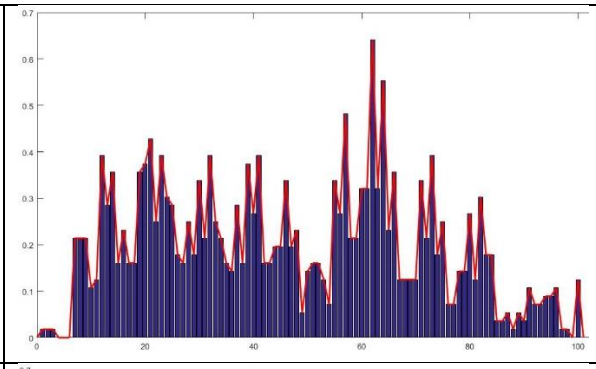
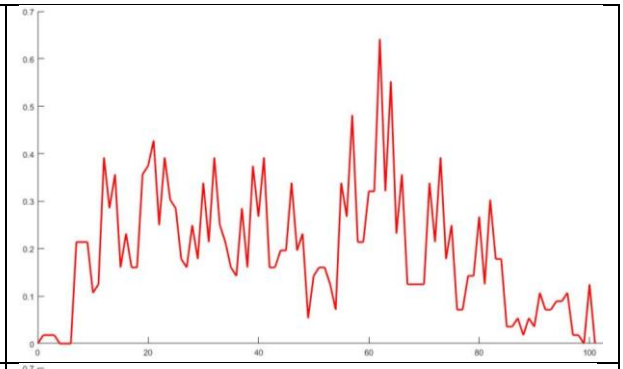
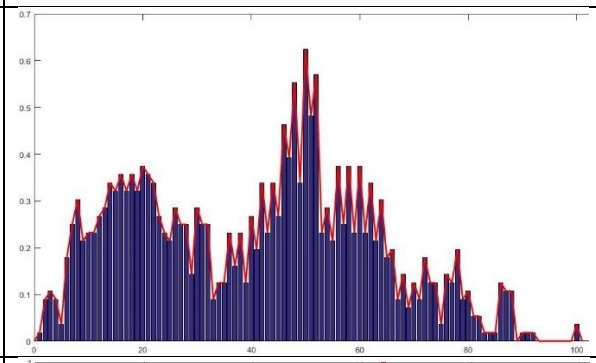
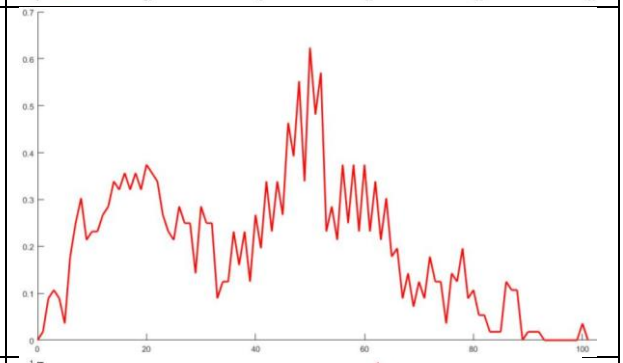
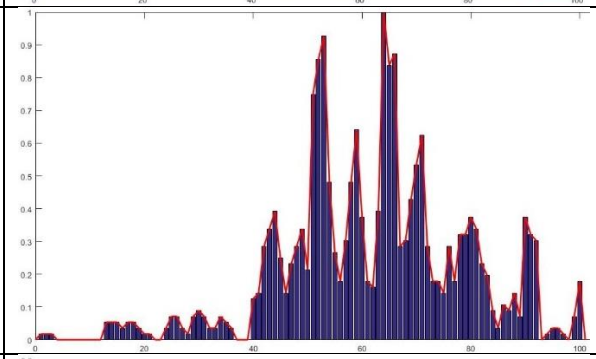
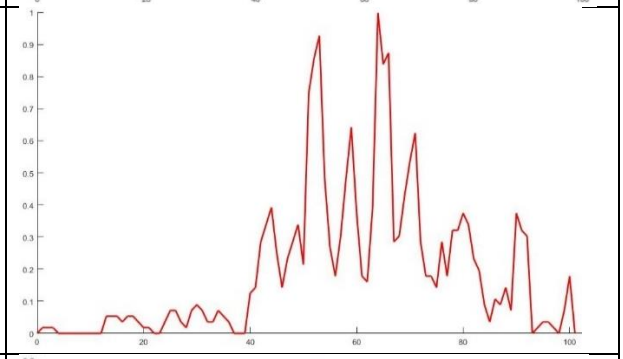
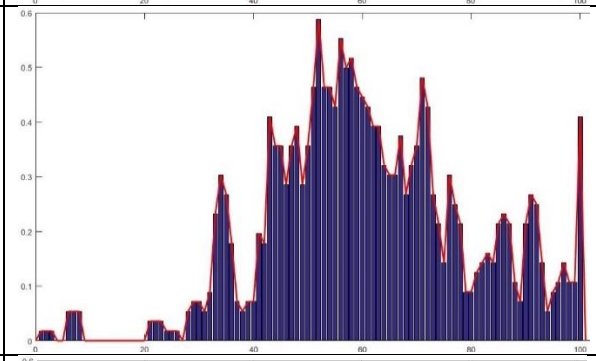
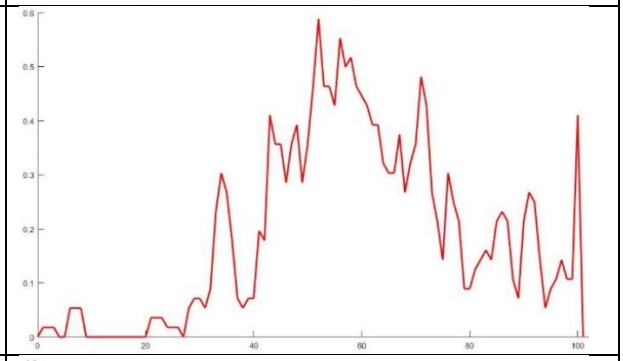
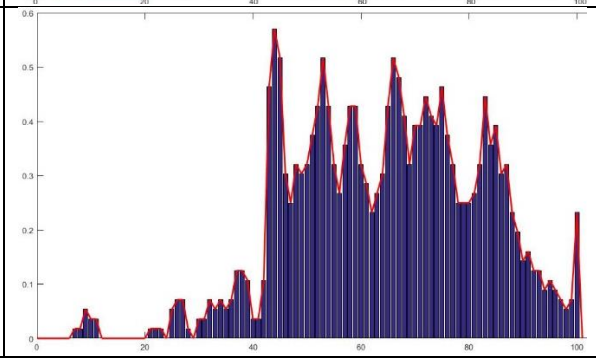
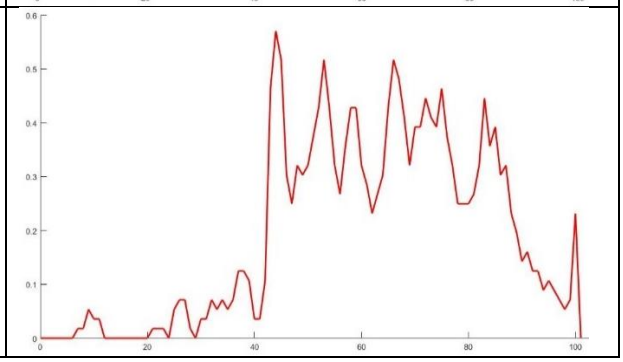
The histogram has many fluctuations; therefore, we slide an average filter with the window size of 3 and 5, the window size of 5 results in better and smoother bar graph. Column 2 and column 3 of the following figure, shows the result of filtering with window size of 3 and 5 respectively.

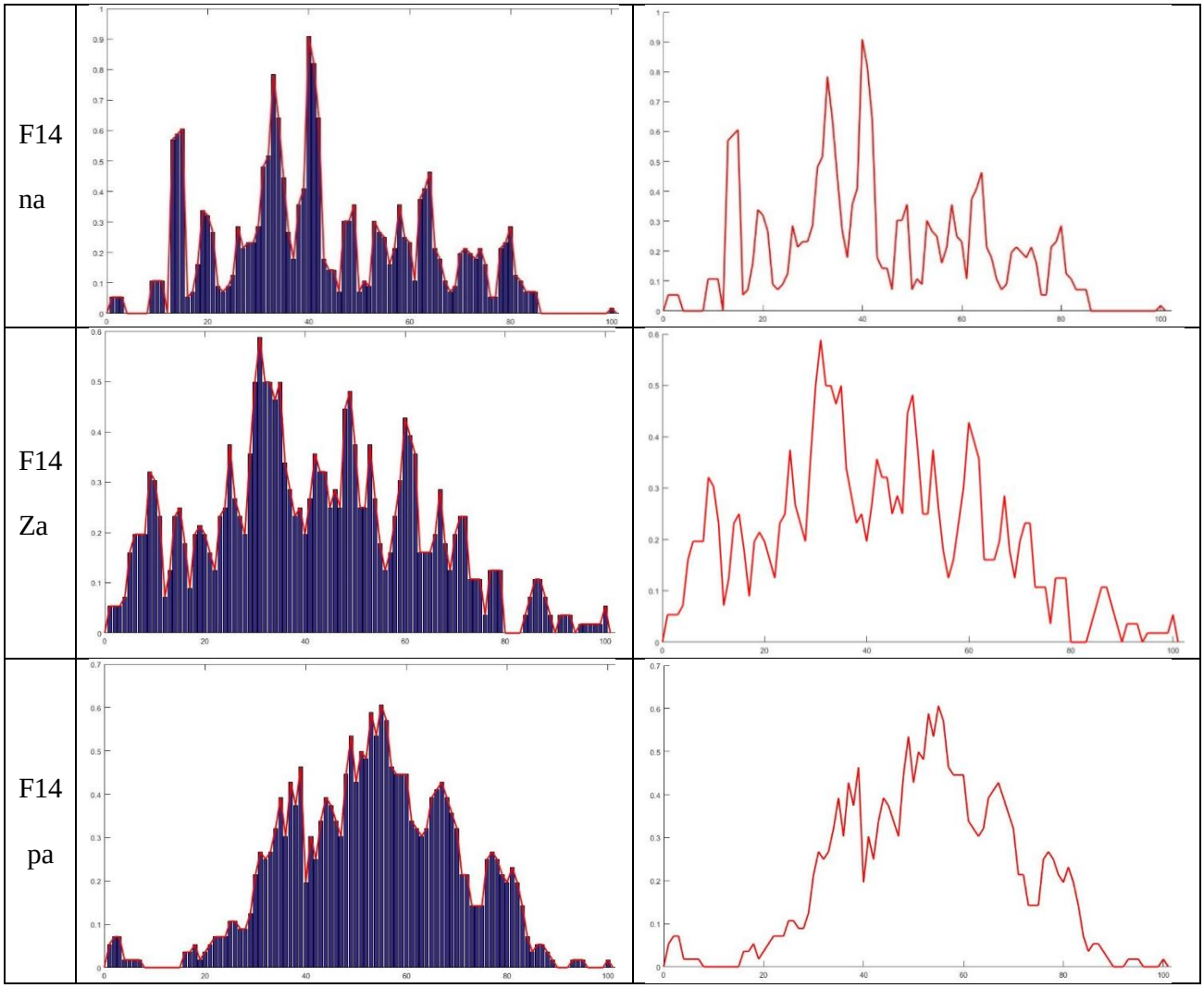
	histogram	Filtered (w = 3)	Filtered (w = 5)
F24 na			
F24 za			
F24 pa			
F29 Na			
F29 Za			

F29 Pa			
F14 na			
F14 Za			
F14 pa			

We have used the histogram filtered with window size of 3. The maximum value of this histogram in all 9 histograms is 18.7. In order to use the histograms as fuzzy membership functions we should scale the amplitude between [0 1]. Therefore, we divide all histogram values by 18.7.
Then we draw lines from center to center of each histogram values, and compute the line equations. Then we use these line equations as the fuzzy membership functions, and use them in the inference engine of the Mamdani fuzzy system.
The membership functions obtained from the histogram (which is filtered by average filter with window size of 3) is shown in the following figure.

F24 na		

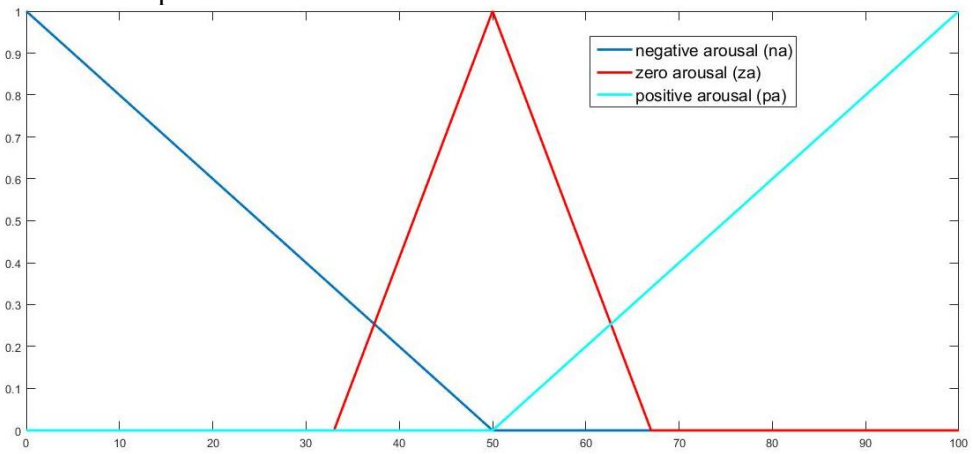
F24 za		
F24 pa		
F29 Na		
F29 Za		
F29 Pa		



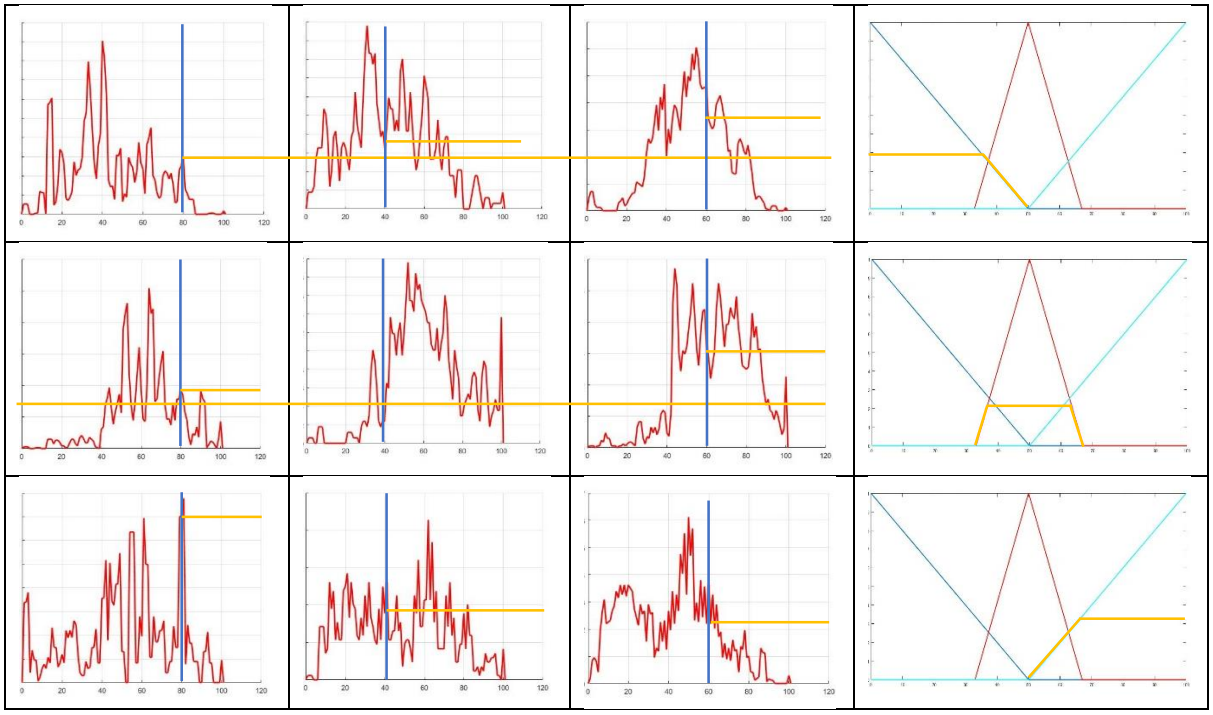
The number of fuzzy rules are three rules as follows:
Suppose we have a point with $data_{in} = (x24, x29, x14)$ feature values,

1- $If\ x24\ \in F24_{na} and\ x29 \in F29_{na} and\ x14 \in\ F14_{na} then\ data_{in} \in na$
2- $If\ x24\ \in F24_{za} and\ x29 \in F29_{za} and\ x14 \in\ F14_{za} then\ data_{in} \in za$
3- $If\ x24\ \in F24_{pa} and\ x29 \in F29_{pa} and\ x14 \in\ F14_{pa} then\ data_{in} \in pa$

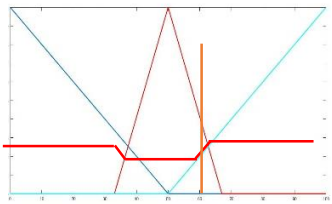
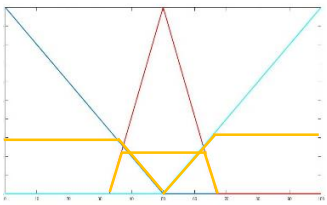
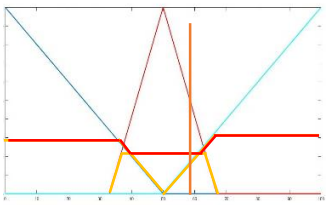
The output membership functions are as follows:



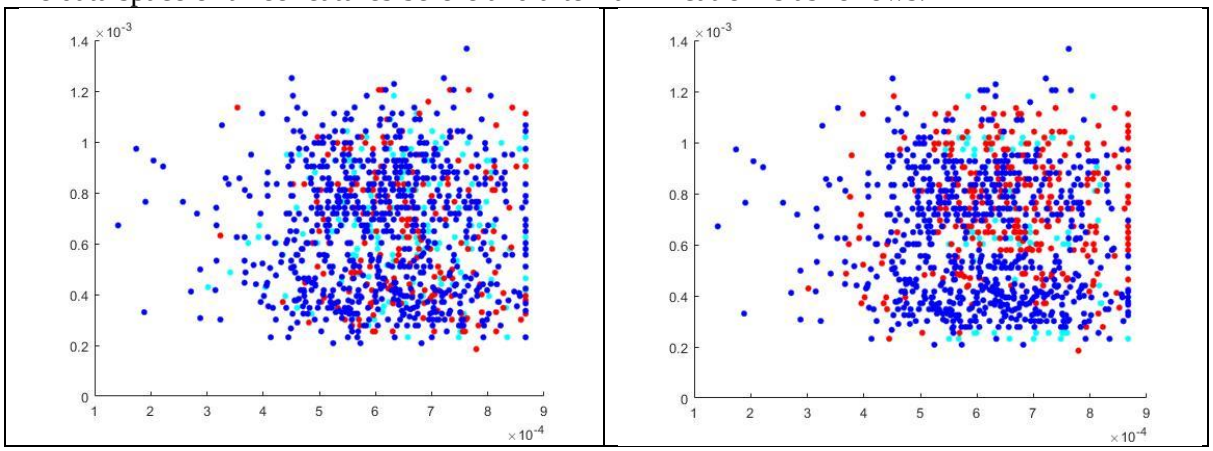
We have used minimum operator and product operator as a t-norm operator in our code.
We have used max operation as s-norm operator in our code.
In order to defuzzify the output we have used the maximum, and also center of area in our code.
The final result is achieved by considering min as t-norm, max as s-norm and center of area as defuzzificaton. The procedure of inference is shown in the following figure:

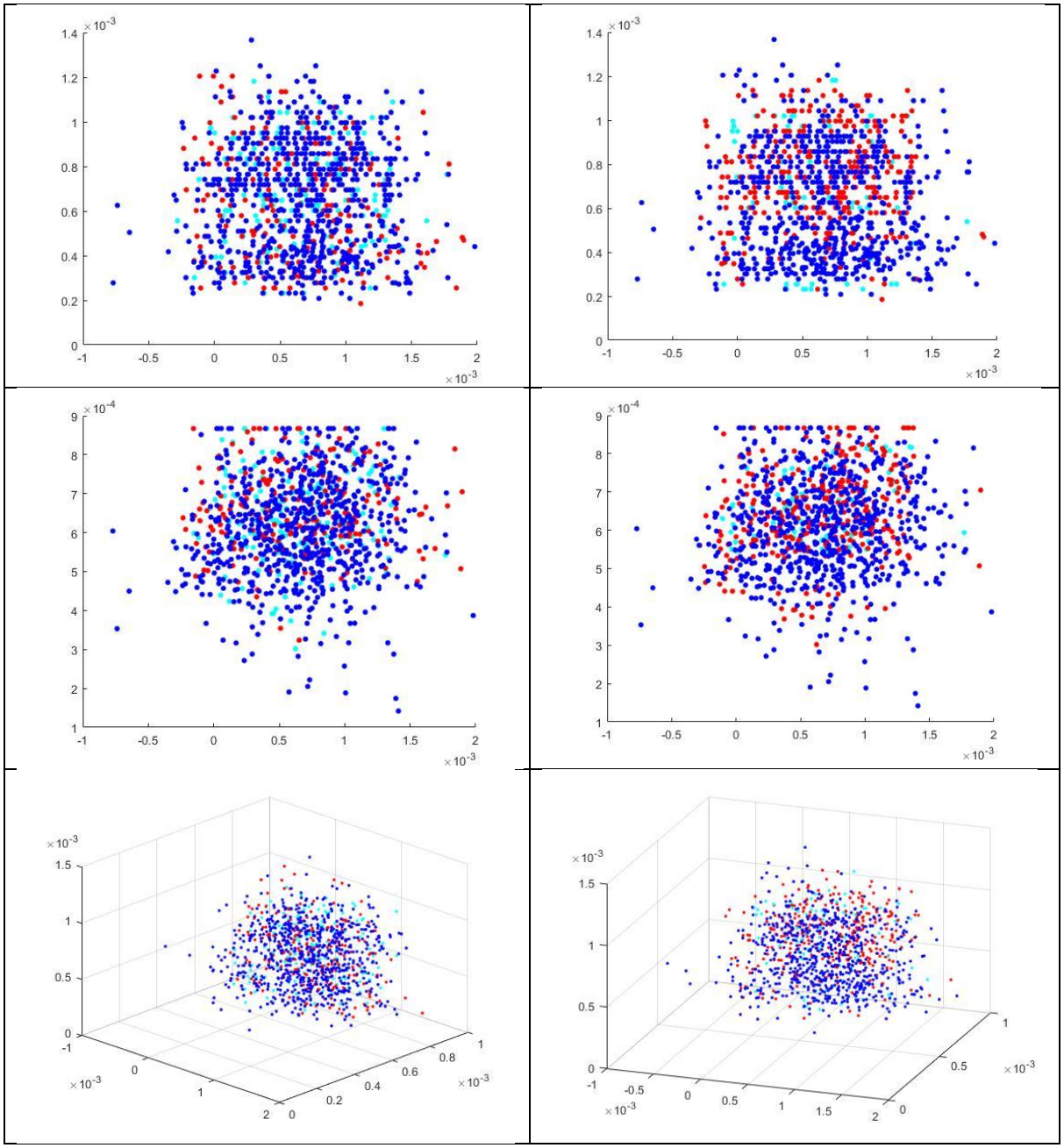


The defuzzification methods which we write the code for them is shown in the following figure:

Method 1 (na, 0.2) (za, 0.14) (pa, 0.23) Defuzzification(max)=>class=pa	Method 2 (na, 0.2) (za, 0.14) (pa, 0.23) P = 0.2*0 + 50*0.14 + 100*0.23 /(0.2+0.14+0.23) = 61  ⇨ Class = za ⇨ (na,0), (za,0.4),(pa,0.3)	  Method 3, center of area ⇨ Class = za ⇨ (na,0),(za,0.6),(pa,0.2)

The data space of three features before and after fuzzification is as follows:



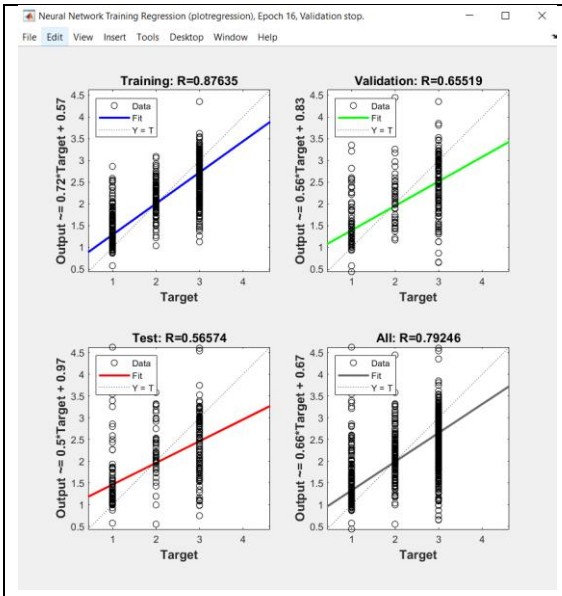


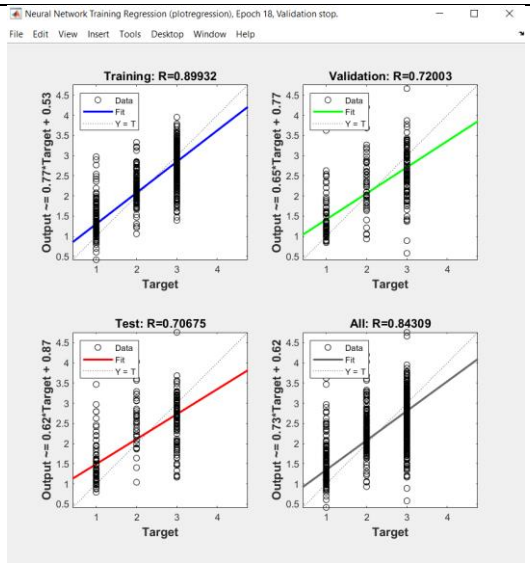
1.5 Task 3.4: Improving MLP's performance using the fuzzy inference system

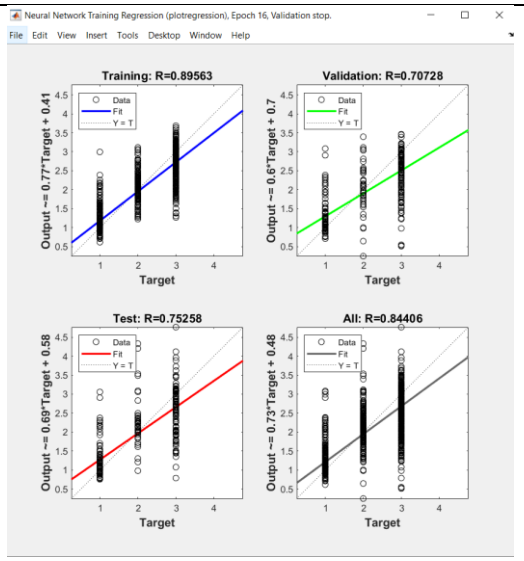
Use the output of the fuzzy inference system to train the MLP described in Section 3.1 to estimate the arousal level. This should be done with the aim of improving the accuracy, thereby reducing the bias. In order to achieve this result, the output of the fuzzy inference system must first be defuzzified and then used to replace the desired arousal level, where needed.

We have used k-fold cross Validation and the result of each fold is shown in the one row of the following figure.
The first column is the result of MLP before applying the fuzzy Rules and the second column is the result of MLP after applying the fuzzy Rules.
The ANN MLP network has Two hidden layers with 60 and 30 hidden neurons respectively.

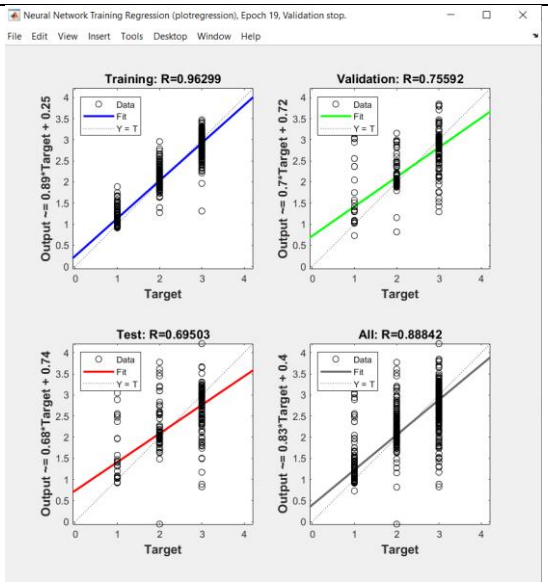
Before Fuzzy:

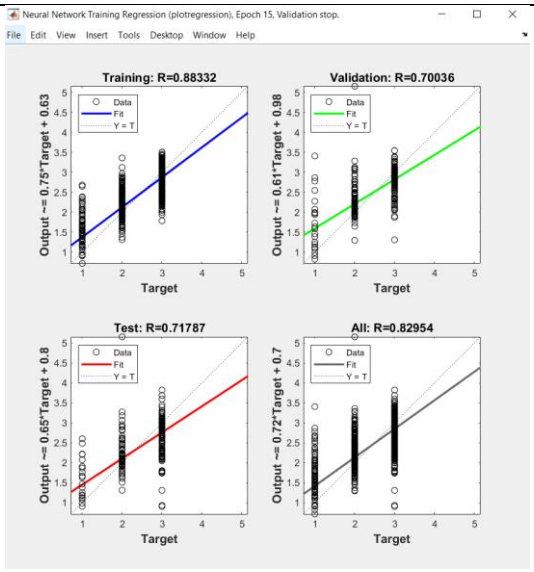


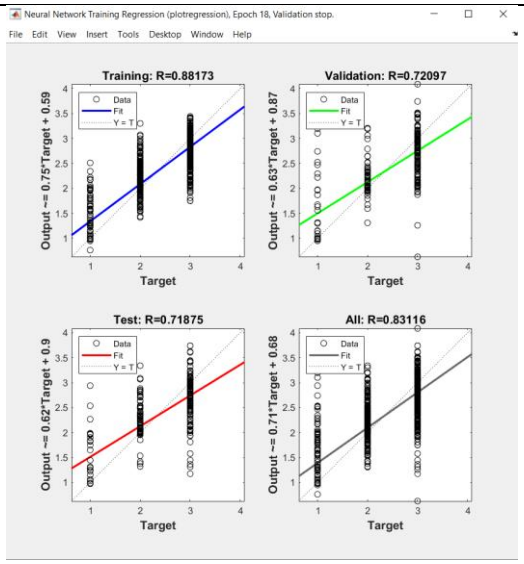


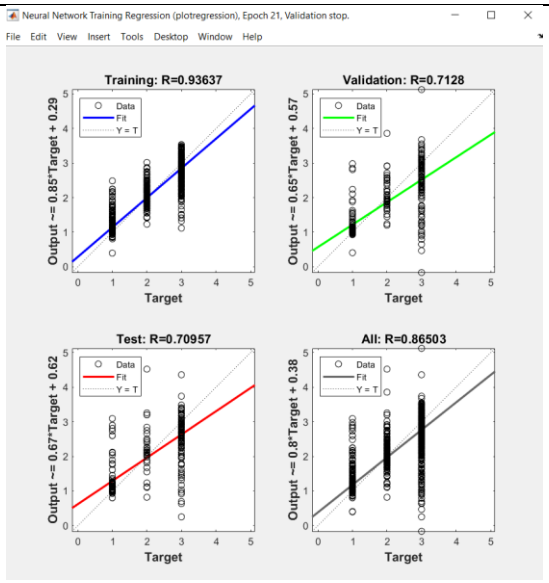


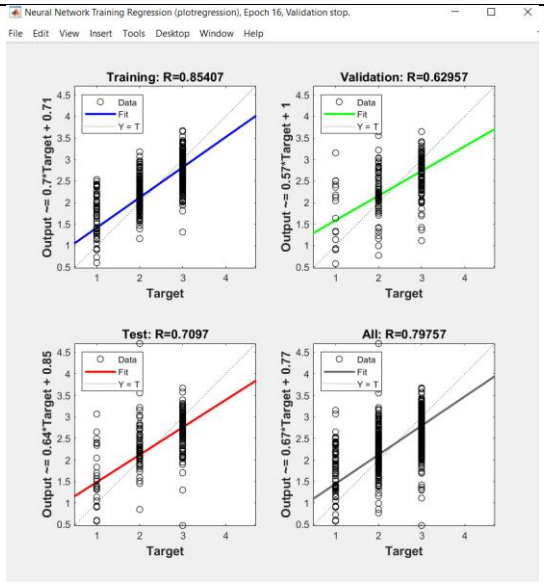
After Fuzzy:

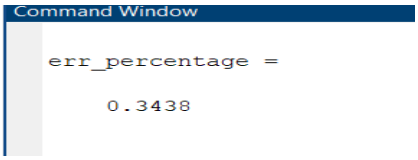


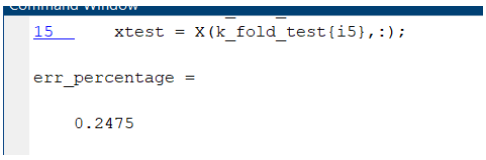












You can see from the Final Results We have improved the Error Percentage while we use Fuzzy Rules.

2 Second part of project:

The purpose of the this part is the classification of emotions on the basis of images depicting faces of people with different facial expressions related to a given emotion.

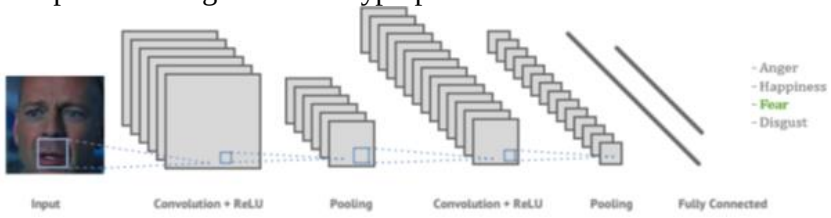
2.1 Dataset of images

This dataset contains photos that show faces of people expressing different emotions. As a result of a labeling process, each image is associated with one of the following emotions:
Anger, Happiness, Fear, Disgust.
Figure 2-1 shows four examples of images included in the dataset. All images have a resolution of *224x224* pixels, and are in *RGB* format. The images are in folders named with the labels of the emotions. In the *happiness* folder there are images depicting happy faces, in the *fear* folder there are images depicting frightened faces, and so on.
Note that it is not required to use the entire set of images, but rather it is recommended to select a representative set of images so that the number of images per emotion is balanced among the four emotions. 4
Since the training of a convolutional network is computationally demanding and can take a lot of time, it is suggested to start considering two sets of images labelled with different emotions (e.g., *happiness* and *disgust*) and train the network to classify the images into the two chosen emotions. The proposal of solutions for the classification in three and four classes will be highly appreciated.

2-1 Figure

2.2 Task 4.1: Classifying facial expressions with convolutional neural networks

The aim of this part of the project is to design and develop a convolutional neural network (CNN) that accurately classifies a person's emotion, based on facial expression. The CNN takes an image as input and returns a class that represents an emotion. The goal is to find the best architecture for the CNN with the optimal configuration of hyperparameters.



Using CNNs for deep learning is popular due to three important factors:

- CNNs eliminate the need for manual feature extraction—the features are learned directly by the CNN.

- CNNs produce highly accurate recognition results.
- CNNs can be retrained for new recognition tasks, enabling you to build on pre-existing networks.

CNNs provide an optimal architecture for uncovering and learning key features in image and time-series data. CNNs are a key technology in applications such as:

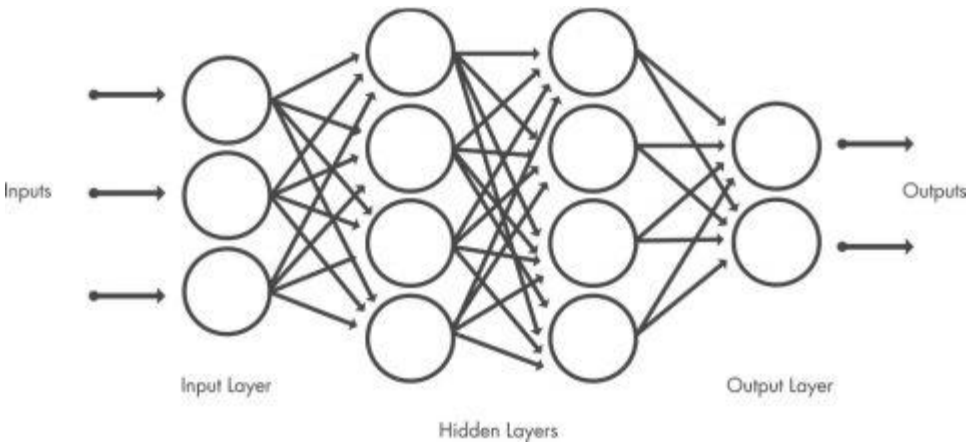
- **Medical Imaging**: CNNs can examine thousands of pathology reports to visually detect the presence or absence of cancer cells in images.
- **Audio Processing**: Keyword detection can be used in any device with a microphone to detect when a certain word or phrase is spoken - ('Hey Siri!'). CNNs can accurately learn and detect the keyword while ignoring all other phrases regardless of the environment.
- **Stop Sign Detection**: Automated driving relies on CNNs to accurately detect the presence of a sign or other object and make decisions based on the output.
- **Synthetic Data Generation**: Using Generative Adversarial Networks (GANs), new images can be produced for use in deep learning applications including face recognition and automated driving.

2.3 How CNNs Work

A convolutional neural network can have tens or hundreds of layers that each learn to detect different features of an image. Filters are applied to each training image at different resolutions, and the output of each convolved image is used as the input to the next layer. The filters can start as very simple features, such as brightness and edges, and increase in complexity to features that uniquely define the object.

2.4 Feature Learning, Layers, and Classification

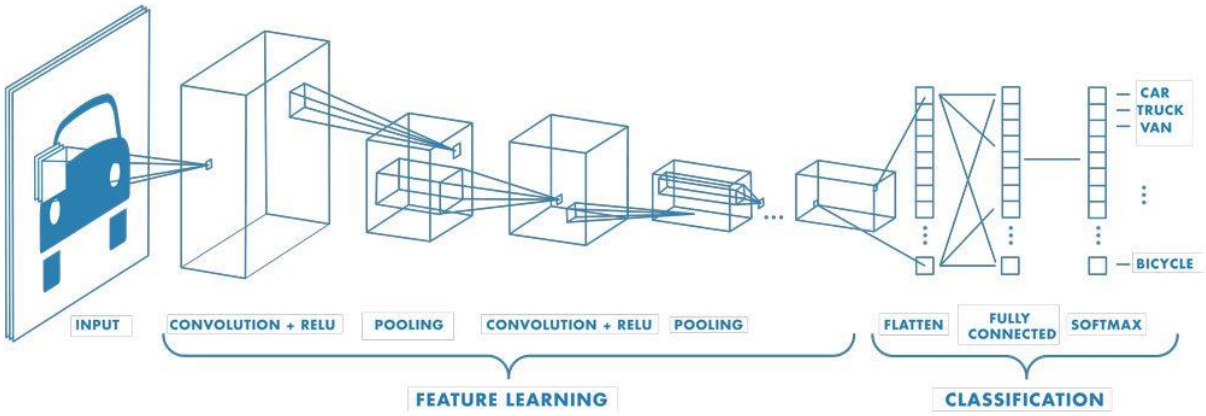
Like other neural networks, a CNN is composed of an input layer, an output layer, and many hidden layers in between.



These layers perform operations that alter the data with the intent of learning features specific to the data. Three of the most common layers are: convolution, activation or ReLU, and pooling.

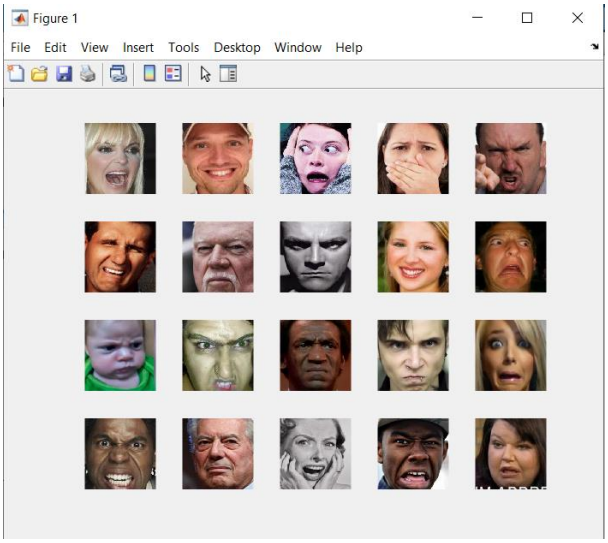
- **Convolution** puts the input images through a set of convolutional filters, each of which activates certain features from the images.
- **Rectified linear unit (ReLU)** allows for faster and more effective training by mapping negative values to zero and maintaining positive values. This is sometimes referred to as *activation*, because only the activated features are carried forward into the next layer.
- **Pooling** simplifies the output by performing nonlinear downsampling, reducing the number of parameters that the network needs to learn.

These operations are repeated over tens or hundreds of layers, with each layer learning to identify different features.



Example of a network with many convolutional layers. Filters are applied to each training image at different resolutions, and the output of each convolved image is used as the input to the next layer.

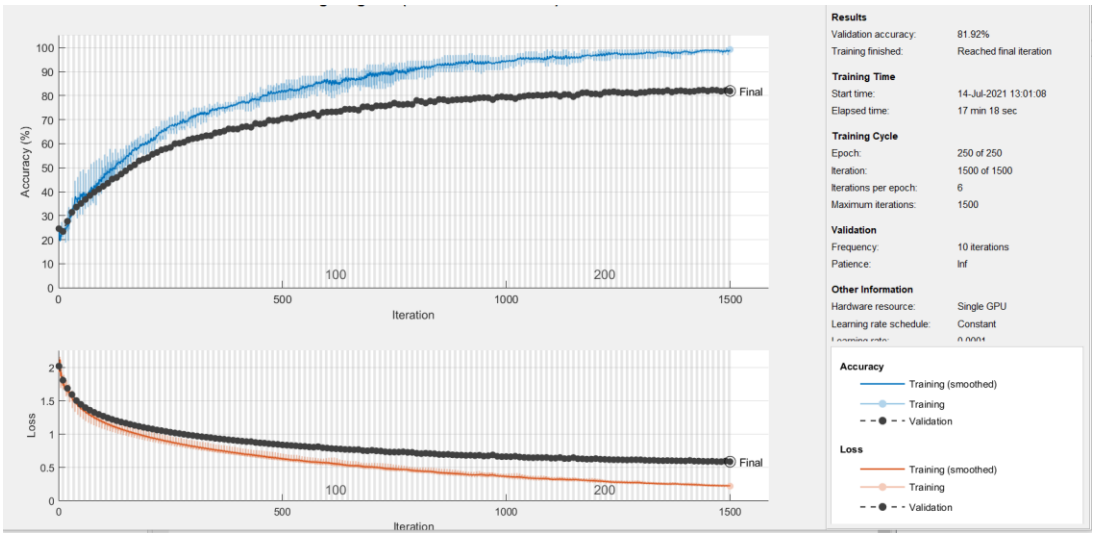
2.5 4 classes with Two layer CNN:



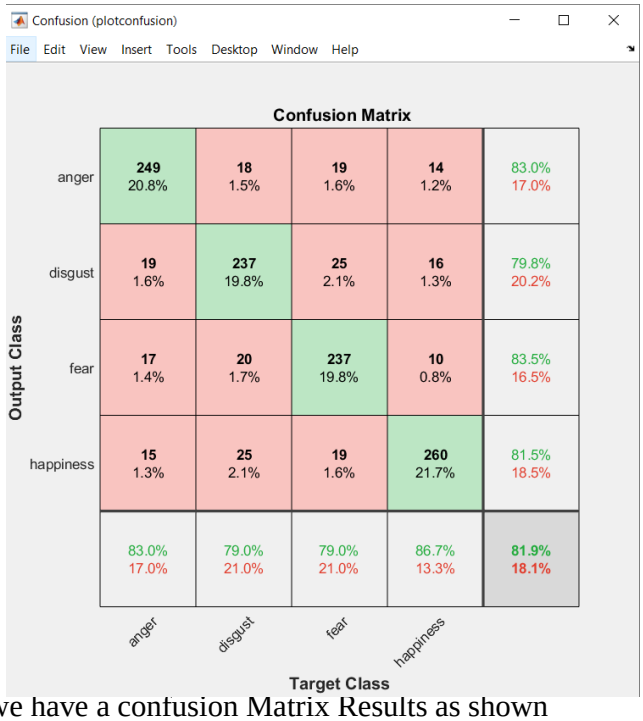
20 pictures that have been chosen randomly

```
layers = [ ...
    imageInputLayer([224 224 3])

%    convolution2dLayer(5,20)
    convolution2dLayer(20 ,6 , 'Stride' ,4,'Padding','same')
    batchNormalizationLayer

    reluLayer
%     leakyReluLayer

    maxPooling2dLayer(2,'Stride',2)
    %****************************************%
    convolution2dLayer(10 ,10 , 'Stride' ,2,'Padding','same')
    batchNormalizationLayer
    reluLayer

    maxPooling2dLayer(2, 'Stride' ,2)
    %****************************************%
%     convolution2dLayer(32 ,8 , 'Stride' ,2,'Padding','same')
%     batchNormalizationLayer
%     reluLayer
%
%     maxPooling2dLayer(2, 'Stride' ,2)
    %****************************************%
    fullyConnectedLayer(60)
    fullyConnectedLayer(4)

    softmaxLayer

    classificationLayer];
```

Configuration of our proposed CNN

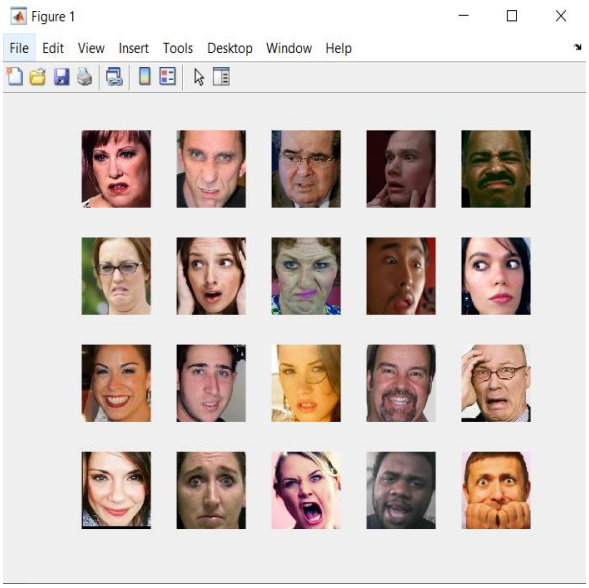


The result of our CNN network for 4 classes with Two layer CNN



As you can see we have a confusion Matrix Results as shown

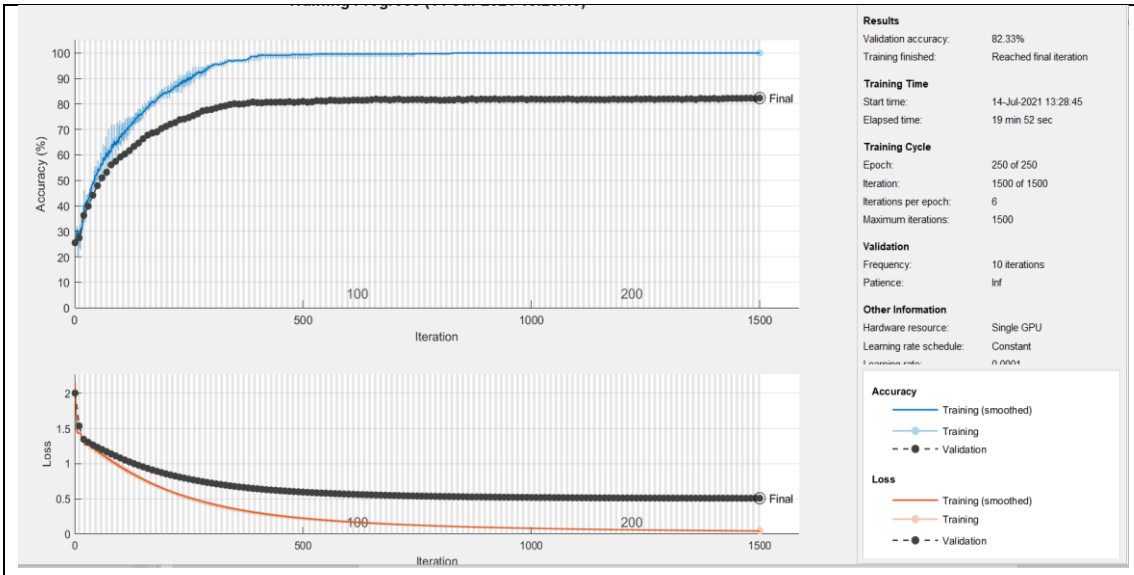
2.6 4 classes with Three layer CNN:



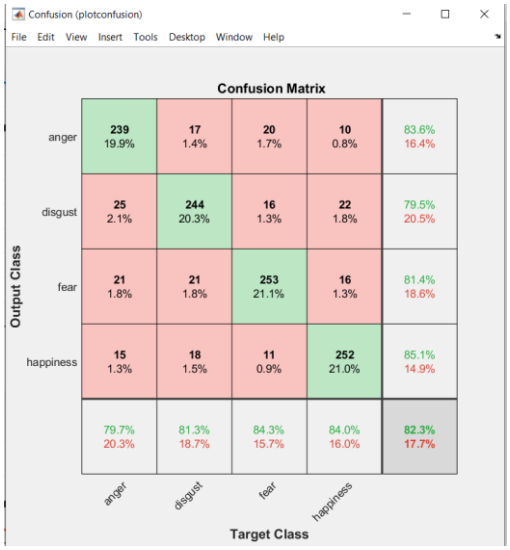
20 pictures that have been chosen randomly

```
layers = [ ...
    imageInputLayer([224 224 3])

%    convolution2dLayer(5,20)
    convolution2dLayer(20 ,6 , 'Stride' ,4,'Padding','same')
    batchNormalizationLayer

    reluLayer
%     leakyReluLayer

    maxPooling2dLayer(2,'Stride',2)
    %****************************************%
    convolution2dLayer(10 ,10 , 'Stride' ,2,'Padding','same')
    batchNormalizationLayer
    reluLayer

    maxPooling2dLayer(2, 'Stride' ,2)
    %****************************************%
    convolution2dLayer(32 ,8 , 'Stride' ,2,'Padding','same')
    batchNormalizationLayer
    reluLayer

    maxPooling2dLayer(2, 'Stride' ,2)
    %****************************************%
    fullyConnectedLayer(60)
    fullyConnectedLayer(4)

    softmaxLayer

    classificationLayer];

```

Configuration of our proposed CNN

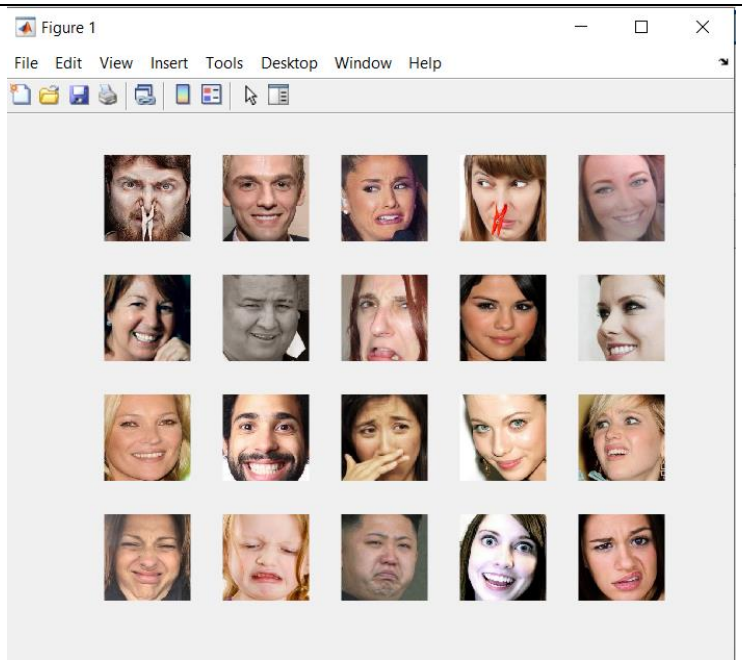


The result of our CNN network for 4 classes with Three layer CNN

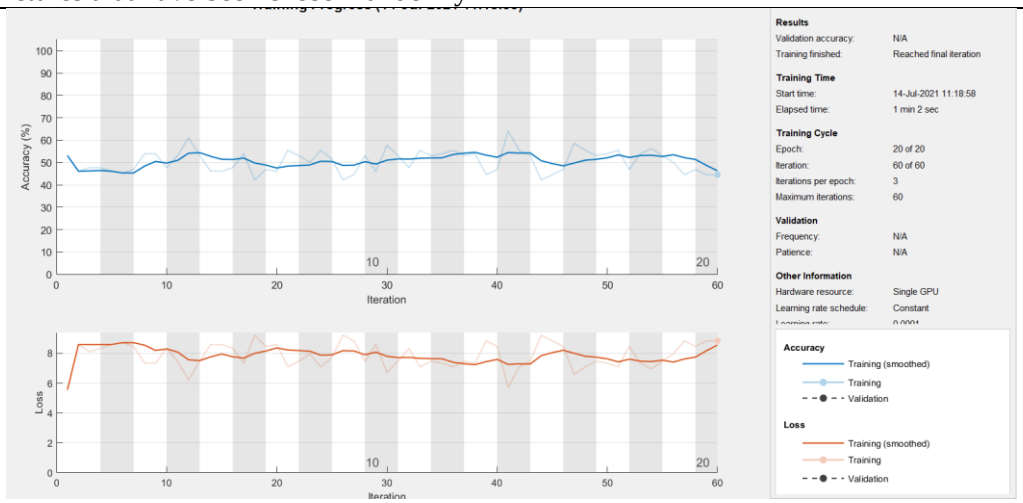


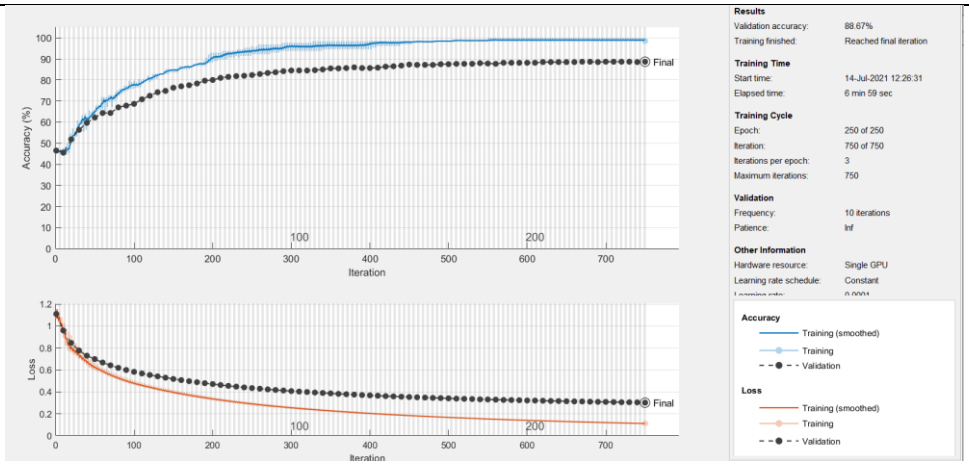
As you can see we have a confusion Matrix Results as shown

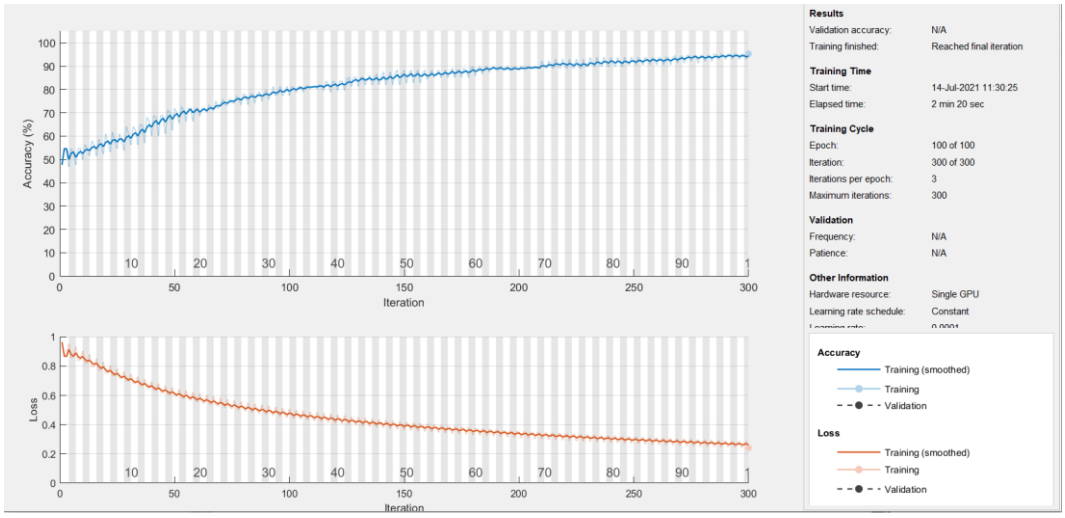
2.7 2 classes with two layer CNN:

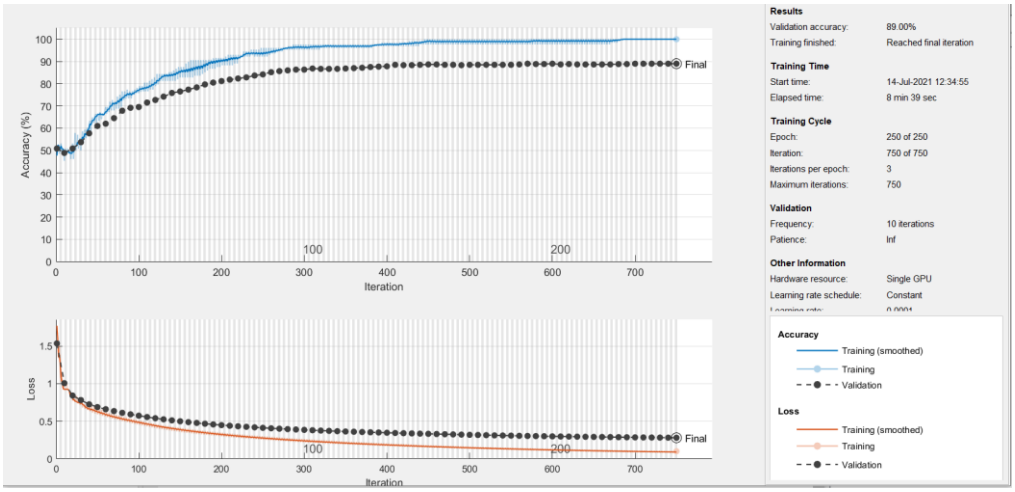


20 pictures that have been chosen randomly





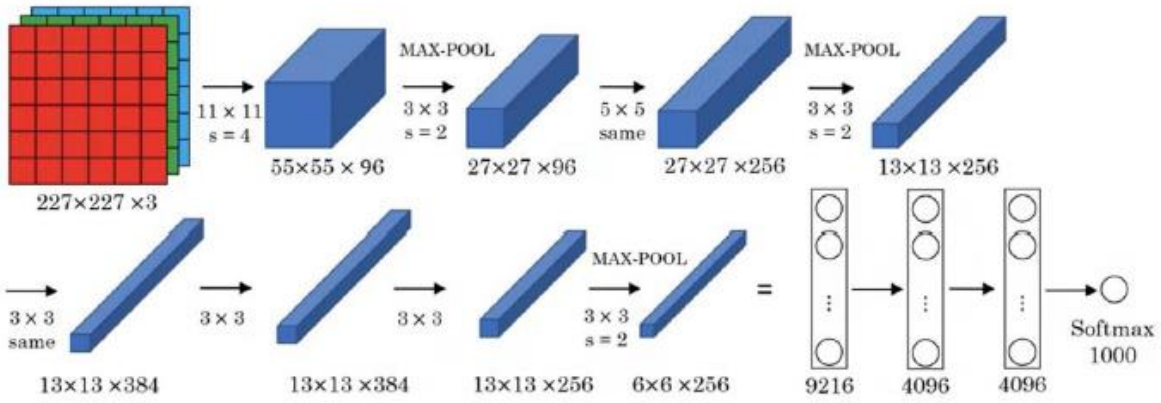




The result of our CNN network for 2 classes with two layer CNN

2.8 Task 4.2: Classifying facial expressions with a pretrained CNN

The aim of this part of the project is to fine-tune a pretrained CNN to perform the task described in the previous section. The suggested pretrained network is A*lexNet* (*https://it.mathworks.com/help/deeplearning/ref/alexnet.html*). *AlexNet* is a CNN for object recognition in images, trained with a dataset of images that contain objects belonging to *1000* classes.

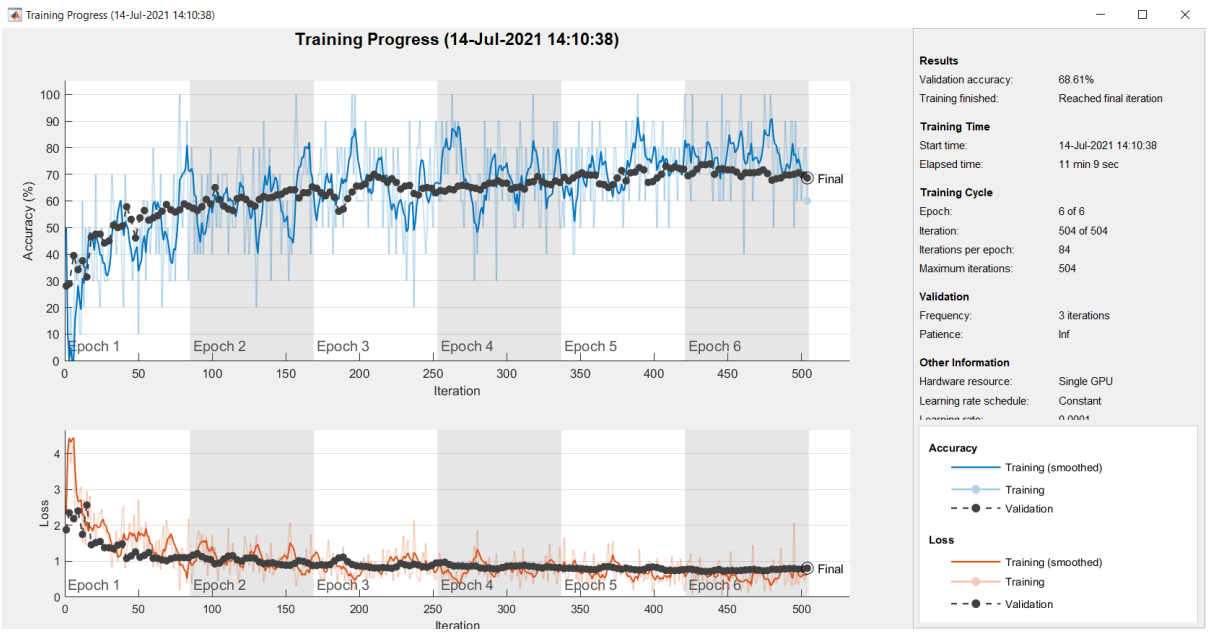


2.8.1 Transfer learning using Alexnet CNN

We have used from the transfer learning that the name of this Network is Alexnet CNN which is we just used from the first layers also we just trained the last layer for achieving our goal.

At the following you can see the results:

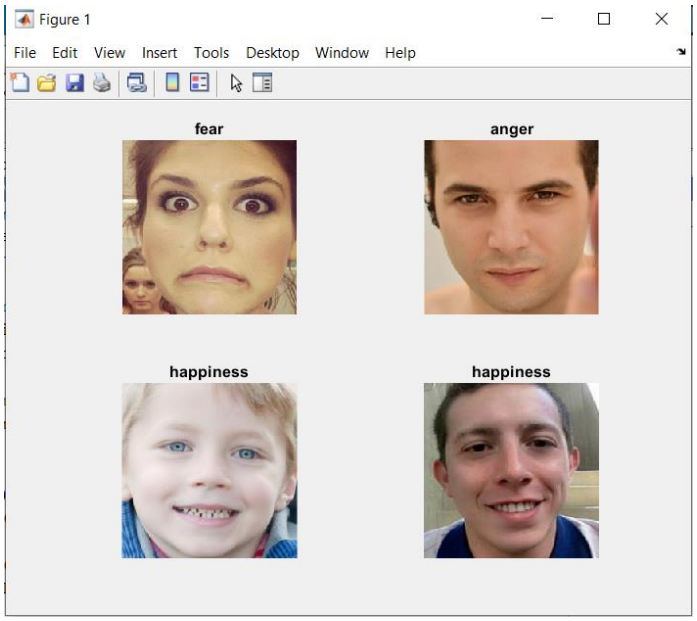
at the first we have Tested with default Epoch which is 6 :



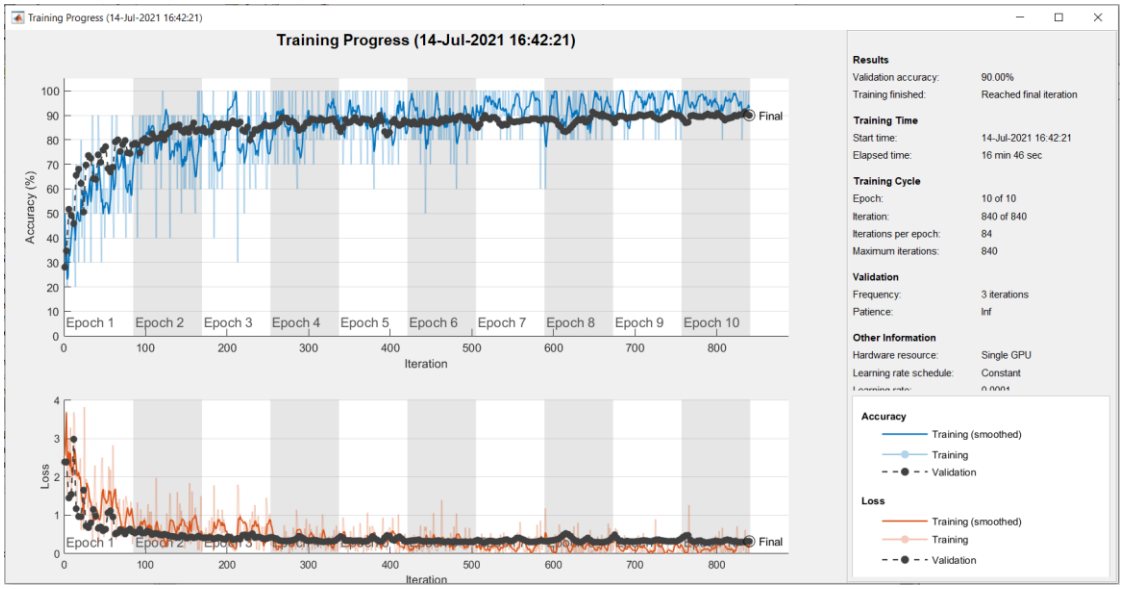
the default configuration of our transfer Network is:

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

    options = trainingOptions('sgdm', ...
    'MiniBatchSize',10, ...
    'MaxEpochs',6, ...
    'InitialLearnRate',1e-4, ...
    'Shuffle','every-epoch', ...
    'ValidationData',augimdsValidation, ...
    'ValidationFrequency',3, ...
    'Verbose',false, ...
    'Plots','training-progress');

```

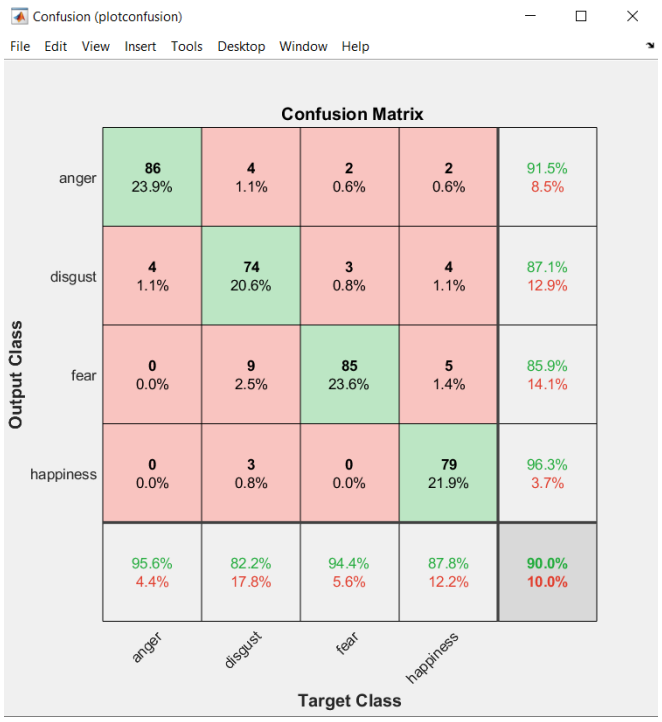
For the Result of this configuration we have the result that shown below :



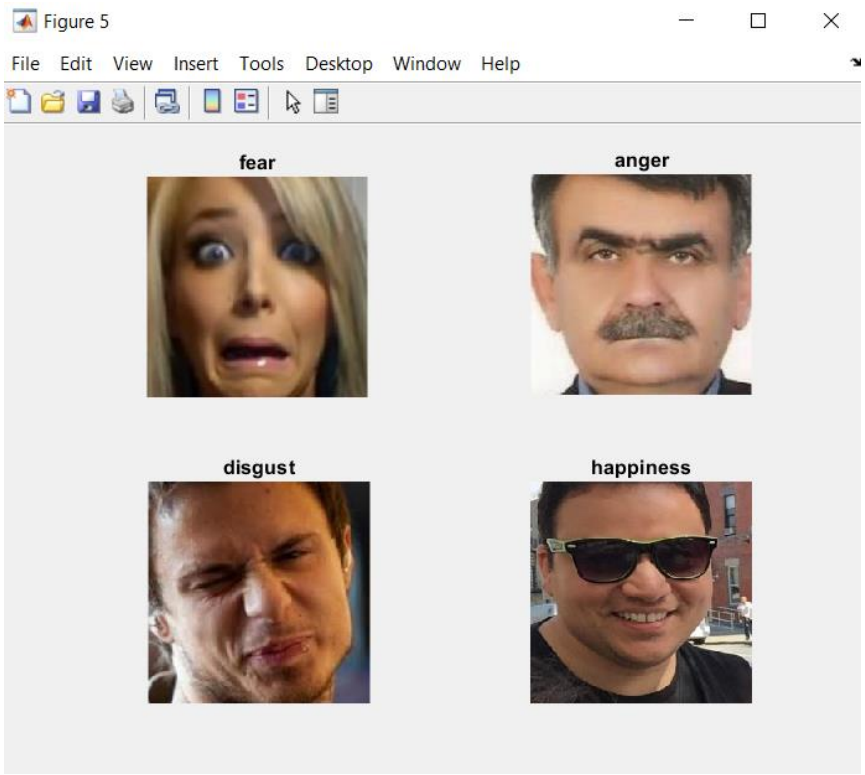
in the following you can see that we have used different Epochs which is 10 at the Below:



As you can see we have a confusion Matrix Results as shown below:



At the end you can see the Final result:



3 Addendum of the Project

3.1 Using MATLAB Cloud and Amazon Aws infrastructure

For reaching better performance when I want to Execute the code I have used parallel computing and Amazon virtual machines instead of using my own Hardware.

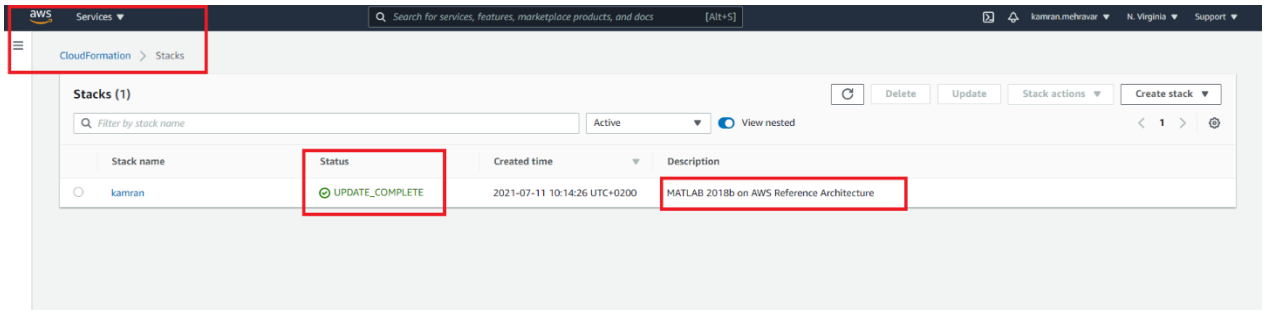
In addition, for Implementing this kind of computing and calculation I had to configure MATLAB Cloud Cluster and Amazon AWS in the correct way.

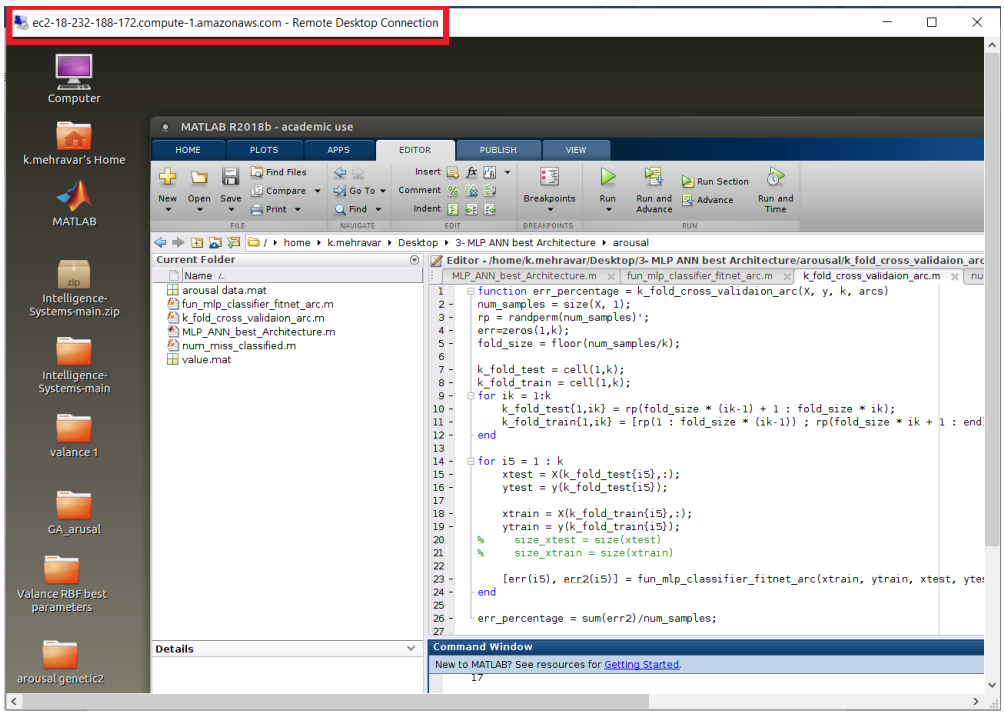
This is simply because the best way also the standard way for doing a project perfectly you have to save your time and find other ways to run and get the results as soon as possible to accelerate the process.

Furthermore, after searching and reading Structures and solutions I have decided to do this work and I have done it perfectly.

In conclusion, you can see the Pictures which Proves my implementation below:

3.1.1 This Picture Shows that my Amazon AWS Stacks machine (virtual machine) works well and it's ready to run codes.





3.1.2 At the second Part because of the Hardware issue and limitations I had to use Parallel Computing for having much Faster result, so this Pictures Proves my Work: