IN SUPREMÆ DIGNITATIS 1343

UNIVERSITÀ DI PISA

MSc in Computer Engineering

Intelligent Systems Project Report

Student: Luca Tartaglia

Academic year 2020/21

Contents

1. Introduction

For this project has been carried out tasks 3.1, 3.3, 4.1 and 4.2.

Below are explained the procedures to satisfy the required tasks.

Task 3.1: For this task is executed a dataset cleaning process, the selection of the most relevant features and a data augmentation on the obtained values (Matlab files `data_preprocessing.m` and `data_augmentatio.m`).
After, is performed the design and the development of two MLP artificial neural networks that accurately estimate a person's valence and arousal.
As a last point the development of two RBF networks that do the same thing as the previously developed MLPs (Matlab files `arousal_neural_networks.m and valence_neural_netoworks.m`).

Task 3.3: This part is reserved to design and develop a Mamdani-type fuzzy inference system to estimate a person's arousal. (Matlab file `mamdani_FIS_evaluation.m` and `mamdani_FIS.fis`)

Task 4.1: This part of project is dedicated to design, develop, and train a CNN to classify facial expression. Have been created two different CNN, one to perform 2-class classification problem and another one to perform 4-class classification problem (Matlab files `CNN_2_classes.m` and `CNN_4_classes.m`).

Task 4.2: For this section has been performed a fine-tune of a pretrained CNN (Alexnet) to obtain a convolutional neural network (CNN) that accurately classifies a person's emotion, based on facial expression. Have been created two different CNN, one to perform 2-class classification problem and another one to perform 4-class classification problem (Matlab files `CNN_2_classes_alexnet.m` and `CNN_4_classes_alexnet.m`)

2. Artificial Neural Networks (Task 3.1)

2.1 Dataset Preparation

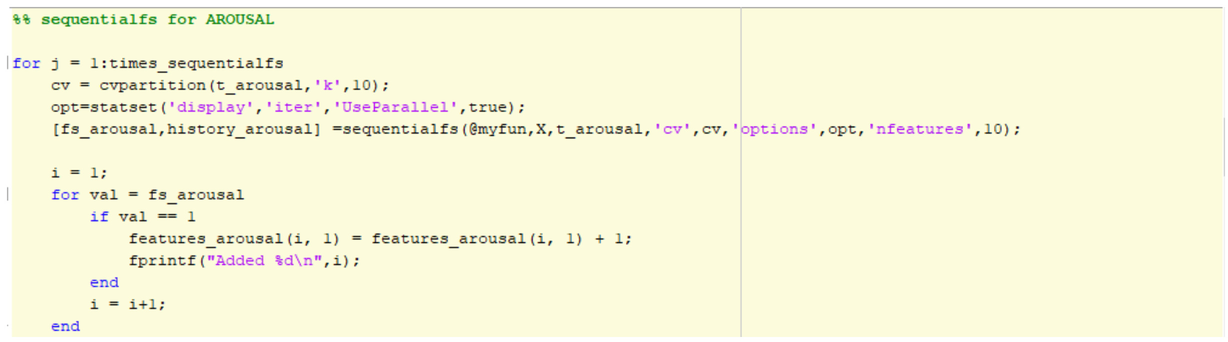
As starting point for task 3.1 (also for 3.3 task) a dataset preparation process it was carried out.

This preparation process consisted in:

- Remove non-numerical values
- Remove outliers
- Select best 10 significant features (and best 3 for 3.3 task)
- Data Augmentation

First of all, a **cleaning process** has been done in order to remove non-numeric data and infinite numbers into the dataset. To do so *isinf()* function was used.
Then a procedure of **outliers removal** has been carried out. To do so, some experiments have been done out with the way to perform outlier removal. The *rmoutliers()* function with *median* method has been selected as optimal by comparing the results.

To best predict the output, it was decided to **select the most 10 significant features** (and the most 3 significant for task 3.3).
This operation was performed by the implementation of the suggested function, **sequentialfs**. After some experiments it was decided to use a ten-fold cross validation to split the dataset for a better evaluation and use a fitnet with 25 hidden neurons as a criterion function that was able to properly assess the accuracy of each subset.
The sequentialfs has been ran 5 times. Each feature selected
has been counted in a counter vector and at the end this vector has been sorted in descending order to obtain the list of most important features. Two different
sorted vector were obtained, one for arousal and another for valence.



```
%% sequentialfs for AROUSAL

for j = 1:times_sequentialfs
    cv = cvpartition(t_arousal,'k',10);
    opt=statset('display','iter','UseParallel',true);
    [fs_arousal,history_arousal] =sequentialfs(@myfun,X,t_arousal,'cv',cv,'options',opt,'nfeatures',10);

    i = 1;
    for val = fs_arousal
        if val == 1
            features_arousal(i, 1) = features_arousal(i, 1) + 1;
            fprintf("Added %d\n",i);
        end
        i = i+1;
    end
```

Fig.1 sequentialfs function for arousal

Once executed the feature selection step, the obtained dataset needed a **data augmentation operation**, that was executed with some ad-hoc operations.

The data augmentation was performed using an autoencoder, that takes several samples in input and gives the latter samples augmented in output (tested by changing the autoencoder parameters).
To do so the dataset was split in 7 different class, according to the values that arousal and valence assume (that are 1, 2.3, 3.6, 5, 6.3, 7.6 and 9).
So, it was implemented a system that let decide the maximum number of samples to have in the final dataset for each class.
After some experiments it was decided to generate a maximum of 300 augmented samples for each class, thus producing an amount of 2168 samples for arousal and 2108 samples for valence, all balanced through the available classes. In the Fig.2 and Fig.3 we can see the results.

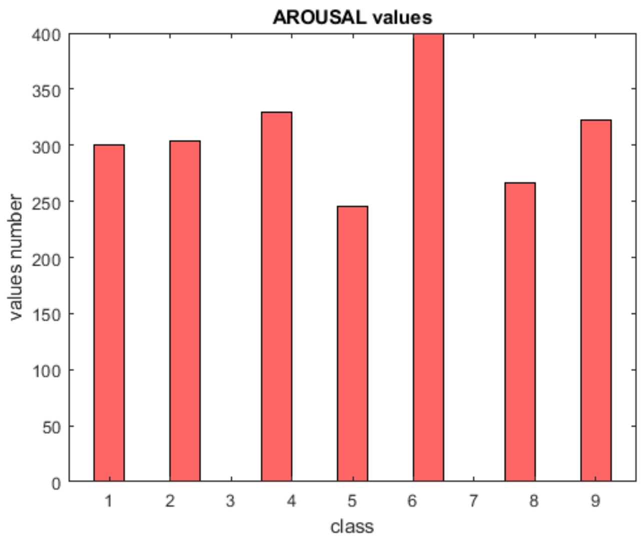


Fig.2 Histogram of arousal distribution

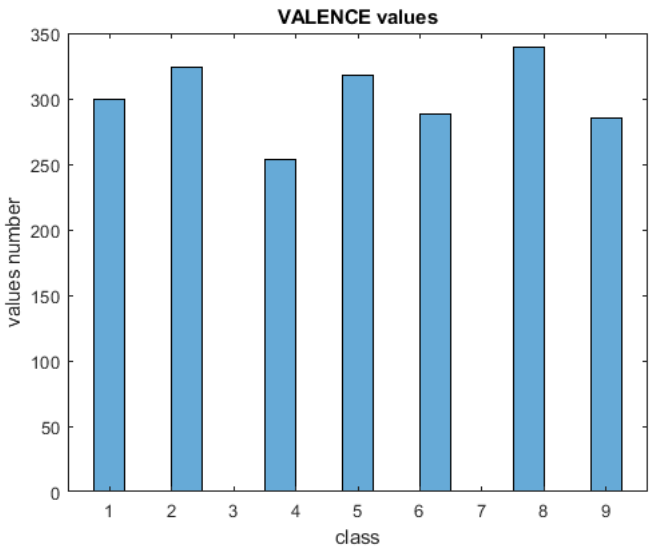


Fig.3 Histogram of valence distribution

2.2 Arousal MLP and RBF

After some experiments to find the best ANN architecture, it was decided to use the **fitnet** for the arousal (also valence) regression.

To divide the dataset, it was chosen to use **holdout** partition, to have a totally random division. This decision is due to the fact that computationally and for the time run make a holdout partition is better than the k-fold cross validation method. Having found good results it was decided to use those.
For arousal regression the best split is to have 80% of data for training and 20% of data for test.
some experiments to find the best architecture for MLP was by changing the number of neurons (i.e for the first hidden layer in the interval [5,60], trying to minimize this number as soon as good results started to appear), by changing the training function, by modifying the maximum number of validation fails (from 3 to 10), the number of epochs and by increasing/decreasing the validation set.
In the Fig.4 we can see the final neural network structure chosen.

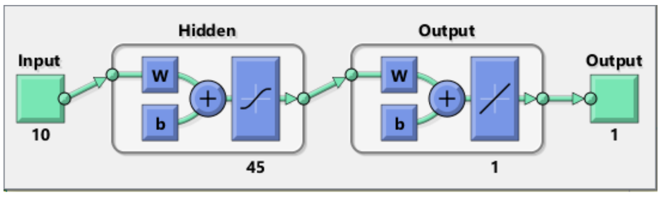


Fig 4. Arousal MLP structure

Good results were obtained using 45 number of neurons, using 80% of data for training and 20% for validation, also the max failures value was increased compared to the default value.
In the Fig.5 and we can see respectively the regression plot and the performance plot.
In the Fig.7 we can see also the good results achieved testing the network with the test set, obtained through the holdout partition.

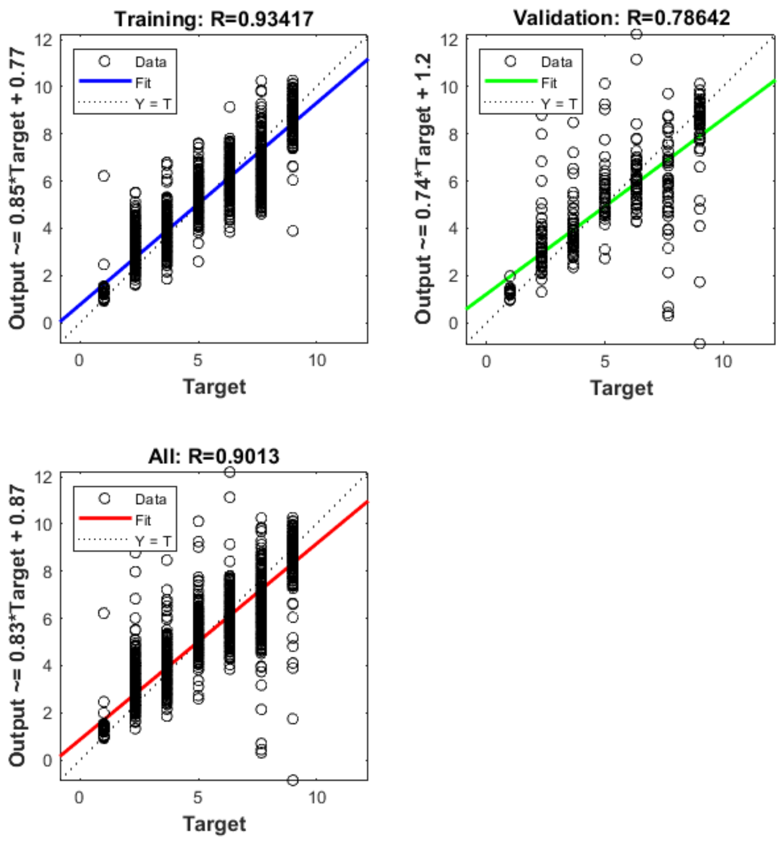


Fig 5. Arousal training and validation regression plot

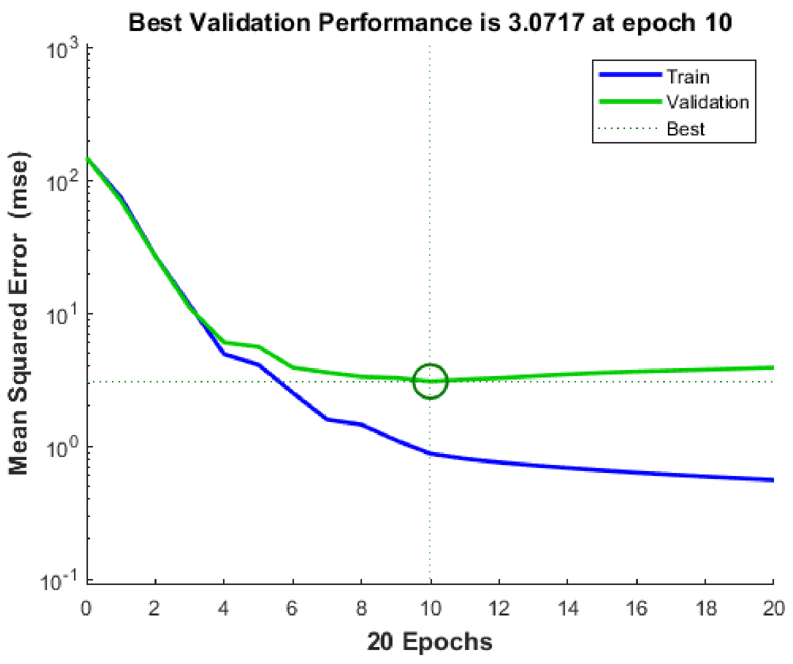


Figura 6. Arousal performance plot

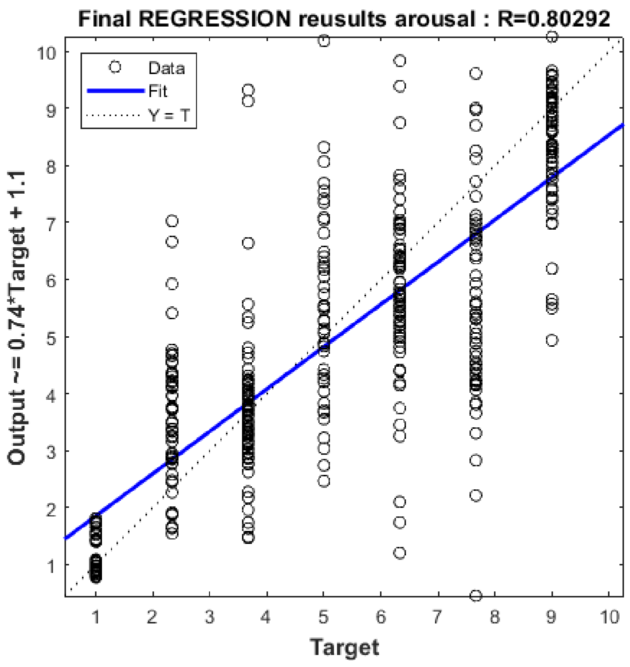


Fig.7 Arousal regression results plot

The last step for the arousal values estimation was to design and train a **Radial Basis Function Network**. This task was performed by the implementation of the network through the *newrb* function.

In particular, the experiments were made by changing not only the spread values, that was fundamental to obtain good results, but also by changing the number of maximum neurons.

The best results were obtained with **spread=1.83**, in the Fig.8 we can see the results

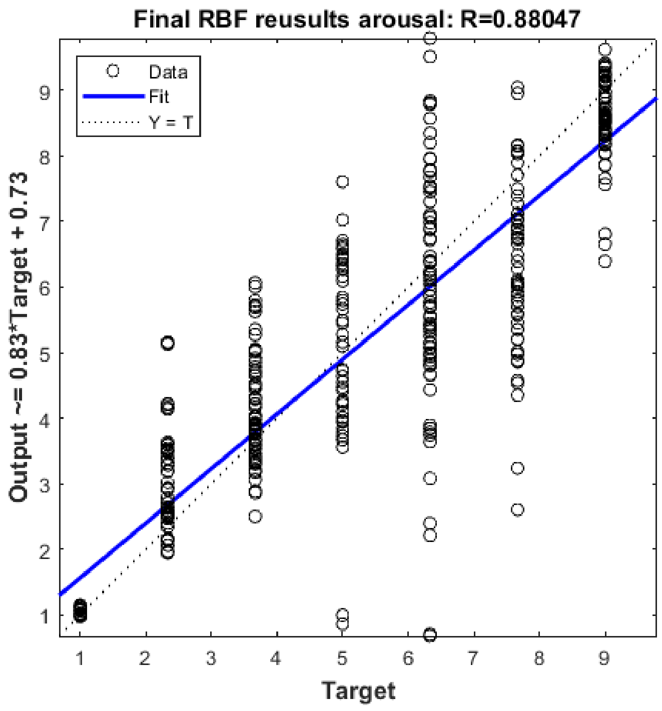


Fig.8 Arousal RBF regression results plot

2.3 Valence MLP and RBF

As for the arousal values, also for valence value estimation was implemented a **fitnet** network.

The dataset was split with the same method as for arousal values, producing a holdout partition, and after several experiments the best percentage of the feature sets found was 80% for training and 20% for testing, that divided the dataset with a good distribution.

The neural network structure was changed as well, to best perform the regression. It was reached the conclusion to use 20 neurons for the hidden layer and to use a large value for max failures (25 instead 10). The percentage for training and validation is unchanged, 80% and 20%.

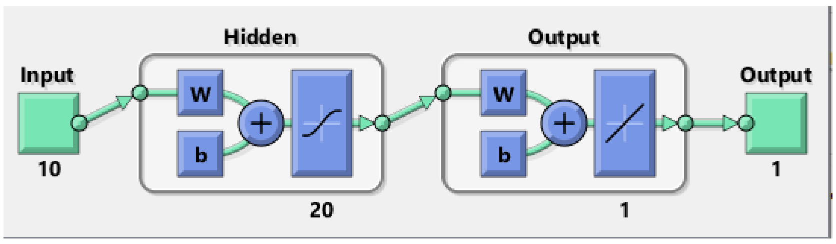


Fig.9 Valence MLP structure

In the Fig.10, Fig.11 and Fig.12 we can see the results as for the arousal network.

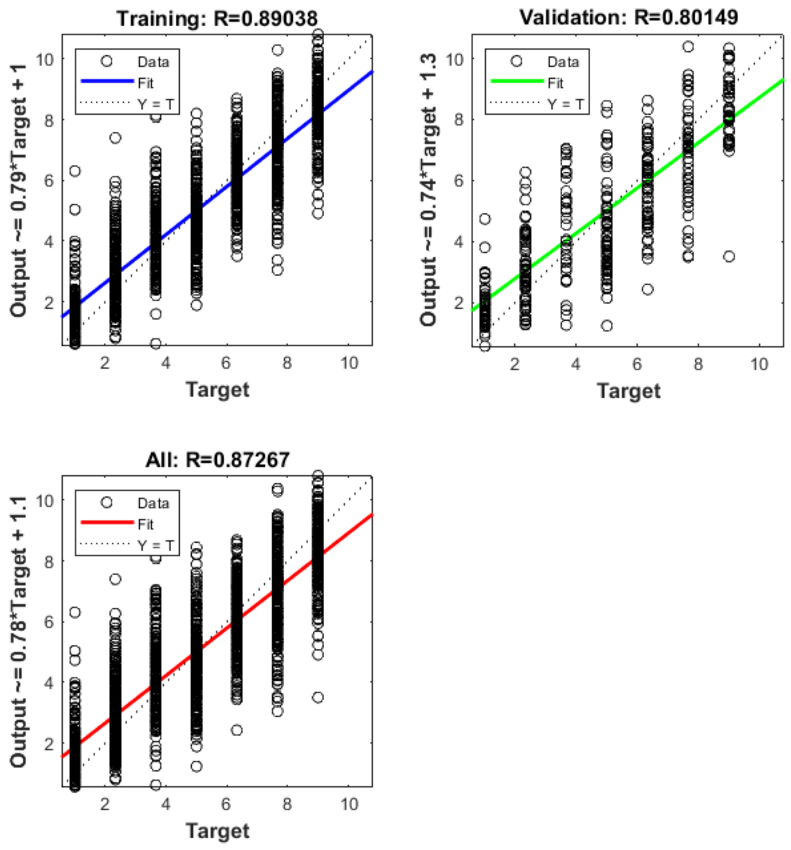


Fig.10 Valence training and validation regression plot

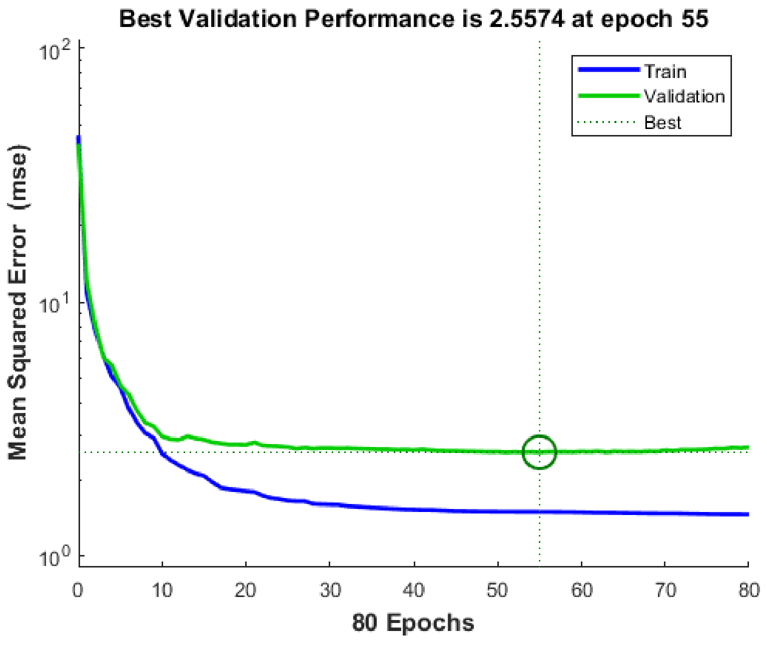


Fig.11 Valence performance plot

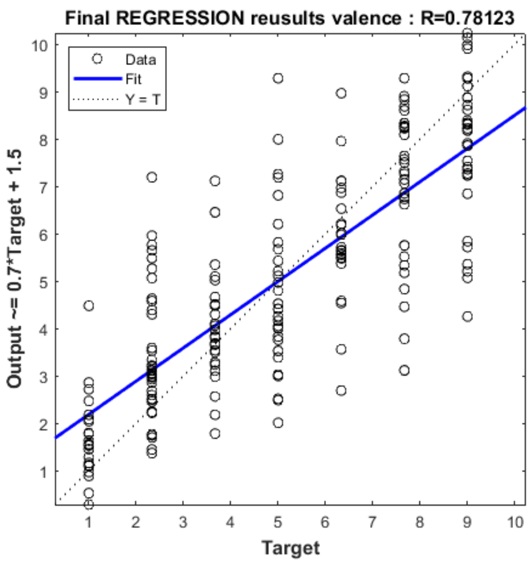


Fig.12 Valence regression results plot

To conclude the MLPs section, the arousal neural network behaves much better than the valence, comparing training, validation, and test plots; but the results can be reasonable also for valence regression.

For what concerns the valence RBF network, the procedure was the same as for arousal. Were tested different values for spread, changing the numbers of maximum neurons. The most relevant parameter is the spread equal to 1.05.

In comparison with the arousal RBF, here we have obtained much better results, as we can see from the Fig.13.

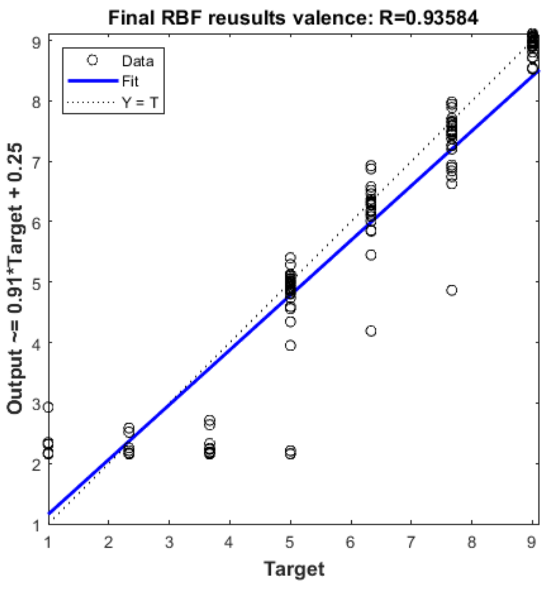


Fig. 13 RBF valence results plot

3. Fuzzy Inference System (Task 3.3)

3.1 Mamdani Fuzzy Inference System

The aim of this task was to design and develop a Mamdani-type fuzzy inference system to fix the deficiencies regarding the arousal values (i.e., to not correctly evaluate arousal values).

The goal was reached by an observation step of the most 3 significant features (obtained in the task 3.1), studying their distributions and values. To build the FIS it was used the Fuzzy Logic Designer graphical tool.

In specific were studied the distributions of the features through their histograms, that we can see in Fig.14.

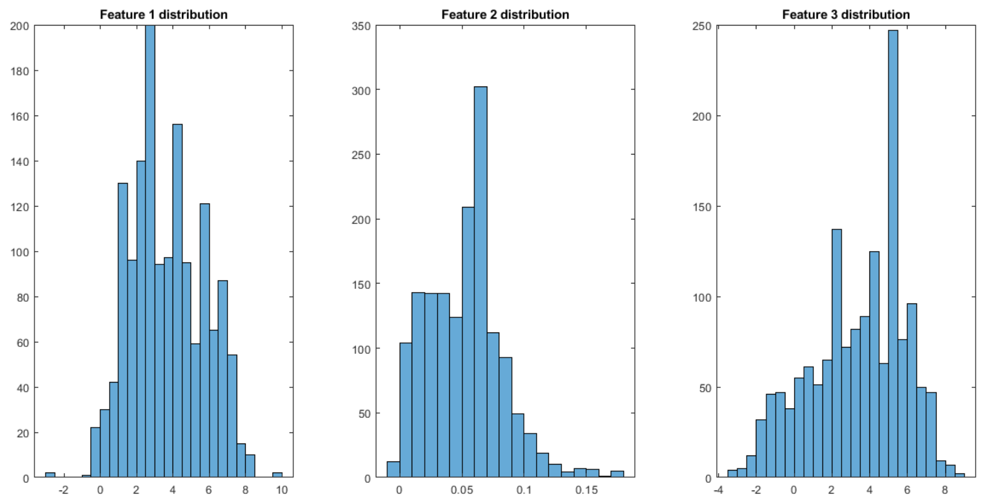


Fig.14 Histograms of features distributions

These distributions have been used to model the linguistic variables of the three features. The approach adopted was the following: since the distributions tends to have peaks in some points, those peaks were considered as central point of a particular input variable.

For set the membership functions, first has been calculated the ranges for each feature that defines the universe of discourse, then, through the histograms, it was decided to apply as a triangle the central membership function and as trapezoidal the two on the sides. As mentioned before, the centre of the inputs is in correspondence of the peaks studied in the histograms of the features.
In the Fig.15, Fig.16 and Fig.17 we can see graphically how the input membership functions are modelled.

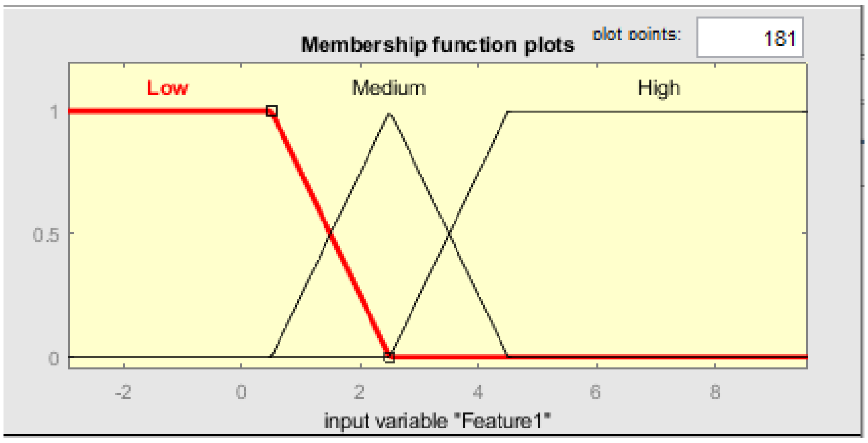


Fig 15 Feature 1 membership function

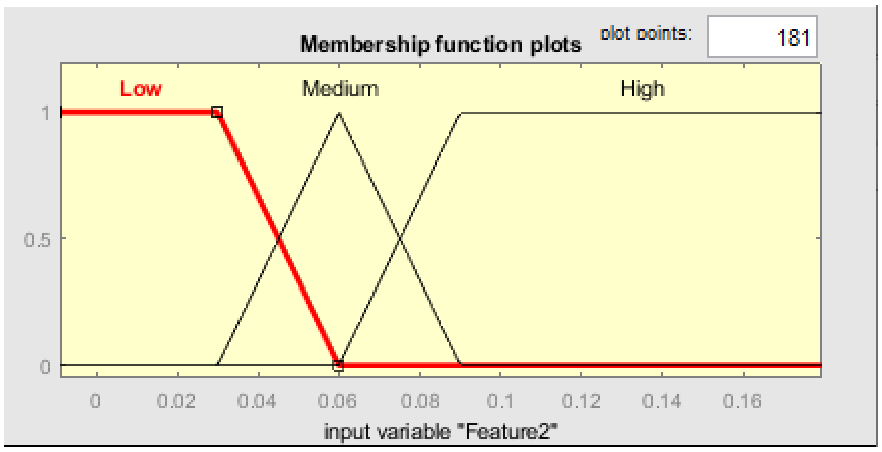


Fig 16 Feature 2 membership function

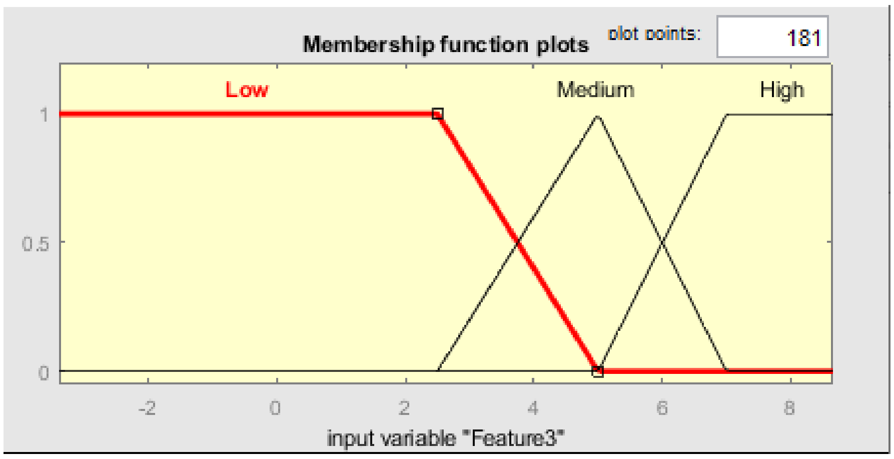


Fig 17 Feature 3 membership function

For the output, watching the arousal values it was decided to model it as three triangular membership functions. In the Fig.18 we can see the results.

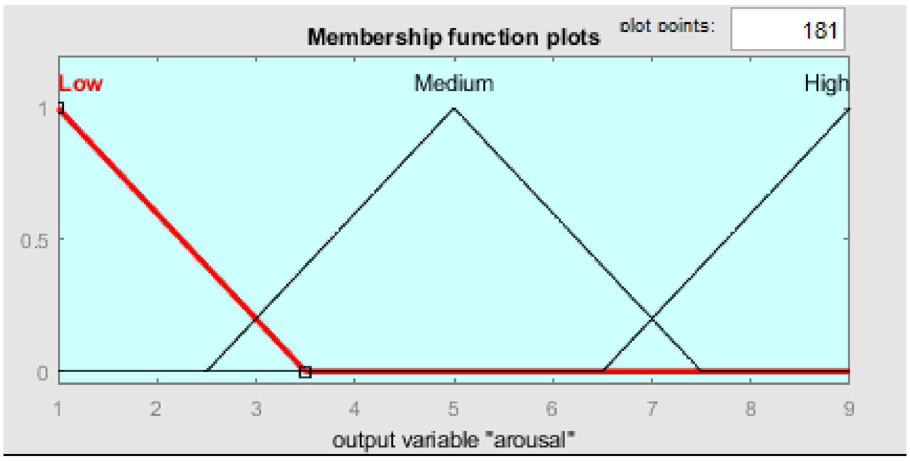


Fig.18 Arousal membership function

Finally, the rules have been written thanks to other histograms also produced at the observation stage.
These histograms, that we can see in the Fig.19, show the feature distribution separated by range of arousal values.
In the first line we can see the feature distributions when they assume a low arousal value (in the range [1,3]), in the second level the feature values when they assume a medium arousal (in the range [3,7]) and in the last line the distribution of the features that assume high values (range [7,9]).

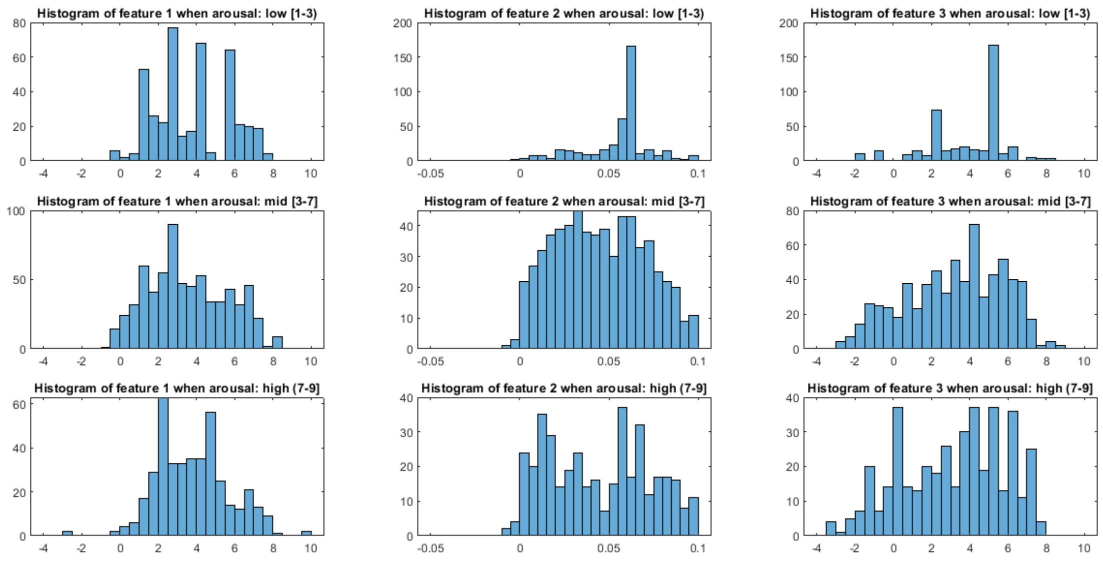
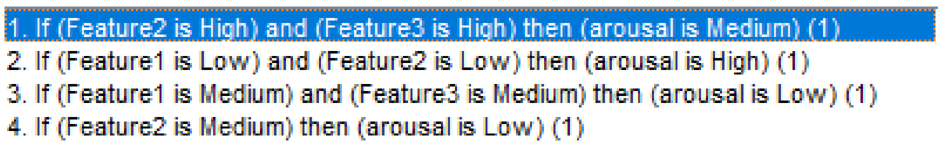


Fig. 19 Histograms of features distribution divided by arousal ranges

In the fig.20 we can see the written rules.



1. If (Feature2 is High) and (Feature3 is High) then (arousal is Medium) (1)
2. If (Feature1 is Low) and (Feature2 is Low) then (arousal is High) (1)
3. If (Feature1 is Medium) and (Feature3 is Medium) then (arousal is Low) (1)
4. If (Feature2 is Medium) then (arousal is Low) (1)

Fig.20 FIS Rules

To understand the reasoning behind the rules an example is explained. It considers the rule number 3:

"If (Feature1 is Medium) and (Feature3 is Medium) then (arousal is Low)".

This rule can be inferred analyzing the two graphs corresponding to feature 1 and 3 when they assume a low arousal value.
In fact, is possible to see from the Fig.19 that feature 1 and in specific the feature 3 have much more samples in correspondence of their medium values (we can see distinct peaks).

In general, the rules were written comparing the position of the peaks and watching the feature values distributions.

4. Convolutional Neural Networks (Task 4.1)

For this section the aim is to design, develop and train a Convolutional Neural Network (CNN) to accurately classify person's emotions based on facial expression, using the dataset provided (images of people divided in four classes, who express anger, fear, disgust, happiness).

This task was accomplished by developing 2 different CNN, one that solve the "2-class problem", the classification of the two most different classes, anger and happiness, and another one that classify all four given image classes, solving the "4-class problem".

An important observation to highlight is that being too big, the dataset has been reduced to 300 images per class, selecting those that best express the desired emotion. As we shall see this was fundamental to obtain appreciable results.

4.1 CNN for 2-class classification problem

As stated before, this CNN perform a person emotion classification of two different classes:

- Anger
- Happiness

Once the starting dataset was selected (300 images per class) it was split in a randomized way into subsets used for training and validation operations. The percentage chosen were:

- 70% for training
- 30% for validation

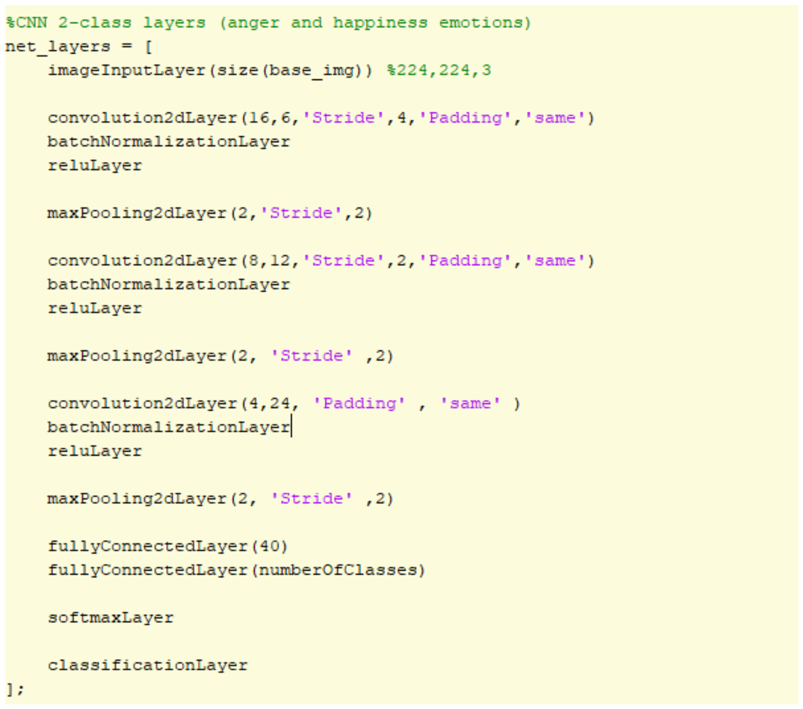
Then, a data augmentation step is necessary before train the network.

To do so it was decided to perform a data augmentation process, reflecting, translating, and rotating the images.

After this step, it was defined the Network Architecture, this was one of the most crucial steps.

After some experiments, like testing different numbers of convolutional layers and different filters dimension, using different types of ReLu layers (tested also Leaky ReLu layer instead ReLu), testing test different fully connected layers the best architecture was find.

In the Fig. 21 we can see the structure used.

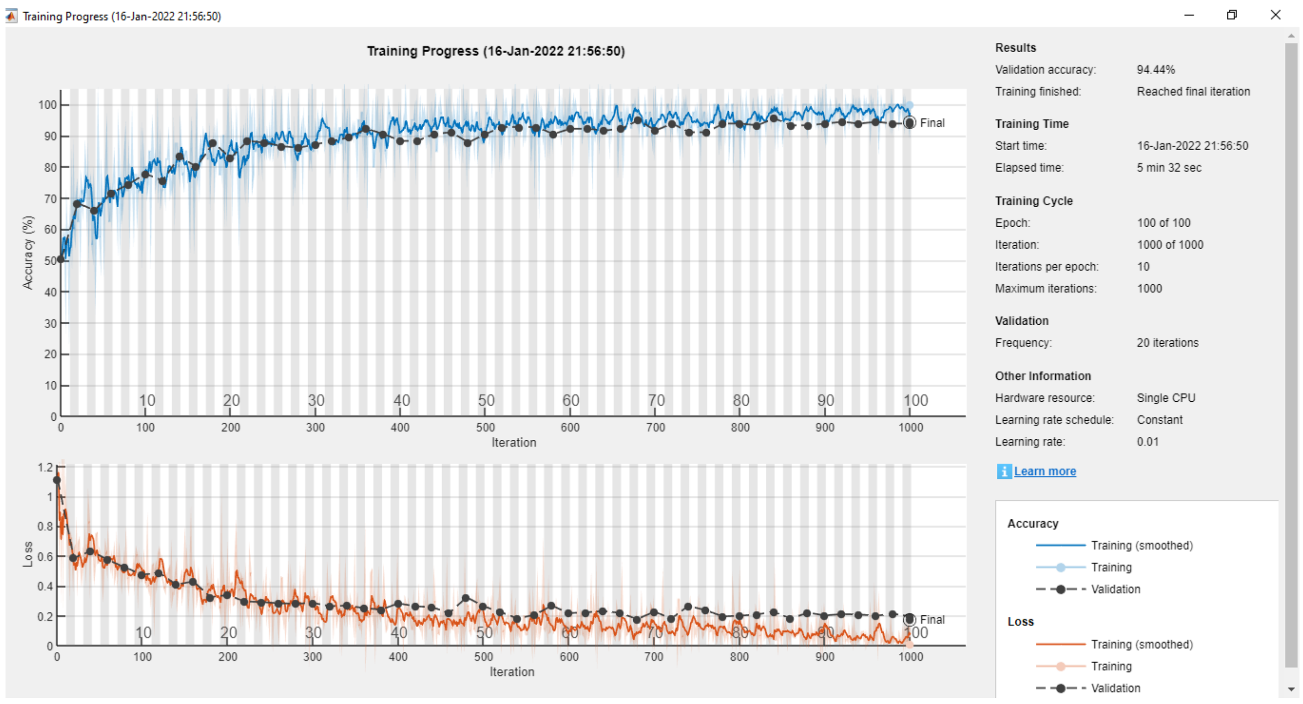


```
%CNN 2-class layers (anger and happiness emotions)
net_layers = [
    imageInputLayer(size(base_img)) %224,224,3

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

    maxPooling2dLayer(2,'Stride',2)

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

    maxPooling2dLayer(2, 'Stride' ,2)

    convolution2dLayer(4,24, 'Padding' , 'same' )
    batchNormalizationLayer
    reluLayer

    maxPooling2dLayer(2, 'Stride' ,2)

    fullyConnectedLayer(40)
    fullyConnectedLayer(numberOfClasses)

    softmaxLayer

    classificationLayer
];
```

Fig. 21 CNN 2-class network structure

At this point the training step is performed. Also, some option changes were tested, like the initial learning rate (set to 0.01), the max epochs (set to 100) and the mini batch size (set to 40).

From the training process results is possible to see that with the studied architecture satisfactory results were achieved (94.44% validation accuracy). In the Fig.22 is described the training process.



Training Progress (16-Jan-2022 21:56:50)

Training Progress (16-Jan-2022 21:56:50)

Results

Validation accuracy:	94.44%
Training finished:	Reached final iteration

Training Time

Start time:	16-Jan-2022 21:56:50
Elapsed time:	5 min 32 sec

Training Cycle

Epoch:	100 of 100
Iteration:	1000 of 1000
Iterations per epoch:	10
Maximum iterations:	1000

Validation

Frequency:	20 iterations

Other Information

Hardware resource:	Single CPU
Learning rate schedule:	Constant
Learning rate:	0.01

Fig. 22 CNN 2-class training process

It was analysed the validation accuracy in depth by plotting the
confusion matrix and by computing the accuracy. In the Fig.23 we can see the results.

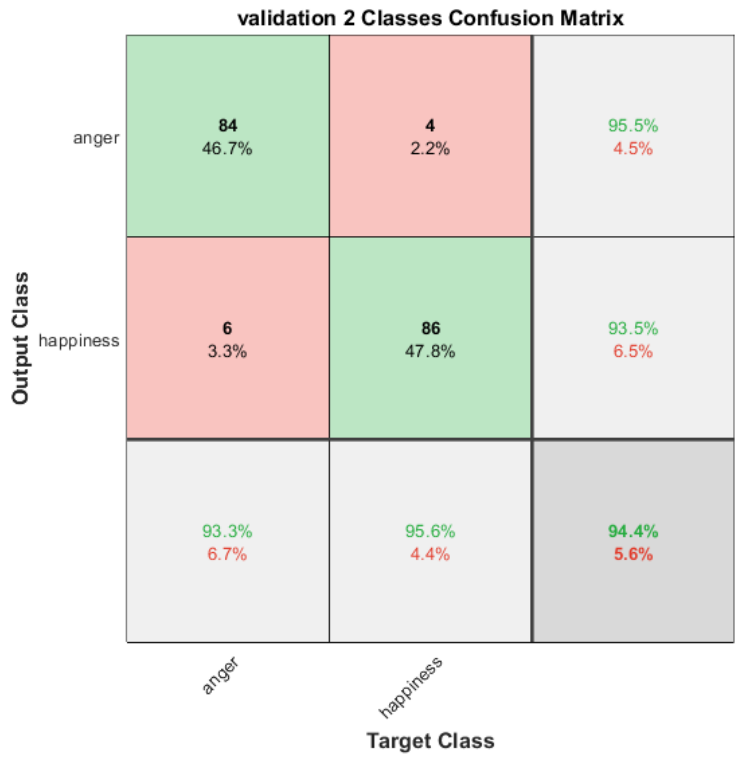


Fig. 23 CNN 2-class confusion matrix

As a last point was produced a set of classified samples, using the previously extracted validation images, to really show how the CNN worked. In the fig 24 we can see 12 extracted samples from the validation set.

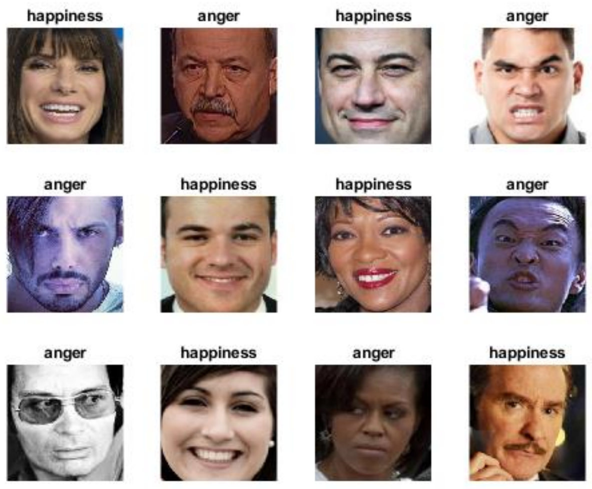


Fig 24. Classification samples

4.2 CNN for 4-class classification problem

The aim of this experiment was to observe the difference between classify 2 classes and 4 classes, how the network architecture must change and highlight the difficulties identified.

This architecture was built to classify all four available image classes:

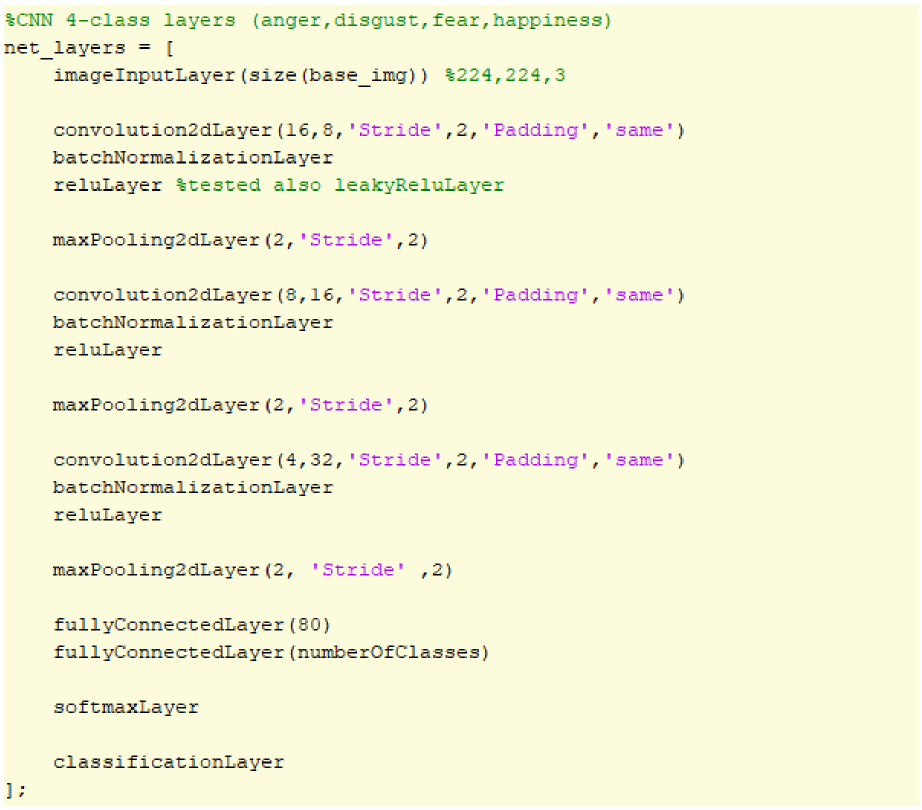
- anger
- disgust
- fear
- happiness

The number of images used is the same, 300 per class and the division also is the same (70%, 30%)

As might be expected, this time the classification is much more complicated, not only because there are 4 classes instead 2, but also because anger, fear and disgust images can resemble each other.

The approach was to start from the network created for 2-class problem and modify it based on several experiments.

After several tests, like increase the number of convolutional layers, use more than two fully connected layer, change the max pooling layer size, the final architecture was produced.

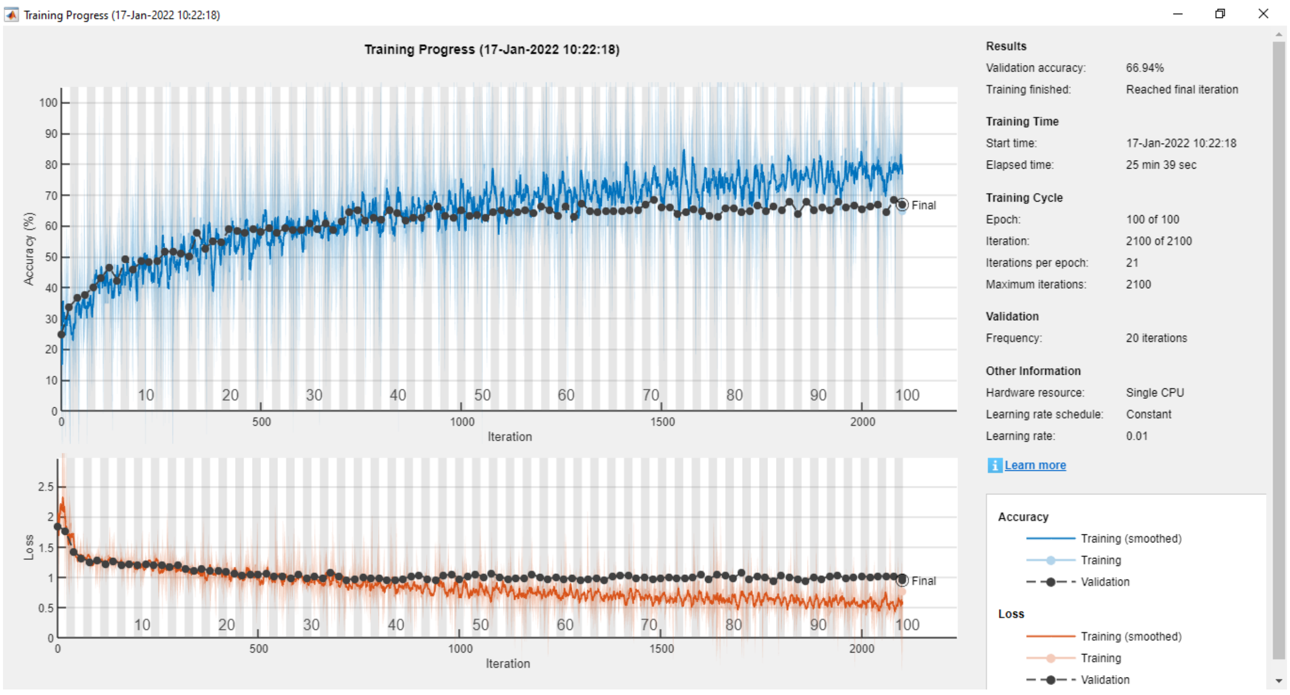


```
%CNN 4-class layers (anger,disgust,fear,happiness)
net_layers = [
    imageInputLayer(size(base_img)) %224,224,3

    convolution2dLayer(16,8,'Stride',2,'Padding','same')
    batchNormalizationLayer
    reluLayer %tested also leakyReluLayer

    maxPooling2dLayer(2,'Stride',2)

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

    maxPooling2dLayer(2,'Stride',2)

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

    maxPooling2dLayer(2, 'Stride' ,2)

    fullyConnectedLayer(80)
    fullyConnectedLayer(numberOfClasses)

    softmaxLayer

    classificationLayer
];
```

Fig. 25 CNN 4-class network structure

The most significant changes were the increase of the convolutional layer filters and the increase of the fully connected layer output size.

With the training process plot and the confusion matrix on the validation data is possible to see that, even if not better than the previous, the results are satisfying,.



Results
Validation accuracy: 66.94%
Training finished: Reached final iteration

Training Time
Start time: 17-Jan-2022 10:22:18
Elapsed time: 25 min 39 sec

Training Cycle
Epoch: 100 of 100
Iteration: 2100 of 2100
Iterations per epoch: 21
Maximum iterations: 2100

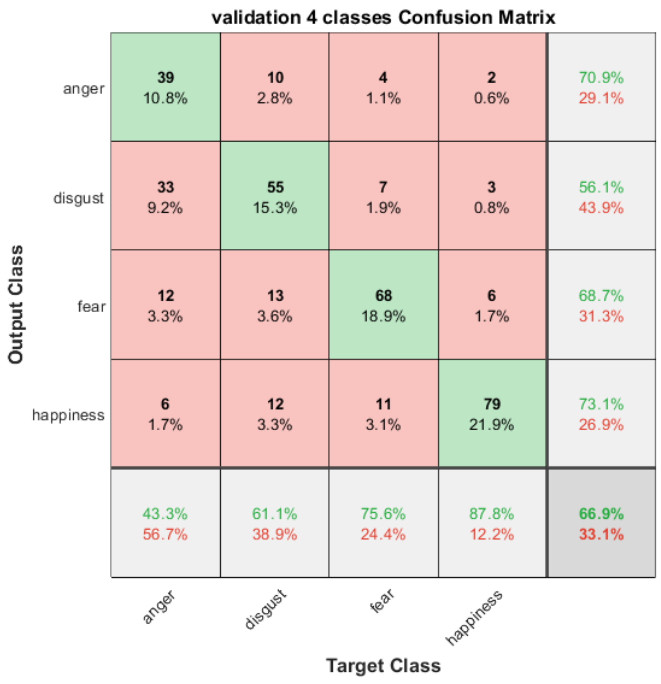
Validation
Frequency: 20 iterations

Other Information
Hardware resource: Single CPU
Learning rate schedule: Constant
Learning rate: 0.01

Accuracy
Training (smoothed)
Training
Validation

Loss
Training (smoothed)
Training
Validation

Fig. 26 CNN 4-class training process



	anger	disgust	fear	happiness	
anger	39 10.8%	10 2.8%	4 1.1%	2 0.6%	70.9% 29.1%
disgust	33 9.2%	55 15.3%	7 1.9%	3 0.8%	56.1% 43.9%
fear	12 3.3%	13 3.6%	68 18.9%	6 1.7%	68.7% 31.3%
happiness	6 1.7%	12 3.3%	11 3.1%	79 21.9%	73.1% 26.9%
	43.3% 56.7%	61.1% 38.9%	75.6% 24.4%	87.8% 12.2%	66.9% 33.1%

Fig. 27 CNN 4-class confusion matrix

5. Pretrained CNN (Task 4.2)

For this section the aim is to fine-tune a pretrained CNN to accurately classify person's emotions based on facial expression, using the dataset provided (images of people divided in four classes, who express anger, fear, disgust, happiness).

For this task has been fine-tuned in two 2 different modes the suggested CNN, AlexNet, to solve the 2-class classification problem (anger, happiness) and 4-class classification problem (anger, disgust, fear, happiness).

The dataset used was the same, 300 images per class, selecting those that best express the desired emotion.

5.1 AlexNet fine-tune for 2-class classification problem

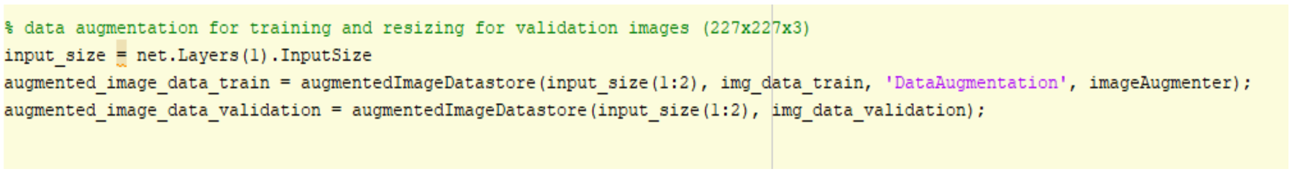
The emotion to classify are the same that in the section 4.1 (anger, happiness).

The dataset was split in the same manner, and it was applied the same data augmentation to the samples (reflection, rotation, translation).

To recap it were produced these subsets:

- 70% training
- 30% validation

The first difference with the CNN built from scratch is that here is necessary to change the input size of the images, because in AlexNet the images used have size 227x227. To fix the size it was used the augmented image datastore, that automatically resize images.



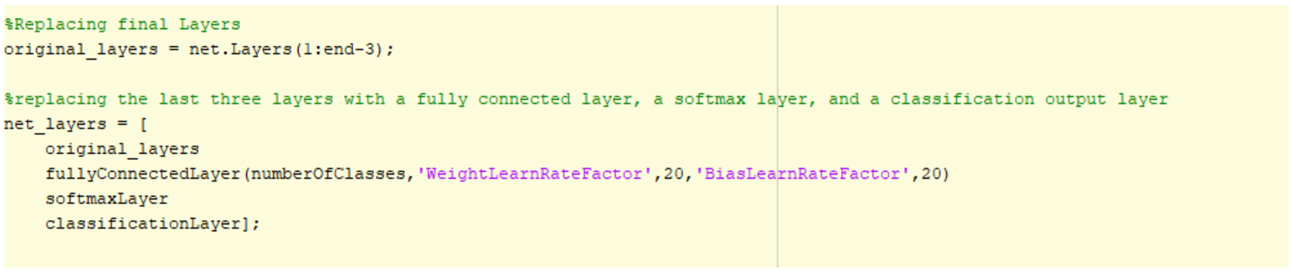
```
% data augmentation for training and resizing for validation images (227x227x3)
input_size = net.Layers(1).InputSize
augmented_image_data_train = augmentedImageDatastore(input_size(1:2), img_data_train, 'DataAugmentation', imageAugmenter);
augmented_image_data_validation = augmentedImageDatastore(input_size(1:2), img_data_validation);
```

Fig. 28 Data augmentation and resizing

To use AlexNet is also necessary to replace the final layers.
The last three layers of the pretrained network are configured for 1000 classes, instead, for this problem is needed to configure it for 2 (and 4 classes after).

The fine-tuning consisted in extract all layers, except the last three, from the pretrained network, transfer the layers to the new classification task by replacing the last three layers with a **fully connected layer** of the same size as the number of classes (2 and 4 in our cases), a **softmax layer**, and a **classification output layer**.

To learn faster in the new layers than in the transferred layers,
the *WeightLearnRateFactor* and *BiasLearnRateFactor* values of the fully connected layer were increased to 20.

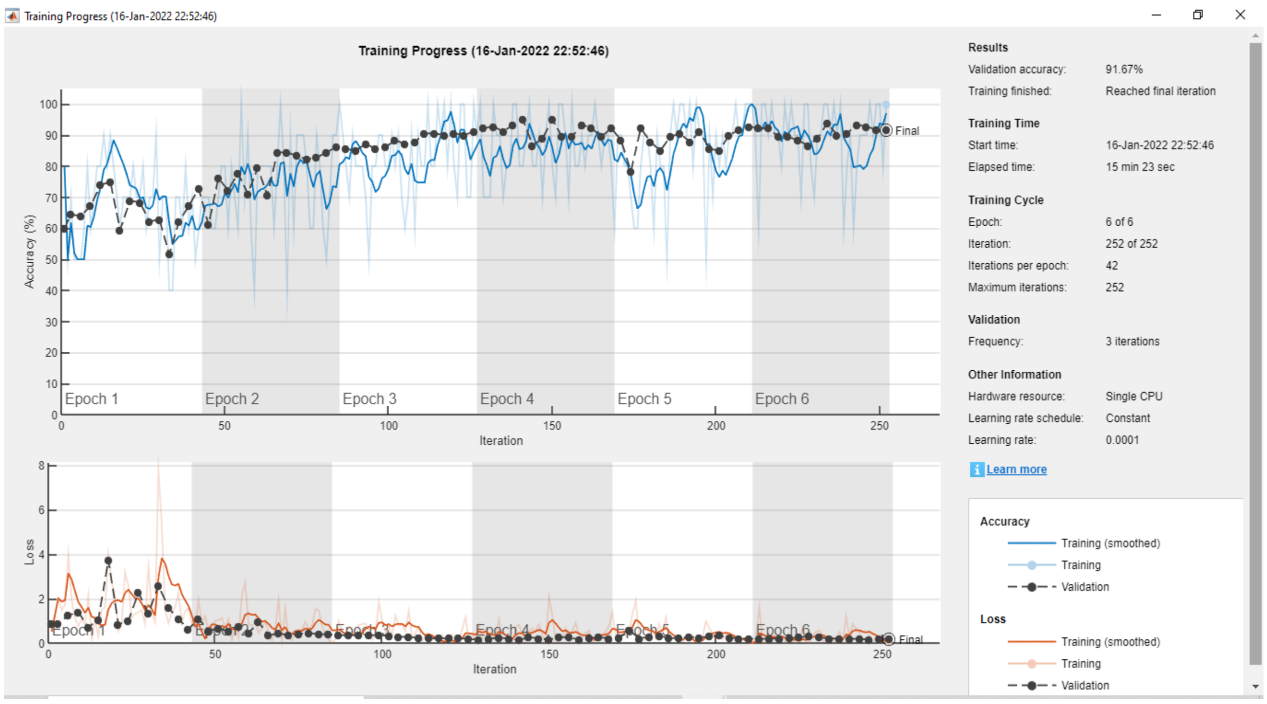


```
%Replacing final Layers
original_layers = net.Layers(1:end-3);

%replacing the last three layers with a fully connected layer, a softmax layer, and a classification output layer
net_layers = [
    original_layers
    fullyConnectedLayer(numberOfClasses,'WeightLearnRateFactor',20,'BiasLearnRateFactor',20)
    softmaxLayer
    classificationLayer];
```

Fig. 29 AlexNet based CNN 2-class new layers structure

Several experiments were made changing not only the architecture of the network but also the number of epochs. For this task 6 number of epochs (default for AlexNet) was sufficient to obtain good results (91.67% accuracy).

As last step is observed the results of the network: the training process results, through its graph produced, and the classification of validation images, through its confusion matrix. In the Fig. 30 and Fig.31 are shown these latter results.



Results
Validation accuracy: 91.67%
Training finished: Reached final iteration

Training Time
Start time: 16-Jan-2022 22:52:46
Elapsed time: 15 min 23 sec

Training Cycle
Epoch: 6 of 6
Iteration: 252 of 252
Iterations per epoch: 42
Maximum iterations: 252

Validation
Frequency: 3 iterations

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

Accuracy
Training (smoothed)
Training
Validation

Loss
Training (smoothed)
Training
Validation

Fig. 30 AlexNet based CNN 2-class training process

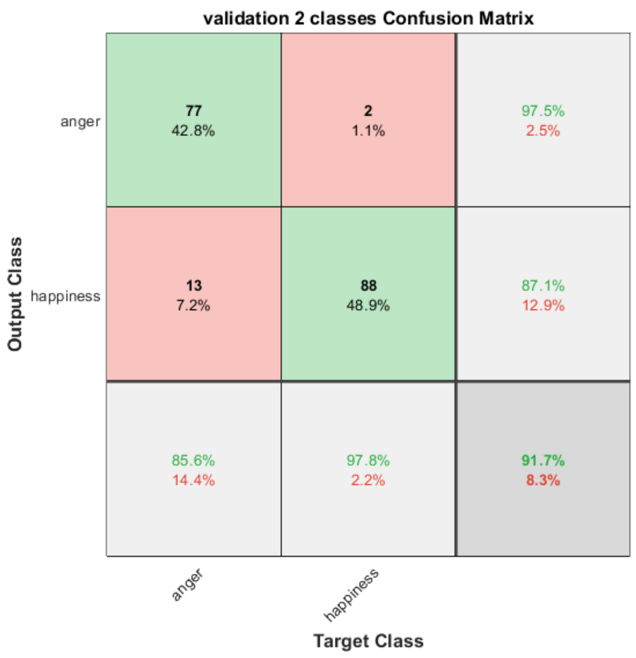


Fig 31. AlexNet based CNN 2-class confusion matrix

In addition, in Fig.32, are displayed several classification samples with their predicted labels, taken from the validation set.

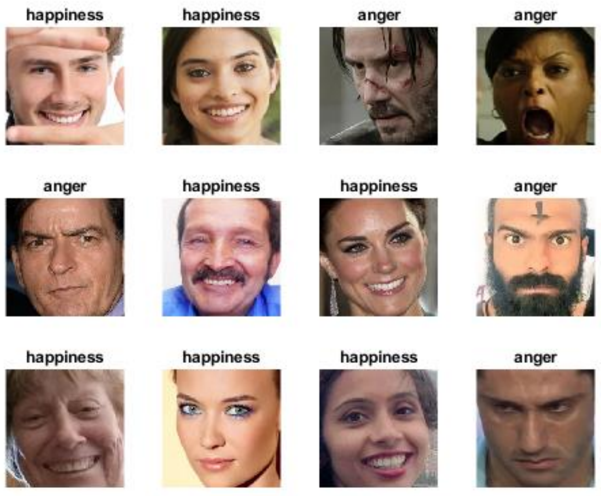


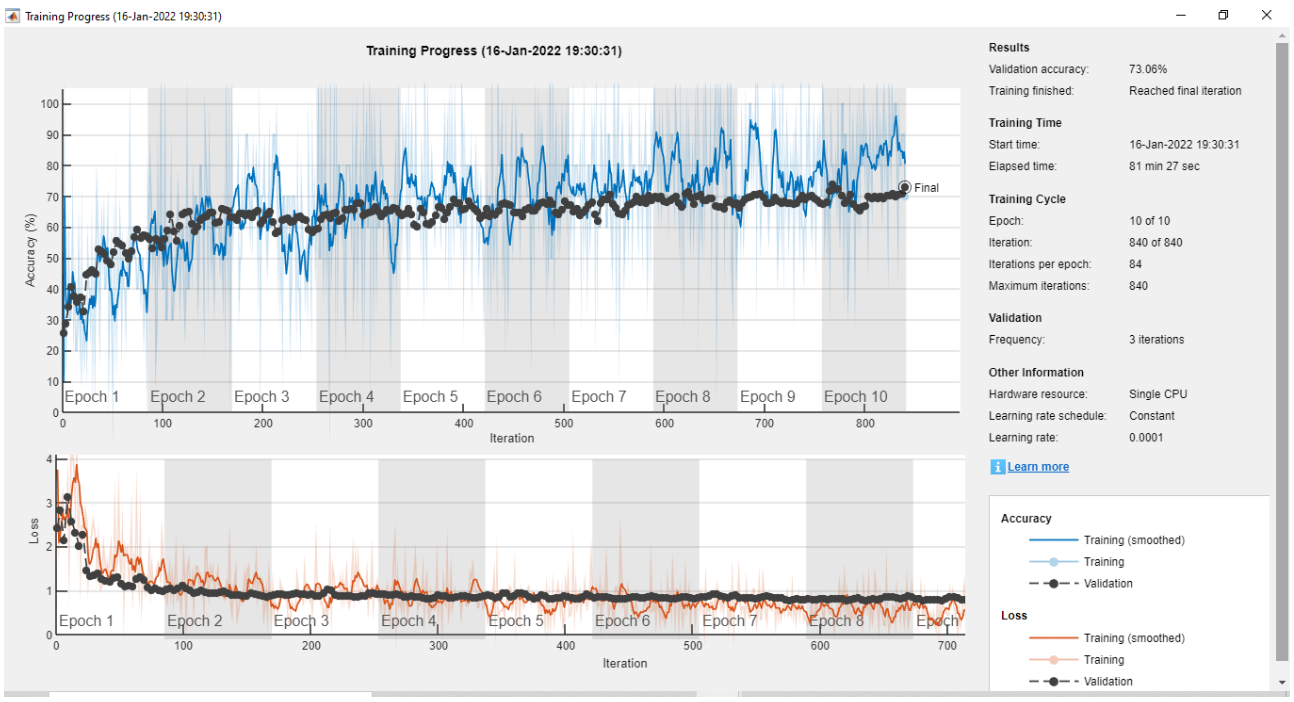
Fig.32 AlexNet based CNN 2-class classification samples

5.2 AlexNet fine-tune for 4-class classification problem

The aim of this section is the same of the section 4.2, classify all the available classes (anger, disgust, fear, happiness).
Starting from the CNN built for 2-class classification with AlexNet, some other experiment was made, in order the obtain good results also for this task.

Has been concluded that the number of epochs equal to 6 was no longer sufficient, and after some test it was found that with 10 max epochs good results were achieved (73% of accuracy).

In the Fig.33 and Fig.34 is possible to see the training process and the final confusion matrix that shows the results.


Training Progress (16-Jan-2022 19:30:31)

Results

Validation accuracy:	73.06%
Training finished:	Reached final iteration

Training Time

Start time:	16-Jan-2022 19:30:31
Elapsed time:	81 min 27 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

Fig.33 AlexNet based CNN 4-class training process

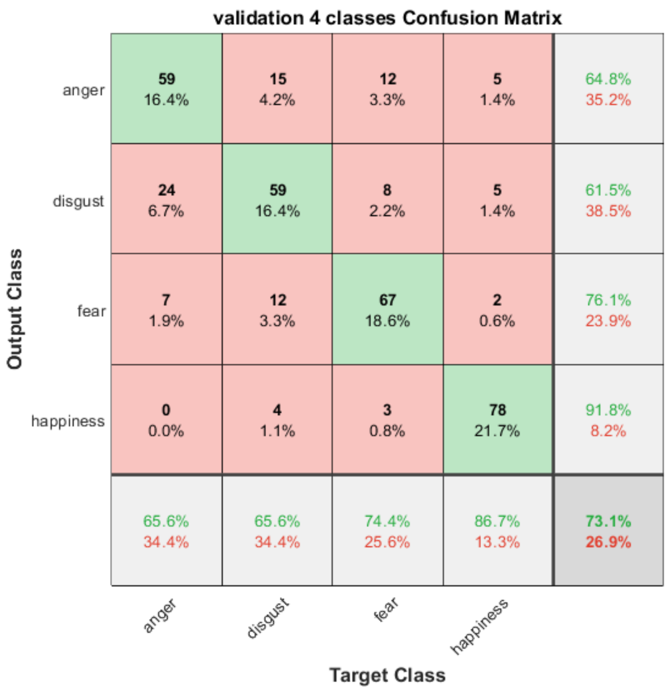


Fig.34 AlexNet based CNN 4-class confusion matrix

To conclude the section about CNNs some conclusion was made.

First, comparing the 2-class and 4-class emotion classification problems is possible to conclude that the workload is much more in the second case, not only for the number of classes to classify, but also because that disgust with anger is easily equivocal (it is possible to see by the confusion matrix).

The considerations on the produced CNNs are the following: for the easiest task (2-class classification problem) both, built from scratch CNN and fine-tuned CNN based on AlexNet, have performed in a good way, achieving satisfying results within a reasonable training time.
For the harder task (4-class classification problem) both CNNs have reached not very good results but acceptable, but the biggest difference was on the training time (81 minutes for fine-tuned CNN and 25 minutes for built from scratch CNN).

In conclusion, even though a build from scratch CNN is a very powerful tool, which is possible to change in all its parts, also using a pretrained CNN is a good way to solve classification problems without having to change a lot the main structure.