Intelligent Systems Project Report

Author:

Francesco Iemma

M.Sc. in Computer Engineering

Academic Year 2020/21

Contents

Introduction

The tasks performed in this project are the following:

- *3.1* "Design and develop two MLP artificial neural networks that accurately estimate a person's valence and arousal, respectively" and "two RBF networks that do the same thing as the MPLPs"
- *3.3* "Design and develop a fuzzy inference system to fix the deficiencies in the arousal dimension"
- *4.1* "Design and develop a convolutional neural network (CNN) that accurately classifies a person's emotion, based on facial expression."

The dataset at our disposal are two, one for tasks *3.1* and *3.3* and another one for task *4.1*. For what concern the first dataset, i.e. the one with biomedical signals for estimate arousal and valence, it is important to perform a cleaning of the data in order to obtain better performance for the neural networks that will be trained on it. This process, which is performed by the script `/matlab/data.m` is explained in the chapter 1.
After this chapter for each task is dedicated a chapter in which are explained the choices done and the results obtained in terms of performance.

Chapter 1

Data Cleaning

In this chapter we will see the data cleaning procedure performed in order to obtain better performance for the NNs. The steps done are:

- Remove non numeric values
- Remove outliers
- Balance the data among the different values of arousal and valence
- Features selection

All the procedure is contained in the file `/matlab/data.m`. The first two steps are performed thanks two matlab functions:

- `isinf(A)` that given a matrix returns a logic matrix is indicated if the correspondent element of the input matrix are infinite (1) or not (0)
- `rmoutliers(dataset, method)` that given a dataset remove the outliers found using the method specified in input that is 'median' by default (i.e. "Outliers are defined as elements more than three scaled MAD from the median. The scaled MAD is defined as $c \times median(abs(A - median(A))$")

Then after the first two steps there are the most interesting part: data balancing and features selection.

1.1 Data Balancing

The dataset is composed by samples and each sample contains biomedical signals: to each set of biomedical signals (that we will call *features*) correspond a value for arousal and a value for valence. The possible values are 7 (1, $2.\bar{3}$, $3.\bar{6}$, 5, $6.\bar{3}$, $7.\bar{6}$, 9), thus we can divide the dataset according 7 class for arousal and valence. The distribution of the samples among the classes is represented in the histograms in figure 1.1 and 1.2.

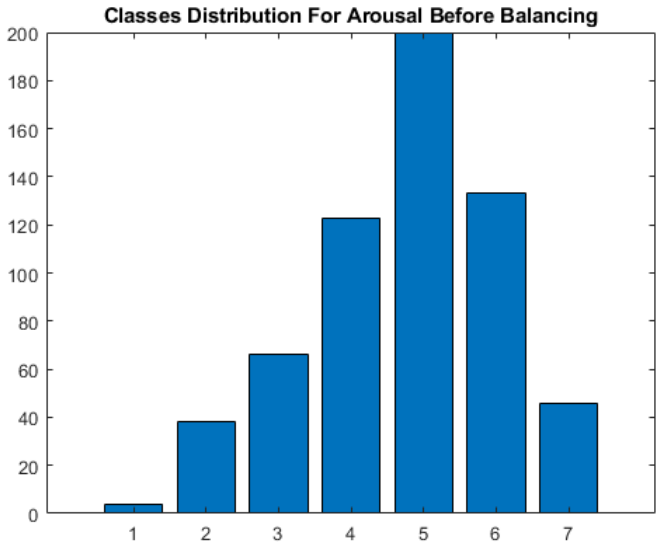

Figure 1.1: Classes Distribution For Arousal Before Balancing

Figure 1.2: Classes Distribution For Valence Before Balancing

As we can see the samples are heavily unbalanced, for this reason an algorithm to balance the data has been used. The algorithm is based on the concepts of undersampling, oversampling and data augmentation.
The steps are the following:

1. I augment the samples that belong to the majority class of arousal and don't belong to the majority class of valence and viceversa (i.e. the samples that belong to the majority class of valence and don't belong to the minority class of arousal).

2. I remove the samples that belong to the majority class of arousal and don't belong to the minority class of valence and viceversa (i.e. the samples that belong to the majority class of valence and don't belong to the minority class of arousal).

3. I repeat steps 1 and 2 for a $n = 40$ (40 after some experiments this is the number that gives the best results) times and for each repetition I compute the new majority and minority class both for arousal and valence.

4. After the end of the repetitions I perform an undersampling on the first class because it is unbalanced for what concern the valence. Thus I remove some samples from this class, after some experiments removing 30 samples results in a balanced distribution for both arousal and valence.

At the end of this procedure the data are balanced as we can see in figure 1.3 and 1.4.

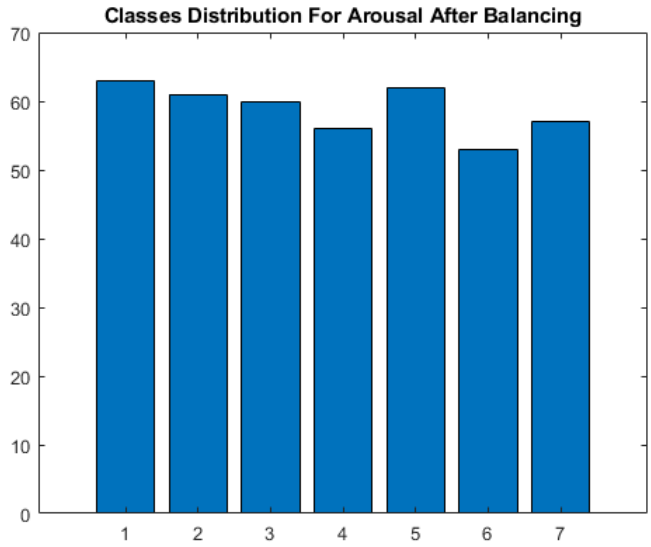

Figure 1.3: Classes Distribution For Arousal After Balancing

Figure 1.4: Classes Distribution For Valence After Balancing

1.2 Features Selection

Before the features selection is necessary to divide the data in two set: one for training and one for test. This is very important because if we use all the data to perform feature selection we have a bias because the test data have been already seen by the net.
Thus after the extraction of the holdout partition we perform x CAMBIARE times `sequentialfs` for arousal and x time for valence. Then we select the first `FEATURES_TO_SELECT` (constant set at the beginning of the script) features that appear most times in the different runs of sequentialfs, this operation is performed separately for arousal and valence.
At the end we save the data obtained into three `.mat` files:

- `/matlab/data/biomedical_signals/dataset_cleaned.mat`

  It contains the entire dataset without infinite values, outliers and with balanced class distribution.

- `/matlab/data/biomedical_signals/training_data.mat`

  It contains a `struct` with the training input (only the selected features) and the correspondent target output.

- `/matlab/data/biomedical_signals/test_data.mat`

  It contains a `struct` with the test input (only the selected features) and the correspondent expected output.

Chapter 2

Neural Networks

In this chapter we will see two types of neural networks that resolve the same problem, that is to estimate the values of arousal and valence given a set of biomedical signals.

2.1 Fitnet

2.2 RBF

Chapter 3

Fuzzy Inference System

Chapter 4

Convolutional Neural Networks

In this chapter the development and training of a CNN based on the pre-trained Alexnet which has the aim of classifying facial expressions. The possible classes are four:

- Happiness
- Anger
- Disgust
- Fear

Two CNN have been developed (both are based on Alexnet). The first one is able to classify images in two classes (happiness and anger), instead the second one is able to classify images in all four classes.

4.1 CNN For 2-Classes Classification Problem

The matlab code for this CNN is in the file `/matlab/cnn_2classes.m`. Due to the fact that the images of happiness and the ones for anger are very different we are able to obtain good result (accuracy equal to 82.67%) without a proper selection of the images. In fact as first experiment 500 images for each class are selected at random from the dataset. This experiment return the results shown in figure 4.1:

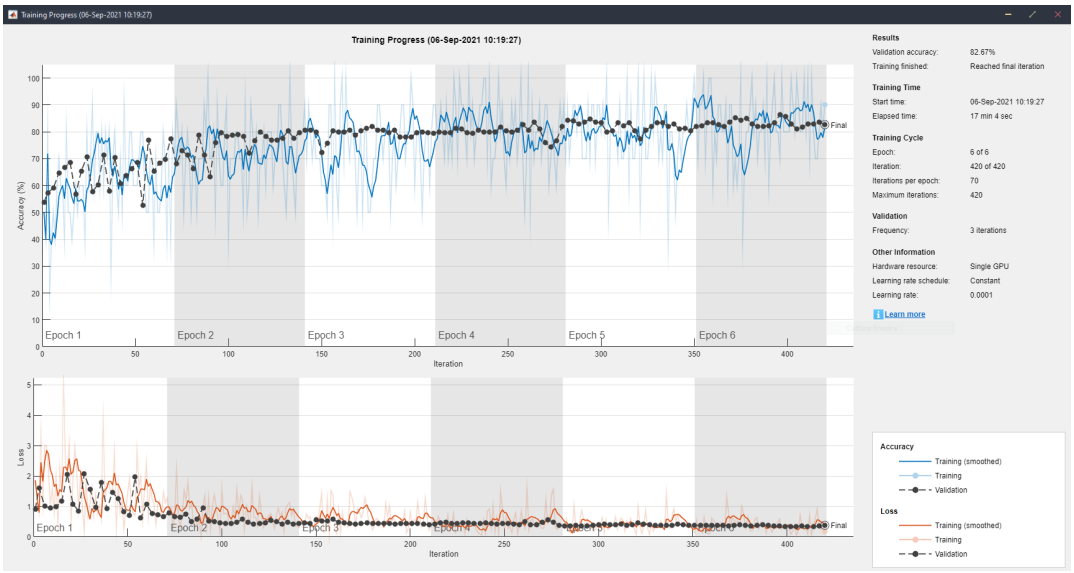

Figure 4.1: First experiment: no images selection

Then a selection of the images has been performed and from the 500 images only 300 have been selected. The selection consist in the removal of the images with ambigous facial expressions. After this selection the results in figure 4.2.

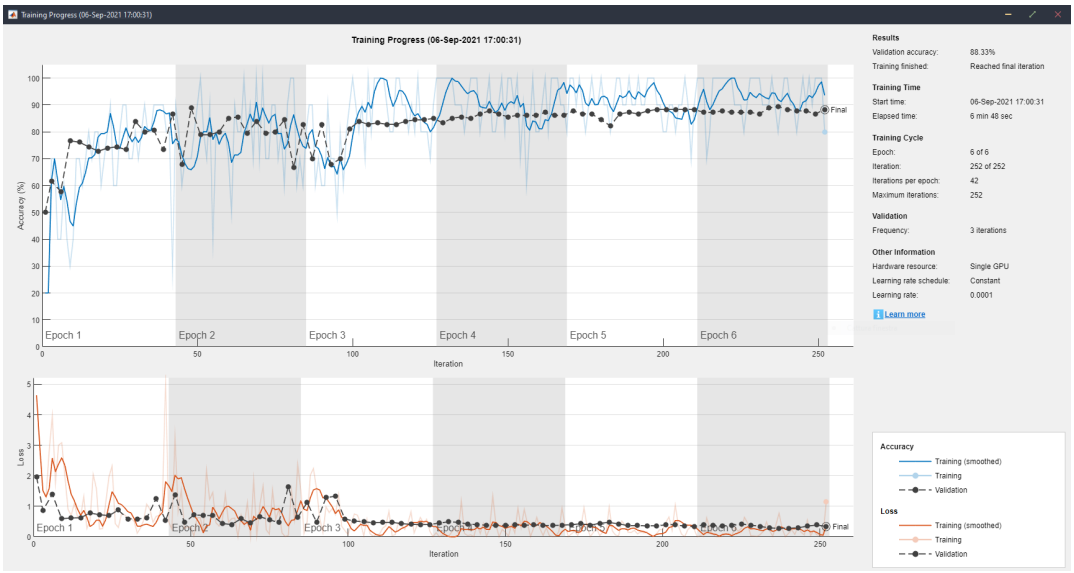

Figure 4.2: First experiment: after selection

Thanks to the selection of the images the accuracy has increased by $\approx 6\%$: from 82.67% to 88.33%.

4.2 CNN For 4-Classes Classification Problem

The matlab code for this CNN is in the file `/matlab/cnn_4classes.m`. As in the previous case two experiments have been performed, the first one with 500 images per class selected at random, the second one with 300 images selected removing the images classified wrong or with ambiguous facial expressions.
The result for the experiments are respectively in the figures 4.3 and 4.4.
Analyzing the two results it is possible to see that, thanks to the selection, we have improved the accuracy from 58.17% to 63.61% that can be considered a reasonable result for a classification problem with 4 classes.

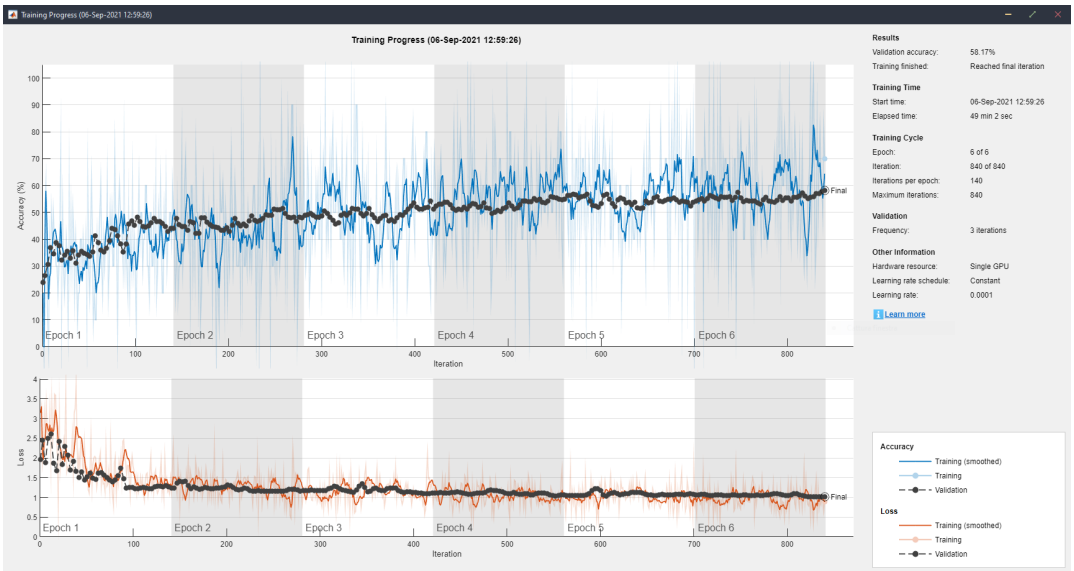

Figure 4.3: First experiment: no images selection

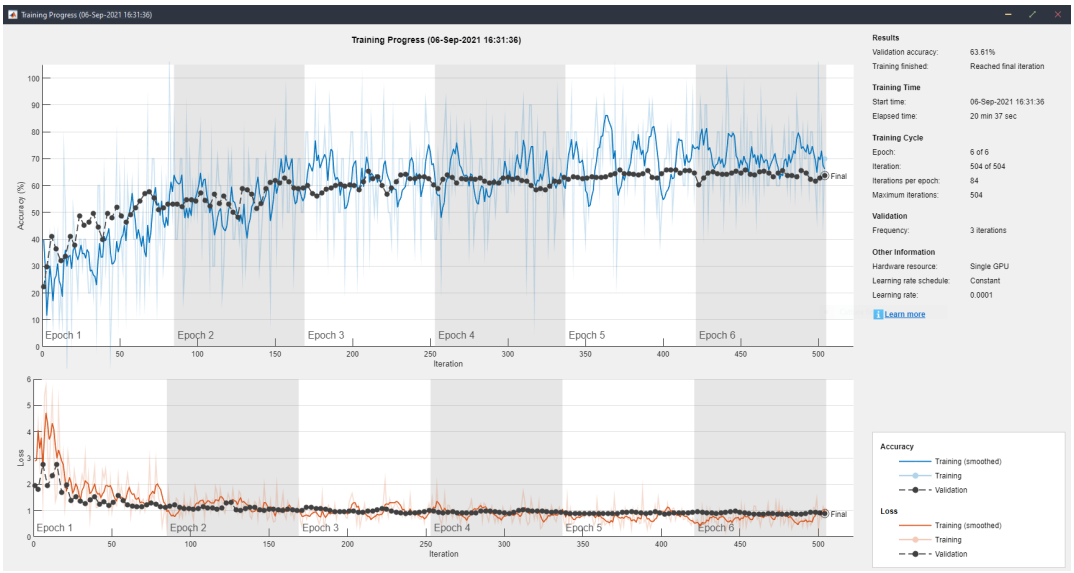

Figure 4.4: Second experiment: after images selection

4.3 Design Choices

The dataset of images is very huge, but in order to maintain a reasonable training time only 500 images have been selected at random for each classes, then in order to improve perfomance a selection has been performed reducing this number to 300 images per class.
In all experiments the images used in training have been augmented:

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

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);
```

The reason why also the image for validation are augmented is because the method for augmentation (that is `augmentedImageDatastore`) automatically resize the image in order to fit the input size requested from AlexNet. In fact you can see that there is no `imageAugmenter` in the `augmentedImageDatastore` for validation because it is used only for resizing, instead in the case of test images the same method perform both resizing and augmentation.
For what concerns the CNN architecture only the last three layers of AlexNet have been removed and substituted with other three layer:

```
net_layers = [
original_layers
fullyConnectedLayer(
  numberOfClasses,'WeightLearnRateFactor',20,'BiasLearnRateFactor',20)
softmaxLayer
classificationLayer];
```

4.4 Final Results

At the end after some experiments the best architectures is the one in the two matlab scripts. Some hyper-parameters have been changed (for instance the initial learning rate, the weight learn rate factor, the bias learn rate factor ecc..), but at the end the parameter that improve the performance is the the maximum number of epochs, in fact increasing this parameter the

training of the network require more time but it is more effective also because at every epoch the data are shuffled and so the presentation order of the images is different. The results obtained are the ones in figure 4.5 and 4.6. For what concern the test accuracy:

- Test Accuracy 2-classes problem: 97.08%
- Test Accuracy 4-classes problem: 80.31%

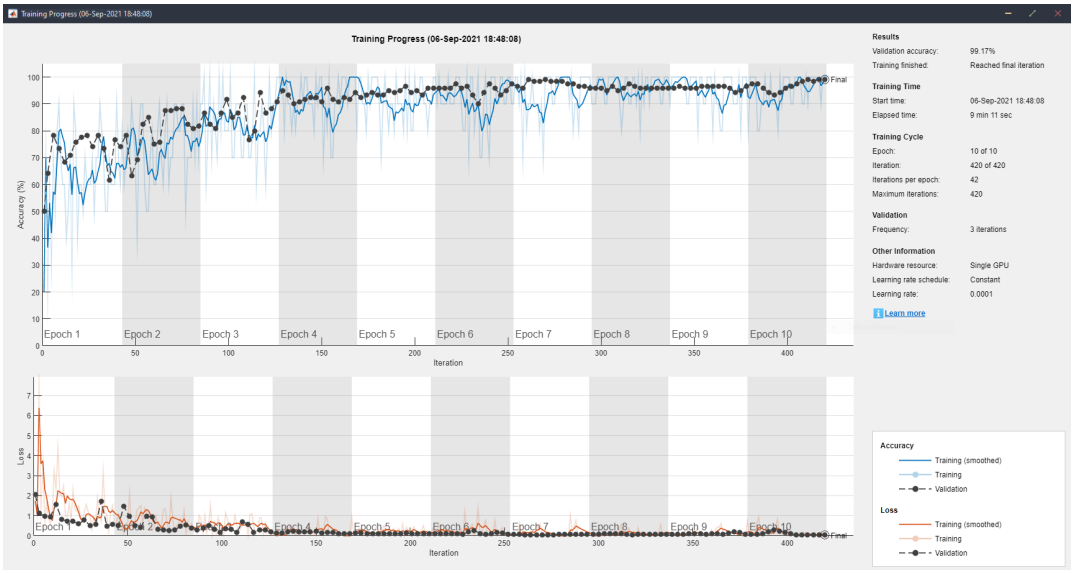

Figure 4.5: Final result for two-classes classification problem

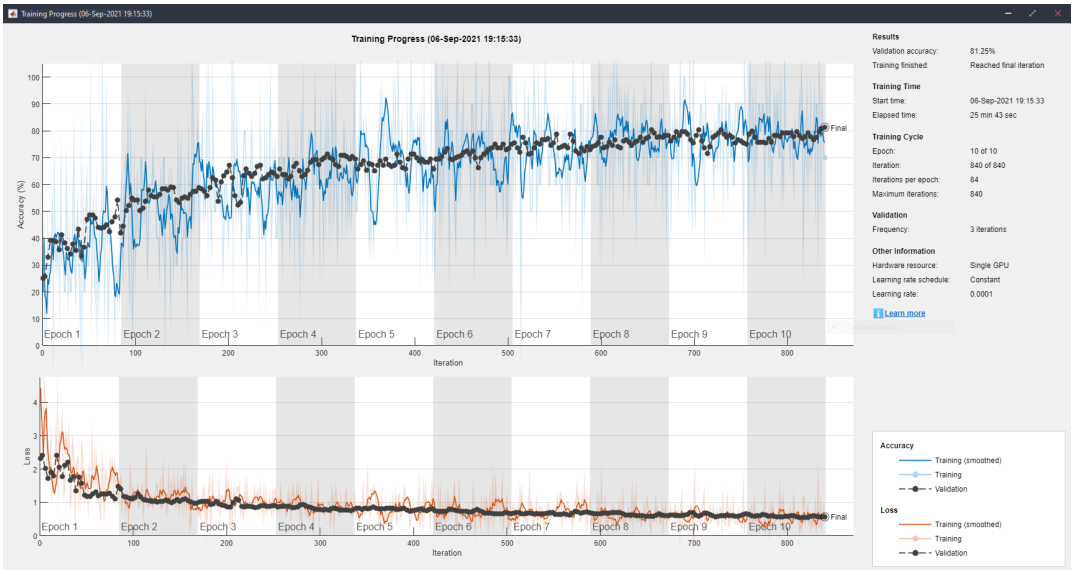

Figure 4.6: Final result for four-classes classification problem

For what concern the confusion matrix the results in figure 4.7 and 4.8 have been obtained.

Figure 4.7: Confusion matrix for test images in 2-classes classification problem

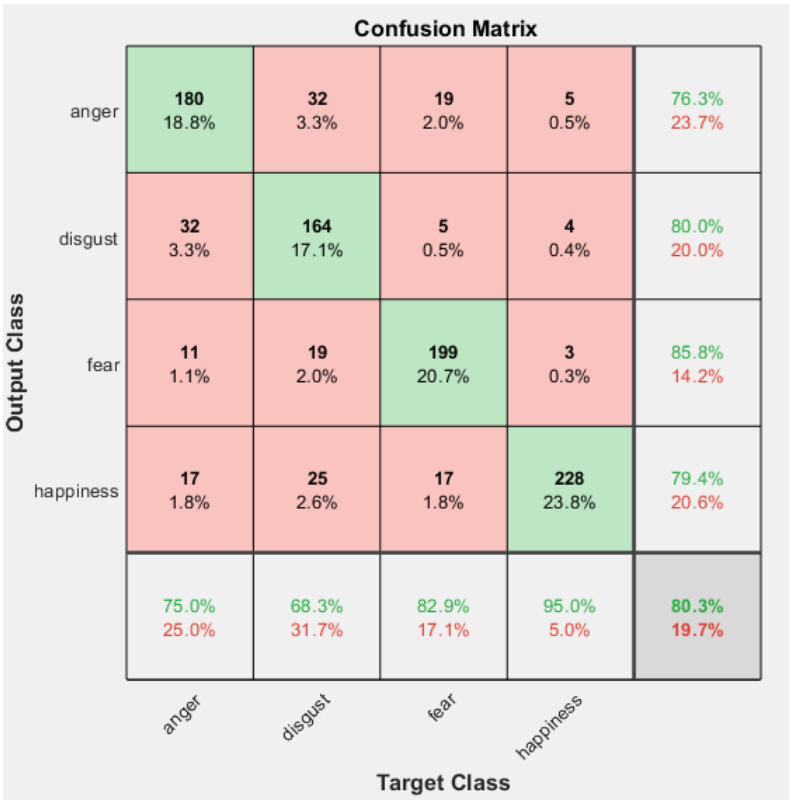

Figure 4.8: Confusion matrix for test images in 4-classes classification problem