Face and Number Recognition on Images and Videos

Project Report and Source Code - Computer Vision IMN460

By Francisco Javier Medel Medina

Introduction: This project report describes the Application of Machine Learning and Neural Networks techniques applied in the field of Computer Vision. The main objectives of this project are to detect faces in individual Images, group of people and Identify the correct individual in individual Images or group Images. We will also perform number recognition on Individual image and video where each person is holding a number that will help to label the individual, which in turn will help train the classifiers. To reach to goal we must implemented different feature extraction on the Images and also apply a variety of filters to recognize the numbers. The description of how this was implemented is explained in the next sections.

Motivation: This is the real scenario: Imagine that you went to a party of your best friend and you have you ever been introduced to new people, but after hanging out around, you forget few of their names? It is complete normal, we are so good recognising faces that we see before but remembering their names can be difficult. But you are a Data Scientist and you know one of the coolest technologies of Machine Learning. Then you have the idea of creating an app that can recognize people that you met before and prompts you with their name, to help you avoid asking "I'm sorry but what is your name? I forgot it".

1. User guide: For the scripts name

1.1 MATLAB scripts that create the **Models**

Matlab_CNN.m	- generates the CNN
Matlab_CNN_Emotions.m	- generates the CNN used for Emotion Detection
Matlab_MLP_HOGF.m	- create MLP with HOG feature extraction
Matlab_MLP_SURF.m	- create MLP with SURF feature extraction
Matlab_SVM_HOGF.m	- create SVM with HOG feature extraction
Matlab_SVM_SURF.m	- create SVM WITH SIRF feature extraction

1.2 MATLAB scripts to make the **pre-processing** and the auxiliary functions.

Pre-processing_Create_Subfolders	- create the subfolder after each test execution
Pre-processing_Faces_Group_Image	- get Faces from Group Image Using CNN
Pre-processing_Faces_Group_Video	- get Faces from Group Video Using CNN
Pre-processing_Faces_Individual_Image	- get Faces from Individual Image Using CNN
Pre-processing_Faces_Individual_Video	- get Faces from Individual Video Using CNN
Pre-processing Resize	- resize all the face data source image to [128 128]

2. Function Explanation (Inputs and Outputs)

2.1 Face Recognition:

This function recognizes the face in individual Image or group Images. The input parameters are: I, which is an Image, feature Type can either be SURF or HOG and the third parameter is classifier name which can be SVM, MLP or CNN, for CNN you do not need to specify feature type because the CNN extracts features by its own from the data.

Face Recognition Function: P = RecogniseFace(I, featureType, classifierName)	
Inputs: I: Indivual Image, Group Image featureType: SURF, HOG classifierName: MLP, SVM, CNN (feature type is no needed)	**Outputs**: The method will return a matrix with id, x, y z. Where id is the label of the face found in the image, x is the location of the person detected in the individual image or group image in the x axis and y is the location of the person detected in the individual image or group image in the y axis. Z is the label for the emotion. P: [id x y z …] Example: P = 37 244 353 0 40 512 723 1
Comments: This function returns an Image labelled with the number over the face of everyone in the individual image or group image, you can use the same function in images or videos indistinctly. Z can take values from 0 to 3 where 1 = happy, 2 = sad, 3,= surprised and 4 = Angry.	

2.2 OCR:

This function takes as an input an Image and return a message with the number found. This function also works in videos where each frame is taken as an individual Image. We need to wait until all the video is processed.

OCR: detectNum (filename)	
Input: **Filename**: Image of Video file	**Output**: Message: `'This video is for the person num: 1'.`
Comments: To use the function and apply it to videos, the function works in the same way, but for videos, where not all the frame have good quality, the program needs to process all the video to find the correct number.	

3. Task 1 – Face Recognition (50%):

3.1 How the face detection works

To recognize the faces on the Individual Images and Group Images, I used the function CascadeObjectDetector, the cascade object detector is a function in MATLAB that uses the **Viola-Jones algorithm** [1] too detect people's face, upper body, noses or mouth in an image. To implement this method we set up the properties of ClassificationModel as FrontalFaceCART, we are looking that are upright and forward facing, also the *MergeThreshold property* that defines how sensible the detector is to the area of interest, a low value will recognize any object that can look like a face and a high value that will defect just objects that are well define in the images. I was playing with this number and I found that the correct one for **individual photos is 25.**

```
FaceDetector = vision.CascadeObjectDetector('FrontalFaceCART','MergeThreshold',25);
```

3.2 How to solve the problem of creating the Classifiers:

We have 53 different folders where there we have image and videos for everyone, we are going to use this data source to create our face data source that will use to train all the models.
We have the next type of data sources- Individual Photos, Individual Videos, Group Photos and Group Videos. We will extract as many face as possible from every source, for everyone to train the models. In the world of Machine Learning more data is always better, with more data we extract more features and create a more reliable model.

Figure 1. shows a flow diagram of what the process looks like. First step is to load the Image, find the face in the Image by applying the CascadeObjectDetector function, extract the region where the face is found, get the coordinates, with these coordinates create a new sub Image from the original one. The next step is to save all these faces in the correct folder to create a Face Database Gallery. Using this Face Database Gallery to create the test and train set, extract features that the models SVM and MLP are going to use to create the classifiers. For CNN, no feature extraction is need, we can see this in the section 5.5

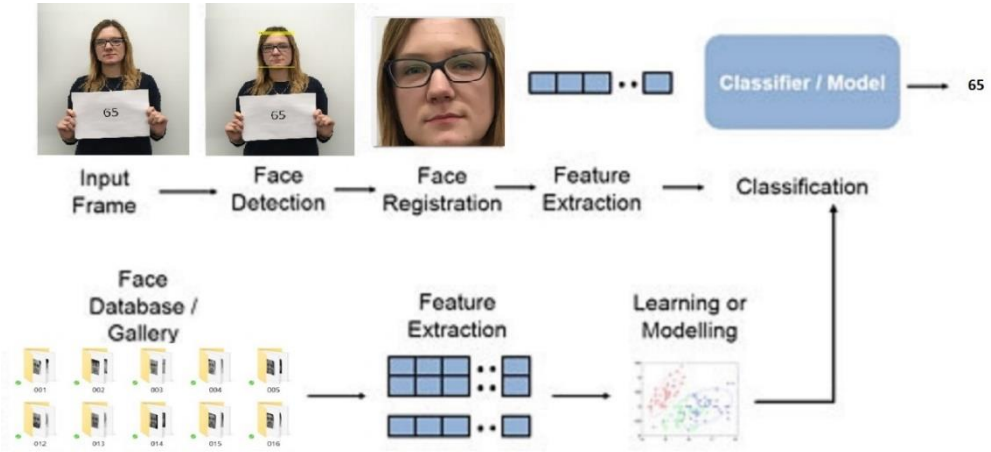


Figure 1 Model Creation Process

4 Data Pre-processing - Create the database of Images (10%)

4.1 Get face from Individual photo

The first step is to extract the faces for all the individual images Figure 1.1, you can find the code of this step in the script **Pre-processing_Faces_Individual_Image.** While we were iterating across all the folder we found some challenges.

Issues in the Face Extraction.

4.1.1 No all the image is the same size.

Solution Implementation: One the image is read and sent to the method, I evaluate the same of the image if the image is bigger than 2500 pixels in X and Y, I resize to a scale of 20% if not I resize the Image any way to a scale of 40%. We do not need big images and it also helps to improve the performance.

4.1.2 When the image is loaded it is loaded horizontally.

Solution Implementation: I first applied the Face Detection in the image to validate if the image is vertically or not, if the method does not return a face, I rotated the phot in 270 grades and I applied the face detection again to the new image in vertically positon this time.

4.1.3 The face detector is detecting some objects in the individual images that are not faces.
Solution Implementation: As we are using the method CascadeObjectDetector I noticed that in some Images we are getting more than one face even when the object is no face, to avoid this, I updated the threshold to be more restrictive by applying a value of 25.

Figure 1.1 Faces from Group Image

4.2 Get face from Individual video

I resuse the script from of the function "get face from individual image" only that this time, I am going treat each frame as an image, I create this new script "Pre-processing_Faces_Individual_Video". There are a few people from 64 to 70 labels that do not have video, this is a problem because without the video, the number of faces for these people are so limited to around 4 to 6 images for each person.

Issues in the Face Extraction

4.2.1 Need more faces for the people that does not have video.

Solution Implementation: I create a script that takes the original data source for these few people, I apply a rotate on the image between 10 grades to left and 10 grades to right, then I apply a blur from 1 to 7 divided in 10 scales. And then to ensure that we still have a face, I apply the face detector on the new images. As a result, for a simple image, I got (10 X 10 = 100) 100 new images per individual.

4.2.1 Create an extra category for unknow people

Solution Implementation: Once I get all the face from a group Image, I found some people that do not have category. I decide to take all this new people and put in a new category '999'. This like defatul category for people that do not exist in the original data source for 53 categories. Aa result we have now 54 categories.

4. 3 Get faces from Group photo

To get the faces from the group photo, I implement the method CascadeObjectDetector to recognize all the faces in the image group, then I iterate them one by one to create a sub image for the size [128, 128] and I save it in a folder to user to train the models. I created the script "Pre-processing_Faces_Group_Image".

Issues in the Face Extraction

4.3.1 No all the faces are detected.

Solution Implementation: The main problem here was that no all the faces are detected in the group Image, to solve this, I tried detecting with different thresholds and I found that the best one is with a threshold of 15 for the CascadeObjectDetector. Even when not all the faces that are in the behind section of the photo are detected, it still shows some reasonable results.

4. 4 Get faces from Group video

To get the faces from the group video. I reuse the function Pre-processing_Faces_Group_Image and create a loop to read the video and use each frame as an image. It results in a new script Pre-processing_Faces_Group_Video.

Issues in the Face Extraction

4.4.1 Iterate across all the videos in all the folders: The main issue was to iterate across all the videos in a folder and to not over write the individual images that the loop creates while it is running. Solution Implementation: I must take care on the iteration of all the folders and videos. I use the name of the video and global variable to count the number of faces by group images. Also, I must work on how to get all the videos in a subfolder and create a list to iterate across them.

4.5 Face Database / Gallery is created

The last results after applying all the pre-processing techniques and take all the data from images and Videos. We got the Face database / Gallery that will be using to train the Model Figure 2.

After applying all the function for data pre-processing, we got that the largest number of faces that is 2576 for the face number 045 and the smallest number is 445 for the face 001, this smallest value of 445 is our threshold Figure 3., because we are going to randomly select 445 images from each label to train our models.

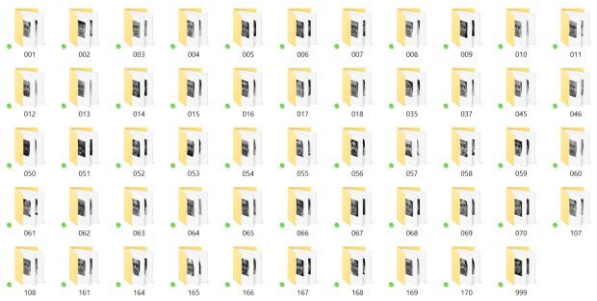


Figure 2. Face database / Gallery

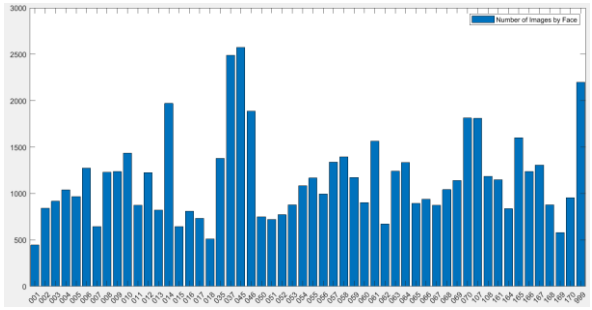


Figure 3. Histogram of Number of Faces

4.5.1 Challenges, What methods did I try

The main challenge was to get enough data from people that does not have more that few individual images and no video. I try to implement the Data Augmentation method that MATLAB provides through imageDataAugmenter. I got some issues to implement this method and also, I wanted to see the images in folders before to train the model, for this reason I decided to create my own method that rotates and blurs the existing images to create more training samples. The function is in the script "**Pre-processing_Faces_Individual_Video"**. This new data really helps to avoid overfitting and create more accruable models. Another challenge was that, At the begging I started the images with a size of [256, 256], but I notice that with that size, to train the model will take so much time, them I decided to create a function to resize the images faces that already have been extracted and I create the script "Pre-processing Resize", this script resize all the images of the subfolder in a path. I like this function as it is quite powerful, useful and simple. To run again the methods to extract all the faces will take more time, that just run this new function.

5. Features Extraction Technique (HOG, SURF) and 3 Classifiers MLP, SVM, CNN

5. 1 Extract histogram of oriented gradients (HOG)

The histogram of oriented gradients (HOG) is a feature extraction method used in computer vision for object detection. The technique takes a reference the gradient orientation in an image. HOG is computed on a dense grid and return shape information by regions of the images.

5.2 Speeded-Up Robust Features (SURF)

The SURF algorithm is future extraction technique and It is based on the same principles of SIFT, but details on the process detection are different. The algorithm works in three different states: local neighbourhood description, interest point detection and matching. As a result, the standard version of SURF performs several times faster than SIFT in some scenarios, SURF is more robust that SIFT when SURF use different image transformations.

Figure 4. Surf features Example

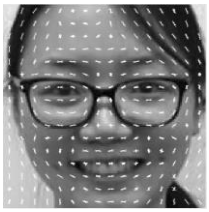


Figure 5. HOG features Example

Bag of features.
It is used in computer vision to treat each feature as word. It is usually implemented in image classification. This is one the reason why we must create one for each type of feature to be used by the Classifiers MLP and SVM.

5.3 Multilayer Perceptron (MLP)

The multilayer perceptron is one type of feedforward artificial neural network into the world of machine learning. The MLP must have at least 3 layers of nodes. The hidden and the output layer use and activation function the input layer only receive the data and transfer this data to next layer. Basically, the structure of the MLP consist on:

- An input layer.
- One ore more hidden layer that are interconnected
- An output layer.

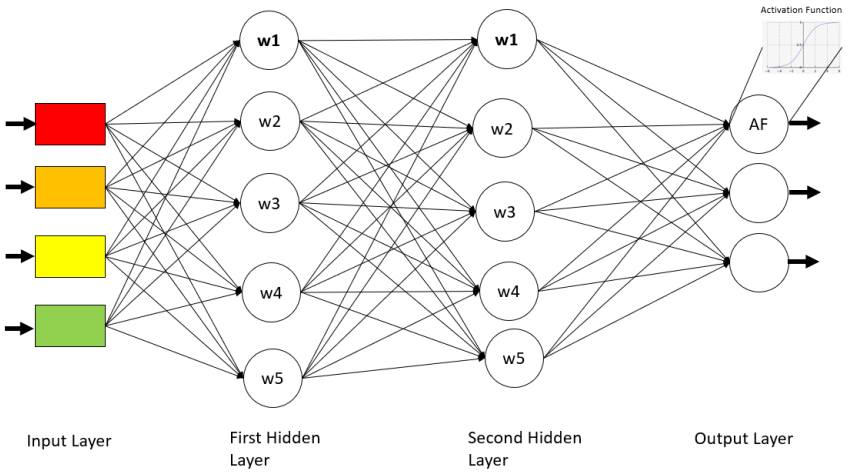


Figure 6. Architectural graph of a multilayer perceptron with two hidden layers

Back Propagation: algorithm – 2 passes:
Forward pass: Calculates the values for the of the output layers from the input data through all the neuron, from the first layer to the last layer.
Backward pass: Starts at the output layer, refers to the process of counting weights (learning factor) for each neuron, from output layer towards the first layer (backward direction). The weights are updated to minimize the error for the next iteration. The goal is to repeat this process until convergence or a number or iteration is reached.

5.4 Support Vector Machines (SVM)

Support vector machine is a binary classifier. Multiple classes can be classified using one SVM per class:

• One against all: It is a term introduced by Vladimir Vapnik in 1995 [5]. It is defined when a data point would be classified under a certain class, only if that class's SVM accepted it, and all other classes must have rejected it.
• Pairwise: Each pair has a class; the classes are trained to separate must as they can from each other.
The main idea of support vector machines is given an accountable method for separating the data "optimally" into the classes [6]. Taking as a reference the vectors as a separation measure.

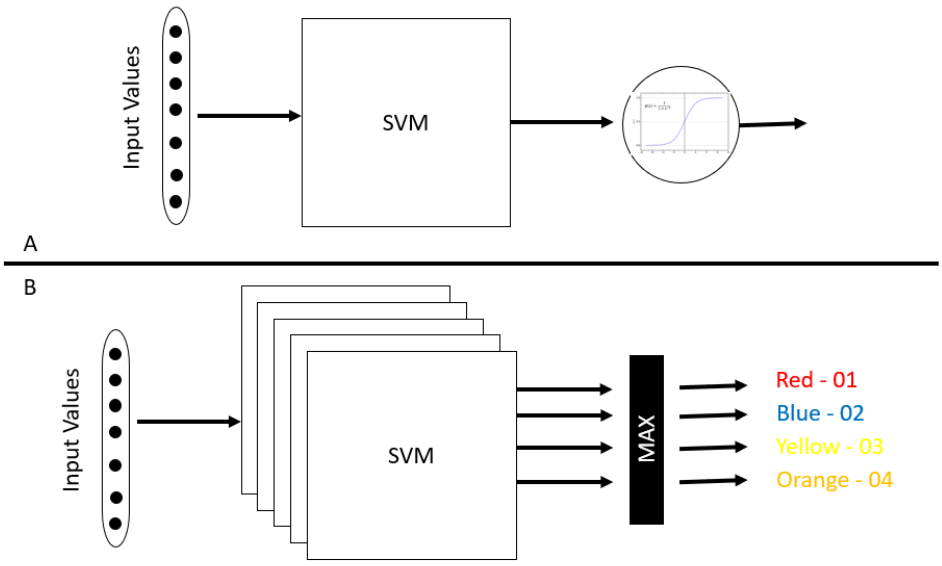


Figure 7. Binary Classification and Multiclass models for support vector machines

5.5 Convolutional Neural Network (CNN)

A convolutional neural network (CNN), CNNs are mostly used to recognize objects and perform object detection in images. CNNs does not need to feature extraction, because CNN lean itself this features with the help of layers and pooling. The use of CNN has been taking a lot of attention by the high power processed that we have these days and the easy way to train the CNNs with good level of accuracy. The CNN has three important aspects [10][11].

- No need manual feature extraction.
- Produce results with high level of accuracy.
- One of the mayor advantage is that CNN can be retrained to learn new features applying transfer knowledge.

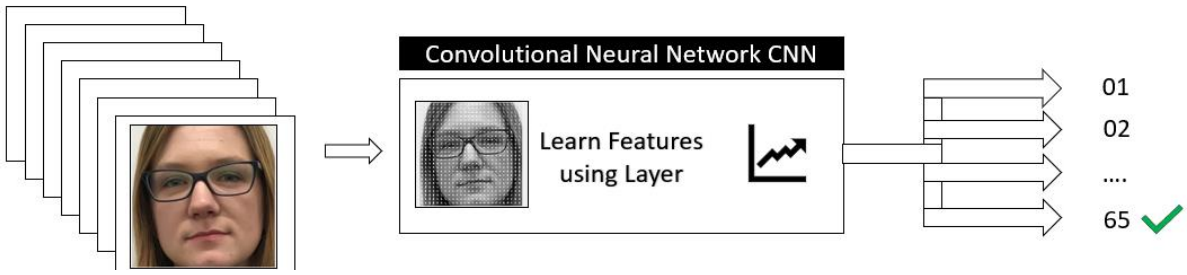


Figure 8. CNN Structure.

Usually a convolutional neural network has tens or hundreds of hidden layers where each layer learns different features of the image. One method is to apply Filters to each training image with different resolutions, the output of each layer is convolved to be used as the input to the next layer Figure 10. The filters start learning simple features, edges or corners and the next layers learns mores complex features that will help distinct to recognize the object from similar objects.

One to the best ways to create a convolution neural network is from scratch, but this approach need high quantities of data also the data science must define the hyperparameters of the networks structurers as the number of layers, learnings weights, activation function, which kind of filters to extract the correct features with other tuneable parameters of the neural net. I decide to create the CNN from starch using or face database images. In the Figure 10 you can the structure of the CNN layers.

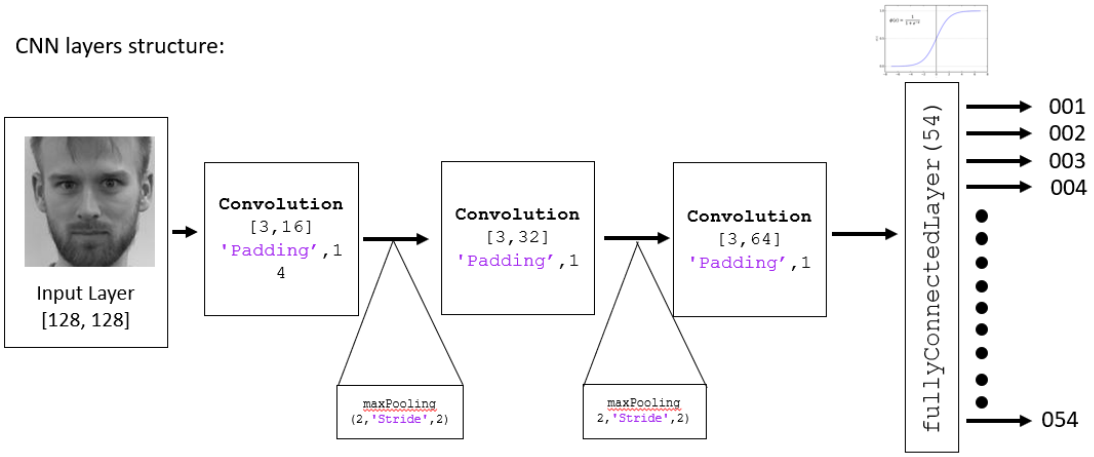


Figure 10. Architecture Layers

Once we have the structure to train the CNN in MATLAB. We need to run the script but with the tuned hyperparameters. I just set up the MaxEpocs to 1 and the MiniBatchSize to 64 and ask MATLAB to display the training process window as shown in Figure 11. IN MATLAB 2018a [8] To complete this CCN takes just 12 mins and 4 secs and 375 Iterations. I tried before to train the same model but in MATLAB 2017b and the same CNN took around 35 mins. Then the upgrade from the version 2017b to 2018a really helped in the creating the model.

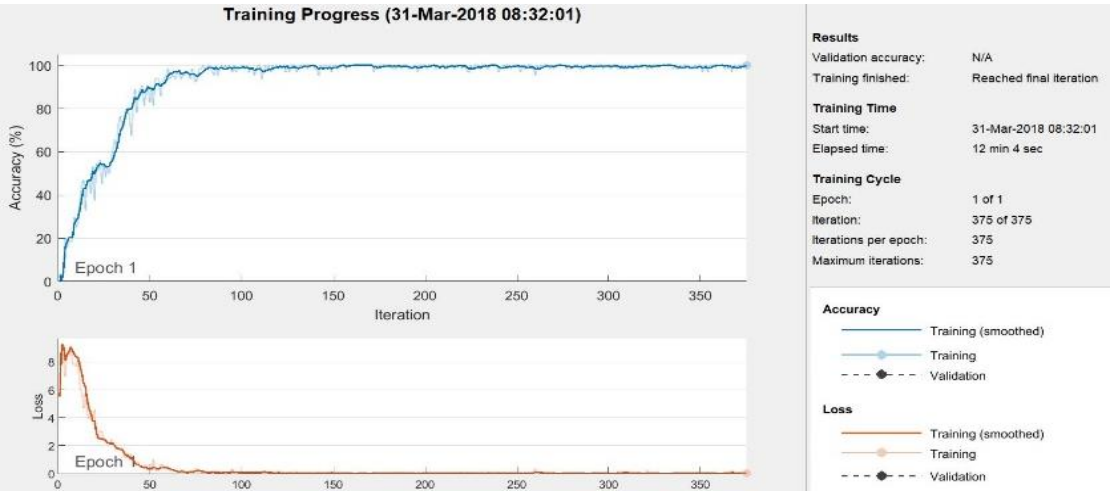


Figure 11. Training Progress CNN

Model	Future Selection	Train Set	Test Set
CNN		97.67	96.98
MLP	HOG	97.48	96.18
MLP	SURF	99.49	99.12
SVM	HOG	99.00	98.00
SVM	SURF	99.00	98.00

Figure. 12 Performance Models

6. Confusion Matrix: The Figure 13 show the confusion matrix for the best Model MLP-SURF on the Train set. We can see how almost all the categories are getting between (306-312) images correctly labelled of the 313 of the Train Set. Particularly in the category 999 number was 254. But this is intuitive because this category holds faces from few different people, even then this is a good accuracy under this circumstance.

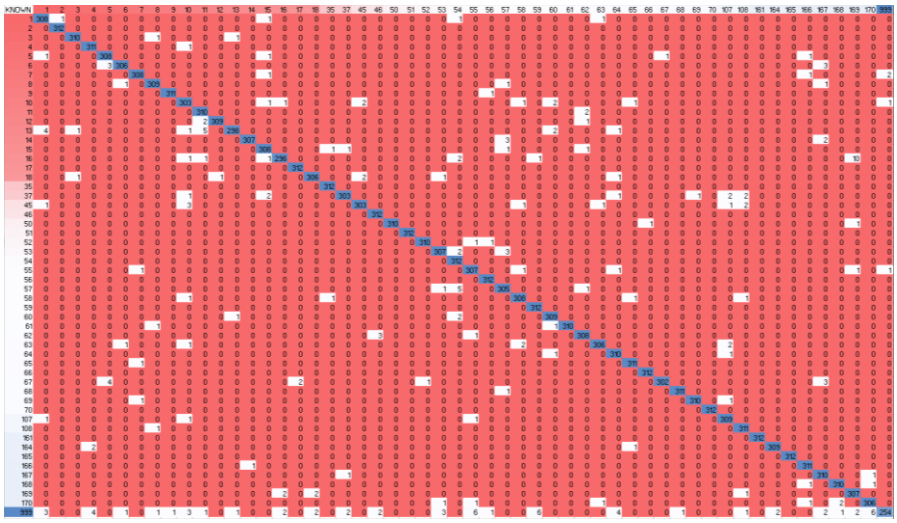


Figure 13. Confusion Matrix MLP - SURF

7. Final Results on the group Image of the Classifiers MLP, SVM and CNN

In the Figure 14, we can see what the results after applying any classifier on the group Images. I tried with 5 different models[CNN, SVM-SURF, SVM-HOG, MLP-SURF, MLP-HOG], I cannot see and different in the label result. I assume that this is because the accuracy of all the classifiers are over 97% and there is no difference that I can perceive. In the Figure 14.1 we can see an example of incorrect classification for the category 7 a face for the label 15, even when the accuracy are so high the model are no perfect.

Figure 14. Final Results for the CNN in Group Image

Figure 14.1 Incorrect Pediction

8. Innovation for the Couse Work

I found the results of labelling the correct person on images and Videos so interesting, with a high level of accuracy in the test and the train set Figure 12, For individual images that existed in the image data base, I was wondering how accurate can this method be if we can capture a real time video input taking into account environment factors such as the light or movements of a person. I tried to produce this **Face real time detection** Figure 15 , it shows good results when the person is just in front of the camera and the room is limited in terms of space.

Fig 15. Face Recognition in Real Time

9. Conclusion Analysis and interesting patterns.

As you can see in the Performance table Figure 12 that contains the accuracy results of each model on the train and test set, in all the scenarios the accuracy was over 95%, even for SVM with any feature, HOG or SURF the results ere almost 100%, This result shows how powerful the future extraction combined with a classification method are, as they really provide excellent results Figure. 12

But what happens if we apply this classifier in really time setting up the web cam, recognize a face of an individual and try to label. We perform this exercise and even when the CNN model was the one that we use to perform this exercise really show good results identify the correct individual that already exist in the database image, for those individual that the classifier does not know, it labelled them with '999', The classifier deals with external factors such as the light and when the person moves the face too fast in front of the camera (we did not try with more professional equipment the resolution of the web camera was just 680 x 480) and the time of software's reaction, we can see some interesting results.

One way to perform a real time classifier, for example- for the face that the model recognizes with an accuracy over 97% we can take this new image and apply transfer knowledge and retrain our CNN to have a better model, after a couple of iteration we can expect so much better results for a real time interaction. This can be a good approach for future work
Missing

10. Task 2 - OCR (25%)

10.1 Problem solution.

To tackle this problem, I decide to cut the image into a small section to apply the function OCR that MATLAB provides. I faced a lot issues to approximate to the correct area of each photo, the problem is that some photos were taken so far and some were too close, and the person is holding the paper with the number in a high or low position. This was a big problem because there is not a consistent area in all the image to apply the OCR function.

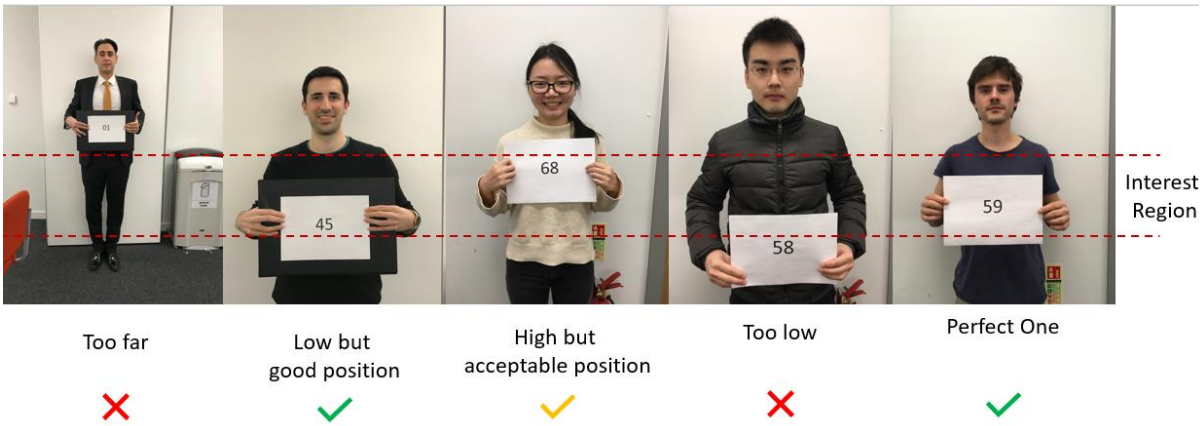


Figure 16. Interest Image for OCR detector

As we can see in the sample image, My plan was to find a patter that is consistence in all the images, Then I thought on apply the face recognition and create an area (this approach limits the solution to only solve this problem "someone holding a number and identify this number"), we know that there are faces in all the cases over the paper and this area is going to be consistent in all the images, even when the photo is far away from the camera.

10.1 Getting the Interest Area

10.1.1 First Try to get the Area: Applying the ClassificationModel: UpperBody

Parameters for the classification method: I tried to use the Upper Body classification model but when I was doing some test on the image, I realised that the classifier was not taking the same area in the images.

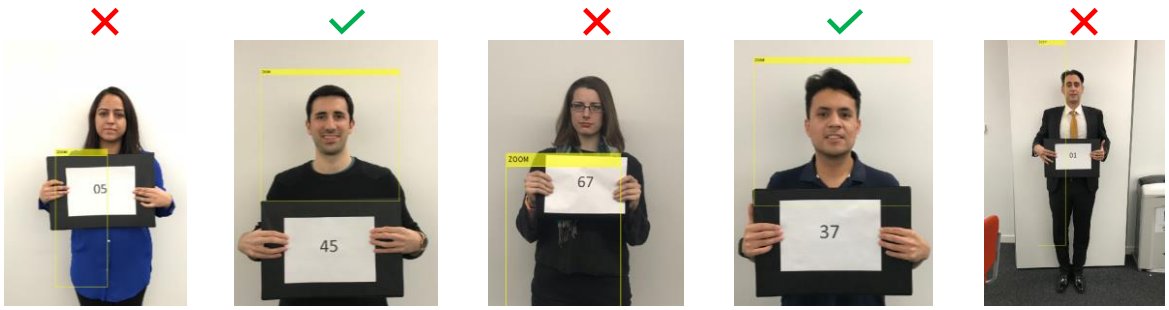


Figure 17. Fail Area applying the Upperbody Argument

10.1.2 Second Try to get the Area: Applying the ClassificationModel: FrontalFaceCART

Then I decided to apply the ClassificationModel with FrontalFaceCART with a MergeThreshold of 15 because to solve this problem all the pictures are taken in front of the individual. The detector found all the faces correctly Figure 17, the next step is to create a replication below this face area.

Figure 17. Correct Area applying the FrontalFaceCART Argument

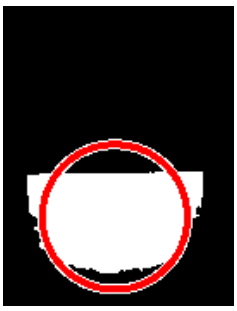
Figure 18. Replication are using the Face Area

10.2 Create the replication Area

Once we found a face area this is replicated to create a "replication area" as shown in Figure 18, we can focus now on this area and perform few pre-process steps to cut any noisy object and apply the OCR once we have the coordinates for the method OCR. I learnt this techniques from the Labs 3,4,5

1. Find the face and create a replication area.
2. Cut this new area and find the centre to apply saturation filter taking the centre as a reference, the objective of this is to make the white the whites area in the photo.
3. Apply colour segmentation to only keep the whitest areas with the below thresholds.
   R > 250, G > 150, B > 145
4. Apply Morphological processing to fill the holes.
5. Apply Region filtering to find the highest white area and calculate the coordinates.
6. Apply OCR method on the new coordinates.
7. Label the Area on the Original Images and Print the Number in console.
8. Steps Fig 19.

1. 2. 3. 4.



5.

6.

7.

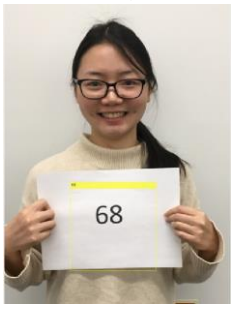
Fig 19. Preprocesing Steps to and OCR Function

11. Results for the OCR:

I applied the new function to the Individual Image and I got good results Figure 20. For the video number detection, the software needs to process all the video to find correct number and perhaps can take a while until process all the frames(It depends of the video duration). While more frames are processed we will have more accuracy which will get us the correct number. Figure 21 shows some results.

11.1 Detect the numbers correctly in test images. (15%)

'IMG_0901.jpg' '46'



'IMG_0340.JPG' '68'

'IMG_0036.JPG' '67'

'IMG_0246.JPG' '15'

'IMG_0246.JPG' '53'

Figure 20. OCR Results on Individual Images

11.2 Detecting numbers in videos. (10%)

'01'

'01'

'111'

'01'

'01'

Figure 21. OCR Results in Individual VIdeo

12. Extra Task: Emotion detection

For the additional task emotion detection, I decided to create a CCN from scratch, I take few images for the Face database that I used to train the models for SVM, MLP and CNN. Then to create labels for the CCN, I organized manually regarding my criteria in the 4 labels, 0-happy, 1-sad, 2-surprised, 3-angry as shown in Figure 22.

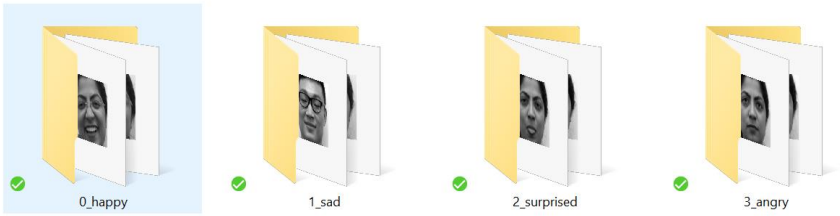


Figure 22. Emotion Image Database

One of the main issue to create this little Emotion Image Database was that even I could not distinguish some expression from my classmates, it was little difficult to find faces for Surprised emotion and something different between happy and sad was not to easy to see.

We could reuse one of the existing CNN like Alexnet [7] applying 'transfer knowledge' [9] but I have the knowledge of how to create an CNN for the previous model. I felt confident on creating a CNN from scratch for emotion detection. The CNN structure is displayed in the Figure 23. It is so similar to the Figure 8, The main difference is the number of output layer we only need 4 classes this time. The script that contain the code is "Matlab_CNN_Emotions.m"

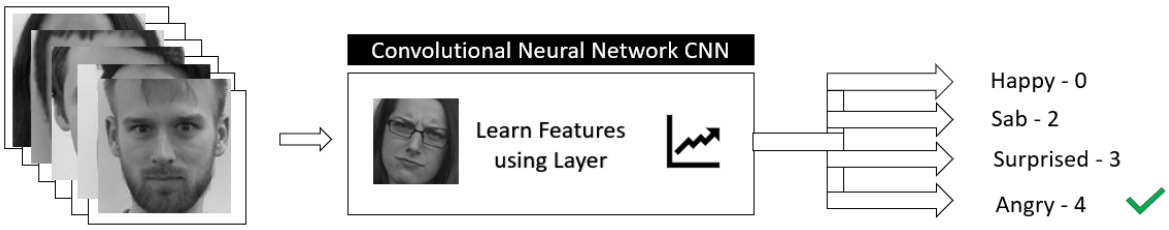


Figure 23. CNN Emotional Structure

Once we have created the CNN for emotional detection we can see that the accuracy if over 96% in the train set and below the 90% for the Test Set Figure 24. Remember that CNN does not need features because the Neural Networks create by its own. It could be a good approach for future work to use a Feature extraction and focus on facial features of the mouth, noise or eye but for now it shows good results with reasonable work.

	Train Set	Test Set
Face Emotion	96.12	86.52

Figure 24. Performance Emotional CNN

12. 1 Results for Emotional CNN

This function is implemented in the Individual and Group Images. I add a 4 four parameter with values from 0 to 4 for 0 = happy, 1 = sad, 2 = surprised, 3 = angry.

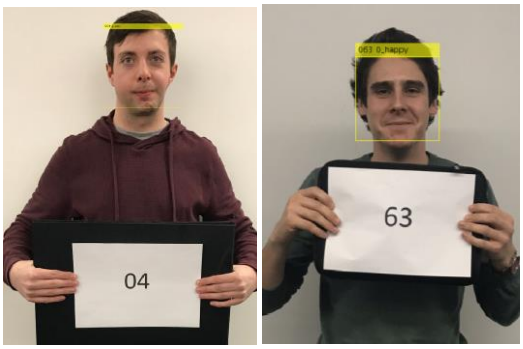


Figure 25. Individual Image Emotions

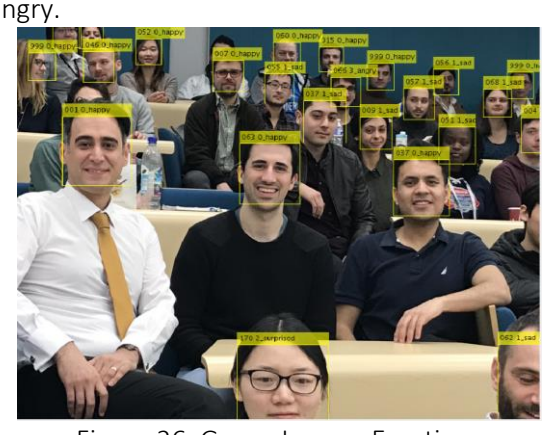


Figure 26. Group Images Emotions

13 Executing the Functions

13. 1 How to call the Classifiers.

Table to call the functions by Model + Future Extraction

Face detection fucntions.	
CNN	`P = RecogniseFace(I, '', 'CNN');`
SVM – HOG	`P = RecogniseFace(I, 'HOG', 'SVM');`
SVM – SURF	`P = RecogniseFace(I, 'SURF', 'SVM');`
MLP – HOG	`P = RecogniseFace(I, 'HOG', 'MLP');`
MLP - SURF	`P = RecogniseFace(I, 'SURF', 'MLP');`

13. 2 How to call the OCR, Image and Video.

OCR Detection Functions.	

Image: You need to put the image in the subfolder 'OCR'. The method is going to read all the files '.jpg' and display the results.
You can put more than one image.
	`%% Recognize Number in Images`
	`clear, clc, close all;`
	`list = dir('OCR\*.JPG');`

Video: You need to give the video name inside of the subfolder 'OCR'.	`%% Recognize Number in videos`
	`clear, clc, close all;`
	`v = VideoReader('OCR\IMG_0882.mov');`

13. 3 How to call the Real Time face detector.

Real Time Face Detector.	

Just execute this code, when you want to finish the execution. Just close the window.	`%% Recognize Face in Real Time with CNN`
	`clear, clc, close all;`
	`face_real_time();`

References:

[1], https://uk.mathworks.com/help/vision/ref/vision.cascadeobjectdetector-system-object.html
[1] https://uk.mathworks.com/help/gpucoder/examples/feature-extraction-using-surf.html
[2] https://uk.mathworks.com/help/vision/ref/extracthogfeatures.html
[3] https://uk.mathworks.com/help/vision/ref/vision.cascadeobjectdetector-system-object.html;jsessionid=e8164f82876dda46eb01e143efa5
[4] https://uk.mathworks.com/help/nnet/ref/imagedataaugmenter.html
[5] Vapnik, V. (1995) The Nature of Statistical Learning Theory. Springer-Verlag, London.
[6] Kijsirikul, B. & Ussivakul, N. (2002) Multiclass support vector machines using adaptive directed acyclic graph. Proceedings of International Joint Conference on Neural Networks (IJCNN 2002), 980-985.
[7] https://uk.mathworks.com/help/nnet/ref/alexnet.html
[8] https://uk.mathworks.com/products/matlab/whatsnew.html
[9] https://uk.mathworks.com/help/nnet/examples/transfer-learning-using-alexnet.html
[10] Vivienne Sze, "Designing Hardware for Machine Learning: The Important Role Played by Circuit Designers", *Solid-State Circuits Magazine IEEE*, vol. 9, pp. 46-54, 2017, ISSN 1943-0582
[11] https://uk.mathworks.com/discovery/convolutional-neural-network.html