INM 460 Computer Vision Report

Kiril Tsarvenkov

kiril.tsarvenkov@city.ac.uk

1. Introduction

The aim of this report is to show the development of Matlab programs for face recognition and numbers detection from raw images and videos. The main approach of this paper is the use of Convolutional Neural Networks CNN for both problems, however, handcrafted methods such as Histogram of Oriented Gradients are also implemented as required by the coursework specification. This report is structured as follows: Section 2 explains the dataset and the required steps to create a face database. Section 3 outlines the deployed algorithms for the development of the RecogniseFace program and their implementation. Section 4 proposes a CNN solution to the Optical Character Recognition task for number detection and Section 5 provides a summary of the strengths and weaknesses of the developed solutions.

2. Dataset

The data consists of 4 folders with a total of 69 folders with 3 to 7 images and videos with a person holding a white sheet of paper with a number on it. The images are taken from different angles and the number on the sheet is used as a name for the respective folder. Additionally, there is one extra folder which contains group photos and videos consisting of the people in the individual sub folders. The formats of the images are jpg and HEIC, and for the video the format is mp4. The variation in the image formats is due to the three different devices that have been used in the creation of the dataset. Nonetheless, this is not a problem for the current study as an entirely new face database is created for the development of the RecogniseFace program, while the development of the OCR function, detectNum() does not require uniform image formats. However, the HEIC file images from camera two needed to be converted in a suitable format such as jpeg in order to use the images. The conversion software iMazing was used for this . More on this topic in section 3.2.1 Face Database creation and section 4.1.1 Ground Truth.

3. Face Recognition

A main deliverable of this project is the development of a face recognition system. Various approaches were undertaken and the underlying methods are discussed in the following subsections.

3.1 Methods

The task of face recognition is not trivial, however, there are two major steps that need to be performed. Firstly, a system should be able to detect faces and secondly, the system should be able to compare the detected face to an existing database in order to perform classification. The next subsection discusses the Viola-Jones algorithm as this is the chosen approach to detect faces in images for the RecogniseFace function.

3.1.1 Face Detection

3.1.1.1 Viola – Jones Algorithm

The Viola-Jones framework is motivated primarily by the problem of face detection although it is extendable to object tracking. In terms of face detection, the algorithms requires full frontal faces pointing towards the camera which are not tilted to either side.

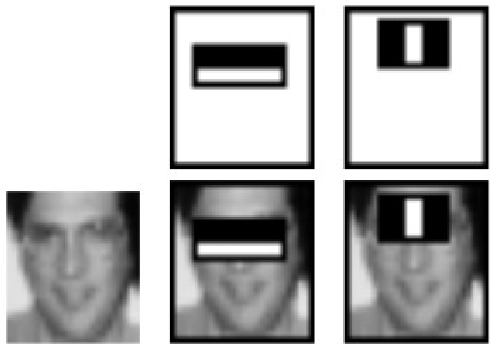
Figure 1: Haar features and Integram Images *[1]*

The algorithm consists of four stages which facilitate object detection [1]. The first is Haar Feature selection which is based on the premise that human faces share similar characteristics. For example, the nose bridge region is brighter than the eyes. Grouping similar features in this manner permits the creation of a feature set which aims to form a detectable pattern which is able to detect faces. This is achieved through the creation of an integral image (row one in Figure1) which generates the sum of values in a rectangular subset in an image. However, this creates a large feature space which poses computational constraints. Therefore, the third stage is the training of an Adaboost algorithm which selects the best features from the feature space and train classifiers to use them. This is achieved through the combination of ‘’weak’’ classifiers which form a linear combination, in order to construct a ‘’strong’’ classifier. Finally, a Cascading classifier architecture is applied in order to achieve better detection rates and improve computational efficiency, especially for real time applications. In the cascade architecture each classifier is strong and all the detected features are grouped into stages. Then each stage determines if the groped features represent a face or not and If a set of features fails at a particular stage then this sample of features is discarded as a face. If a face is detected then the second step of face recognition can be performed.

3.1.2 Face Classification

After a face is detected, it should be compared to a database with known labels. This is a basic machine learning task under the Supervised Learning domain known as classificaiton. Depending on the problem, a number of preprocessing steps need to be performed. For example, a typical machine learning algorithms such as Decision Trees DTs and Support Vector Machines SVMs can be implemented only after a set of features have been extracted from the images. This is where handcrafted feature extraction methods such as Histogram of Oriented Gradients are used in order to facilitate ‘’learning’’. Alternative to this approach, Convolutional Neural Networks CNN provide an end to end solution which uses the architecture of the CNN to extract features and provide a classification output. There are many versions of CNN but this paper will focus on two prominent architectures which have delivered excellent results in the recent years. They are AlexNet and ResNet50. However, it is important to note that the use of ResNet50 is limited to the extraction of features. Separately a classifier such as SVM and DTs are used on top of the extracted features. AlexNet is the choice for an end to end CNN of this paper and the expectation is that this approach should provide the most prominent results. As a result, this paper adopts two feature extraction methods – HOG and ResNet, two machine learning classifiers – SVM and DT, and an end to end CNN – AlexNet.

3.1.2.1 AlexNet

AlexNet is a pretrained CNN and in this paper it will be used for face classification. This is an example of Transfer learning where a pretrained CNN has already fine tuned parameters and it is trained on various categories. This is much faster computationally as it avoids the burden of starting with random initial weights. Moreover, the network has a proven performance record as it won the 2012 ImageNet competition by a substantial margin and sparkled a new research era [2].

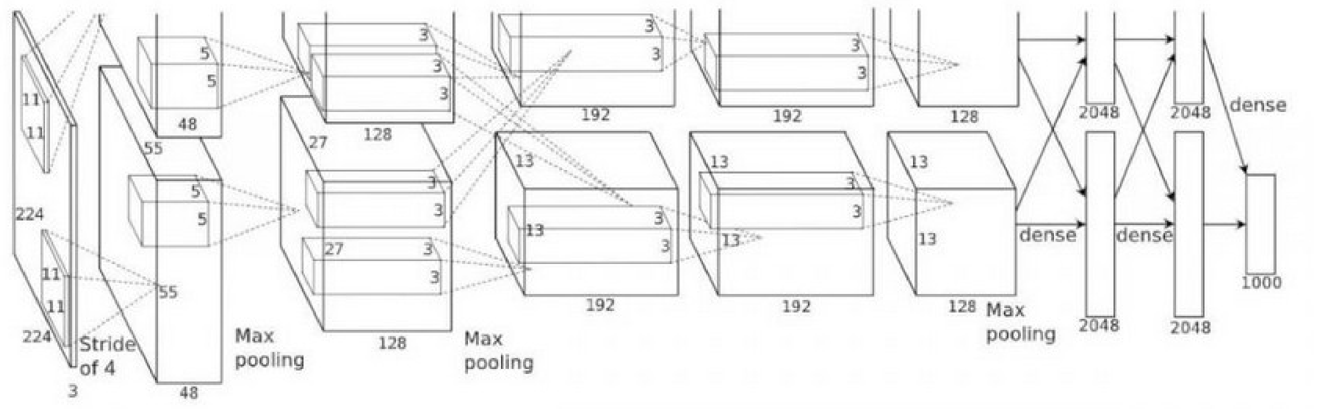

Figure 2: AlexNet Architecture with 8 layers [3]

The architecture of AlexNet supports an end to end solution for image classification[3]. It contains eight layers, where the first five are convolutional and the last three are fully connected (Figure 2). The output of this layers is feed forward to a 1000-way softmax classifier which produces a distribution over the class labels. For the purposes of this paper, the final layers and the classifier will be adjusted to suit the face recognition task. Moreover, a Relu activation function is used after every layer and dropout tackles the problem of overfitting between the first and the second fully connected layers. The network contains 62.3 million parameters and requires 1.1 billion computation units in a feed forward pass.

3.1.2.2 ResNet50

ResNet is another pretrained CNN with a proven track record of success. In 2015, it won first place in the ILSVRC classification competition with an error rate of 3.57% [4]. The network architecture aims to solve the problem of saturation in accuracy as network grows excessively deep [5]. ResNet tackles this problem using Deep Residual learning framework (Figure 3).

The idea is to render the Residual function F(x) = 0, thus making x = y (Figure 3 right). The hypothesis is that it is easier to optimize the residual function rather than the unreferenced mapping H(x). The creators of the network have carried numerous test to verify that this approach produces beneficial results, which is indeed the case [5].

The design of the network uses 3*3 filters and down samples the CNN layers with stride 2. Shortcut connections are used to turn the network into Residual form, as shown in Figure 3 and Figure 4. In order to prevent the vanishing gradients a ReLu activation function is used and batch normalization is implemented after every convolution layer.

ResNet has different versions dependent on the depth of the network. This paper adopts ResNet50 which consists of 50 layers. The original implementation has 34 layers but the additional layers aim to increase the efficiency of the network. Figure 4 shows a ResNet residual network with 34 layers against a plain network without residual blocks. However, it is important to note, that for the purposes of this paper, the ResNet architecture will be used for feature extraction only and a supervised machine learning algorithm such as SVM and DT will be applied on top of the extracted features in order to perform classification. In addition, ResNet will be used in the Optical Character Recognition task in Section 4 of this paper.

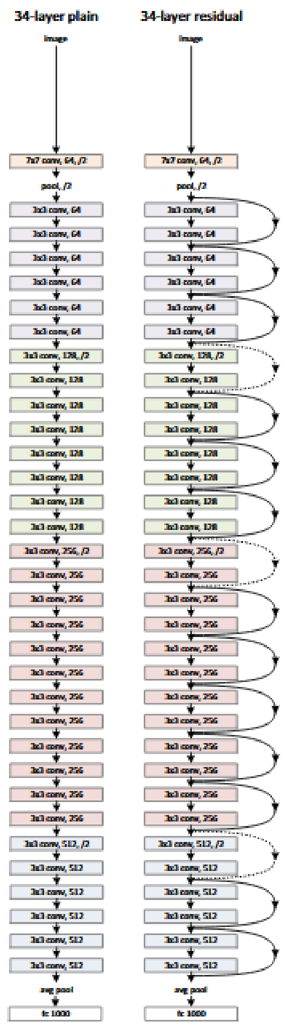

Figure 3: 34 Layer Plain CNN vs Resnet34 [5]

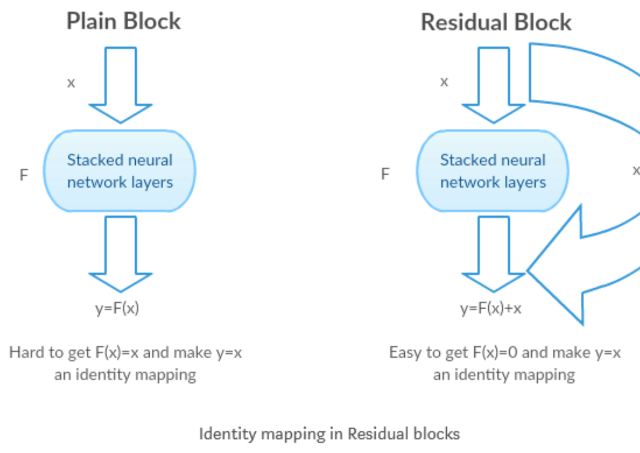
Plain Block
x
F
Stacked neural network layers
y=F(x)
Hard to get F(x)=x and make y=x an identity mapping
Residual Block
x
F
Stacked neural network layers
x
y=F(x)+x
Easy to get F(x)=0 and make y=x an identity mapping
Identity mapping in Residual blocks

Figure 4: The Residual Block on the right shows the stricture of ResNet, whereas the block on the left shows a plain net without a resudual block [4]

3.1.2.3 Histogram of Oriented Gradients

Histogram of Oriented Gradients HOG is a general feature extractor used in image processing for object detection. The fundamental idea behind this approach is that the appearance of an object can be described by the edge directions which can be inferred from the distribution of gradients[6]. This is achieved by dividing the image into small regions which contain a given number of pixels. For every region a histogram of gradient directions are computed and the direction with the highest intensity is used as a descriptor for this region. As a result, the image is represented with the directions of the descriptors for all regions. The resulting feature map can be observed from Figure 5.

3.1.2.3 Support Vector Machines SVM

A popular machine learning algorithm is the Support Vector Machines SVM. It is often used for classification and the most appealing characteristic of the algorithm is the ability to find optimal separation line/hyper-plane between the classes [7]. This is achieved by finding support vectors which are also know as the data points which are most difficult to classify. Moreover, the algorithm solves the classification problem by increasing the dimensionality of the problem, thus looking for a dimension where the classes are linearly separable. Therefore the algorithm bypasses the curse of dimensionality problem. Before the introduction of CNN, SVM were considered state of the art in the computer vision domain therefore, the expectation is that the classifier should perform well in this task. Nonetheless, the performance largely depends on the feature extraction method used to feed into the SVM.

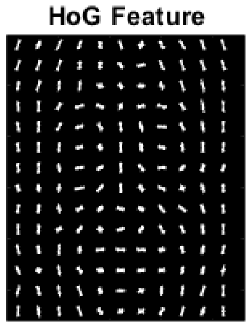

Figure 5: HOG Feature example of gradient directions in an image

3.1.2.3 Decision Trees DT

Decision trees DT is a popular method for approximation of discrete-valued target functions [8]. They classify examples by sorting them down a tree. The algorithm starts from a root and it is propagated through leafs nodes until a final node is reached which provides the classification for the instance. The root represents an attribute from the feature space and it should contain the highest information gain from all attributes. The information gain represents the difference in entropy from partitioning the feature space. Any subsequent leaf node is selected based on the highest information gain until all attributes are exhausted from the feature space.This algorithms is not very popular in computer vision application however, it will be interesting to inspect what results will be achieved.

3.2 Implementation of the RecogniseFace Function

A main deliverable of this project is the development of a RecogniseFace function which is able to classify faces. It is constructed with the methods discussed above and this section of the report outlines how these approaches have been adapted to the task.

3.2.1 Face Detection

The face detection part of the project is performed by the Viola-Jones algorithm. It is implemented with the Matlab built in function vision.CascadeObjectDetector(). This function is used in two stages of the development of the program. Firstly, it is used in the creation of a face database (see file Data_Base.mat) where faces are detected in the raw images and then in the actual RecogniseFace() function as a preprocessing step before feeding it into pretrained classifiers.

3.2.1.1 Face Data Base Creation

The code for this was obtained from [9]. Initially, a raw image from the original dataset is feed to the Viola-Jones algorithm in order to detect a face, after that the algorithms is applied again on a cropped image of the detected faces searching to detect a nose/mouth. This approach aims to deal with false positives, where ‘’detected faces’’ are not actually faces but other random objects which mimic the features of a face. Also, this ensures that occlusion such a glasses do not hinder the detection process. When a nose/mouth is detected the initial bounding box of the face is cropped to a new folder in a .png file format with a size of [112 to 92]. This assures that all of the images are in the same file format in the database and that they have the same dimensions. However, the number of raw images from the original dataset is very low for most individuals (3 to 5 images per person) which creates a very small database of images. Furthermore, the Viola-Jones algorithm was not able to detect faces in profile images (Figure 6) which reduced significantly the number of usable faces. Therefore, the videos provided in the original dataset were used in order to extract frames (see file videos.m [10]). They were feeded into the face detection algorithm (see file Data_Base.m) to detect and crop the faces in the new folder of the individual. The extracted frames and subsequently face images are also in a .png file format in order to keep the face database homogeneous. Only one of the videos per person was used and the number of extracted frames varied between 60 to 122 frames for a video. The reason for this variation is the use of different devices. Nonetheless, this was not material for the database as only 3 to 9 images from the videos where used for the creation of the final database. Furthermore, face images from the group photos were obtained in order to increase the variation in facial expressions and positions as most of the frames from the videos displayed identical faces which were helpful for learning and induce problems such as overfitting. As a result, each individual has their own folder, as in the original data, with 10 images per person in the folder.

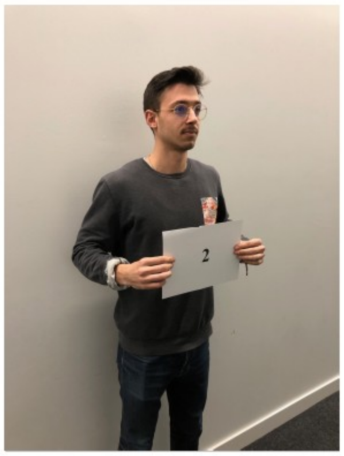

Figure 6: Example of an image where the algorithm failed to detect a face

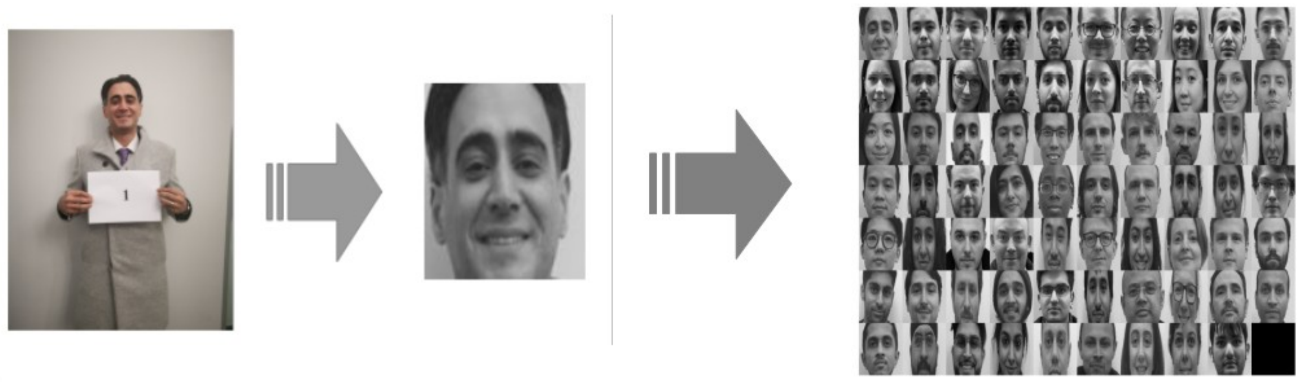

Figure 7: Face Database Creation Process: Raw image > Cropped Face > For all people in the data set

The 10 images are a mixture of the cropped faces from the raw images, the cropped images from the video frames and the cropped faces from the grouped photos. The proportions of the different types of images varies for the different people as the face detection algorithm was not able to detect faces for all of the raw images. Finally, significant time and effort was invested to create a database with as diverse images as possible in order to create a quality dataset rather than merely creating large quantity database with identical images which produces spurious results.

3.2.2 Face Classification

After the database was created the images are ready for classification. In this section the various feature extraction and classification combinations, and CNN algorithms are developed. The end result is trained classifiers which can be adapted into the RecogniseFace program.

3.2.2.1 Training ResNet

A pretrained ResNet model was used for feature extraction. The model is trained on the ImageNet dataset. Moreover, the face database is split into 60% training and 40% test sets (see file ResNet50.m). Additional data augmentation is applied converting the images from gray to color as the network requires color images. Additionally, random rotation was performed by 20 degrees in either directions [-20 20], random translation of the x axis by [-3 3] and random translation of the y axis by [-3 3]. Also the input size of the images is adjusted to 224x224 as the network requires these dimensions. The weights of the first convolutional layer as displayed in Figure 8 and the feature layer is set as fc1000 (Figure 9). The last two layers fc1000_softmax and ClassificationLayer_fc1000 are not used. Instead of them, SVM and DT classifier are trained. As a result, this a hybrid system of a convolutional neural network approach and supervised machine learning algorithms used for classification.

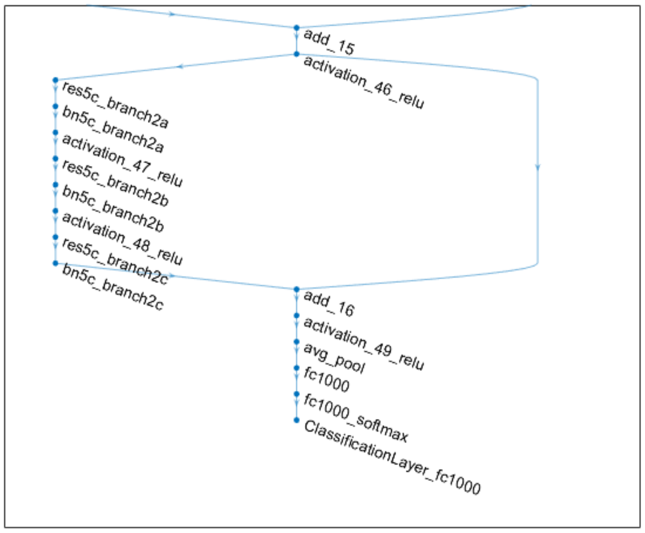

Figure 9: The Last layer of ResNet (note the classification layer is removed in the implementation)

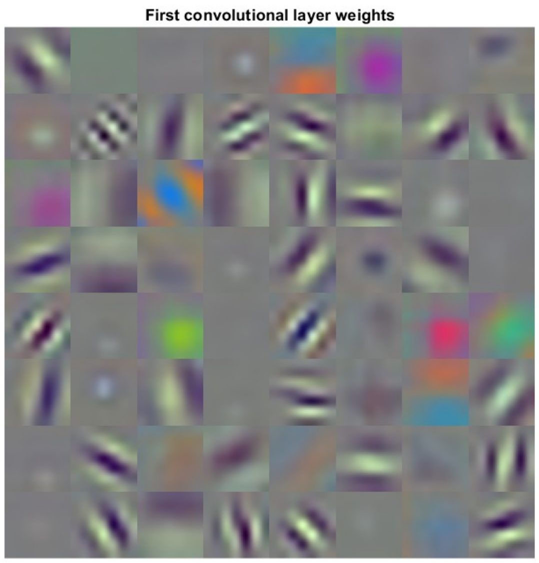

Figure 8: Weights of the first convolutional layer

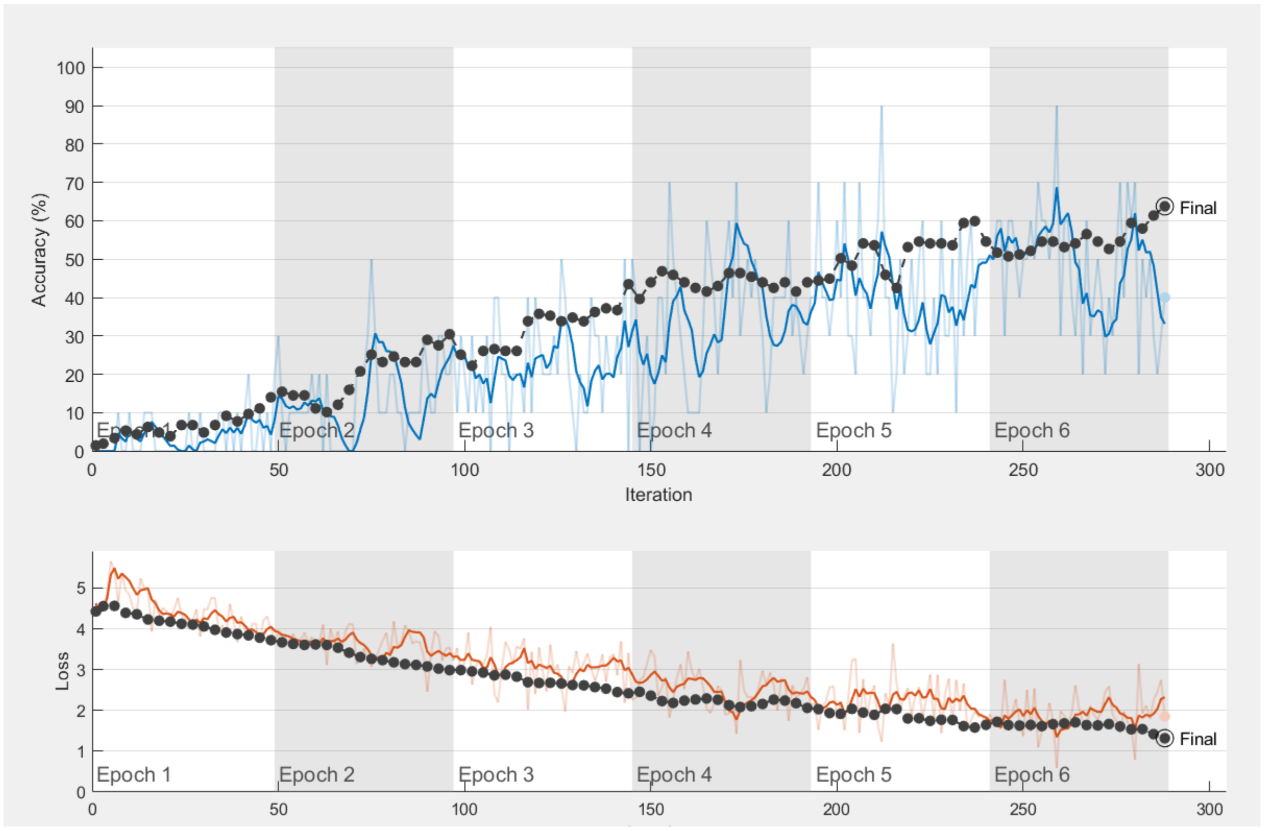

Figure 10: The training process for the Alex Net atchitecture for 6 epochs, 288 iterations and 48 iterations per epoch – training time 33mins and 32 sec on a single CPU

3.2.2.2 Training AlexNet

Similar to ResNet, the AlexNet model is pretrained on the ImageNet dataset(source code[13]). The training and test sets are split 60% and 40% in order to maintain the degree of comparability between the models. The difference between the ResNet and the AlexNet model, in this paper is the use of the softmaxLayer and ClassificationLayers for the AlexNet implementation (see file AlexNet.m) while ResNet is used as feature extractor only. The data for AlexNet is augmented using a pixel range of [-30 30]. This range is applied to the translation on the x and y axis. The data images are also converted from gray to RGB color space. The training parameters are set as follows: the Learning rate set at 0.004, MiniBatch size is set to 10, the maximum number of epochs is set to 6 and the data is shuffled at every epoch. The time taken for training is 33 mins and 32 sec, and the highest accuracy is achieved in epoch 6 at 61.97% (Figure 10)

3.2.2.3 Extraction Histogram of Oriented Gradient HOG Features

Histogram of Oriented Gradients HOG is performed following a standard Matlab procedure, where an empty features matrix is initialized [11]. The rows are the number of the classes (faces in this case) multiplied by the training examples - 6. The number of the columns is the count of the total HOG features for the training set. Therefore there are 414 rows and 4680 columns. As a result, the training matrix represents all the training images with their corresponding features. An example of a feature extraction for an individual image can be observed in Figure 11. Having extracted the features of the training set a supervised classifier can be trained, in order to perform predictions about new instances. In this paper, the classifiers are SVMs and DT.

3.2.3 Workflow of the Function

The workflow of the RecogniseFace function is shown in Figure 17 at the end of this paper after the references. An image is passed through as an argument and featureType and Classifier options are specified. The options can be one of the aforementioned algorithms and methods. (Valid way to call the functions can be seen at the end of this paper and the README.txt file) After that the function uses the Viola Jones algorithm to detect a face. If no face is detected the function returns and empty matrix P = []. If a face is detected, the face region is cropped into a separate image which is passed to one of the input arguments of the function such as classifier and feature type.

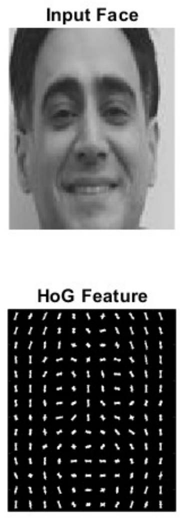

Figure 11: HOG Feature extraction for a query face

In the case of end to end CNN - AlexNet, the feature type is set at '0' and the function loads the pretrained classifier AlexNet which is used for prediction of the input image. Also, the bounding box locations are taken and calculated, in order to find the central face regions for x and y.

The process is similar for the ResNet architecture with the only difference that the classifiers are SVM (SVM_ResNet50.mat [12]) and DT (DT_ResNet50.mat[12]). Once the an image is input into the function a face is detected or not. If there is a face, the face region is cropped to a new image, which is also resized in [112 92] dimensions. Then a DT(DT_HoG.Mat) or an SVM (SVM_HoG.mat) classifier is loaded with pretrained HOG features. After that HOG features are extracted from the cropped face image. The features are used for query in the pretrained classifier which delivers a classification of the input features.

3.2.4 Drawbacks

Before the discussion of the results is presented, several points need to be made about the developed RecogniseFace() function.

Firstly, the function accepts only single images as loaded in Matlab standard procedures (i.e. I = imread('test1.jpg');). The function does not accept imageSet or ImageDatastore objects as arguments and it is difficult to use for a large number of images. Therefore the evaluation of the function is based on the quality of the pretrained classifiers which are developed in AlexNet.m, ResNet50.m and MDL_HoG.m scripts. This should not undermine the quality of the function to classify correctly new instances as It is logical to assume the function will perform as good as the classifiers which constitute the function.

Secondly, the face database is rather small, only 10 images per person which is a problem for CNN approaches which require large amounts of data. Also, the training and test set are specified at 60% and 40% which means that only 6 image are used for training which deteriorates the problem even more.

Thirdly, the problem of false positives is present, which means that the function can detect a face region but also other regions which have similar features. Thus the function will chose classify instances which might not even be faces which inevitably will deteriorate the performance of the function compared to the pretrained classifiers.

Finally, the ID of an individual in the database is not identical to the number that they hold in the picture. Thus, if a person holding a number 22 in the picture, their corresponding place in the database is not 22 due to the structure of the imageSet data type in Matlab and the fact that numbers (such as 18) were skipped in the creation of the dataset (taking pictures at university). This makes is difficult to track the ID of the person between the output of the function and the input image.

3.3 Results

The results for the various feature/classifier combinations are shown in Table 1. The best performing model is the SVM with HOG feature extractors at 62.68% percent. The second best model is AlexNet with 61.97% correct classification followed closely by ResNet Feature extractor with SVM classifier at 59.06 %. The Decision Tree classifier is last with a significant variation in performance for the different feature extraction methods. For the HOG feature method the results is dismal 18.48% percent however, when ResNet is used the result deteriorates further to 8.33%. The results of the experiments clearly demonstrate some of the drawbacks outlined in the previous section. The models for the classifiers with HOG features are potentially overfitting the data which explains their rather strong performance compared to the CNN implementations. However, when ResNet is used their performance drops slightly which could be attributed to the data augmentation step in the ResNet implementation. For the DT the performance drop is by almost 10% which raises the suspicion of overfitting for this classifier. The difference in performance across the models is due to the fact that data augmentation was used for the ResNet and AlexNet implementations which reduces the probability of overfitting by creating images with different rotations, translations and colors. The better performance of HOG with SVM is somewhat surprising however, given the small database this results is not all that surprising. Nonetheless, the slightly worse performance of the CNN approaches demonstrates their superiority in dealing with image classification problems as the results do not tend to lead to misleading conclusions. As a result, confusion matrix is plotted for the SVM classifier with the HOG feature extractors. Individuals with IDs 23, 24, 29, 35, 47, 54, 57, 61, 65, 66 and 68 seem to have a strong bias towards a particular class with incorrectly classified instances. Overall the results are satisfactory, not great but plausible. Nonetheless, the inherent drawback of the face database should keep one mindful of the results.

Classifier	**SVM**		**DT**		**AlexNet**
Feature	ResNet	HOG	ResNet	HOG	-
Accuracy	59.06%	62.68%	8.33%	18.48%	61.97%

Table 1: Results for the various classifer/feature type combinations

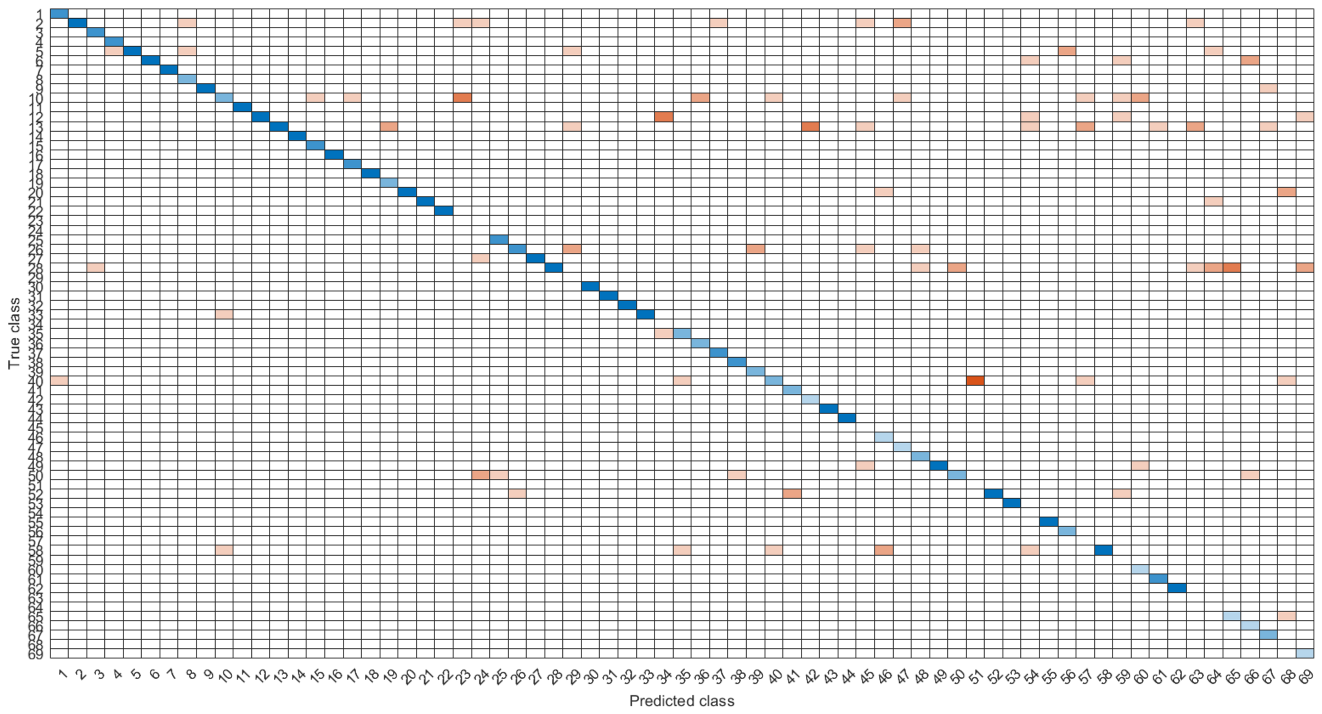

Figure 12: Confusion Matrix for the SVM/HOG combination

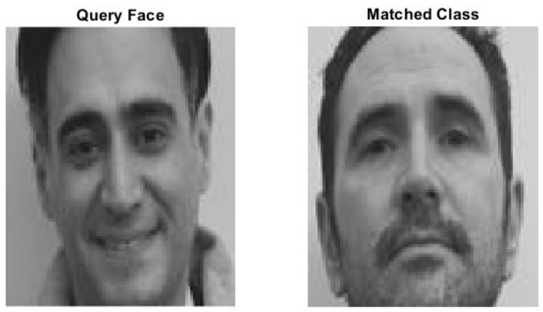

Figure 13: Example of misclassification by the SVM/HOG classifier

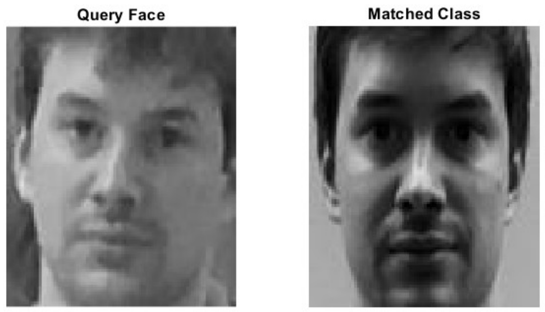

Figure 14: Example of correct classification by the SVM/HOG combination

4. Optical Character Recognition

The number detection task could be tackled in a number of different ways. One of them is through image processing which manipulates the pixels of an image in order to prepare a photo for object detection. An alternative is using deep learning or pretrained CNNs such as ResNet in Region CNN framework, where the architecture learns the features of a particular region and aims to match them into new images. The former approach was used for initial experimentation but the results were rather poor as there was a considerable variation in the objects in the images which meant that there is no single segmentation that works ‘’best for all''. Therefore the latter approach of RCNN was adopted in order to follow a more robust framework which could produce worthwhile results in even more challenging environments such as in real time videos and multiple numbers in an image, compared to the rather uniform database of images for this project.

4.1 Network Training

As mentioned previously, the pretrained CNN used for this task is the ResNet50 architecture[14]. However, the network still needs to be trained for the regions of interest in the current task. Therefore, the final three layers of the network fc1000, fc1000_softmax and classification_fc1000 are removed and new layers are initiated suitable for the RCNN task. The new layers are fully connected rcnnFC, rcnnSoftmax and rcnnClassification layers with the number of classes set at 1 for the bounding boxes of interest plus one for the background. The bounding boxes for the problem are specified in the next subsection. Once the bounding boxes are established they are cropped and passed to the ocr() function in Matlab which detects text and number.

4.2 GroundTruth

After the network is adjusted for the needs of the task, a ground truth needs to be established, according to which the network can adjust its weights. This task is performed in the ImageLabeler app in Matlab where individual images are uploaded and regions of interest are selected. The regions selected for this task are rectangles around the number with the aim to capture the variation in the background of the number. Initially, the region of interest was selected to be the entire white paper with the number, however, this produced dissatisfactory results. The algorithms was matching regions of the wall and other areas which are also white but never matching the white A4 paper. Therefore, a narrow region of interest around the number was selected which improved slightly the results. The total number of images where a region of interest was specified is 72 which approximately corresponds to the number of the people in the database. Each individual has at least one image used in the establishment of the Ground Truth for this task.

4.2 Results

The adopted approach produces some mixed results. For example, Figure 15 shows a test image where a bounding box is plotted around the number 3. However, the bounding box fails to recognise the other number 3 next to it, which results in a wrong number detection. Another test image is shown in Figure 16 where the RCNN completely fails to detect the number and the bounding box is plotted on a part of the coat. Therefore, the OCR function is mistakenly takes the features of the bounding box as numbers and produces spurious results. As the experiments suggest, this approach has its merit, however, more development is needed mainly on the ground truth specification of the problem as CNN implementations require large quantities of data.

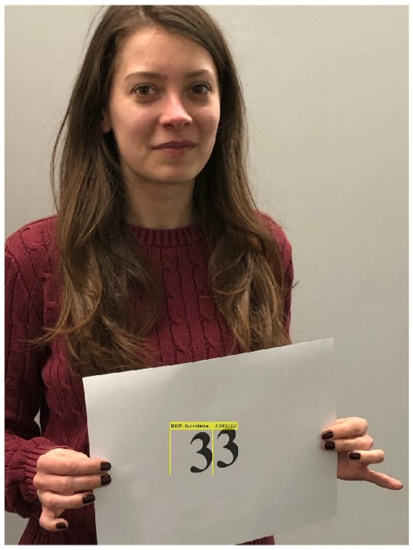

Figure 15: Example where RCNN produces promising results

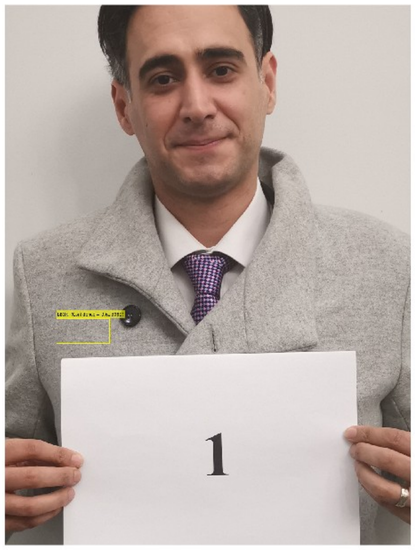

Figure 16: Example where RCNN fails to detect a region of interest

4.3 Drawbacks

Similar to the RecogniseFace function, detectNum exhibits some drawbacks which can be addressed in future work. Firstly, the computational time for the detector (see the code line below) is prohibitively long, around 5 mins which makes it difficult to test the function over a large sample of test images.

```
[bbox, score, label] = detect(rcnn, filename, 'MiniBatchSize', 32);
```

Also the function does not accept multiple files at one attempt which is prohibitive in large datasets. Future work can focus on training Fast RCNN implementations which should provide a more efficient computation. Finally, the network architecture can be improved by providing a better ground truth reference by using more examples images from the individuals as the current number of training examples is only 72 images with different numbers in the bounding boxes and the CNN architectures require large samples of data in order to work effectively on test images.

5. Conclusions and Future Work

In this paper a Deep Learning approach was developed to solve the problems of face recognition and number detection. The deployed architectures are ResNet and AlexNet. In addition, a handcrafted approach such as Histogram of Oriented gradients was used in order to increase the scope of the research problem. The deployed machine learning algorithms for the various tasks were SVM and DT. The best performing model in this paper was SVM with HOG features where the achieved accuracy is 62.68%. This somewhat surprising result could be traced to the fact that the engineered face database is not sufficiently large, only 10 images per person, and the size of the training set was very low (only 60%). However, significant time and effort was invested in developing a database with diverse images in terms of pixels, angles and resolutions rather than forming a large database with identical images which creates an imparatus for overfitting. As a matter of fact, such an approach was initially undertaken but the algorithms were severely overfitting the data with close to 100% accuracies for the SVM implementations and AlexNet and more than 65% for the DT algorithm.

Despite the inferior results for the CNN architectures, the performances are very close to the HOG features approach and a larger data set is almost certain to produce better results for the CNN approach. Moreover, data augmentation was implemented only for the CNN methods which renders the classification problem slightly more challenging but also mitigates to certain extend the low number of training examples.

The developed RecogniseFace solution can be improved in a number of ways. Firstly, retraining the classifier with larger train and smaller test set could improve the results by a small but significant margin. Secondly, the function can be improved to accept multiple images or object types such ImageSet or ImageDatastore which would scale up the practical applications. Thirdly, a larger face database with diverse images in terms of angles, pixels and resolutions will certainly improve the accuracies. Finally, the numbers on the sheets can be used as IDs in the face database as way to track the individuals from the database to the output of the function

The developed detectNum solution does not perform as promising as the RecogniseFace program, however, developing a larger database with ground truth will improve the performance of the function as only 1 training example was take for a given number. The reason for the low amount of training examples in the ground truth database is the time consuming manual labour involved with drawing rectangles around the numbers in the images. An already established dataset could be used for this task, however, the matlab requires a specific format which was not possible to obtain for already created numbers databases such as MNIST.

Valid ways to call the RecogniseFace function:

```
RecogniseFace(I , 'AlexNet', '0')

RecogniseFace(I , 'SVM', 'ResNet')

RecogniseFace(I , 'SVM', 'HOG')

RecogniseFace(I , 'DT', 'ResNet')

RecogniseFace(I , 'DT', 'HOG')
```

References:

[1] Viola, Jones: Robust Real-time Object Detection, IJCV 2001

[2] https://neurohive.io/en/popular-networks/alexnet-imagenet-classification-with-deep-convolutional-neural-networks/

[3] https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks

[4] https://medium.com/@14prakash/understanding-and-implementing-architectures-of-resnet-and-resnext-for-state-of-the-art-image-cf51669e1624

[5] Deep Residual Learning for Image Recognition, Microsoft Research: https://arxiv.org/pdf/1512.03385.pdf

[6] N Dalal and B.Triggs, ''Histogram of oriented gradients for human detection'' in 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition vol.1 Jun 2005 885-893 vol

[7] Haykin, S.S., 1931 2009, *Neural networks and learning machines,* 3rd, International edn, Pearson

Education, Upper Saddle River, N.J.

[8] Mitchell, T. 1997, *Machine learning,* [International]. edn, McGraw-Hill, London;New York;

[9] Source Code for Face Database: https://www.youtube.com/watch?v=b4Q_N39u5k4

[10] Source Code for frames extraction from videos: https://uk.mathworks.com/matlabcentral/answers/48797-how-to-extract-frames-of-a-video

[11] Source Code for HOG Features: https://uk.mathworks.com/matlabcentral/fileexchange/53849-code-for-face-recognition-with-matlab-webinar

[12] Source Code for ResNet: https://uk.mathworks.com/help/deeplearning/examples/train-deep-learning-network-to-classify-new-images.html

[13] Source Code for AlexNet: https://uk.mathworks.com/help/deeplearning/examples/transfer-learning-using-alexnet.html

[14] Source Code for RCNN: https://uk.mathworks.com/help/vision/ref/trainrcnnobjectdetector.html#bvcpk41-3

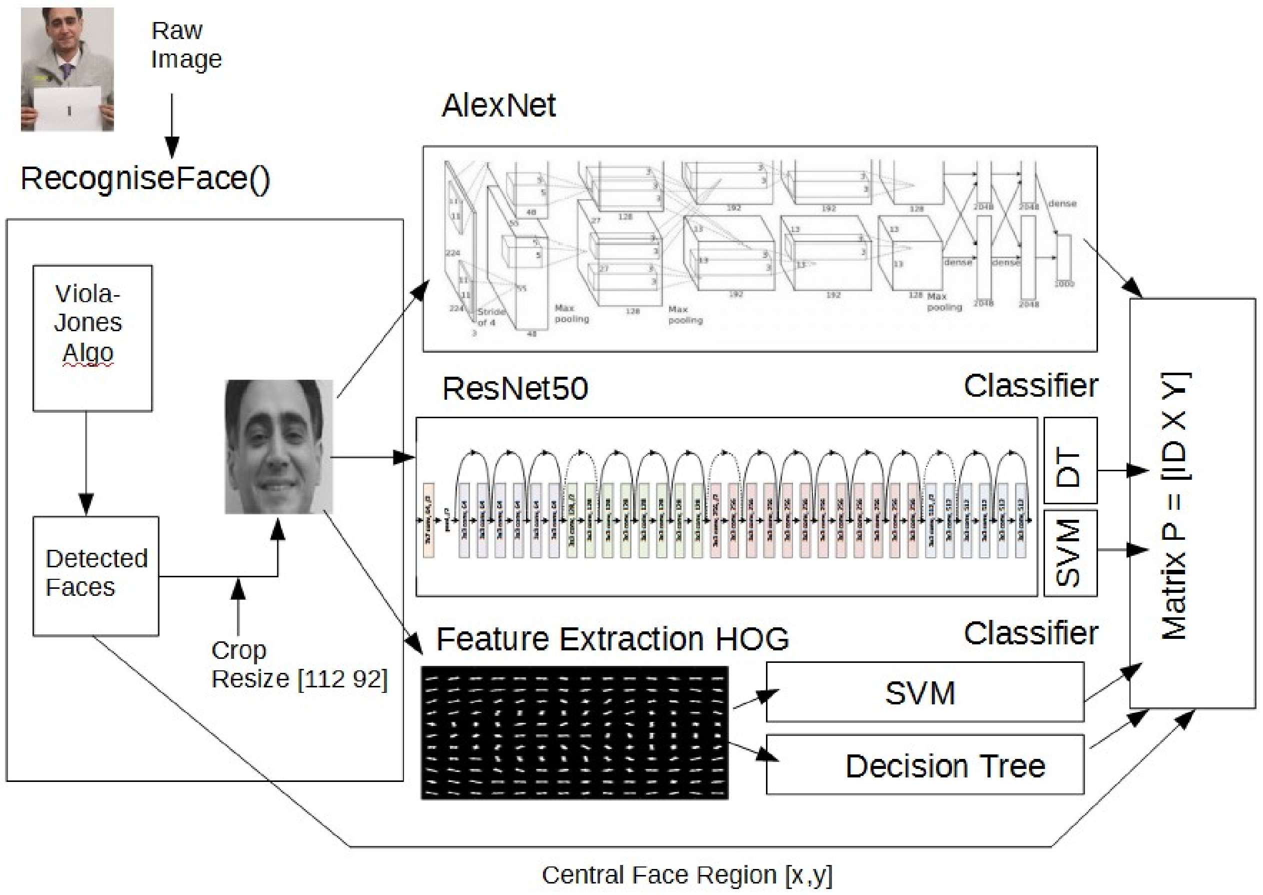

Figure 17: Workflow of the RecogniseFace Function