Computer Vision: Face Recognition

Vithurshan Vijayachandran

Introduction

Humans are exceedingly capable of recognising faces previously seen before, but matching names to faces has been a difficult task. Smartphones however are brilliant at this as technology giants have gone to great lengths to make the devices more user friendly, with camera's auto focusing on detected faces when taking a picture and identifying those faces based on users' previous tags. China has gone a step further by making facial recognition mandatory for mobile phone users [1]. This shows the tremendous progress the field of computer vision has made over the last few years notably with facial recognitions.

In this paper the author will report on the computer vision system developed to detect and identify individuals using a database of known face images. Images comprise of individual and group pictures of the author's classmates. The author chose MATLAB as the environment to create the facial recognition system.
The report is structured in the following manner: following the introduction, author report on the purpose of the study (RecogniseFace function) and initial plan of approaches to be taken (data collection and pre- processing of the data, feature extraction and classifiers to be used). Following that, author reports on how each approach was implemented and, on any changes made to the initial plan. Next, the implementation of the function followed by creative mode is also discussed. Finally, the report is concluded by reflecting on the work conducted and of future work.

Face Recognition

The purpose of the study is to create a RecogniseFace function. The function should take four input arguments, I (image), featureType (HOG, SURF or None), classifierType (SVM, MLP or CNN) and CreativeMode (1 or 0). The function will return a matrix P of size Nx3, where N is the number of people detected in the image and three columns will be represented by ID (unique label associated with each person, for individual who are not in the training data), X (the x coordinate of centre of the face of the person detected) and Y (the y coordinate of centre of the face of the person detected).

Approach

Graphical representation of the approach to be undertaken during training is shown in Fig.1.

Data Collection

The data used for this study was provided by the lecturer. Computer vision students (those who volunteered) were photographed by the lecturer to create a collections of group images and group videos. Individuals who appeared in the group photo were photographed and filmed individually from different angles, albeit in portrait mode. To be assigned to a label, each individual was requested to hold a randomly chosen A4 sheet of paper consisting a number (01 to 80) printed on it while they were being photographed and filmed. It is to be noted that some individuals who appeared in the group photos opted against taking individual photos/videos.48 students were photographed. The number of photos and videos per person varied with range being 5 to 12. The photos were in JPG format and videos were in MP4 format.

Data Labelling

In order to separate images and assign it to the given class (ID), Optical Character Recognition is to be performed by the author. MATLAB built-in function *OCR* is to be applied to extract digits from the images and frames extracted from videos, and folder will be created based on the extracted digit to add images and frames.

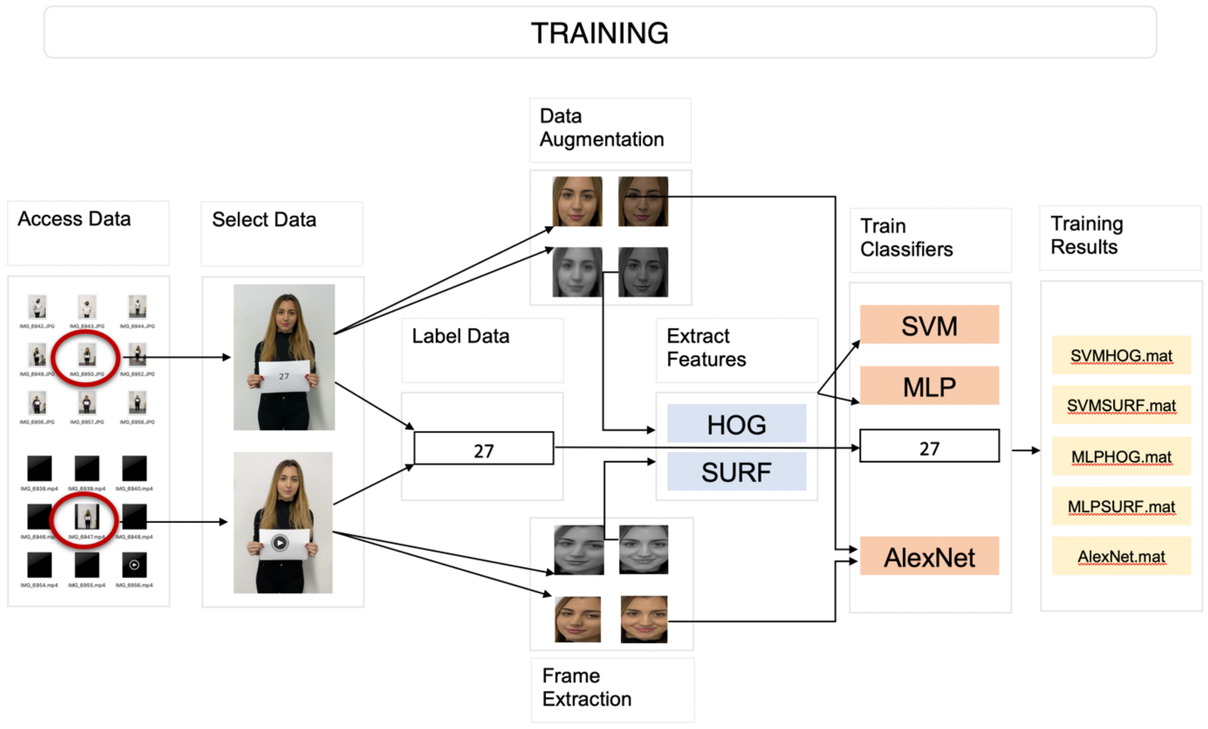


Fig. 1. Training approach

Data Augmentation

Iterate over separated labels to extract faces from the pictures provided of each individual. To further enrich the data samples, extract frames from each video by iterating over each labelled folder. Upon extracting the frames, resize the detected faces (227 x 227 for CNN) and convert all samples into greyscale for MLP and SVM and keep RGB copies for CNN. In order to avoid overfitting while training and to ensure the conditions of the images matched with that of group images, gaussian blur, image rotation, brightness adjustment to be performed on each detected face. Considering the limitation of access to computational power, aiming for training data in the range of 6,000 to 8,000 samples, ideally 150 images per class.

Feature Extraction

HOG

Histogram of Oriented Gradients (HOG) is a feature descriptor used in computer vision to extract features from an image. It focuses on the shape of an object and extracts useful information from the image while removing extraneous information. It is a concept that was first described by Robert K. McConnell [10] but gained popularity within the field of computer vision due to the paper published by Navneet Dal and Bill Triggs on Histograms of Oriented Gradients for Human Detection [11].

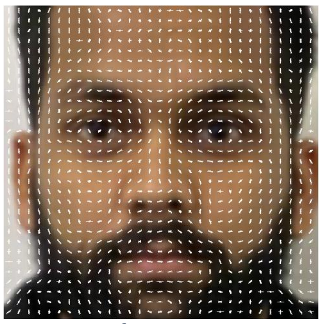


Fig. 2. HOG features on training image

HOG features provide directions of the edges and can be calculated by breaking down the image into smaller regions and extracting the gradients and orientation for that region. Histogram is then generated for each region based on the gradients and orientations of the pixel values. [12] Hog features overlaid on an image from our training data is shown in Fig. 2.

SURF

Speeded-Up Robust Features (SURF) is another feature descriptor used in computer vision to extract features from an image. It is based on Scale Invariant feature transform (SIFT) algorithm but relies on integral images for image convolutions and known to be faster and more robust. It was introduced by Herbert Bay, Tinne Tuytelaars and Luc Van Gool in the paper "SURF: Speeded Up Robust Features" [13]. SURF is good at handling blurred and rotated images but performs poorly when illumination is changed. Blob features overlaid on the image from our training data is shown in Fig. 3.

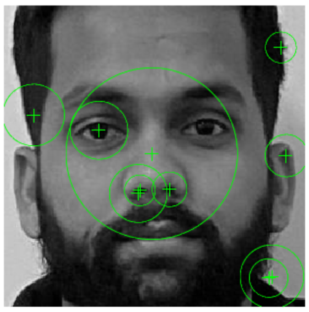


Fig. 3. SURF blobs on training images

Classifiers

Multilayer Perceptron

A multilayer perceptron (MLP) is a deep, artificial neutral network. It is composed of an input layer to receive the signal, an output layer to that makes decision or prediction about the input and in-between those two, an arbitrary number of hidden layers that are the true computational engine of the MLP [2]. Even though it is deemed insufficient for modern computer vision tasks, it is still widely used for both regression and classification tasks.

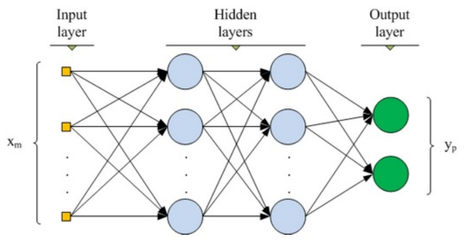


Fig. 4. Typical architecture of MLP with two hidden layers.

Pros	Cons
• Learns and model nonlinear relationships that is easy to use and understand. • Features can be extracted from the neurons in hidden layers • Can achieve high level of learning with low supervision. • Performs well on large volume of data with volatility.	• Cannot predict influence of independent variables on dependent variables • Computationally expensive and time consuming to train with traditional CPUs. • Heavily reliant on labelled training data. • Very prone to overfitting due to the high complexity. • Images have to be in greyscale

Support Vector Machines

Support Vector Machines (SVM) is a non-probabilistic classifier. Given a binary classification task, SVM separates the dataset into two different classes by maximising the margin and finding the decision boundary knows as the hyperplane. Data points nearest to the hyperplane are known as the Support vectors are the critical elements of the classifier as the position of hyperplane has to be changed if those points are to be removed.

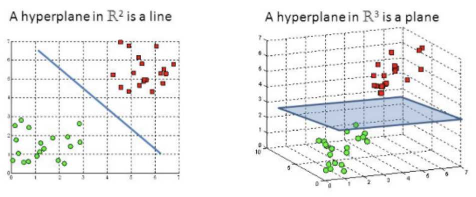
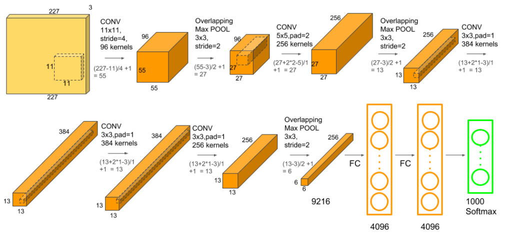


Fig. 5. Hyperplane of SVM in different dimension space

Pros	Cons
• Performs well with fewer data • Great generalization performance without prior domain knowledge • Convex optimisation problem so can always find global minimum error	• Ineffective on noise day with overlapping classes • Images have to be in greyscale • Performs poorly when there is no distinct separation between labels

AlexNet

AlexNet is a convolutional neural suggested in the paper "ImageNet Classification with Deep Convolutional Neural Network" [4] and the winning entry in the first *ImageNext* Large Scale Visual Recognition Competition in 2012. Network consisting of three fully connected layers and five convolutional layers which uses multiple kernels to extract interesting features from an image. [3]



227
3
11
11
227
CONV 11x11, stride=4, 96 kernels
(227-11)/4 +1 = 55
96
55
55
Overlapping Max POOL 3x3, stride=2
(55-3)/2 +1 = 27
96
27
27
CONV 5x5,pad=2 256 kernels
(27+2*2-5)/1 +1 = 27
256
27
27
Overlapping Max POOL 3x3, stride=2
(27-3)/2 +1 = 13
256
13
13
CONV 3x3,pad=1 384 kernels
(13+2*1-3)/1 +1 = 13
384
13
13
CONV 3x3,pad=1 384 kernels
(13+2*1-3)/1 +1 = 13
384
13
13
CONV 3x3,pad=1 256 kernels
(13+2*1-3)/1 +1 = 13
256
13
13
Overlapping Max POOL 3x3, stride=2
(13-3)/2 +1 = 6
256
6
6
9216
FC
4096
FC
4096
1000 Softmax

Fig. 6. Architecture of AlexNet

Architecture of AlexNet can be seen in Fig.6. It applies ReLU to add non-linearity after the very first convolutional layer and fully connected layer. ReLU increases the speed by six times without affecting the accuracy. In order to deal with overfitting, it uses dropout layers over regularisation and overlap pooling, which is also used to reduce network size.

Pros	Cons
• Performs with high accuracy for image recognition tasks • Image processing is built into the network structure eliminating the need for feature extraction methods • Accepts RGB images	• Pre-trained with images • Can be very time consuming to train with traditional CPU's • Require large amount of training data to perform well

Implementation

Data Augmentation

Due to the disruption caused by the pandemic and taking time constraint into consideration, author opted against performing OCR and, instead, manually separated the files into sub folders based on the label.

CascadeObjectDetector function in MATLAB which uses the Viola-Jones [5] algorithm was used to detect faces in the images. Detection threshold was set to 10. If there were multiple detections around an object, detections that meet the threshold were merged to produce on bounding box around the target object. [6] Size of smallest detectable object was set to 70 x 70 and largest was set to 700 x 700 to retrieve only relevant objects.

Images

The sizes of the faces in group images varied depending on where the individual was seated as faces at the front were larger (around 250x250) than faces at the back (around 80x80). However, faces in the individual images were in the range of 500x700. Therefore, each detected face from images and frames was resized to 120x120 for SVM and MLP classifier and 227x227 for AlexNet.

In order to avoid the black edges, each image in the labelled folders were rotated 15 degrees clockwise and anti-clockwise, and then the following augmentations techniques were initially applied on the rotated and original images:

- Gaussian filter was applied using MATLAB built-in function *imgaussfilt* with standard deviation of 2 to blur the images.
- Brightness of each images was reduced with a 1.5-fold reduction
- Gaussian blur and brightness reduction were combined to create dark and blurred images
- Brightness of each images was increased with a 1.5-fold enhancement

Upon extracting frames and running our classifiers, although models performed well on the individual pictures, the models performed poorly on group photos. Hence, further augmentation techniques were applied to match the conditions of the group photos. However, this approach further affected the accuracy of the model as augmentation such as dilating the images (shown in Fig.7.), added noise to the photos. This resulted in further misclassification. Hence, author removed some of the extreme augmentation methods and only the selected techniques below were applied.

Fig. 7. Dilated augmentation

- *Imguidedfilter* was applied for smoothing of image while preserving edged
- *Imguidedfilter* was applied and brightness was increased with 1.3-fold enhancement
- Image was down sample by ¼, brightness increased (1.3-fold) and resized to the required scale

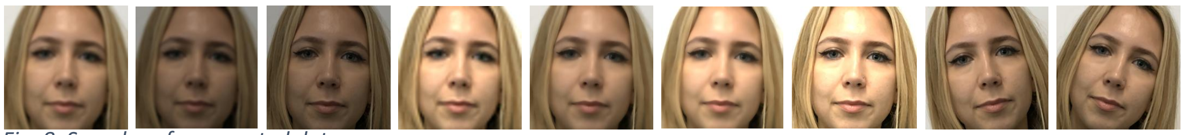


Fig. 8. Samples of augmented data

Frames

MATLAB *videoReader* function was used to read the video files and extract frames. Brightness threshold was set to exclude the dark frames at the start and end of the video. Number of frames on videos varied between 45 to 70. 30 frames were initially selected from each available video per label. However, after manually inspecting the extracted frames, it was clear that a lot of the frames were identical to each other. This could result in overfitting during training. Therefore, author opted to extract 7 frames per video (every 5th frame till the 35th frame). This resulted in much better results with very minimal duplicates. As per our earlier approach conducted on images, face detection was performed, and each detected face was resized accordingly. Author opted against further augmentation on frames as initially planned. In significant amount of the extracted frames, eyes were closed, some frames were blurred, and all frames were compressed as show in Fig. 9. Therefore, it was decided that the frames currently match the condition of the faces to be extracted from group images and further augmentation could potentially add unwanted noise in the data.

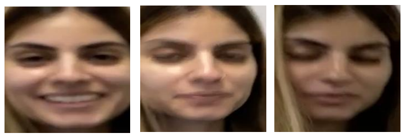


Fig. 9. Samples of extracted frames

By reducing the number of frames extracted per video, the amount of data samples per ID was reduced and aimed to be in a range of 100 to 150. Upon completing the augmentation on images and frame extracting, 32% of the data (15 labelled folder) were below 100 images and about 9% (4) was less than 50 images. This is due to the individuals who were glasses, hoods and other challenging conditions. Therefore, further augmentation was performed on these selected ID's. Initially different classification models (*profileFace* and *frontalFaceLBP)* of *cascadeObjectDetection* were applied. This resulted in more data samples for some folders but failed to do so for others. Hence, more frames were extracted, and threshold was reduced from 10 to 6 and 3. While this resulted in more data samples and allowed us to meet the minimum requirement of 100 samples for each ID, it also resulted in identical images. This could potentially result in overfitting during validation and also lead to poor performance on the unseen data. The author also manually inspected each labelled data folder to remove all faulty images wrongly detected by the object detector as shown in Fig. 10.

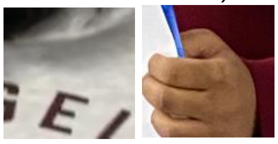


Fig. 10. Faulty data produced by object detector

Even with the number of measured techniques applied to enrich the data samples, the models failed to predict well on the unseen group photos (discussed further below in the classifiers section). As the final throw of dice, the author opted to extract faces from 20 selected group photos and manually assigned it to those extracted faces to the designated ID's. This resulted in a total of 8901 data samples for our study. Even though this is a higher number than author's earlier plan, in order to further improve the performance accuracy of the models, the number of data samples had to be increased at the cost of computational complexity. The number of data samples per class is shown in Fig. 11. All of the ID's met the minimum requirement of 100 data samples with an average of 185 (25 more than as planned) data samples per ID. ID's 42 and 6 contained the highest number samples but as it was below 250, the author opted against removing any extra samples.

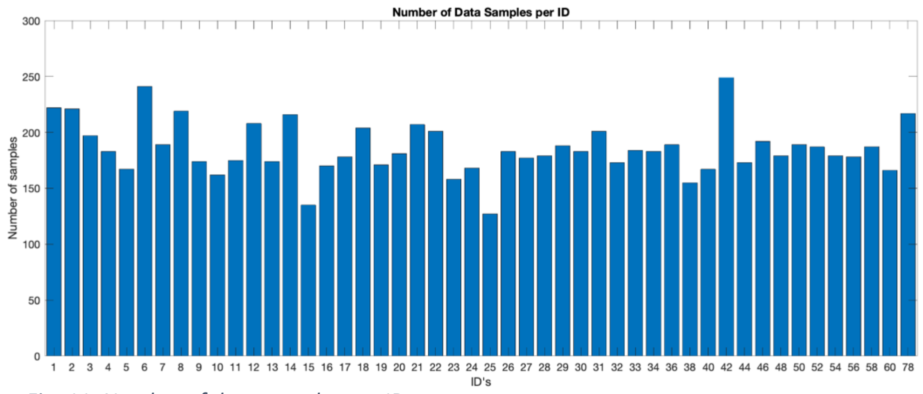


Fig. 11. Number of data samples per ID

Although the author initially opted for an 80:20 train test split ratio, due to the poor performance of the model during the initial stages, the author decided to increase the training size. This allowed the models to be trained on a maximum possible number of data samples, using a more common 90:10 train test split ratio. This resulted in a training size of 8011.

Feature Extraction

MATLAB built-in function *extractHOGFeatures* was used to extract HOG features for the training and validation images. MATLAB built-in function *bagOfFeatures* was used to extract SURF features for the training images and construct the visual vocabulary by reducing the number of features through quantisation of feature space using K-mean clustering. [14] 500-word visual vocabulary was created using K-means. Extracted HOG and SURF features on greyscale images, were used to train SVM and MLP classifiers.

Classifiers

SVM

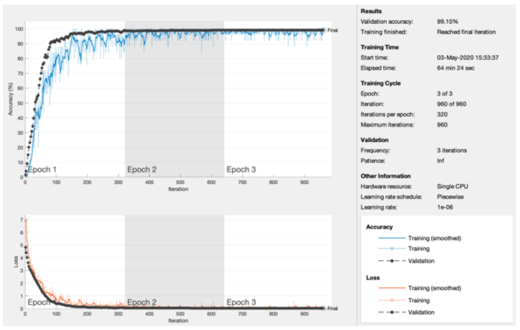
SVM models (HOG and SURF) were trained using *fitcecoc* function. Hyperparameter optimization was initially set to auto mode to find the optimal parameters for our model and 5-fold cross validation was used. However, due to the size of the data and running on CPU, there was no results returned after running for over 20 hours and, ultimately, the laptop crashed. Therefore, the author opted to train the models with the default MATLAB settings. Nevertheless, the classifier was cross-validated using 10-fold cross validation and resulted in generalization error of 0.75% for HOG features and 1.16% for SURF features, which indicates our model generalize fairly well. SVM performed excellently during initial validations, with accuracies of 99.87% on SURF and 99.36% on HOG features. However, when tested on group photos, the classifier only managed to obtain an accuracy of 47.56% during the early stages. Upon increasing the augmentation and inclusion of faces from group images, validation accuracy dropped slightly to 98.54% with SURF features and 97.76% on HOG features. This is could be explained by the increase in distinct separation between labels with the added data.

MLP

MLP was trained using *patternet*. *trainscg* was chosen as the training function and upon testing on multiple number of hidden neurons [1 to 200], 65 was chosen for the final model. It was shown to perform well with HOG features with accuracy of 80.29% compared to 78.37% on SURF features. Like SVM, validation accuracy prior to the inclusion of group phots and further augmentation were higher. MLP was the worst performing model on validation and when tested on group photo.

AlexNet

MATLAB function *trainNetwork* was used to train AlexNet. Initially MATLAB function *imageDataAugmentar* was used instead of manual augmentation to train the model. However, due to the poor performance of the model, manual augmentation was performed, as previously conducted for the SVM and MLP models. Learning rate of 0.0001 was used with drop factor of 10% after every epoch as an effective regularisation method. Epoch was initially set to 8. However, after the first epoch, there was very little change in the training and validation accuracy, as shown in Fig. 12. Therefore, author reduced the epoch to 3 which also significantly reduced the running time. AlexNet achieved the best accuracy of 99.10% during validation and was consistently the best performing model. Upon displaying the validation results of wrongly classified faces from randomly chosen samples, it was evident that objects (glasses) and human emotions significantly influenced the misclassified predictions of the model as shown in Fig. 13. This highlights the limitation of the model and with the faces in the group photos expected to be varied with emotions and objects covering faces, it is expected to have a drop-in performance in the testing phase.



Results
Validation accuracy: 99.10%
Training finished: Reached final iteration

Training Time
Start time: 03-May-2020 15:33:37
Elapsed time: 64 min 24 sec

Training Cycle
Epoch: 3 of 3
Iteration: 960 of 960
Iterations per epoch: 320
Maximum iterations: 960

Validation
Frequency: 3 iterations
Patience: Inf

Other Information
Hardware resource: Single CPU
Learning rate schedule: Piecewise
Learning rate: 1e-06

Accuracy
Training (smoothed)
Training
Validation

Loss
Training (smoothed)
Training
Validation

Fig. 12. AlextNet training results

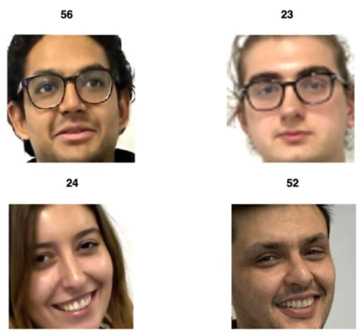


Fig. 13. AlextNet validation results samples

RecogniseFace

Best performing models during validation stage were saved and tested using the *RecogniseFace* function. An algorithmic workflow of the function is shown if Fig. 14. As per the requirement of the study, the function accepts 4 arguments as inputs and returns P matrix as the output along with the input image with detected face(s). If the user input invalid queries, the function will display an error message accordingly. All the files required to test the classifiers are submitted as part of the submission and all submitted files must be downloaded and located in one location for the function to run successfully.

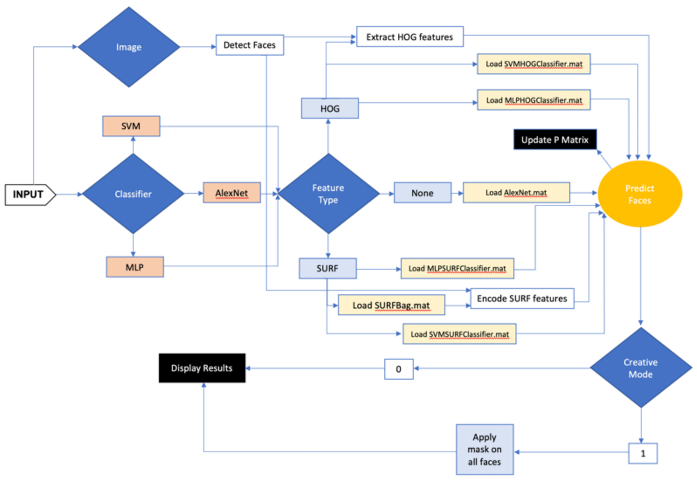


Fig. 14. Algorithmic workflow of RecogniseFace function

Classifiers were successfully tested on five different group images and the results are shown in Fig. 16. Validation accuracy is from earlier stage is also included to further evaluate the drop-in accuracy on unseen image, which in turn indicate the limitation of the models. AlexNet is our best

Fig. 15. Left: Correctly classified image, Right: Misclassified image

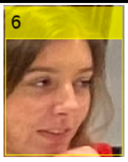
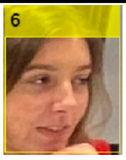
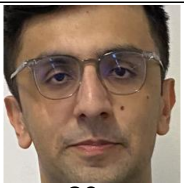
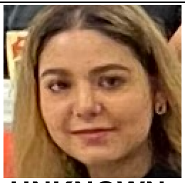
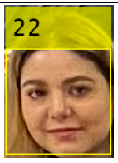
performing model with an accuracy of 86.64%. Training the model by extracting faces form group photos during validation contributed significantly towards the results. If the unknown faces were not to be considered, AlextNet was mainly misclassifying on faces that is not easy to distinguish even for the human eye as shown in Fig. 15.

Classifier	Feature Type	Validation Accuracy	Test Accuracy
SVM	HOG	97.76%	76.12%
SVM	SURF	98.54%	79.55%
MLP	HOG	80.29%	68.45%
MLP	SURF	78.37%	62.19%
AlexNet	None	99.10%	86.64%

Fig. 16. Validation and Testing results of classifiers

Unknown Labels

In order to add an extra layer to the model prediction, author implemented the additional task of labelling unknown faces. Threshold was set on posterior probability of the prediction to identify weaker classifications and label them as unknown. Due to SVM being a linear classification model, author was unable to obtain posterior probabilities and, therefore, could not successfully evaluate unknown labels prediction on SVM classifier.
Setting a threshold proved to be successful in some cases as shown in Fig.17. as it helped us identify which of the predictions were weaker (lower probability). Ultimately, it resulted in some correctly predicted classification labelled as unknown. The error rate was low in AlexNet compared to MLP, which indicates AlexNet is strong in its predictions. Nevertheless, it affected the overall accuracy of both models significantly on some images compared to other. Therefore, author opted to remove the threshold for the best performing model (AlexNet) to ensure that true accuracy is achieved when tested on unseen images.

Ground truth	Prediction	Prediction with Threshold	Result
24	 ✗	 ✗	Falsely classified
 **20**	✓	✗	Falsely classified as unknown
 **UNKNOWN**	 ✗	✓	Successfully classified as unknown
05	✓	✗	Falsely classified as unknown

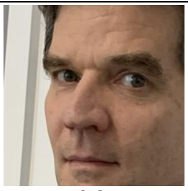
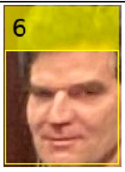
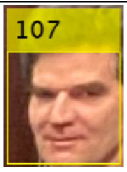
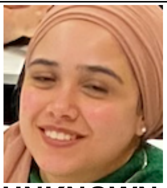
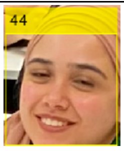
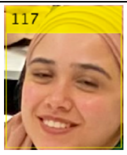
 **06**	 ✓	 ✗	Falsely classified as unknown
 **UNKNOWN**	 ✗	 ✓	Successfully classified as unknown

Fig. 17. Evaluation of unknown labels

Creative mode

As per the requirement of the study, *creativeMode* was implemented by the author in the *recogniseFace* function. When input variable is set to 1, faces detected by the function is altered by superimposing a cartoon face expression on face(s). Cartoon face expression was resized using the properties of bounding box detected by the face detector to ensure the best possible fit was to be achieved. Cartoon face expression was a transparent, PNG file with black background colours which was replaced by the pixel values of detected face to impose the cartoon face expression on the face(s) as shown in Fig. 18.

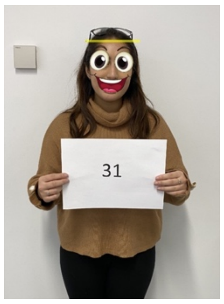


Fig. 18. Cartoon face expression imposed on a test image

Reflection

In this study, we have successfully implemented three different classifiers for our face recognition task. We successfully trained the SVM and MLP models using HOG and SURF features while pre trained AlexNet model was the standout performer. HOG features were more efficient with prediction for SVM classifiers and SURF were more efficient with MLP classifiers. From our study it is also evident why MLP's are not deemed sufficient enough for modern computer vision tasks. Overall, our approach provides to be very effective in most scenarios.

During the augmentation stage, we understood that augmentation is only beneficial to an extent; any extreme augmentation that does not match the condition of the training and testing samples could potentially be seen as an outlier and affect the overall accuracy of the performance. Nevertheless, models require large amount of data to sufficiently distinguish different labels. Lack of computational powers proved to be costly factor in finding the optimal parameters for our models.

Limitation of the models was evident during the initial validation stages. When models are trained on individual photos and tested on group photos, that are in different condition to that of individual photos (such as different brightness and lighting, different face sizes, different face angles, etc.), the models performed poorly. Performance can only be improved by including faces from group photos in the training set. This is a common problem in computer vision tasks and further studies is required in produce a model that can recognise similar images regardless of the condition of the images.

Further Work

Further study is required to successfully label unknown images. For the SVM model, posterior probabilities can be extracted by retrieving SVM scores and training a logistic regression to turn them into classification probabilities to predict unknown labels successfully. Hyper parameter turning by performing grid search for the classifiers could make significant impact on improving the model accuracy. The author was unable to do so for this study due to lack of access to GPU-powered computers.

The faces in the front of the group photo are of bigger than the faces in the back photo. Therefore, an experiment can be conducted in using training images of different dimensions to compare the generalisation capabilities between a model trained on bigger training images on small faces in the back and a model trained on smaller training images on large faces in the foreground.

Upon implementation of the unknown label, it is clear that the models failed to distinguish between individuals that looked similar or had strong identical features. To build on that, further studies can be conducted in using different models such as YOLO [7] to further distinguish these similar characteristics. Furthermore, our models failed to recognise and identify faces that were partly covered or tilted. Therefore, models could be trained by covering parts of the face, such as mouth or ear, and results could then be evaluated.

In our study, we have also learnt that the displayed human emotions play a part in model predication. Therefore, by identifying these human emotions from extracted images or combining text and images by labelling the image as “happy” or “sad” and training the models with this, it could to more supportive results, when keeping the broader aim of face recognition accuracy.

References

[1] Kuo L, “China brings in mandatory facial recognition or mobile phone users”, The guardian, Dec. 2019.
[2] Nicholson C, 2019, “A Beginner’s Guide to Multilayer Perceptrons (MLP)”, Pathmind Wiki.
[3] Sunita Nayak, “Understanding AlexNet”, LearnOpenCV, Jun. 2018.
[4] Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton, “ImageNet Classification with Deep Convolutional Neural Network” Neural Information Processing Systems, pp. 4824, 2012.
[5] Paul Viola, Michael Jones “Robust Real-time Object Detection”, International Journal of Computer Vision, 2001.
[6] Vision.CascadeObjectDetector, MathWorks, 2020.
[7] Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi, “You Only Look Once: Unified, Real-Time Object Detection”, Computer Vision and Pattern Recognition, May. 2016.
[10] “Histogram of oriented gradients”, Wikipedia, 2020.
[11] Navneet Dalal, Bill Triggs, “Histograms of Oriented Gradients for Human Detection”, Conference on Computer Vision and Pattern Recognition, pp. 886-893, Jun. 2005.
[12] Aishwarya Singh, “Feature Engineering for images: A Valuable Introduction to the HOG Feature Descriptor”, Analytics Vidhya, Sept. 2004.
[13] Herbert Bay, Tinne Tuytelaars and Luc Van Gool in the paper “SURF: Speeded Up Robust Features”, European Conference on Computer Vision, 2006.
[14] “Image Category Classification Using Bag of Features”, MathWorks, 2020.