Computer Vision Coursework Report

Author: Tareq Galala

Abstract: In this report, we will present the methods and tools used to build a face recognition pipeline to recognize faces for a classroom. In addition to using several combinations of SURF, SIFT, SVM & LR, we will use a VGG16 convolutional neural network-based architecture pre-trained on VGG-Face dataset and fine-tuned on our custom face dataset. In addition, we will use three creative modes on the detected faces. The report will analyze methods, classifiers and implementations used in the project along a critical analysis of the results obtained from the experiments with future directions and real-life applications.

Index Terms: Face Recognition, OCR, Face Filters, VGG Face, MTCNN, VGG16, SVM, LR, SIFT, SURF

1. Introduction

Face recognition is one of the many wonders that AI research has brought forward to the world. According to Wikipedia a facial recognition system is a technology capable of identifying or verifying a person from a digital image or a video frame from a video source. There are multiple methods in which facial recognition systems work, but in general, they work by comparing selected facial features from given image with faces within a database. It is also described as a Biometric Artificial Intelligence based application that can uniquely identify a person by analysing patterns based on the person's facial textures and shape. [1][2][3]

1.1 Problem Statement

Have you ever been introduced to a number of people, but forget their names? Humans are exceedingly capable of recognising faces previously seen before, but matching names to faces can be difficult. Imagine having an app running on a device (e.g. smartphone, glasses) that recognises people and prompts their names, helping to avoid blurting an awkward “I’m sorry but I forgot your name?”. Person identification is important for a variety of practical tasks, including cybersecurity and authentication. In this project, we will develop a computer vision system to detect and identify individuals using a database of known face images (specifically, students in this module).

1.2 Dataset structure

The data used in this project contains videos and images for students taking module INM460: Computer vision. The data are categorized in two groups, First, the individual photos which are a group of photos and videos taken individually for the students whom participated in this special photo shooting, which are 48, all of them holding a white sign with their corresponding ID’s. A number of front-view and side-view photos and video were taken for the 48 individuals with the same setup of a white background. The group class photos were a total of 60 files taken from different angles in the class. There were 8 students that participated in the group photos but not the individual shoot. The total number of students in the group raw material are 8 in addition to the previous 48 students. However, one student from the individual photoshoot was not in the group photos which brings the total to 55, although the total labels are 56.

The group class material unlike the individual photos have different photo settings, faces had different sizes and scales. The photos types are in jpeg format and the videos are in MP4 format. The videos are short bursts of 2 seconds each. It’s worth noting that the number of files taken for each student is different.

Creating a data set is about data preparation which is such an important step in machine learning projects. Gathering and labelling data takes most of our time, especially data sets that are accurate enough to reflect a realistic vision of the market. Datasets are required to train our machine learning algorithm(s). We structured the data as following:

1. Organise a folder structure for each student with their corresponding ID, this means that each person will have a separate folder. We created 48 folders for all students from the individual photoshoot and were labelled according to their numbers identification.

2. The students that didn’t participate in photoshoot but were present in class photos will be called unknowns and will have also folders. A total of 8 students fall in this category and their folders are labelled starting from unknown1, and ending with unknown8.

Total number of folders is 56 photos labelled with their perspective class name.

2. Face Recognition

In this section, we will describe all the tools and methods used to execute our project. The goal of this project is to develop a function that returns a matrix P describing student(s) present in an RGB image I. The matrix P should be of size Nx3, where N is the number of people detected in the image. The report will contain 3 columns, an ID, which is the unique number assigned to that person, x-coordinate of the centre of the face of that face and lastly the y-coordinate of the centre. Fig. 1. demonstrates our face recognition pipeline.

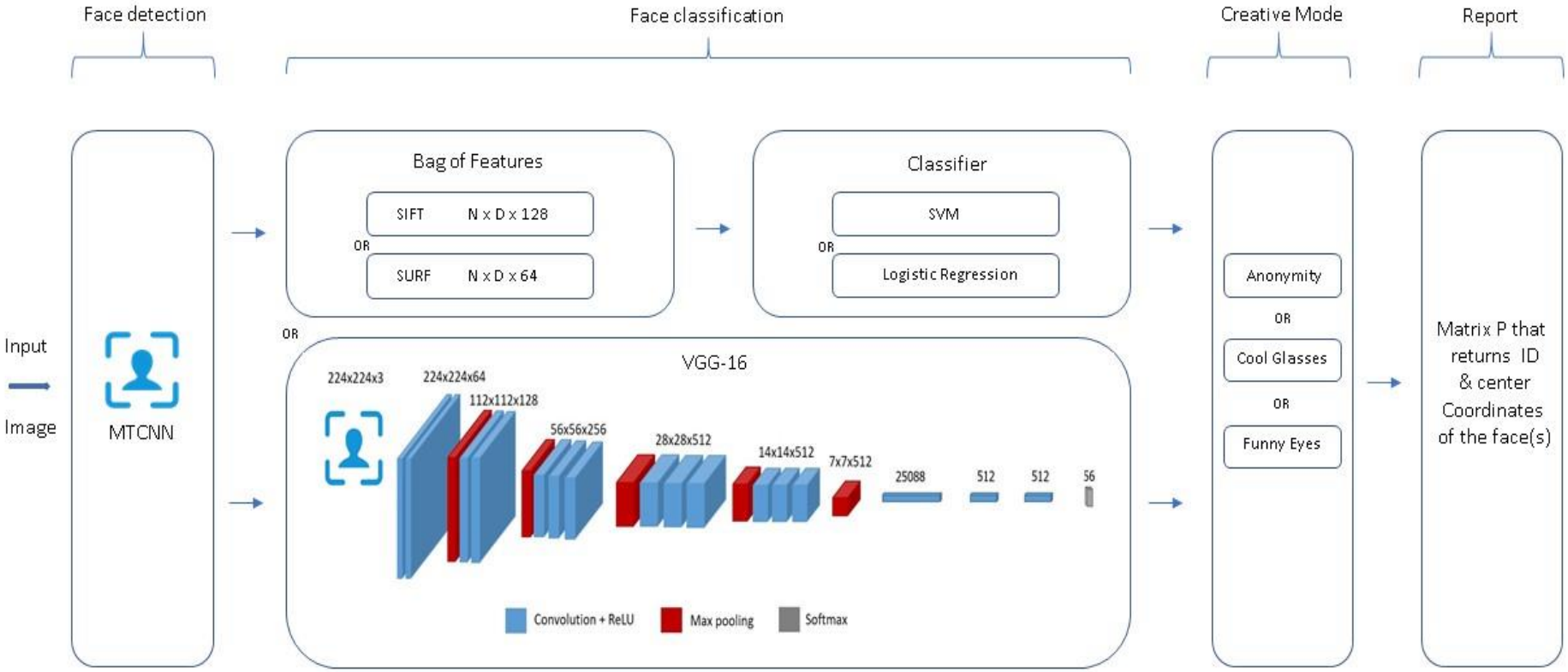

Fig. 1. An overview of the RecogniseFace.py pipeline. An input image goes through four stages. The first stage is the face detection with MTCNN. In stage two the function accepts three arguments, SIFT, SURF or NONE. The classifier stage is stage 3, the argument accepts either of three arguments, Support vector Machines (SVM), Logistic Regression (LR) or a CNN network VGG16. The last stage is creative mode which has a choice of three options, option 1: anonymity, option 2: cool glasses and final option 3: funny eyes. The function outputs a report matrix

2.1 Face Detection

Object detection is a computer technology related to computer vision and image processing that deals with detecting instances of objects such as human faces, cars, etc. Face detection can be considered as a specific case of object detection. The primary aim of face detection algorithms is to determine whether there is any face in an image or not. This is an essential step for face recognition. It is widely used in law enforcement, security and even unlocking phones and personal computers. There are several methods in face detection, the one we are interested in, is appearance based which depends on training images of faces and techniques such as machine learning finding features in images and also extracting faces. The one we are introducing in our project is MTCNN which is neural network appearance based.

2.1.1 MTCNN

Multi-Task Cascaded Convolutional Neural Network (MTCNN), was presented by Kaipeng Zhang, et al. in 2016. In their paper, they proposed a new framework to integrate these two tasks using unified cascaded CNNs by multi-task learning. The proposed CNNs consist of three stages. In the first stage, it produces candidate windows quickly through a shallow CNN. Then, it refines the windows to reject a large number of non-faces windows through a more complex CNN. Finally, it uses a more powerful CNN to refine the result and output facial landmarks positions. [7]

Fig. 2 from Kaipeng Zhang, et al. demonstrates the three stages (P-Net, R-Net, and O-Net). These convolutional networks outperform many face detection benchmarks and works simultaneously in real-time, allowing great performance for this network.

Stage 1:

In stage 1 an image pyramid is created from the input; the goal is to detect faces in all possible sizes. All copies are passed to the first Network P-Net which returns bounding boxes for any possible faces. Non-Maximum Suppression, or NMS then refines the boxes and keep the ones with the highest confidence scores and deletes the ones that are overlapping. The coordinates of the surviving bounding boxes are then sent to stage 2.

Stage 2:

R-Net is the second stage and the process are repeated by refining bounding boxes and finding more accurate coordinates. Similar to stage one the bounding boxes with lower confidence gets removed and NMS eliminates the redundant boxes. The boxes get reshaped to 48 x 48 pixels, and standardized and then passed to the final stage.

Stage 3:

O-Net stage is the final stage and this network have three outputs:

1. Bounding box coordinates
2. Facial landmarks (‘left eye ‘, ‘right eye ‘, ‘nose ‘, ‘mouth left ‘, and ‘mouth right ‘) coordinates.
3. Bounding box confidence level

Same as the previous stages, lower confidence boxes get deleted and NMS runs one last time to find a box for each face.

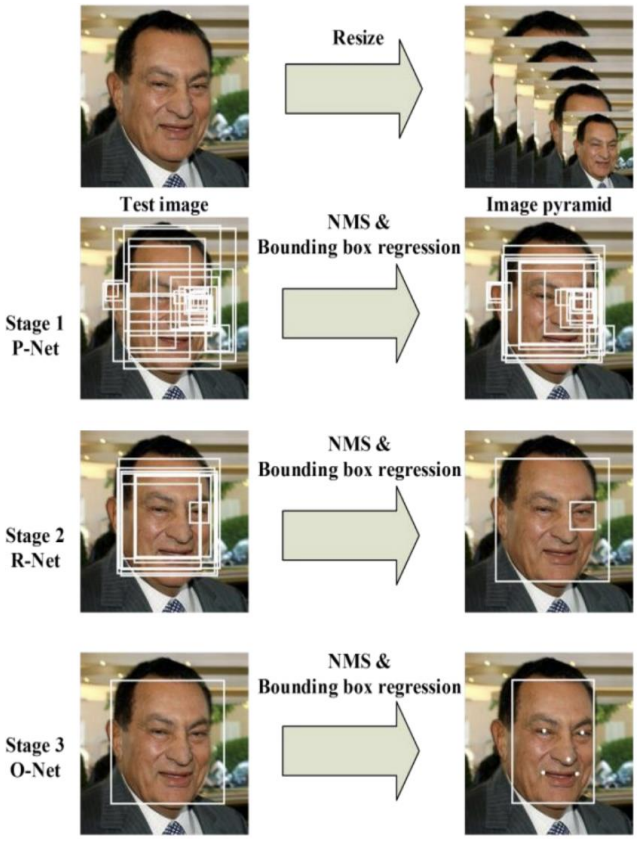

Fig. 2. Pipeline of our cascaded framework that includes three-stage multitask deep convolutional networks. First, candidate windows are produced through a fast P-Net. After that, we refine these candidates in the next stage through a R-Net. In the third stage, the O-Net produces final bounding box and facial landmarks position. [7]

MTCNN performs very well as the 3 networks and its layers fine tune the results. The image pyramid plays a very important role in finding faces in all regions of the image. The three main contributions for performance improvement are carefully designed cascaded CNNs architecture and joint face alignment learning. The version will be using in our project is MTCNN with TensorFlow by Iván de Paz. The team that developed this model used the WIDER-FACE dataset to train bounding box coordinates and the CelebA dataset to train facial landmarks.

2.2 Feature Extraction

Feature extraction is a process of dimensionality reduction by which an initial set of raw data is reduced to more manageable groups for processing. Feature extraction is the name for methods that combine variables into features, effectively reducing the amount of data that must be processed, while still accurately and completely describing the original data set. One of the practical uses is Bag of Words, a technique for natural language processing that extracts the features used in a sentence, document, website, etc. and classifies them by frequency of use. This technique can also be applied to image processing. SURF & SIFT probably are the most common in image processing feature extraction.

2.2.1 Scale-Invariant Feature Transform (SIFT)

SIFT is a feature detection algorithm that detects and describes local features in images. It was published by David Lowe in 1999. It uses the Difference of Gaussians (DoG) filter, which approximates the normalised Laplacian of Gaussian (LoG). SIFT algorithm, which according to [4] consists of four computational stages: (i) scale-space extrema detection, (ii) removal of unreliable keypoints, (iii) orientation assignment, and (iv) keypoint descriptor calculation.

One of its more recent uses also includes face recognition, where it was shown to deliver encouraging results. SIFT-based face recognition techniques found in the literature rely heavily on the so-called keypoint detector, which locates interest points in the given image that are ultimately used to compute the SIFT descriptors [5].

A feature descriptor is calculated for each keypoint. First a small window is positioned around each keypoint at the relative scale, secondly, a gradient magnitude and orientation of the image pixels are calculated, thirdly the gradient orientations are rotated relative to the keypoint dominant orientation, then a window is divided in subregions. For each subregion, an orientation histogram with 8 bins (each covering 45°) is created, each pixel votes for an orientation bin, with a weight equal to the gradient magnitude. Fig. 3 describes the process.

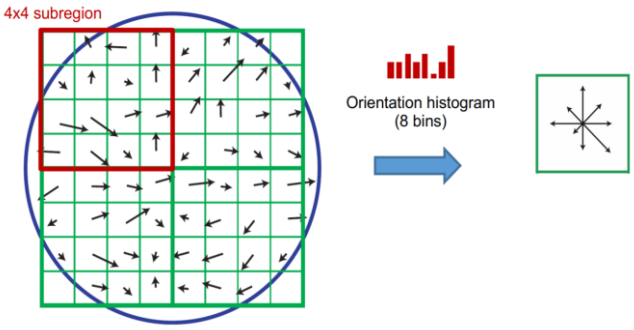

Fig. 3. For each subregion, an orientation histogram with 8 bins (each covering 45°) is created: each pixel votes for an orientation bin, with a weight equal to the gradient magnitude

The number of elements of the SIFT descriptor was 16 subregions x 8 bins = 128. We use a feature vector of 128 elements to describe each keypoint

2.2.2 Speeded Up Robust Features (SURF)

SURF algorithm is based on the same principles and steps as SIFT; but details in each step are different. The algorithm has three main parts: interest point detection, local neighbourhood description, and matching. SURF is much faster to compute than SIFT, and does not sacrifice performance much.

Bay et al. [6] introduced the SURF feature descriptor which is based on the sum of the Haar wavelet response around the point of interest.

A small window is positioned around each keypoint and a window is divided in subregions. For each subregion, 25 sample points (in a 5x5 grid) are selected. For each sample point, two Haar wavelets are applied, with a size depending on the scale associated with each keypoint. Haar wavelets sum the pixel intensities underneath the white portion and subtract the ones underneath the black one, the wavelet responses are summed up (both with and without sign) within each subregion

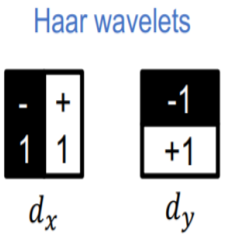

Fig. 4. Haar wavelet filters to compute the responses in x (left) and y direction (right). The dark parts have the weight -1 and the light parts +1

$\sum dx, \sum dy, \sum |dx|, \sum |dy|$

The descriptor for this subregion is defined by these 4 numbers. Fig. 5 shows the properties of the descriptor for three distinctively different image intensity patterns within a subregion. One can imagine combinations of such local intensity patterns, resulting in a distinctive descriptor. [6]

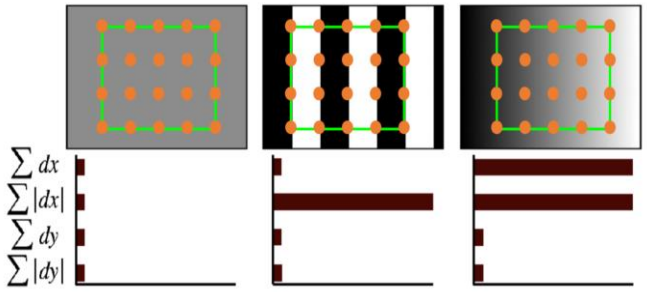

Fig. 5. The descriptor entries of a sub-region represent the nature of the underlying intensity pattern. Left: In case of a homogeneous region, all values are relatively low. Middle: In presence of frequencies in x direction, the value of ∑|dx| is high, but all others remain low. If the intensity is gradually increasing in x direction, both values ∑dx and ∑|dx| are high. [6]

The number of elements of the SURF descriptor was 16 subregions x 4 numbers = 64. We use a feature vector of 64 elements to describe each keypoint.

2.3 Classifiers

2.3.1 Logistic Regression

Logistic regression is a statistical method for binary classification. Logistic regression is not regression, but It is referred to as logistic regression as it works very similar to linear regression.

In statistics, the logistic model is used to model the probability of a certain class or event existing such as yes or no and win or lose. This can be extended to model several classes of events such as determining whether an image contains a dog or a cat or even recognising faces. Each object being detected would be assigned a probability between 0 and 1. The logistic function, also called the sigmoid function which looks like a S-shaped curve that can take any real-valued number and map it into a value between 0 and 1.

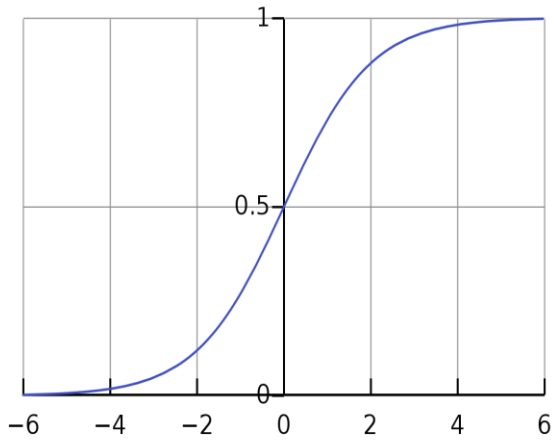

Fig. 6. Standard logistic function

The coefficients of the logistic regression algorithm can be estimated from the training data by using maximum-likelihood estimation. Maximum likelihood estimation is a learning algorithm that is used often in a number of machine learning algorithms. The best coefficients would result in a model that predicts a value close to one for the default class and a value close to zero for the other class. The goal is to minimize the error resulted from the model predicted probabilities.

$$\ell(\beta_0, \beta) = \prod_{i:y_i=1} p(x_i) \prod_{i:y_i=0} (1 - p(x_i)).$$

We use the maximum likelihood to estimate the parameters. This gives probability of observed zeroes and ones in the data. Solvers like BFGS are used in an iterative method for solving unconstrained nonlinear optimization problems with support to L2 regularization. Logistic regression is widely used because it is very efficient, interpretable and does not require computational resources. In our model we combine SIFT or SURF features with our classifier logistic regression.

2.3.2 Support Vector Machines

In machine learning, support-vector machines (SVMs) are supervised learning models with associated learning algorithms that analyse data used for classification and regression analysis.

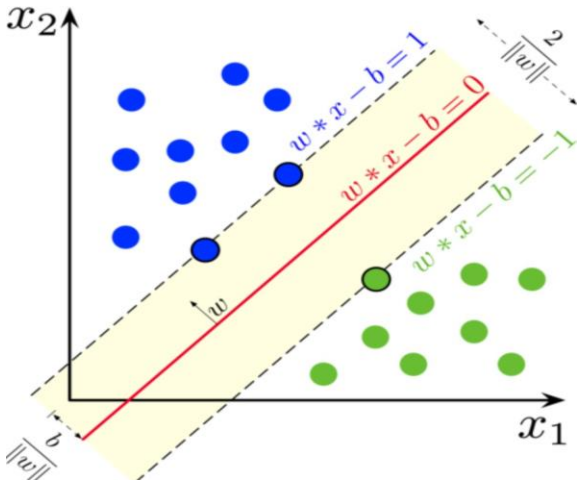

Fig. 7. Maximum-margin hyperplane and margins for an SVM trained with samples from two classes. Samples on the margin are called the support vectors.

SVM looks for the hyperplane that separates classes and can be also used in multiclass cases. In addition to performing linear classification, SVMs can efficiently perform a non-linear classification using what is called the kernel trick, implicitly mapping their inputs into high-dimensional feature spaces.

According to Chittora et al. [8] they presented the face recognition experiments using linear support vector machines with one against all multiclass classification strategy and Radial Basis Function as kernel function for SVM. As shown in the comparison with other techniques, it appears that the RBF SVMs can be effectively trained for face recognition. The experimental results show that the RBF SVMs are a better learning algorithm than other kernel function approaches of SVM for face recognition. [8]. The following function is the Gaussian radial basis function (RBF).

.

$$K(\mathbf{x}, \mathbf{x}') = \exp(-\gamma \|\mathbf{x} - \mathbf{x}'\|^2)$$

The last part of the equation may be recognized as the squared Euclidean distance between the two feature vectors. A multi-class pattern recognition system can be obtained by combining two class SVMs. There are many available schemes for this purpose. One is the one-against-all strategy to classify between each class and all the remaining; The other is the one-against-one strategy to classify between each pair. [8] In this project will be using the latter and combine it with SURF & SIFT.

2.3.3 VGG16

VGG16 is a convolution neural net (CNN) architecture which was used to win the ILSVRC-2014 competition. It achieved a 92.7% test accuracy on ImageNet, which is a dataset of over 14 million images belonging to 1000 classes. VGG16 was proposed by K. Simony an and A. Zisserman from the University of Oxford in the paper “Very Deep Convolutional Networks for Large-Scale Image Recognition”. It is considered till date one of the excellent computer vision model architectures. The structure has convolution layers of 3x3 filter with a stride 1 and always used same padding and MaxPooling layer of 2x2 filter of stride 2, this simplicity was interesting as they lowered the hyper parameters to around 140m. The block arrangement is consistent, followed by two fully connected layers and a SoftMax for output. [8] The original architecture is shown in fig. 8.

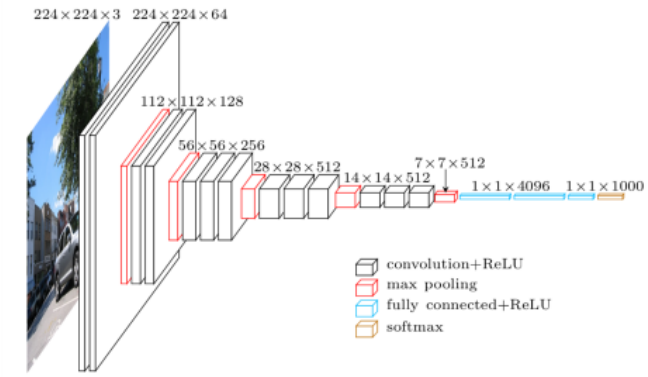

Fig. 8. Original VGG16 architecture used in ImageNet.

Although the original VGG16 was trained on millions of images, the classes were very generic. We opted for a more specific approach, VGGFace is a state-of-the-art model developed by researchers at the Visual Geometry Group (VGG) at the University of Oxford. The VGGFace refers to a series of models developed for face recognition and demonstrated on benchmark computer vision datasets. There are two main VGG models for face recognition, VGGFace and VGGFace2. The model that we will be using is VGGFace which is described by parkhi et al. in the 2015 paper titled “Deep Face Recognition.” They describe the process of training a face classifier first that uses a SoftMax activation function in the output layer to classify faces as people. This layer is then removed so that the output of the network is a vector feature representation of the face, called a face embedding. The model is then further trained, via fine-tuning, in order that the Euclidean distance between vectors generated for the same identity are made smaller and the vectors generated for different identities is made larger. This is achieved using a triplet loss function. [10] The model is pretrained on the VGG Face Dataset which has 2,622 identities.

Fig. 9. Custom simple VGGFace structure for our face recognition

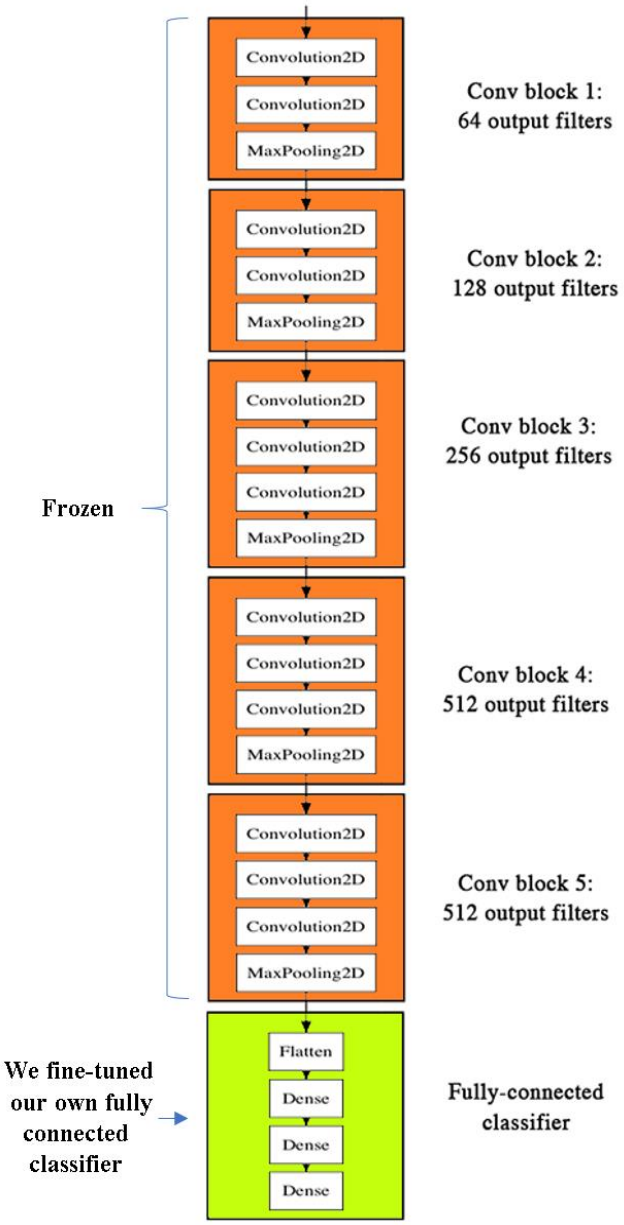

Fig. 10. Fine tuning our custom VGG16 to our custom dataset

Fig.9. shows a custom structure of the VGG-Face model. We demonstrate the fine-tuning process in fig. 10, the process was divided into three steps. First step, build vgg-16 and load the weights, next step, freeze the top layers of the network (orange colour) and the final step is editing the final FC layers (in green) to the number of outputs desired and then do model training. The input is a face image of size 224×224, the embeddings learning in the fine-tuned section uses triplet loss. Parkhi et al. [10] described the architecture as 11 blocks where they separated the conv layers from non-linearities such as ReLU and max pooling. Our figure shows the blocks combined. Fig. 10 shows we have 5 conv blocks with MaxPooling followed by fully-connected block and a SoftMax activation where we fine-tuned our model.

2.3.4 K-means

K-means is an unsupervised learning algorithm. The k-means algorithm adjusts the classification of the observations into clusters and updates the cluster centroids until the position of the centroids is stable over successive iterations. In this implementation of the algorithm, the stability of the centroids is determined by comparing the absolute value of the change in the average Euclidean distance between the observations and their corresponding centroids against a threshold. This yields a code book mapping centroid to codes and vice versa.

The number of centroids to generate. A code is assigned to each centroid, which is also the row index of the centroid in the codebook matrix generated. The initial k centroids are chosen by randomly selecting observations from the observation matrix. Alternatively, passing a k by N array specifies the initial k centroids.

Fig. 11 demonstrates a k-means implementation. A step by step guide is showed in the figure. The assignment step is referred to as the expectation step, while the update step is a maximization step, making this algorithm a variant of the generalized expectation-maximization algorithm.

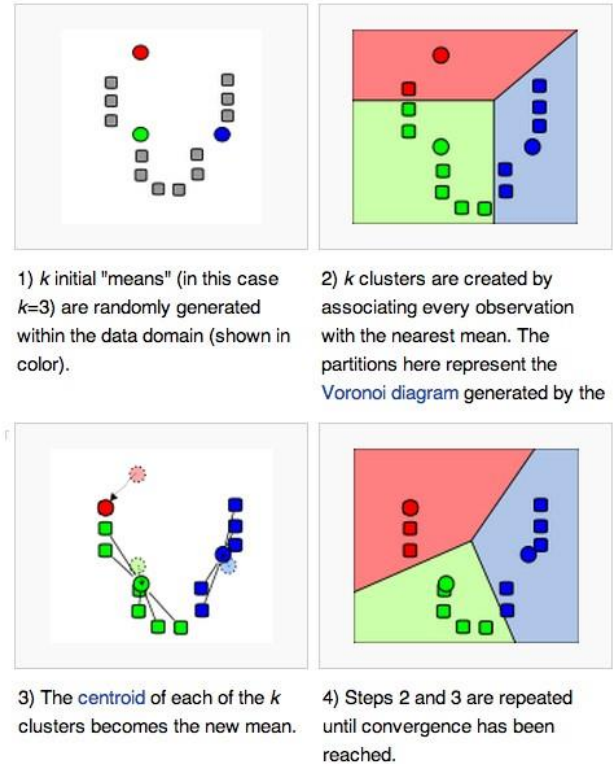

Fig. 11. Example of K-means algorithm with k=3, credit Wikipedia

SciPy k-means returns a Nd array codebook. A k by N array of k centroids. The i'th centroid codebook[i] is represented with the code i. The centroids and codes generated represent the lowest distortion seen, not necessarily the globally minimal distortion from k-means models.

2.4 Experiment Set-Up

2.4.1 Dataset Creation

In section 1.2 we structured our dataset to 56 folders. Next is creating the dataset. Since this a face recognition task then our database will contain faces. We chose state-of-the-art MTCNN face detection and iterated through all folders that have the individual photo shoots using Batch MTCNN and extracted faces from all pictures and video frames. The frame extraction was set at every 8 frames. Next, we extracted all faces from the class photos and videos and placed them to their perspective class label. The final step was cleaning up the data and removing all faces images with no faces or repetitive images. The goal was to create 20 face images for each student. At this point our face database creation was complete.

We applied image augmentation using scikit-image which is a collection of algorithms for image processing. We applied randomly to our images the following: horizontal flip, vertical flip, gaussian blur, noise, different rotations and finally gamma which performs gamma correction on the input image. For gamma greater than 1, the histogram will shift towards left and the output image will be darker than the input image. For gamma less than 1, the histogram will shift towards right and the output image will be brighter than the input image. We generated random 80 extra images each in addition to the 20-original bringing the total to 5600 images. Our dataset was divided to 90% training set and 10% validation set with balanced classes using Stratified Shuffle Split.

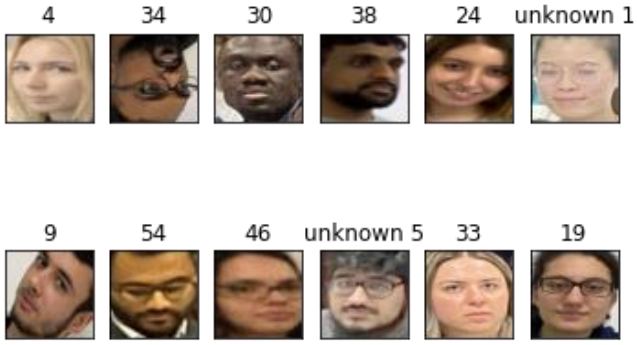

Fig. 12. Dataset of faces after applying augmentation

In addition to the dataset of 5600 images, we applied the Image data generator method from Keras on our CNN model, which is a real-time data augmentation for both sets. The benefits of this is model generalization, saving disk space, faster and increasing the dataset by a large magnitude. We applied rescaling and shearing in this stage for both sets.

2.4.2 Hyperparameters Tuning for ML models

The classifiers used in this project is support vector machines (SVM) and logistic regression (LR) using feature extractors SIFT and SURF with bag of words approach. The feature vector size array for SIFT and SURF are, 64 and 128, respectively.

The steps we followed in this section is:

1. Extract image features and descriptors on the training set with feature Type set to SURF or SIFT and stack the descriptors into NumPy array vertically.
2. Create a codebook or vocabulary of k=750 using k-means clustering.
3. Apply vector quantitation for each image.
4. Standarize the features by scaling which happens independently on each feature by computing the relevant statistics on the samples in the training set. Mean and standard deviation are then stored to be used on later data using transform.
5. Training the classifiers

A gird search of 5-fold cross validation was applied to both algorithms for hyper parameter tuning and model selection. In SVM we choose our kernel to use radial basis function (RBF) according to Chittora et al. [8] experiment results which shows that the RBF SVMs are a better learning algorithm than other kernel function approaches of SVM for face recognition. Class weight was set to balanced and the grid search fine-tuned the gamma and C hyper parameters. Intuitively, the gamma parameter defines how far the influence of a single training example reaches, with low values meaning 'far' and high values meaning 'close'. The C parameter trades off correct classification of training examples against maximization of the decision function's margin. For larger values of C, a smaller margin will be accepted if the decision function is better at classifying all training points correctly. A lower C will encourage a larger margin, therefore a simpler decision function, at the cost of training accuracy. In other words, C behaves as a regularization parameter in SVM.

Logistic regression (LR) multiclass was set to auto and the grid search was fine-tuning the solver which is an algorithm to use in the optimization problem, C, which is the Inverse of regularization strength, like in support vector machines, smaller values specify stronger regularization. The penalty used to specify the norm used in the penalization. The 'newton-cg', 'sag' and 'lbfgs' solvers support only l2 penalties. 'elastic net' is only supported by the 'saga' solver. If 'none' no regularization is applied.

2.4.3 CNN parameters & transfer learning

The convolutional neural network (CNN) model used in our project is VGG16 is pretrained on the VGG face dataset. We froze the top layers and fine tuned the fully connected layers, each followed with a ReLU activation, some of the advantages of using a ReLU is being fast, sparsity, and a reduced likelihood of vanishing gradient. The input images were set to 224x224. A loss function is one of the two parameters required to compile a model. The loss function used is sparse categorical cross entropy. We use it together with the Adam optimizer, which is one of the standard ones used today in very generic scenarios, and use accuracy as an additional metric, since it is more intuitive to humans. SGD optimizer was also used in our experiments but accuracy was low.

The output layer size was changed to the number of classes in our project which is 56 with a SoftMax activation function. The model was trained for 10 epochs with a batch size of 16.

2.5 Results and Discussion

This section will present the results from the hyperparameter grid search in SVM and LR, validation performance of the all models build, validation and training performance of VGG16 and misclassified faces. We will also present a comparison visualization for our models and classification report and confusion matrix for the performing model.

The face detector used in our function is MTCNN created by Iván de Paz Centeno for TensorFlow inspired by MTCNN implementation of David Sandberg for Face net. The detector worked extremely well, however we had few issues in detecting non-faces. Figure 13 demonstrates the misclassification of number 5 and 17 which is mistaken for a hair tie and hand, respectively. To solve this issue, we forked MTCNN and changed the confidence thresholds for P-Net, R-Net, O-Net From 0.5, 0.6, 0.6 to 0.7, 0.8 and 0.8, respectively. The issue was rectified to fit our requirement and the face detector worked extremely well after that.

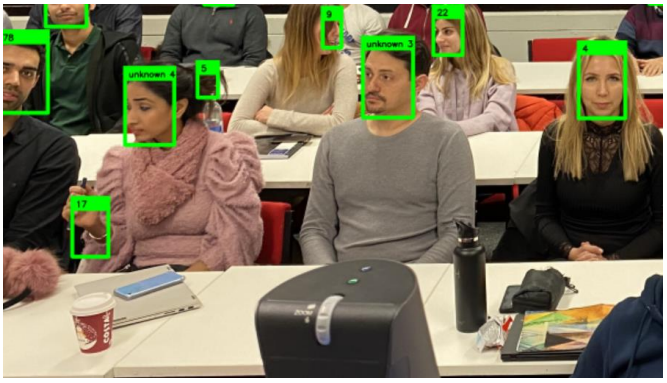

Fig. 13. Face image Misclassification example for a hand

SURF & SIFT were excellent feature extractors. In Table 1 we notice that SURF performed better than SIFT for both algorithms which was expected. The best machine learning classifier performing on the validation set was SURF feature extractor combined with a radial basis function SVM as shown in table 1 with a C of 100 and gamma of 0.001. The best solver for logistic regression that worked on our dataset was L-BFGS which Stands for Limited-memory Broyden–Fletcher–Goldfarb–Shanno. It approximates the second derivative matrix updates with gradient evaluations.

Classifier Hyper-parameters					
Classifier	Feature Extractor	C	gamma	Solver	Validation Accuracy
SVM with RBF kernel	SURF	100	0.001	-	95.1%
	SIFT	100	0.001	-	88.0%
LR	SURF	0.04	-	lbfgs	91.6%
	SIFT	5000	-	lbfgs	86.2%

Table 1. Classifier hyper-parameters results from the cross-validation grid search training on the dataset. The table also presents the accuracy on the validation set to evaluate the models.

The vgg16 performed extremely well on the validation set with

96.8% as shown in table 2. Our experiments show that Adam optimizer did better than SGD optimizer. The fully connected number of nodes played an important role in the accuracy, with 4096 nodes, the total parameters was 145m and validation accuracy 91.4%. 2622 nodes and 82m total parameters the accuracy was 93.1% and finally 512 nodes and total parameters 28m a validation accuracy of 96.8%. Table 2 displays the best accuracy results for our VGG16 model.

CNN classifier				
Classifier	Optimizer	FC size	Training Accuracy	Validation Accuracy
VGG16	Adam	512	97.5%	96.8%

Table 2. VGG16 model results

A summary of our classifier's performance is displayed in fig 14. It's clear that vgg16 ranked first, followed by SURF & SVM then SURF & LR. Our conclusion here is SURF is a better feature extractor than SIFT and faster.

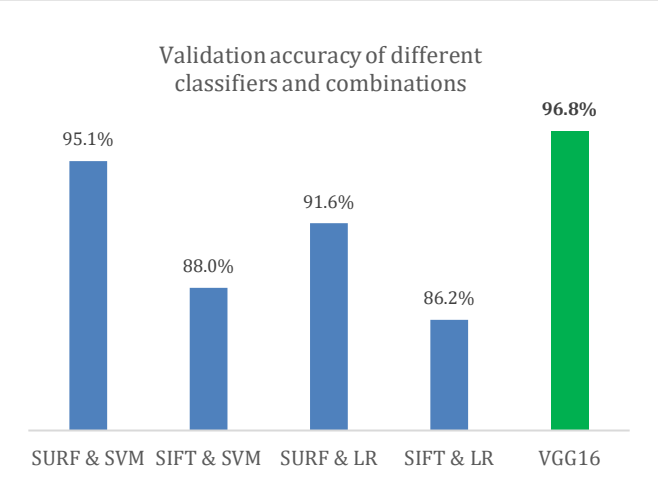

Fig. 14. Bar chart for different classifiers trained on our face dataset

There are a number of metric types to measure our face recognition performance. The first measure is the recognition rate, which is probably the simplest measure. It relies on a list of gallery images and a list of probe images of the same identities. For each probe image, the similarity to all gallery images is computed, and it is determined, if the gallery image with the highest similarity is from the same identity as the probe image. Finally, the recognition rate is the total number of correctly identified probe images, divided by the total number of probe images. The second measure is the verification rate. It relies on a list of image pairs, where pair with the same and pairs with different identities are compared. But in our case, we want to train on multiple images of several people and then test on different images of the same people. This would be a multi-class classification problem and a confusion matrix would be helpful in evaluating this sort of test. A confusion matrix is included in the Appendix for the vgg16 model on our validation dataset. The total number of images in the validation set is 560, 10 for each person. Fig 15 shows examples of misclassified face images from the VGG16 evaluation.

Fig. 15. Misclassified face images examples from the validation set

The confusion matrix in the appendix displays successfully all labels and shows the misclassification clearly with ID # 21 having the lowest correct classification at 7 out of 10. The left image from fig. 15 demonstrates one of the 3 misclassifications for class ID # 21. We notice that the image is completely blurred. This misclassification can be rectified by removing the unclear images from the dataset, and/or generating more through image augmentation. Another solution for CNN networks will be removing all augmented images locally and maximizing real time implementation with different setting experimentation.

A Jupyter notebook is submitted showcasing all possible combinations with reports and images with labelled bounding boxes. Also, an OCR notebook showcase some examples which is not discussed here. The OCR used is frozen EAST from OpenCV as a text detector and a tesseract engine as a text recognition system. The system was applied successfully on the individual photoshoots recognising the numbers from an A4 paper sheet. However, there were a few wrong detections on some other images used outside of this project in order to display limitations.

The results in Fig. 16. demonstrate two things, first the accuracy increases on both sets till epoch 4 and then stabilizes. Secondly, the highest accuracy on validation set is at epoch 5 and 8. Since, the accuracy gap between the two sets is not increasing considerably then we can assume that our model generalized very well. This is important as our accuracy reached 100% on training set on some epochs and we want to conclude no overfitting on the training set.

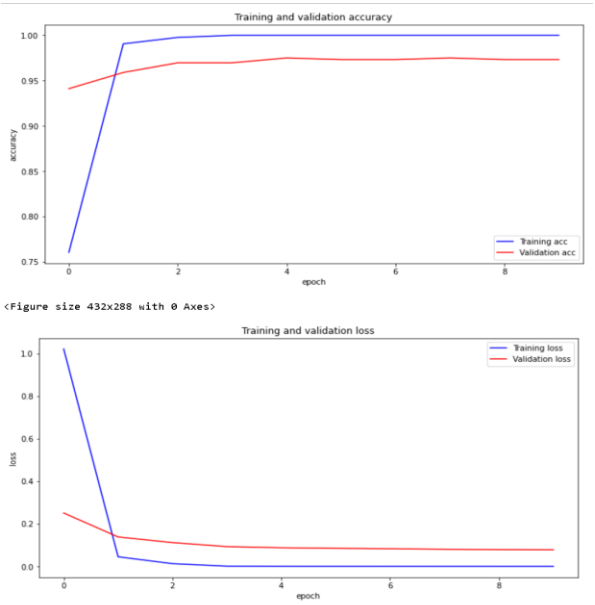

Fig. 16. Training and validation accuracy and loss plots for the VGG16 model.

Creative mode is one of the function arguments used on the function Recognise Face, we built 4 options (0,1,2,3). If set to zero then no creative mode will be displayed. If set to 1, an anonymity option is applied to the face within the bounding box only and the person will be anonymous. Option 2 will apply cool glasses and option 3 funny eyes as in fig 16. The last two options use MTCNN landmarks left eye and right eye and apply transparent images in PNG format masks.

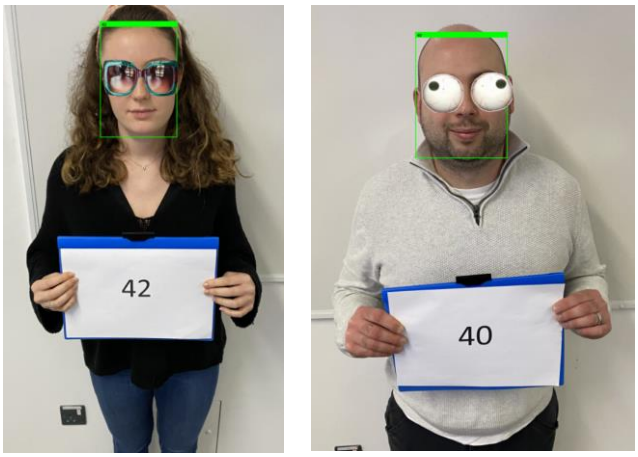

Fig. 17. Examples of creative modes. Left: Cool glasses creative mode, Right: Funny eyes creative mode. Labelled bounding box in green and face landmarks are displayed on the image as red dots.

3. Conclusion and Future Work

In this report, we addressed the face recognition pipeline, starting with an overview of the Recognise Face function in a workflow chart and the theory behind the tools used in this project. We started with a dataset creation then face extraction, and then applying feature extraction to the input images so our classifier can predict the identity of the person in an image with their face central coordinates. SURF performed better than SIFT in feature extraction. 5 classifier combinations were tested, and the highest accuracy on a validation set was a VGG16 backbone pretrained on VGG Face dataset and fine tuned on our custom dataset achieving a 96.8% on the validation set.

Future work would be classifying unknown people not in the 56 classes as unknowns. Also, building an OCR from scratch using Resnet and K-NN algorithm. Experimenting with One-Shot Learning with Siamese Networks. Also, we would like to explore and experiment with deeper backbones like SENet-50, ResNet-50 or ResNet-101 on a GPU as that was not possible on our local machine. Another interesting future work, will be using Tensor Board or WANDB for experiment tracking, hyperparameter optimization and model management for our CNN models.

Our work can be extended to real life scenarios, an example would be a face recognition-based attendance system for schools and universities or finding missing children and victims of human trafficking. The missing individuals can be added to the face database and law enforcement can become alerted as soon as they are recognized by face recognition in public spaces, airports, etc. In fact, thousands of missing children were discovered in the world using this technology.

4. References

[1] "What is Facial Recognition? - Definition from Techopedia". Techopedia.com. Retrieved 2018-08-27

[2] Andrew Heinzman. "How Does Facial Recognition Work?". How-To Geek. Retrieved 2020-02-28.

[3]"How does facial recognition work?". us.norton.com. Retrieved 2020-02-28.

[4] Lowe, D.G.: Distinctive Image Features from Scale-Invariant Keypoints. International Journal of Computer Vision 60, 91–110 (2004)

[5] Krizaj, J., Struc, V., and Pavesic, N., Adaptation of SIFT features for robust face recognition. In: International Conference Image Analysis and Recognition, pp. 394–404, 2010.

[6] Bay H., Tuytelaars T., Van Gool L. (2006) SURF: Speeded Up Robust Features. In: Leonardis A., Bischof H., Pinz A. (eds) Computer Vision – ECCV 2006. ECCV 2006. Lecture Notes in Computer Science, vol 3951. Springer, Berlin, Heidelberg

[7] K. Zhang, Z. Zhang, Z. Li and Y. Qiao, "Joint Face Detection and Alignment Using Multitask Cascaded Convolutional Networks," in IEEE Signal Processing Letters, vol. 23, no. 10, pp. 1499-1503, Oct. 2016.

[8] Chittora, Ashish & Mishra, Om. (2012). Face Recognition Using RBF Kernel Based Support Vector Machine. International Journal of Future Computer and Communication. 280-283.

[9] Simonyan, Karen & Zisserman, Andrew. (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition. arXiv 1409.1556.

[10] Parkhi, Omkar & Vedaldi, Andrea & Zisserman, Andrew. (2015). Deep Face Recognition. 1. 41.1-41.12. 10.5244/C.29.41.

Appendix