Low-Resolution Face Detection & Identification

Face detection & identification done on faces on image with resolution of 30x30 pixels.

Contents

[**1.** **Introduction** 0](#_Toc77108120)

[1.1. Problem Background 0](#_Toc77108121)

[1.2. Objectives/Aims 1](#_Toc77108122)

[1.3. Motivation 1](#_Toc77108123)

[1.4. Timeline/Milestone 2](#_Toc77108124)

[**2.** **Research Background** 2](#_Toc77108125)

[2.1. Background of the applications 2](#_Toc77108126)

[History of Face Recognition 2](#_Toc77108127)

[Super Resolution 4](#_Toc77108128)

[2.2. Analysis of selected tool with any other relevant tools 5](#_Toc77108129)

[2.3. Justify why the selected tool is suitable 8](#_Toc77108130)

[**3.** **Methodology** 9](#_Toc77108131)

[3.1. Description of dataset 9](#_Toc77108132)

[Dataset 1: LFW3D (for testing) 9](#_Toc77108133)

[Dataset 2: LFW (for training) 9](#_Toc77108134)

[Dataset 3: Personal images (for both testing & training - real life) 10](#_Toc77108135)

[3.2. Applications of the algorithm(s) 10](#_Toc77108136)

[Image Acquisition 10](#_Toc77108137)

[Image Enhancement 10](#_Toc77108138)

[Face Detection 11](#_Toc77108139)

[CNN (Convolutional Neural Network) 13](#_Toc77108140)

[Face Analysis 13](#_Toc77108141)

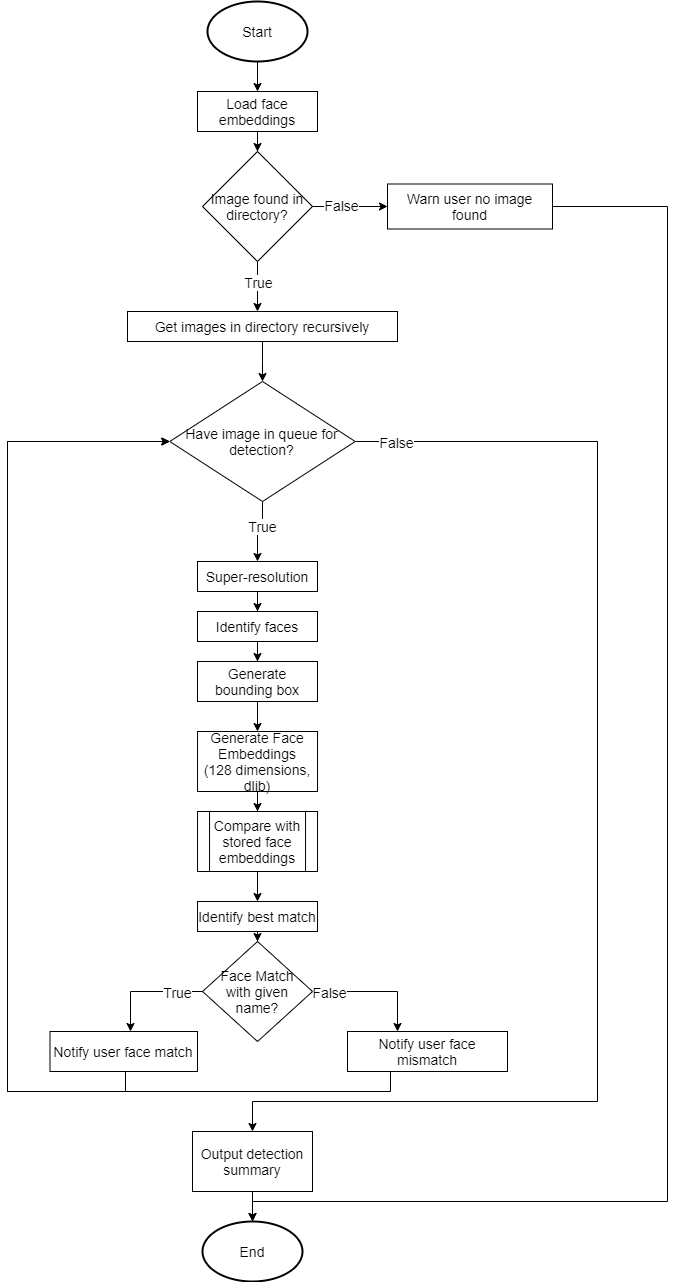
[Data Conversion 13](#_Toc77108142)

[Face Identification 14](#_Toc77108143)

[3.3. System flowchart/activity diagram 17](#_Toc77108144)

[Step 1: Face Encodings (Building dataset) 17](#_Toc77108145)

[Step 2: Face Identification 18](#_Toc77108146)

[ 18](#_Toc77108147)

[3.4. Proposed test plan/hypothesis 19](#_Toc77108148)

[Detect & identify face (benchmark of 30 people in LFW/3DLFW + myself + 3 unknown people) - classifier model 19](#_Toc77108149)

[Identify face (benchmark of 30 people in LFW/3DLFW + myself + 3 unknown people) 19](#_Toc77108150)

[Detect & identify face (in 3DLFW) 19](#_Toc77108151)

[Detect & identify face (Real Life) 19](#_Toc77108152)

[Identify face (Unknown) 19](#_Toc77108153)

[Identify unknown face (benchmark of 34 people) 19](#_Toc77108154)

[Validation 20](#_Toc77108155)

[Improvement 20](#_Toc77108156)

[**4.** **Result** 20](#_Toc77108157)

[4.1. Results 20](#_Toc77108158)

[Disclaimer 20](#_Toc77108159)

[4.2. Discussion/Interpretation 28](#_Toc77108160)

[Detect face (benchmark of 34 people) 28](#_Toc77108161)

[**5.** **Discussion and Conclusion** 31](#_Toc77108162)

[5.1. Achievements 31](#_Toc77108163)

[5.2. Limitations and Future Works 32](#_Toc77108164)

[Time Limitation 32](#_Toc77108165)

[Hardware Limitation 32](#_Toc77108166)

[Future Improvements 32](#_Toc77108167)

[**Reference & Source** 33](#_Toc77108168)

# **Introduction**

## Problem Background

In recent years, Malaysia has become increasingly dangerous with crimes happening all around the year. Crimes such as snatch thief, hit-and-run, as well as robbery are unfortunately quite common in Malaysia. In 2020, The U.S. Department of State has assessed Kuala Lumpur as being a HIGH-threat location for crime directed at or affecting official U.S. government interests. This includes around-the-clock street crime that occurs primarily in densely populated urban centers and affects locals and foreigners alike. The most common crimes include petty theft (particularly purse snatching and pickpocketing), smash-and-grab thefts from vehicles, and residential burglaries. Violent and more serious crimes are considerably less common. Other types of common non-violent criminal activity include credit card fraud, ATM-skimming, and cybercrime (OSAC 2020).

As of today, traditional face recognition has received worldwide attention and achieved very high accuracy. In particular, face recognition models have scored extremely well, reaching 99.63% accuracy on the LFW benchmark images under constrained conditions with good quality (Zhiyi Cheng, Xiatian Zhu & Shaogang Gong 2018). However, despite the high performance, it is commonly noted as well that this is under “perfect conditions”. Any deviations in terms of image quality will deteriorate the detection significantly. This can be seen when the images are acquired by surveillance cameras, and subjected to pose, resolution, brightness, and other variations (Zhiyi Cheng, Xiatian Zhu & Shaogang Gong 2018).

Omid had done research on low resolution face identification as well, and although achieved satisfactory results, by cropping and resizing to 224 × 224 or 112 × 112 pixel resolutions depending on the input size of the deep learning models (Omid Abdollahi Aghdam et al. 2019). However, there is a lack of code samples available. Another research done by Pei Li and researchers indicated that Although there is no broadly accepted single criterion for labeling a face in an image as low resolution, but many works have recognized that face images with a tight bounding box smaller than 32 × 32 pixels begin to present significant accuracy challenges to face recognition systems in both human and computer vision (Li et al. 2018). Finally, HenningsYeomans and researchers suggested using super-resolution algorithms to enhance the image, but as resolution decreases, super-resolution becomes more vulnerable to environmental variations, and it introduces distortions that affect recognition (HenningsYeomans et al. 2008).

Some research has already been done to improve face recognition & detection in low resolution environments, however, there is a lack of research in augmenting existing face recognition & identification systems with image optimizations, to evaluate the performance of low-resolution face detection. This is especially important because most of the time, we will only have a high-resolution version of a person’s picture. However, we are then forced to use the high-resolution version to try and recognize people in low-resolution environments.

Hence, there is a need to improve face recognition, especially in low-resolution situations. Furthermore, there is also room for improvement by using super-resolution algorithms.

## Objectives/Aims

The main objective that we wish to achieve are the following:

1. To effectively recognize a person’s face under low resolution conditions. This helps with criminal activities because most of the criminals are situated far from the camera, which can be a problem for face recognition algorithms to work.
2. To achieve at least 60% accuracy at 30x30 pixels. The system must at least be able to recognize faces correctly the majority of the time, even in low resolution. Due to time constraints, further enhancements can be done in the future. However, the system must be at least usable and can be trusted by others.
3. To have high performance, and able to recognize a person’s face in a short time. Usually, low resolution cameras are also paired with weak computers. Therefore, it is necessary to build the system with performance in mind.

## Motivation

The main motivations are:

**Decreased cost of hardware**

1. By being able to resolve people’s faces in low resolution, in small environments, it is possible to deploy cheaper hardware, and still be able to detect people’s faces efficiently. This helps to reduce costs, especially in large buildings with multiple small rooms, where hundreds or thousands of cameras can be deployed across multiple angles to ensure no detection slips through.

**Improved public safety**

1. Public security can be improved as people can be matched even when located further away from the camera, and thus might not be in good resolution. This helps to reduce the amount of crime events as more robbers will be disincentivized from robbing due to the increasing risk of getting caught.

**Assist in disaster rescue events**

1. Sometimes, disaster rescue events rely on artificial intelligence to help identify people trapped in buildings or debris. By being able to recognize people’s faces among dust and debris, more people can be identified quicker. This results in decreased fatalities during disasters.

## Timeline/Milestone

This is the planned timeline. However, the time might deviate depending on situation and workload.

|  |  |
| --- | --- |
| Date | Task |
| 18/1/2021 - 31/1/2021 | Select the assignment topic |
| 1/2/2021 - 14/2/2021 | Planning & set focus |
| 15/2/2021 - 28/2/2021 | Research on background |
| 1/3/2021 - 14/3/2021 | Research |
| 15/3/2021 - 28/3/2021 | Implementation & check results |
| 29/3/2021 - 4/4/2021 | Optimization |
| 25/1/2021 - 31/1/2021 | Document Outcome |

# **Research Background**

## Background of the applications

### History of Face Recognition

#### History

Face recognition is a method of identifying or verifying the identity of an individual using their face (EFF 2019).

Woody Bledsoe, Helen Chan Wolf and Charles Bisson were known as the earliest inventors of face recognition. In 1964 and 1965, Bledsoe, along with Wolf and Bisson began work using computers to recognise the human face (NEC 2020). Due to the funding of the project originating from an unnamed intelligence agency, much of their work was never published (NEC 2020). However it was later revealed that their initial work involved the manual marking of various “landmarks'' on the face such as eye centres, mouth etc (NEC 2020). These were then mathematically rotated by a computer to compensate for pose variation. The distances between landmarks were also automatically computed and compared between images to determine identity (NEC 2020). These earliest steps into Facial Recognition were unfortunately limited by the technology of their era, but it does pave a way in proving that Facial Recognition was a viable biometric (NEC 2020).

#### Methods

1. **Holistic Matching Methods**: In a holistic approach, we take the entire face as input data (Parmar & Mehta 2013). Popular examples of holistic methods are Eigenfaces, Principal Component Analysis, Linear Discriminant Analysis, and Independent component analysis etc (Parmar & Mehta 2013). However, due to the possibility of variations in terms of illumination, expressions and other factors, Wójcik has noted that these methods may be unreliable and fail to represent faces well. They also cited that the main reason is due to face patterns lying on a complex nonlinear and non‐convex manifold in the high‐dimensional space (Wójcik, Gromaszek & Junisbekov 2016).

2. **Feature-based (structural) Methods**: In these methods local features such as eyes, nose and mouth are first of all extracted and their locations and local statistics are fed into a structural classifier (Parmar & Mehta 2013). There are 3 main extraction methods, which are generic methods based on edges, lines, and curves; Feature-template-based methods; Structural matching methods that take into consideration geometrical constraints on the features (Parmar & Mehta 2013). Feature-based methods have some advantages over holistic matching methods (Wójcik, Gromaszek & Junisbekov 2016). Feature-based methods are more stable to local changes such as expression, occlusion, pose variations, and misalignments, compared to holistic matching methods (Wójcik, Gromaszek & Junisbekov 2016). Therefore, this method is more suitable for face resolution where the conditions are not ideal.

3. **Hybrid Methods**: Hybrid methods mix holistic & feature-based methods for detection. Generally 3D Images are used in hybrid methods, and thus, it is not very suitable for 2D images. The image of a person's face is captured in 3D, allowing the system to note the curves of the eye sockets, and other contours of the face (Parmar & Mehta 2013) . Even a face in profile would serve because the system uses depth, and an axis of measurement, which gives it enough information to construct a full face (Parmar & Mehta 2013).

According to Parmar, the 3D system proceeds as per the follows:

1. Detection - Capture the face
2. Position - Determining the angle, size, and the location of the head. This step is unique to 3D systems.
3. Measurement - Assign measurements to each curve of the face. In 3D systems, we make a template with specific focus on the outside of the eye, the inside of the eye and the angle of the nose.
4. Representation - Converting the template into a numerical representation of the face
5. Matching - Compare the received face with our existing dataset.

What’s interesting to note is that the methods used for face recognition are very similar even across 3 methods.

#### Current applications

**Face Identification**: Face recognition systems identify people by their face images. Face recognition systems establish the presence of an authorized person rather than just checking whether a valid identification (ID) or key is being used or whether the user knows the secret personal identification numbers (Pins) or passwords. (Parmar & Mehta 2013)

**Access Control**: In many of the access control applications, such as office access or computer logon, the size of the group of people that need to be recognized is relatively small. The face pictures are also caught under natural conditions, such as frontal faces and indoor illumination. The face recognition system of this application can achieve high accuracy without much cooperation from the user. (Parmar & Mehta 2013)

**Security**: Today more than ever, security is a primary concern at airports and for airline staff offices and passengers. Airport protection systems that use face recognition technology have been implemented at many airports around the world. (Parmar & Mehta 2013)

**Image database investigations**: Searching image databases of licensed drivers, benefit recipients, missing children, immigrants and police bookings. (Parmar & Mehta 2013)

**General identity verification**: Electoral registration, banking, electronic commerce, identifying newborns, national IDs, passports, employee IDs. Surveillance: Like security applications in public places, surveillance by face recognition systems has a low user satisfaction level, if not lower. Free lighting conditions, face orientations and other divisors all make the deployment of face recognition systems for large scale surveillance a challenging task. The following are some example of facebased surveillance (Parmar & Mehta 2013)

## Super Resolution

#### History

In most digital imaging applications, high resolution images or videos are usually desired for later image processing and analysis (Yang & Huang 2017). The desire for high resolution stems from two principal application areas: improvement of pictorial information for human interpretation; and helping representation for automatic machine perception (Yang & Huang 2017). The latter is what we are aiming for. Image resolution describes the details contained in an image, the higher the resolution, the more image details (Yang & Huang 2017).

SR reconstruction has been one of the most active research areas since the

seminal work by Tsai and Huang in 1984 (Yang & Huang 2017). Many techniques have been

proposed over the last two decades representing approaches from frequency domain to spatial domain, and from signal processing perspective to machine learning perspective (Yang & Huang 2017).

#### Methods

There are many different methods in image super-resolution, but they can be broadly classified into two categories.

1. **Non-blind Deconvolution**: By pairing a higher and lower-resolution image, and attempting to enhance the lower resolution image using the known higher-resolution image as comparison (Ren et al. 2018). One of the examples is Deep Non-Blind Deconvolution via Generalized Low-Rank Approximation (Ren et al. 2018). The researchers noted that they first compute a generalized low-rank approximation to a large number of blur kernels, and then use separable filters to initialize the convolutional parameters in the network (Ren et al. 2018).
2. **Blind Deconvolution methods**: By attempting to enhance a lower-resolution image without a higher-resolution reference. Some of the examples are: EDSR (Enhanced Deep Residual Networks for Single Image Super-Resolution), and SRResNet (Lim et al. 2017, pp. 1132–1140). Because in our face identification, we will need to identify unknown people and make them known, our only option is to use blind deconvolution methods.

#### Current Applications

There are many applications of super-resolution, it is used successfully in:

1. Improving medical imaging systems (Malczewski & Stasiński 2009). Super-resolution is required to allow doctors to see any deviations inside the patients, and to reveal early signs of diseases.
2. Satellite imaging (Malczewski & Stasiński 2009). Super-resolution is required to let the users of satellite imaging observe places on earth, and to predict weather conditions.
3. Astronomical imaging (Malczewski & Stasiński 2009). Super-resolution is used to resolve details in astronomical bodies that are light years away.

## Analysis of selected tool with any other relevant tools

*Fill the table below and change the tools’ names. You may add more columns.*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Tools comparison** | **Remark** | **OpenCV** | **SimpleCV** | **BoofCV** |
| Type of license and open source license | State all types of license | Berkeley Software  Distribution (BSD)  License | BSD-3-Clause License | Apache 2.0 |
| Year founded | When is this tool being introduced? | 2000 | 2011 | 2011 |
| Founding company | Owner | Intel Corporation, Willow Garage, Itseez | Sight Machine, Inc | LessThanOptimal |
| License Pricing | Compare the prices if the license is used for development and business/commercialization | Free (Open source) | Free (Open Source) | Free (Open Source) |
| Supported features | What features that it offers? | 1. Read/write images 2. Capture/save videos 3. Process images 4. Feature detection 5. Object detection 6. Video analysis 7. Super-resolution   (TutorialsPoint 2021) | 1. Object detection 2. Segmentation 3. Image Arithmetic | 1. image processing 2. Features detection 3. Geometric vision 4. Calibration 5. Recognitio 6. Visualization 7. IO   (Abeles 2021) |
| Common applications | In what areas this tool is usually used? | 1. Robotics (Localization, Navigation, Obstacles Avoidance) 2. Medicine (Classification & detection, segmentation, 3D organ image reconstruction, vision-guided robotics surgery) 3. Security (Biometrics, Surveillance) 4. Facial Recognition   (TutorialsPoint 2021) | 1. Image effects 2. Chroma Key (Green Screen) 3. Face Detection | 1. Camera calibration 2. QR Code scanner 3. 3D Vision 4. Face Recognition 5. Image Enhancement |
| Customer support | How the customer support is given, e.g. proprietary, online community, etc. | Online community | Online community | Online community |
| Limitations | The drawbacks of the software | Comparatively harder to learn due to a lack of documentation and error handling codes compared to MATLAB. (AnalyticsInsight 2020). | No longer being actively updated (Last release as of 2021-3-16 is on 11-08-2012 on PyPI, which is about 8 years ago (Sight Machine Inc 2011)  Lower amount of resources available compared to OpenCV | Uses Java instead of C++, which means comparatively worse performance when compared to OpenCV.  Not as popular compared to OpenCV (which means less tutorials, more potential booby traps) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Tools comparison** | **Remark** | **dlib (face\_recognition)** | **FaceNet** |
| Type of license and open source license | State all types of license | Berkeley Software  Distribution (BSD)  License | MIT License |
| Year founded | When is this tool being introduced? | 2017 | 2015 |
| Founding company | Owner | Adam Geitgey | Google |
| License Pricing | Compare the prices if the license is used for development and business/commercialization | Free (Open source) | Free (Open Source) |
| Supported features | What features does it offer? | 1. Read/write images 2. Face Encoding 3. Face Detection 4. Face Identificatiob   (Geitgey 2017) | 1. Face Detection 2. Facial Landmark Detection 3. Face Recognition 4. Face Verification 5. Face classification   (Pluralsight 2020) |
| Common applications | In what areas this tool is usually used? | 1. Authentication 2. People identification 3. Alert systems | 1. Human computer interaction, 2. Virtual reality, 3. Computer entertainment |
| Customer support | How the customer support is given, e.g. proprietary, online community, etc. | Online community | Online community |
| Limitations | The drawbacks of the software | Running on dlib which has unofficial Windows support. | More complex to set up, while offering not much improvement in terms of actual face recognition accuracy (already many other ones close to ideal) |

## Justify why the selected tool is suitable

*Explain which tool is used for the development, and justify the suitability of the tool used in your project.*

>

After analysis, OpenCV was chosen as the selected tool for the system. There are a few reasons:

1. Better variety of tutorials online
   1. OpenCV was started at Intel in 1999 by Gary Bradsky, and the first release came out in 2000. Vadim Pisarevsky joined Gary Bradsky to manage Intel's Russian software OpenCV team. As of to date, multiple face identification tutorials can be found on the internet for face recognition & identification in OpenCV. (Rosebrock 2018) & (Rosebrock 2018a) & (Rovai 2019) Due to the short timeframe available for this assignment, it is wise to choose something that is widely supported and has been thoroughly tested, rather than to waste time trying to start from scratch.
2. Mature library
   1. In 2005, OpenCV was used on Stanley, the vehicle that won the 2005 DARPA Grand Challenge (OpenCV 2018). Later, its active development continued under the support of Willow Garage with Gary Bradsky and Vadim Pisarevsky leading the project. (OpenCV 2013). OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library (OpenCV 2018). The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms (OpenCV 2018). These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, stitch images together to produce a high resolution image of an entire scene, find similar images from an image database, remove red eyes from images taken using flash, follow eye movements, recognize scenery and establish markers to overlay it with augmented reality, etc (OpenCV 2018). Although we will not use all of the available algorithms, due to the massive library available, it is easy to find what we are looking for.
3. Built-in APIs to handle super-resolution
   1. As part of our exploratory research, it is vital that we have resources available for super-resolution. OpenCV provides a couple of super-resolution models, and also easy ways to access them through the dnn\_superres library. This allows us to access the powerful EDSR single-blind super-resolution model.

Besides that, dlib’s face\_recognition was chosen as the selected tool for the system as well. There are a few reasons as well:

1. Ease of use
   1. face\_recognition has been ported to Python as a library, and is easy to use and access. Their PyPI page contains very comprehensive documentation, which given the limited time, is ideal to start with (Geitgey 2017).
2. Good accuracy
   1. After some short use, we noticed that the face recognition retains high accuracy even at low resolution (achieving good accuracy even at just 30x30 pixels). Therefore, we decided to pick up and use the library.

# **Methodology**

## Description of dataset

*Describe the source of the dataset, and the data structures/data dictionary*

>

Our source of dataset is from 3 sources. Obtaining a dataset for low-resolution face recognition is much more difficult than expected, requiring the creation of multiple scripts to automate very tedious parts. This process took weeks.

Nevertheless, to try and go beyond the limits, my dataset consists of over 30 people, which is above the 10-20 people requirement stated inside the requirements.

### Dataset 1: LFW3D (for testing)

The first source is LFW3D, which is a collection of frontalized LFW (Labelled Faces in the Wild) images by Tal Hassner (Hassner 2015).

However, due to the fact that all the faces are 90x90, which is still considered as “high resolution”, coupled with the fact that there’s no “low-resolution” face dataset with multiple images available (closest is TinyFace, but only single pictures), we will have to create our own dataset for testing purposes.

I will also take a few pictures from LFW3D that are from people not inside our training sets as “unknown people” to ensure that the system doesn't detect unknown people as one of the known people.

### Dataset 2: LFW (for training)

For feeding into the system, we will be using the original pictures from our second source, which is the original LFW (labelled faces in the wild database) to feed into the system (University of Massachusetts Amherst 2007). The main reason for doing so is because in a typical face recognition system, we will usually have a high resolution picture of someone, and will need to detect someone in low resolution. For testing purposes, I will write my own Python code to resize the picture into 30x30 pixels, using the bicubic scaling algorithm to simulate a low resolution environment.

For our “low-resolution” purpose, we will only be taking pictures of people with above 13 picture samples available. We will then use 10 pictures to build our face database, and test it on 3 pictures that are not inside the database to test the accuracy. This is about a 75/25 split between train/testing. I will also write python code to automate the reduction of pictures. This is particularly significant as unbalanced face recognition training sets will require additional effort to solve, and thus, it is of our best interest to abstract this factor away to reduce the complexity of the project (Leng et al. 2016).

### Dataset 3: Personal images (for both testing & training - real life)

Finally, I will also be using 13 pictures of my own as part of the dataset, which is our third dataset source, for ease of testing. The main reason is because I am able to control the pictures myself. These pictures will be manually resized and adjusted to fit our “low-resolution” purpose.

## Applications of the algorithm(s)

Face recognition is split into 4 parts: face detection, face analysis, data conversion, and finally face identification (PandaSecurity 2019). However, we introduce another element, which is image enhancement.

### Image Acquisition

The first step before any detection can be done is to actually acquire a picture from a camera or a file. In this step, we will supply pictures that are already captured to the detection system from the dataset that we use.

### Image Enhancement

The second step is to perform image optimization. In this step, we will be using three different kinds of image optimization. The first one is nearest neighbour scaling, the second is bicubic scaling, and the third is an image super-resolution algorithm known as EDSR x4.

One big issue in face detection is that most of the previous face detection systems usually exclude tiny faces from their detection, and in this case, is applicable to the one we are using as well. Both dlib and OpenCV’s implementation are incapable of detecting faces that are overly small. As a result, we will have to scale up the faces before we can actually perform detection.

This presents an opportunity to somehow improve the quality. Nearest neighbour is the fastest and simplest to implement (Tabora 2019). This technique replaces every pixel with the nearest pixel in the output. When upscaling an image, multiple pixels of the same color will be duplicated throughout the image (Tabora 2019). The bad news however, is that this usually results in a poor scaling. Due to this, we will be using this method as a “baseline” for our optimization purposes.

The second method is to use bicubic scaling. Bicubic scaling is a system that uses cubic splines or other polynomial technique to sharpen & enlarge images. The good news is that it provides a much better quality in scaling compared to its nearest neighbour, and is still quite fast. Therefore, this approach will be used as our first “non-AI” resolution enhancement used as a benchmark as well.

Finally, the third method that we will use is EDSR, or better known as Enhanced Deep Residual Networks. EDSR utilizes deep neural networks to enhance image quality, and so far outperforms some other techniques such as SRResNet and VDSR (Lim et al. 2017, pp. 1132–1140). The cost is, unfortunately, a large performance hit. We expect to see the performance drop sharply, and we will evaluate whether this method is a suitable approach. This will be the crux of image enhancement.

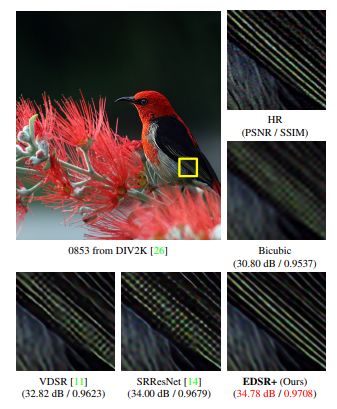


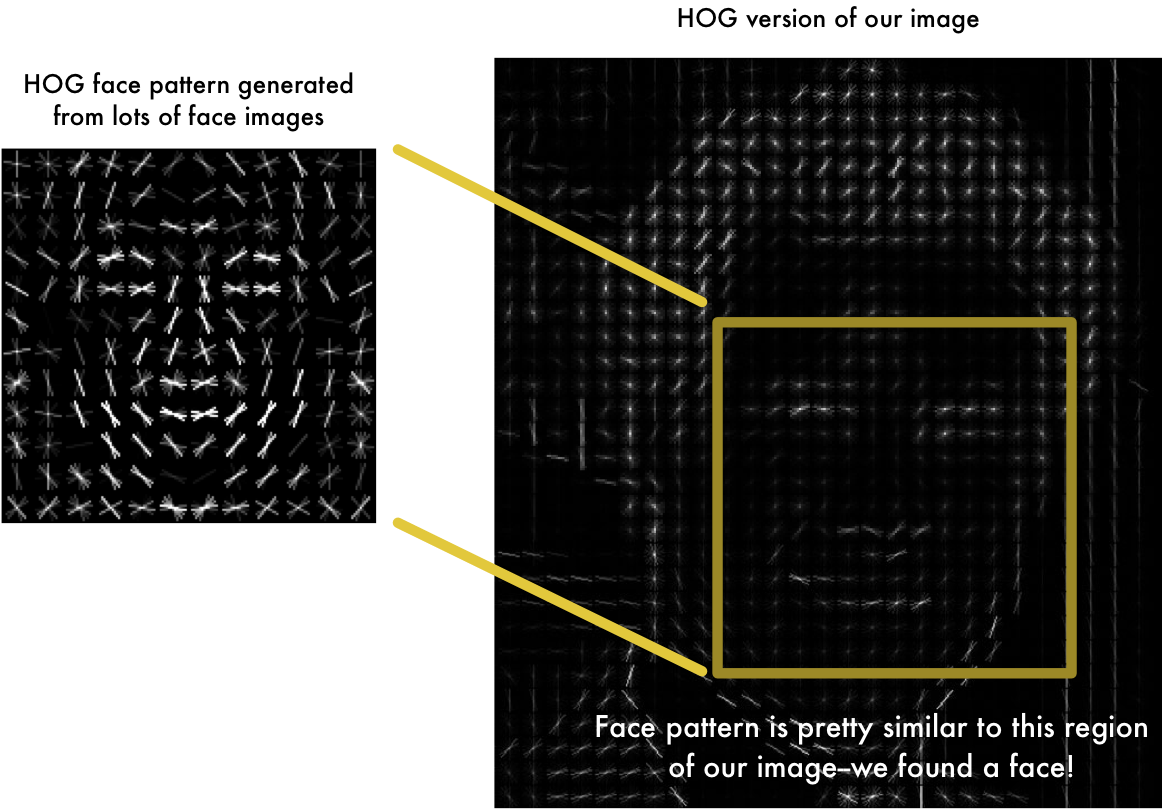
Image shows the performance of EDSR compared to other models of super resolution, proving to be a superior super-resolution model. (Lim et al. 2017, pp. 1132–1140)

### Face Detection

Face Detection refers to the finding of the location of a human face within an image (Face Detection 2020). Face Detection is the first and essential step for face recognition, and it is used to detect faces in the images (Dwivedi 2018). The results of this step are coordinates: In the easiest case it is a bounding rectangle. However, it can also be a set of coordinates for many facial features (landmarks) which are helpful to normalize each face accordingly. (Face Detection 2020).

For Face Detection, we will use the pre-trained model, HoG (Histogram of oriented gradient (HoG) supplied by face\_recognition PyPI library.

#### History of oriented gradients



This photo contains the history of oriented gradients example (Guntupalli 2020)

Histogram of oriented gradients (HOG) is a feature descriptor used to detect objects in computer vision and image processing (Intel 2021). The HOG descriptor technique counts occurrences of gradient orientation in localized portions of an image - detection window, or region of interest (ROI) (Intel 2021).

Implementation of the HOG descriptor algorithm is as follows:

1. Divide the image into small connected regions called cells, and for each cell compute a histogram of gradient directions or edge orientations for the pixels within the cell (Intel 2021).
2. Discretize each cell into angular bins according to the gradient orientation (Intel 2021).
3. Each cell's pixel contributes a weighted gradient to its corresponding angular bin (Intel 2021).
4. Groups of adjacent cells are considered as spatial regions called blocks. The grouping of cells into a block is the basis for grouping and normalization of histograms (Intel 2021).
5. Normalized groups of histograms represent the block histogram. The set of these block histograms represents the descriptor (Intel 2021).

The benefit of HOG is that it is very fast (compared to CNN), however, this comes at an expense of accuracy. For our purposes however, HOG is perfect to evaluate our low-resolution face recognition and super-resolution techniques.

### CNN (Convolutional Neural Network)

Aside from the HOG model, dlib also provides the CNN face detection model (Geitgey 2020).

CNNs have two components:

The first component is the Hidden layers/Feature extraction part. In this part, the network will perform a series of convolutions and pooling operations during which the features are detected (FreeCodeCamp 2018). If you had a picture of a zebra, this is the part where the network would recognise its stripes, two ears, and four legs (FreeCodeCamp 2018).

The second component is the Classification part. Here, the fully connected layers will serve as a classifier on top of these extracted features (FreeCodeCamp 2018). They will assign a probability for the object on the image being what the algorithm predicts it is (FreeCodeCamp 2018).

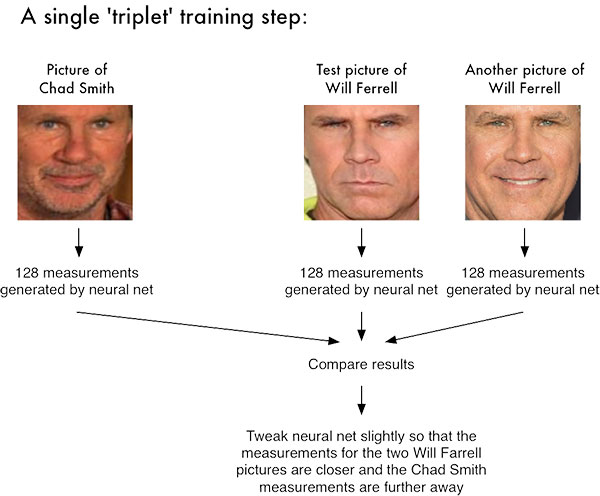
Despite the improved accuracy, we will not be using CNN, as our pre-tests (by changing the “detection\_method” from HOG to CNN inside the code) noted that it is extremely slow compared to HOG (at least 2-3 times as slow), and it is also not as good as HOG in attempting image super-resolution. Furthermore, we also do not have a GPU capable of properly taking advantage of CNN. However, we did leave the option to switch from HOG to CNN in the code. In the future, when we have a better GPU, this will be a focal point for extending this research.

### Face Analysis

Next, the face is analyzed, and landmarks are made on top of the face. In our case, we use the 128-dimensional feature vectors to quantify the face of a person.

### Data Conversion

For data conversion, we will be using the pre-trained output feature vector is 128-d (i.e., a list of 128 real-valued numbers) that is used to quantify the face (Rosebrock 2018a). Training the network is done using a “triplet training step” as shown in the picture below:



This photo contains a picture of the “triplet” training step. (Rosebrock 2018a)

The triplet consists of 3 unique face images — 2 of the 3 are the same person. The NN generates a 128-d vector for each of the 3 face images (Rosebrock 2018a). For the 2 face images of the same person, we tweak the neural network weights to make the vector closer via distance metric (Rosebrock 2018a).

Two of these images are example faces of the same person (Rosebrock 2018a). The third image is a random face from our dataset and is not the same person as the other two images (Rosebrock 2018a).

From there, the general idea is that we’ll tweak the weights of our neural network so that the 128-d measurements of the two identical will be closer to each other and farther from the measurements for the person which is different (Rosebrock 2018a).

### Face Identification

Our network architecture for face recognition is based on ResNet-34 from the Deep Residual Learning for Image Recognition paper by He et al., but with fewer layers and the number of filters reduced by half (He et al. 2016, pp. 770–778).

The network itself was trained by Davis King on a dataset of ~3 million images. On the Labeled Faces in the Wild (LFW) dataset the network compares to other state-of-the-art methods, reaching 99.38% accuracy (Rosebrock 2018a).

We attempt to match each face in the input image (encoding ) to our known encodings dataset (held in data["encodings"] ) using face\_recognition.compare\_faces (Lines 40 and 41).

#### Classification Model - K-NN (or K-Nearest Neighbours)

The K-NN model is a supervised model that assumes that similar things exist in close proximity. In other words, similar things are near to each other. The model will then classify them in the same category.

After recognizing a face, we will then use a K-NN model by first computing the Euclidean distance between the candidate embedding and all faces in our dataset through the compare\_faces function (Rosebrock 2018a). If the distance is below some tolerance then we return True , indicating the faces match (Rosebrock 2018a). Otherwise, if the distance is above the tolerance threshold we return False as the faces do not match (Rosebrock 2018a). Afterwards, we use a voting system to determine which faces in the dataset match the face best, or is there simply no face that matches well.

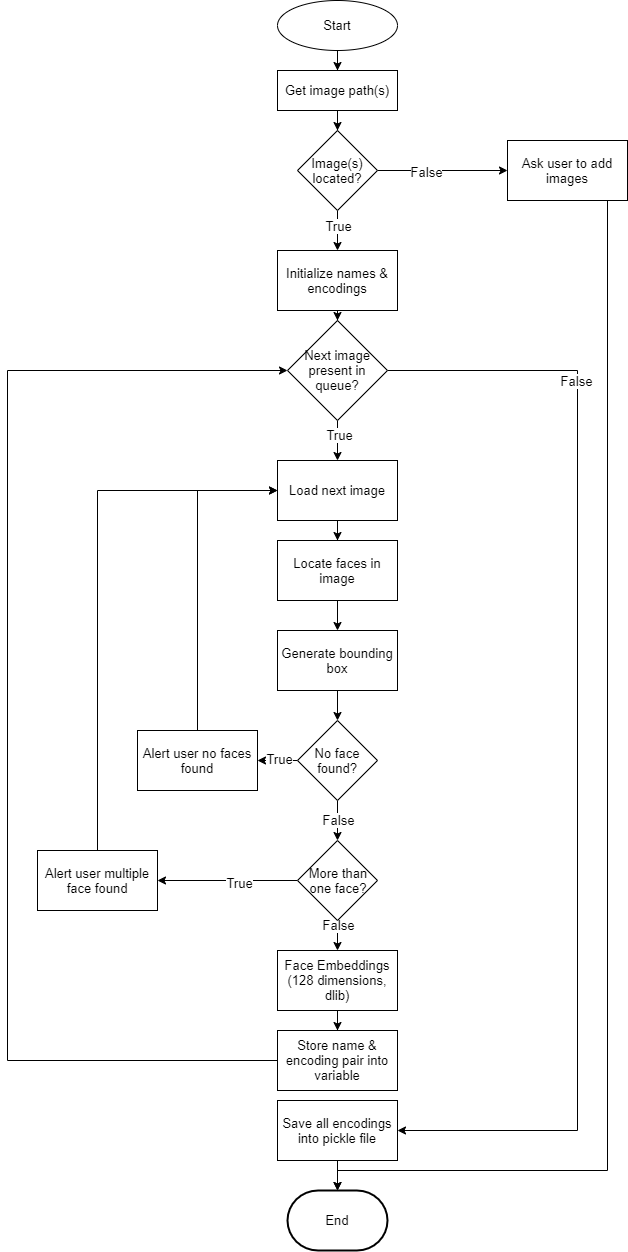
This choice was not arbitrary however. This research first evaluated both K-NN and SVM (Support-vector machine) classifier models by performing a quick test (taking advantage of our hypothesis H1, which is used to evaluate accuracy).

For the quick test, we used the same data for training in the case of SVM, and for encoding in the case of KNN. For the test data we used the exact same data as well. For KNN, we are using the “voting system” inside our system code found in the prototype. Whereas for SVM, we are using the example code with the name “face\_recognition\_svm.py” made by the author of the face\_recognition API (Geitgey 2019).

After performing the testing, we chose the pre-trained K-NN model, with higher performance. For a more thorough explanation, please look at the result and discussion/interpretation of H1.

## System flowchart/activity diagram

### Step 1: Face Encodings (Building dataset)



### Step 2: Face Identification

### 

## Proposed test plan/hypothesis

### Detect & identify face (benchmark of 30 people in LFW/3DLFW + myself + 3 unknown people) - classifier model

H1: The system should be able to detect & identify the face in at least 60% of the sample pictures given at 30x30 pixels.

### Identify face (benchmark of 30 people in LFW/3DLFW + myself + 3 unknown people)

H2: The system should be able to detect the face in at least 60% of the sample pictures given at 30x30 pixels.

H3: The system should be able to identify the detected face in at least 60% of the sample pictures given at 30x30 pixels.

### Detect & identify face (in 3DLFW)

H4: The system should be able to correctly identify ‘Alejandro\_Toledo\_0011.jpg’ as Alejandro\_Toledo at 30x30 pixels.

### Detect & identify face (Real Life)

H5: The system should be able to detect ⅔ of the sample pictures given at 30 width pixels (limitation: can only control one side reliably).

### Identify face (Unknown)

H6: The system should be able to correctly identify ‘Adriana\_Lima\_C’ as an unknown face at 30x30 pixels.

### Identify unknown face (benchmark of 34 people)

H7: The system should be able to correctly identify ⅔ of the unknown people as an ‘Unknown’ at 30x30 pixels.

### Validation

H8: The system should be able to load pictures from the ‘dataset’ folder for encoding.

H9: The system should be able to load pictures from the ‘examples’ folder for identification.

H10: The system should be able to warn the user if the ‘dataset’ folder is not found or empty when encoding.

H11: The system should be able to warn the user if the ‘examples’ folder is not found or empty when encoding.

H12: The system should be able to warn the user if a picture has more than one faces while encoding.

H13: The system should be able to skip encoding if a picture has more than one face while encoding.

H14: The system should be able to display a summary of face recognition & identification.

### Improvement

H15: The system should have improvement in accuracy over the faster methods.

H16: The system should have an “average seconds per picture” of equal or less than 1 second.

# **Result**

## Results

### Disclaimer

Please be noted that the results that are shown (especially the execution time) shown inside the Jupyter Notebook may be slightly different from what is shown here. The reason being non-controllable factors such as current CPU load, (maybe) gravitational time dilation, and so on.

Firstly, H1 is first tested to evaluate which classifier model to be used for the rest of the tests. Both are tested using the “baseline” scaling method, which is `Image.NEAREST` (or nearest neighbour method) (GeeksForGeeks 2019).

The better classifier will be then used for the subsequent hypotheses.

|  |  |  |
| --- | --- | --- |
|  | KNN (Pre-trained) | SVM (Trained) |
| **Detect & identify face (benchmark of 30 people in LFW/3DLFW + myself + 3 unknown people) - classifier model** | | |
| H1: The system should be able to detect & identify the face in at least 60% of the sample pictures given at 30x30 pixels. | Summary (for Nearest Neighbour)  ======================  Total pictures 93  Faces identified correctly: 79  Faces identified wrongly: 9  Faces not identified: 5  Absolute Accuracy (correct / [correct + incorrect + not detected]) \* 100%:  85.0 %  Detection Accuracy [(correct + incorrect) / total] \* 100%:  94.62 %  END, time taken: 46.65 seconds  Avg. secs. per picture: 0.5 seconds  ====================== | Summary (for Nearest Neighbour)  ======================  Total pictures 93  Faces identified correctly: 75  Faces identified wrongly: 11  Faces not identified: 7  Absolute Accuracy (correct / [correct + incorrect + not detected]) \* 100%:  81.0 %  Detection Accuracy [(correct + incorrect) / total] \* 100%:  92.47 %  END, time taken: 47.49 seconds  Avg. secs. per picture: 0.51 seconds  ====================== |
| Since KNN performs 4% better in terms of absolute accuracy, and 2.15% better in detection accuracy, while using less time, we can conclude that KNN is the superior model, and use it for the rest of the hypotheses. If you are interested in the results, you may refer to the Jupyter notebook. | | |

|  |  |  |  |
| --- | --- | --- | --- |
|  | Nearest Neighbour x4 (baseline) | Bicubic x4 | EDSR x4 |
| **Identify face (benchmark of 30 people in LFW/3DLFW + myself + 3 unknown people)** | | | |
| H2: The system should be able to detect the face in at least 60% of the sample pictures given at 30x30 pixels. | Pass.  Summary  ==================  Total pictures 93  Faces identified correctly: 79  Faces identified wrongly: 9  Faces not identified: 5  Absolute Accuracy (correct / [correct + incorrect + not detected]) \* 100%:  85.0 %  Detection Accuracy [(correct + incorrect) / total] \* 100%:  94.62 %  END, time taken: 46.65 seconds  Avg. secs. per picture: 0.5 seconds  ================== | Pass.  Summary  ==========================  Total pictures 93  Faces identified correctly: 83  Faces identified wrongly: 5  Faces not identified: 5  Absolute Accuracy (correct / [correct + incorrect + not detected]) \* 100%:  89.0 %  Detection Accuracy [(correct + incorrect) / total] \* 100%:  94.62 %  END, time taken: 48.7 seconds  Avg. secs. per picture: 0.52 seconds  ================ | Pass.  Summary  ======================  Total pictures 93  Faces identified correctly: 84  Faces identified wrongly: 4  Faces not identified: 5  Absolute Accuracy (correct / [correct + incorrect + not detected]) \* 100%:  90.0 %  Detection Accuracy [(correct + incorrect) / total] \* 100%:  94.62 %  END, time taken: 127.92 seconds  Avg. secs. per picture: 1.38 seconds  ====================== |
| H3: The system should be able to identify the detected face in at least 60% of the sample pictures given at 30x30 pixels. | Pass.  Summary  ==================  Total pictures 93  Faces identified correctly: 79  Faces identified wrongly: 9  Faces not identified: 5  Absolute Accuracy (correct / [correct + incorrect + not detected]) \* 100%:  85.0 %  Detection Accuracy [(correct + incorrect) / total] \* 100%:  94.62 %  END, time taken: 46.65 seconds  Avg. secs. per picture: 0.5 seconds  ================== | Pass.  Summary  =================  Total pictures 93  Faces identified correctly: 83  Faces identified wrongly: 5  Faces not identified: 5  Absolute Accuracy (correct / [correct + incorrect + not detected]) \* 100%:  89.0 %  Detection Accuracy [(correct + incorrect) / total] \* 100%:  94.62 %  END, time taken: 48.7 seconds  Avg. secs. per picture: 0.52 seconds  ================ | Pass.  Summary  ======================  Total pictures 93  Faces identified correctly: 84  Faces identified wrongly: 4  Faces not identified: 5  Absolute Accuracy (correct / [correct + incorrect + not detected]) \* 100%:  90.0 %  Detection Accuracy [(correct + incorrect) / total] \* 100%:  94.62 %  END, time taken: 127.92 seconds  Avg. secs. per picture: 1.38 seconds  ====================== |

|  |  |  |  |
| --- | --- | --- | --- |
|  | Nearest Neighbour x4 (baseline) | Bicubic x4 | EDSR x4 |
| **Detect & identify face (in 3DLFW)** | | | |
| H4: The system should be able to correctly identify ‘Alejandro\_Toledo\_0011.jpg’ as Alejandro\_Toledo at 30x30 pixels. | Pass. | Pass. | Pass. |
| **Detect & identify face (Real Life)** | | |  |
| H5: The system should be able to detect ⅔ of the sample pictures given at 30 width pixels (limitation: can only control one side reliably). | Face 1: Fail. Did not detect any face.  Face 2: Pass  Face 3: Fail. Detected as “Colin Powell” instead of “Hiew Long Shun”  Overall: Fail  PICTURES REDACTED DUE TO PERSONAL INFORMATION | Face 1: Fail. Did not detect any face.  Face 2: Pass  Face 3: Pass  Overall: Pass  PICTURES REDACTED DUE TO PERSONAL INFORMATION | Face 1: Fail. Did not detect any face.  Face 2: Pass  Face 3: Pass  Overall: Pass  PICTURES REDACTED DUE TO PERSONAL INFORMATION |
| **Identify face (Unknown)** | | | |
| H6: The system should be able to correctly identify ‘Adriana\_Lima\_C’ as an unknown face at 30x30 pixels. | Pass | Pass | Pass |
| **Identify unknown face (benchmark of 34 people)** | | | |
| H7: The system should be able to correctly identify ⅔ of the unknown people as an ‘Unknown’ at 30x30 pixels. | Aaron\_Eckhart\_A: Fail.  Detected as ‘Alvaro\_Uribe’    Aaron\_Guiel\_B: Fail.  Not detected.    Adriana\_Lima\_C: Pass    Overall: Fail. ⅓ detection. | Aaron\_Eckhart\_A: Fail.  Detected as ‘Alvaro\_Uribe’    Aaron\_Guiel\_B: Fail.  Not detected.    Adriana\_Lima\_C: Pass    Overall: Fail. ⅓ detection. | Aaron\_Eckhart\_A: Fail.  Detected as ‘Silvio\_Berlusconi’    Aaron\_Guiel\_B: Fail.  Not detected.    Adriana\_Lima\_C: Pass    Overall: Fail. ⅓ detection. |
| **Validation** | | | |
| H8: The system should be able to load pictures from the ‘dataset’ folder for encoding | Pass. | | |
| H9: The system should be able to load pictures from the ‘examples’ folder for identification | Pass. | | |
| H10: The system should be able to warn the user if the ‘dataset’ folder is not found or empty when encoding | Pass. | | |
| H11: The system should be able to warn the user if the ‘examples’ folder is not found or empty when encoding | Pass. | | |
| H12: The system should be able to warn the user if a picture has more than one faces while encoding | Pass. | | |
| H13: The system should be able to skip encoding if a picture has more than one face while encoding. | Pass. | | |
| H14: The system should be able to display a summary of face recognition & identification. | Pass. | | |
| **Improvement** | | | |
| H15: The system should have improvement in accuracy over the faster methods. | N/A.  Summary  ==================  Total pictures 93  Faces identified correctly: 79  Faces identified wrongly: 9  Faces not identified: 5  **Absolute Accuracy (correct / [correct + incorrect + not detected]) \* 100%:**  **85.0 %**  **Detection Accuracy [(correct + incorrect) / total] \* 100%:**  94.62 %  END, time taken: 46.65 seconds  Avg. secs. per picture: 0.5 seconds  ================== | Pass.  Summary  ==========================  Total pictures 93  Faces identified correctly: 83  Faces identified wrongly: 5  Faces not identified: 5  **Absolute Accuracy (correct / [correct + incorrect + not detected]) \* 100%:**  **89.0 %**  **Detection Accuracy [(correct + incorrect) / total] \* 100%:**  94.62 %  END, time taken: 48.7 seconds  Avg. secs. per picture: 0.52 seconds  ================ | Pass.  Summary  ======================  Total pictures 93  Faces identified correctly: 84  Faces identified wrongly: 4  Faces not identified: 5  **Absolute Accuracy (correct / [correct + incorrect + not detected]) \* 100%:**  **90.0 %**  **Detection Accuracy [(correct + incorrect) / total] \* 100%:**  **94.62 %**  END, time taken: 127.92 seconds  Avg. secs. per picture: 1.38 seconds  ====================== |
| H16: The system should have an “average seconds per picture” of equal or less than 1 second. | Pass  Avg. secs. per picture: 0.5 seconds | Pass  Avg. secs. per picture: 0.52 seconds | Fail  Avg. secs. per picture: 1.38 seconds |

## Discussion/Interpretation

### Detect face (benchmark of 34 people)

|  |  |
| --- | --- |
| Hypothesis | Result |
| H1: The system should be able to detect the face in at least 60% of the sample pictures given at 30x30 pixels. | Accepted. |

Discussion/Interpretation

The first hypothesis is meant as a “quick test” for which classifier model to use. As a result, inside the zip file, there are two versions, one is “knn-version”, the other is “svm-version”.

Based on our results, KNN performs 4% better in terms of absolute accuracy, and 2.15% better in detection accuracy, while using less time. Therefore, we have dropped the SVM model.

When I discussed with my teammates, I noticed that their SVM model works better than their KNN model. However, there is one glaring difference is that both their SVM model and their KNN models are trained models. In my case, my KNN model is pre-trained, while only the SVM model is trained.

Therefore, it is vital to do some testing, and since our H1 does not exactly specify what model we must use, we took advantage of that fact, and tested both models first, before moving on to the rest of the hypotheses using the model of our choice.

In the end, K-NN fares superior, compared to a SVM model in our case. One author states that If training data is much larger than no. of features(m>>n), KNN is better than SVM. SVM outperforms KNN when there are large features and lesser training data (Varghese 2019). Since my KNN is already pre-trained, my training data is comparatively much more than my teammate’s. Therefore, KNN pulled ahead in this situation.



Image shows a screenshot of the system using a custom trained SVM model rather than the KNN model, and compared with the KNN model. Note the lower accuracy inside the SVM model.

|  |  |
| --- | --- |
| Hypothesis | Result |
| H2: The system should be able to detect the face in at least 60% of the sample pictures given at 30x30 pixels. | Accepted. |
| H3: The system should be able to identify the detected face in at least 60% of the sample pictures given at 30x30 pixels. | Accepted. |
| H4: The system should be able to correctly identify ‘Alejandro\_Toledo\_0011.jpg’ as Alejandro\_Toledo at 30x30 pixels. | Accepted. |
| H5: The system should be able to detect ⅔ of the sample pictures given at 30 width pixels (limitation: can only control one side reliably). | Accepted. |

These four hypotheses are mainly to show that the system is able to detect faces in low resolution effectively. The H2 & H3 results significantly exceed the initial 60% requirement at 30x30 pixels. All image enhancement works well, with EDSR better than Bicubic, and better than nearest neighbour. Note that in H5, due to the shape of my face, I can only adjust the width reliably.

|  |  |
| --- | --- |
| Hypothesis | Result |
| H6: The system should be able to correctly identify ‘Adriana\_Lima\_C’ as an unknown face at 30x30 pixels. | Accepted. |
| H7: The system should be able to correctly identify ⅔ of the unknown people as an ‘Unknown’ at 30x30 pixels. | Accepted. |

These two hypotheses are mainly to test if the system can properly ignore unknown faces. Adriana\_Lima\_C is not found in the “Unknown” dataset, which confirms that the system can properly identify unknown people as unknown. However, in H6, the system was not able to correctly identify a sufficient number of people as unknown. Therefore, mixed results are obtained here.

This is most likely due to the poor resolution making inference harder than usual.

|  |  |
| --- | --- |
| Hypothesis | Result |
| H8: The system should be able to load pictures from the ‘dataset’ folder for encoding | Accepted. |
| H9: The system should be able to load pictures from the ‘examples’ folder for identification | Accepted. |
| H10: The system should be able to warn the user if the ‘dataset’ folder is not found or empty when encoding | Accepted. |
| H11: The system should be able to warn the user if the ‘examples’ folder is not found or empty when encoding | Accepted. |
| H12: The system should be able to warn the user if a picture has more than one faces while encoding | Accepted. |
| H13: The system should be able to skip encoding a picture has more than one faces while encoding | Accepted. |
| H14: The system should be able to display a summary of face recognition & identification. | Accepted. |

H8 to H14 are all mainly validation & exception handling. Validation & exception handling is very important to ensure the application works well. In this case, all the validations passed, which indicates that the application is good for use, with minimal bugs. Note that in my case, I prevented exceptions from happening in the first place, rather than try-catch them, because it's cleaner (for example, inside the summary part, it is possible to have no images and cause a DivideByZero exception, however, I prevented that by skipping the summary if there are no images in the first place).

|  |  |
| --- | --- |
| Hypothesis | Result |
| H15: The system should have improvement in accuracy over the faster methods. | Accepted. |
| H16: The system should have an “average seconds per picture” of equal or less than 1 second. | Accepted for bicubic. Rejected for EDSR. |

H15 and H16 reflect the outcome that slower image enhancements provide better detection. The biggest jump is from using the bicubic scaling rather than nearest neighbour method to scale. With just an average time increase of 0.02 seconds, the absolute accuracy increased by 4%. The smaller jump is from using the EDSR, which provided another 1% boost to the absolute accuracy. However, the increase of 0.86 seconds which more than doubles the time means that the performance is lower than expected. This does not mean that EDSR is useless however, but it is not suitable for use in performance-demanding situations. Nevertheless, we have to accept H13, and reject H14.

# **Discussion and Conclusion**

## Achievements

The proposed project objectives is as per follows:

|  |  |
| --- | --- |
| Objective | Result |
| To effectively recognize a person’s face under low resolution conditions. This helps with criminal activities because most of the criminals are situated far from the camera, which can be a problem for face recognition algorithms to work. | Successfully accomplished. Managed to enable acceptable detection even at 30x30 pixels. |
| To achieve at least 60% accuracy. The system must at least be able to recognize faces correctly the majority of the time. Due to time constraints, further enhancements can be done in the future. However, the system must be at least usable and can be trusted by others. | Successfully accomplished. System was able to achieve at least 60% accuracy even using the faster available options.  For slower settings, the system was able to achieve better accuracy as well. |
| To have high performance, and able to recognize a person’s face in a short time. Usually, low resolution cameras are also paired with weak computers. Therefore, it is necessary to build the system with performance in mind. | Successfully accomplished. System was able to recognize a person’s face in less than 1 second using HOG + Bicubic.  However, HOG + EDSR, despite having better accuracy, is too slow for video/webcam use. It is only suitable for image use, unless a CUDA-enabled GPU is used. |

## Limitations and Future Works

### Time Limitation

Personally, time is the biggest limiter in this assignment. Due to having dozens of coursework, I need to manage time very strictly. This means that I cannot spend too much time on any particular assignment. As a result, even if there are better, more optimal ways for low-resolution face-recognition, it is not possible for me to explore them in this short amount of time.

### Hardware Limitation

The hardware used for this is an Intel i7-6500U with Intel HD Graphics 520. As an ultra-low-voltage CPU, it is very slow, and incapable of executing face identification at a high speed. Due to the lack of a proper GPU capable of accelerating the work, the speed of the system is very slow, with inference taking more than a second when using the best way to improve resolution (EDSR).

Furthermore, we are also unable to take advantage of the CNN method for face recognition. It was noted by the author that CNN is only suitable for use in GPU. We also need to use EDSR because EDSR has higher accuracy (sometimes better detection, sometimes lower false positive).

### Future Improvements

In the future, I plan to improve this low-resolution face detection & recognition through the following:

1. Implementing an actual face hallucination module, rather than just using a generic image super-resolution module as part of the pipeline. By introducing this, I believe the detection will improve further, and be more accurate
2. Make this system take advantage of GPU, which supports parallel processing, and can dramatically reduce the training & inference time.

## 

# **Reference & Source**

Abeles, P 2021, lessthanoptimal/BoofCV, GitHub, viewed 9 April 2021, <<https://github.com/lessthanoptimal/BoofCV>>.

AnalyticsInsight 2020, OpenCV Vs MATLAB: Which Is Best For Successful Computer Vision Project?, Analytics Insight, viewed 10 April 2021, <<https://www.analyticsinsight.net/opencv-vs-matlab-which-is-best-for-successful-computer-vision-project/>>.

Dwivedi, D 2018, Face Detection For Beginners, Towards Data Science, Towards Data Science, viewed 10 April 2021, <https://towardsdatascience.com/face-detection-for-beginners-e58e8f21aad9>.

EFF 2019, Face Recognition, Electronic Frontier Foundation, viewed 10 April 2021, <<https://www.eff.org/pages/face-recognition>>.

Face Detection 2020, Face Detection: Facial recognition and finding Homepage, Face Detection & Recognition Homepage, viewed 9 April 2021, <https://facedetection.com/>.

FreeCodeCamp 2018, An intuitive guide to Convolutional Neural Networks, FreeCodeCamp, viewed 10 April 2021, <<https://www.freecodecamp.org/news/an-intuitive-guide-to-convolutional-neural-networks-260c2de0a050/#:~:text=Convolutional%20Neural%20Networks%20have%20a%20different%20architecture%20than%20regular%20Neural%20Networks.&text=Every%20layer%20is%20made%20up,layer%20%E2%80%94%20that%20represent%20the%20predictions>>.

GeeksForGeeks 2019, Python PIL | Image.resize() method, GeeksforGeeks, viewed 9 April 2021, <<https://www.geeksforgeeks.org/python-pil-image-resize-method/>>.

Geitgey, A 2017, face-recognition: Recognize faces from Python or from the command line, PyPI, viewed 9 April 2021, <<https://pypi.org/project/face-recognition/>>.

Geitgey, A 2019, face\_recognition\_svm.py, Face\_Recognition, viewed 11 April 2021, <<https://github.com/ageitgey/face_recognition/blob/master/examples/face_recognition_svm.py>>.

Geitgey, A 2020, ageitgey/face\_recognition, GitHub, viewed 9 April 2021, <<https://github.com/ageitgey/face_recognition/blob/master/face_recognition/api.py>>.

Guntupalli, CS 2020, chanddu/Face-Recognition, GitHub, viewed 10 April 2021, <<https://github.com/chanddu/Face-Recognition>>.

Hassner, T 2015, Effective Face Frontalization in Unconstrained Images, Tal Hassner, viewed 9 April 2021, <<https://talhassner.github.io/home/publication/2015_CVPR_1>>.

He, K, Zhang, X, Ren, S & Sun, J 2016, Deep Residual Learning for Image Recognition, pp. 770–778.

HenningsYeomans, PH, S. Baker & B. V. K. V. Kumar 2008, ‘Recognition of LowResolution Faces Using Multiple Still Images and Multiple Cameras’, in 2008 IEEE Second International Conference on Biometrics: Theory, Applications and Systems, vol. , no. , pp. 1–6.

Intel 2021, HOG Descriptor, Intel, viewed 9 April 2021, <<https://software.intel.com/content/www/us/en/develop/documentation/ipp-dev-reference/top/volume-2-image-processing/computer-vision/feature-detection-functions/histogram-of-oriented-gradients-hog-descriptor.html>>.

Leng, B, Yu, K, Liu, Y & Jingyan, Q 2016, ‘Data Augmentation for Unbalanced Face Recognition Training Sets’, Neurocomputing, vol. 235, p.

Li, P, J. Flynn, P, Prieto, L & Mery, D 2018, ‘Face Recognition in Low Quality Images: A Survey’, CoRR, vol. abs/1805.11519, no. , viewed 9 April 2021, <<https://dblp.org/rec/journals/corr/abs-1805-11519.bib>>.

Lim, B, Son, S, Kim, H, Nah, S & Lee, K 2017, Enhanced Deep Residual Networks for Single Image SuperResolution, pp. 1132–1140.

Malczewski, K & Stasiński, R 2009, ‘Super resolution for multimedia, image, and video processing applications’, M Grgic, K Delac & M Ghanbari (eds), Springer Berlin Heidelberg, pp. 171–208.

NEC 2020, A brief history of facial recognition - NEC New Zealand, NEC, viewed 10 April 2021, <<https://www.nec.co.nz/market-leadership/publications-media/a-brief-history-of-facial-recognition/#:~:text=The%20earliest%20pioneers%20of%20facial>>.

Omid Abdollahi Aghdam, Behzad Bozorgtabar, Hazim Kemal Ekenel & Jean-Philippe Thiran 2019, ‘Exploring factors for improving low resolution face recognition’, CoRR, vol. abs/1907.10104.

OpenCV 2013, OpenCV: Introduction to OpenCV-Python Tutorials, Opencv.org, viewed 10 April 2021, <<https://docs.opencv.org/master/d0/de3/tutorial_py_intro.html>>.

― 2018, About, Opencv.org, viewed 10 April 2021, <<https://opencv.org/about/>>.

OSAC 2020, Working Together to Protect U.S. Organizations Overseas, www.osac.gov, viewed 9 April 2021, <https://www.osac.gov/Content/Report/148f55ab-9111-47ef-99e4-1811a5d28a20>.

PandaSecurity 2019, The Complete Guide to Facial Recognition Technology - Panda Security, Panda Security Mediacenter, viewed 10 April 2021, <https://www.pandasecurity.com/en/mediacenter/panda-security/facial-recognition-technology/>.

Parmar, DN & Mehta, BB 2013, Face Recognition Methods & Applications, viewed 10 April 2021, <<https://arxiv.org/ftp/arxiv/papers/1403/1403.0485.pdf>>.

Pluralsight 2020, Face Recognition Walkthrough--FaceNet | Pluralsight, www.pluralsight.com, viewed 10 April 2021, <<https://www.pluralsight.com/guides/face-recognition-walkthrough-facenet>>.

Ren, W, Zhang, J, Ma, L, Pan, J, Cao, X, Zuo, W, Liu, W & Yang, M-H 2018, ‘Deep non-blind deconvolution via generalized low-rank approximation’, Curran Associates Inc., Montréal, Canada, pp. 295–305.

Rosebrock, A 2018a, Face recognition with OpenCV, Python, and deep learning - PyImageSearch, PyImageSearch, viewed 10 April 2021, <<https://www.pyimagesearch.com/2018/06/18/face-recognition-with-opencv-python-and-deep-learning/>>.

― 2018b, OpenCV Face Recognition - PyImageSearch, PyImageSearch, viewed 10 April 2021, <https://www.pyimagesearch.com/2018/09/24/opencv-face-recognition/>.

Rovai, M 2019, Real-Time Face Recognition: An End-To-End Project, Medium, viewed 10 April 2021, <<https://towardsdatascience.com/real-time-face-recognition-an-end-to-end-project-b738bb0f7348>>.

Sight Machine Inc 2011, SimpleCV: Make Computers See with SimpleCV, the Python Framework for Machine Vision, PyPI, viewed 9 April 2021, <https://pypi.org/project/SimpleCV/>.

Tabora, V 2019, JPEG Image Scaling Algorithms, Medium, viewed 10 April 2021, <<https://medium.com/hd-pro/jpeg-image-scaling-algorithms-913987c9d588#:~:text=Nearest%20Neighbor%20Scaling%E2%80%94%20This%20is>>.

Triantafyllidou, D & Tefas, A 2016, Face detection based on deep convolutional neural networks exploiting incremental facial part learning, International Conference on Pattern Recognition (ICPR), pp. 3560–3565.

TutorialsPoint 2021, OpenCV - Overview - Tutorialspoint, www.tutorialspoint.com, viewed 9 April 2021, <https://www.tutorialspoint.com/opencv/opencv\_overview.htm>.

Varghese, D 2019, Comparative study on Classic Machine learning Algorithms, Medium, viewed 10 April 2021, <<https://towardsdatascience.com/comparative-study-on-classic-machine-learning-algorithms-24f9ff6ab222>>.

University of Massachusetts Amherst 2007, LFW Face Database : Main, vis-www.cs.umass.edu, viewed 10 April 2021, <<http://vis-www.cs.umass.edu/lfw/>>.

Wójcik, W, Gromaszek, K & Junisbekov, M 2016, ‘Face Recognition: Issues, Methods and Alternative Applications’, Face Recognition - Semisupervised Classification, Subspace Projection and Evaluation Methods, viewed 10 April 2021, <https://www.intechopen.com/books/face-recognition-semisupervised-classification-subspace-projection-and-evaluation-methods/face-recognition-issues-methods-and-alternative-applications>.

Yang, J & Huang, T 2017, ‘Image super-resolution: Historical overview and future challenges’, Super-Resolution Imaging, pp. 1–33.

Zhiyi Cheng, Xiatian Zhu & Shaogang Gong 2018, ‘Low-resolution face recognition’, CoRR, vol. abs/1811.08965.