

An Exploratory Analysis into Face Recognition, Neural Networks and How They Can Be Used to Create an Attendance Based System

Clare O'Brien

N00180771

Supervisor: John Dempsey

Second Reader: Mohammed Cherbatji

Year 4 2021-22

DL836 BSc (Hons) in Creative Computing

Declaration of Authorship

The incorporation of material without formal and proper acknowledgement (even with no deliberate intent to cheat) can constitute plagiarism.

If you have received significant help with a solution from one or more colleagues, you should document this in your submitted work and if you have any doubt as to what level of discussion/collaboration is acceptable, you should consult your lecturer or the Course Director.

WARNING: Take care when discarding program listings lest they be copied by someone else, which may well bring you under suspicion. Do not leave copies of your own files on a hard disk where they can be accessed by other. Be aware that removable media, used to transfer work, may also be removed and/or copied by others if left unattended.

Plagiarism is an act of fraudulence and an offence against Institute discipline.

Alleged plagiarism will be investigated and dealt with appropriately by the Institute. Please refer to the Institute Handbook for further details of penalties.

The following is an extract from the B.Sc. in Creative Computing (Hons) course handbook. Please read carefully and sign the declaration below.

Collusion may be defined as more than one person working on an individual assessment. This would include jointly developed solutions as well as one individual giving a solution to another who then makes some changes and hands it up as their own work.

DECLARATION:

I am aware of the Institute's policy on plagiarism and certify that this thesis is my own work.

Student: Clare O'Brien

Signed Clare O'Brien

Failure to complete and submit this form may lead to an investigation into your work.

Abstract

An This paper will go into detail about how neural network models are made and why they have grown in popularity over the years. It will also explain how to develop an attendance-based system using face recognition. This application will be able to predict via uploading an image and capturing a snapshot of a person's face. It will also display the attendance to the user. The steps involved in the development were to first gather the requirements and decide which are most important, then research on machine learning and neural networks took place. After this, the model and application were designed. After the design process is complete the implementation step of the process began which included creating and choosing a model and integrating it with an application. For testing the application functional testing, model testing and user testing were carried out. The results of the testing showed that the application worked as intended and the model was as accurate as possible.

Further work that could be carried out includes creating a way for the users to state whether or not the prediction is correct. Automatic addition of new people rather than the image being saved and the model having to be retrained manually. Creating an admin dashboard for all training and manipulation to be done within the application and the ability to download the attendance.

Acknowledgements

I would like to thank my supervisor John Dempsey who gave great guidance and advice throughout the year. I would like to thank my friends for helping with the testing of my application as well as for the general support they gave me throughout the development of the project. I would also like to show my appreciation for my family and mentor outside the college for helping with the development of the project. the lecturers that helped me with various errors while making the application.

Table of Contents

DECLARATION OF AUTHORSHIP	1
ABSTRACT	2
ACKNOWLEDGEMENTS	3
1 INTRODUCTION	7
2 REQUIREMENTS ANALYSIS	9
2.1 REQUIREMENTS INVESTIGATION INTO EXISTING APPLICATIONS	9
2.2 REQUIREMENTS MODELLING	11
2.2.1 <i>Technical Requirements</i>	12
2.2.2 <i>Functional Requirements</i>	12
2.2.3 <i>Non-Functional Requirements</i>	13
2.3 USER RESEARCH AND PERSONAS	14
2.3.1 <i>Survey</i>	14
2.3.2 <i>User Personas</i>	19
2.3.3 <i>Use Case Diagrams</i>	22
2.4 FEASIBILITY STUDY	24
2.4.1 <i>Technologies</i>	24
2.4.2 <i>The Benefits and Issues of the Application</i>	26
2.5 PROJECT PLAN	27
2.6 TEST PLAN	28
2.7 REQUIREMENTS SUMMARY	29
3 RESEARCH	30
3.1 MACHINE LEARNING	30
3.1.1 <i>Supervised learning</i>	30
3.1.2 <i>Unsupervised learning</i>	33
3.1.3 <i>Reinforcement learning</i>	35
3.2 NEURAL NETWORKS	36
3.2.1 <i>Activation Functions</i>	37
3.2.2 <i>Real World ANN Example</i>	38
3.3 DEEP LEARNING	40
3.4 CONVOLUTIONAL NEURAL NETWORKS	41
3.4.1 <i>Example CNN</i>	44
3.5 FACE RECOGNITION	48
3.6 BACKGROUND	49
3.7 APPLICATIONS OF FACE RECOGNITION USING CNN	50

3.8	TENSORFLOW AND KERAS	51
3.9	RESEARCH SUMMARY	52
4	DESIGN	53
4.1	SYSTEM ARCHITECTURE	53
4.2	SEQUENCE DIAGRAM	54
4.3	DATABASE DESIGN	55
4.4	PROCESS DESIGN	56
4.5	MODEL DESIGN	57
4.6	USER INTERFACE DESIGN	59
4.7	DESIGN SUMMARY	62
5	IMPLEMENTATION	63
5.1	DEVELOPMENT ENVIRONMENT	64
5.2	IMAGE CLASSIFICATION AND MODEL PREPARATION	65
5.2.1	<i>Understanding CNN's with Keras</i>	65
5.2.2	<i>Understanding ImageNet</i>	68
5.2.3	<i>Image Manipulation</i>	69
5.2.4	<i>Saving and Loading the Model</i>	72
5.2.5	<i>Model Comparison</i>	73
5.3	MACHINE LEARNING IN FLASK	76
5.3.1	<i>Setting up Flask</i>	77
5.3.2	<i>Testing In insomnia</i>	78
5.4	WORKING WITH A DATABASE	79
5.5	INTEGRATING MODEL WITH DATABASE APPLICATION	80
5.5.1	<i>Integration Issues</i>	80
5.5.2	<i>Creating Templates and Routes</i>	81
5.5.3	<i>Adding Upload Prediction Functionality</i>	82
5.5.4	<i>Database integration</i>	84
5.5.5	<i>Adding Attendance</i>	85
5.5.6	<i>Adding Basic Image Capturing Functionality</i>	86
5.6	MODEL FINETUNING	88
5.6.1	<i>Retraining the model</i>	89
5.6.2	<i>Adding more Manipulations and Images</i>	90
5.7	FINAL ATTENDANCE APP IN FLASK	91
5.7.1	<i>Added Features</i>	91
5.7.2	<i>Issues that Arose</i>	92
5.8	IMPLEMENTATION SUMMARY	92
6	TESTING	93
6.1	FUNCTIONAL TESTING	93

6.1.1	<i>Links</i>	94
6.1.2	<i>Client/Server Communication</i>	94
6.2	MODEL TESTING	94
6.3	USER TESTING	95
7	PROJECT MANAGEMENT	96
7.1	PROJECT MANAGEMENT TOOLS	96
7.2	DIFFICULTIES AND REFLEXION	98
8	CONCLUSION	99
8.1	SUMMARY OF CHAPTERS	99
8.2	FUTURE IMPROVEMENTS	100
8.3	PERSONAL TAKEAWAYS AND PROJECT ACHIEVEMENTS	100
9	BIBLIOGRAPHY	102
10	APPENDICES	106
10.1	APPENDIX A – SURVEY FOR REQUIREMENTS AND EXCEL ANSWERS	106
10.2	APPENDIX B – USER INTERFACE AND SYSTEM MODEL DESIGNS	106
10.3	APPENDIX C – CODE REPOSITORY	106
10.4	APPENDIX D – INTERIM PRESENTATIONS	107
10.5	APPENDIX E – MICROSOFT PLANNER	107

1 Introduction

'With the development of deep learning, face recognition technology based on CNN (Convolutional Neural Network) has become the main method adopted in the field of face recognition.' - Jie Wang and Zihao Li (2018)

Face recognition was once seen as an application of the distant future, but developments in technology in the last 60 years have caused it to be used in our everyday lives. For instance, Facebook used face recognition in their person-tagging system, and security systems have adopted facial recognition like VisionAI to create more advanced security systems. These systems can also be used for tracking down criminals using CCTV cameras. There have been vast developments in face recognition, especially in its evolution from 2D to 3D techniques. The first popular 2D technique was called eigenfaces, which used the angles from different points of a face to create vectors that would be able to recognise if a face was or was not in a particular picture. This was mainly used for front-facing photos. A 3D technique is where multiple images are taken from the front and sides of the face to create a 3D model of a face model, which could be then used for general or specific face detection.

A series of recent studies, including 'Face recognition as a Biometric Application' (Petrescu, 2019), have indicated that deep learning techniques have become one of the fastest and most efficient ways to create a face recognition system mainly involving Convolutional Neural Networks (CNN). This research paper will discuss the architecture of CNN models and some widely used models used in face recognition. These models will be shown in real-world applications and how they improve different industries, including attendance systems for companies. A research paper was written about how face recognition and how CNNs could be used to create attendance systems. These have been implemented and documented in research papers, including a smart attendance system (Kumar Chauhan & Pandey, 2018) that was part of the inspiration for this application.

For this project, face recognition within attendance systems will be investigated. Through this an application will be created to demonstrate how this will work. The central part of this project will be implementing the recognition model and how it will be made and trained to develop an accurate attendance system. Ways to improve this will be investigated, including creating accurate predictions using a small dataset using image manipulation. An essential part of this process will be researching how current image recognition models work and how elements of

them can be used or studied to aid in the implementation of this project. This research paper aims to create a system that automatically gathers attendance while also looking into how users feel about this type of technology.

The following research paper is structured as follows. Requirements analysis will discuss the feasibility study, requirements modelling, user research and project plan. The research chapter is a literature review on how machine learning and deep learning work and how this can be used in face recognition applicators for attendance. The design chapter describes the system architecture, model architecture and the user interface design for the proposed application. The implementation chapter will describe the phases and how the components came together for the final application. The testing chapter discusses the functional, model and user testing undertaken for the application. The project management chapter will discuss how the project was managed and what tools were used. The last chapter is the conclusion, which will summarise the project and its achievements and future developments.

2 Requirements Analysis

This chapter aims to describe the requirements and feasibility of this application. The purpose of gathering requirements is to understand what type of application is being implemented and why it will be built. It identified each of the requirements needed from the start to the end of the project. These can be gathered using various means and tend to be split into sections; the sections of requirements gathering done in this chapter are requirements modelling, user research and feasibility.

Requirements modelling details the functional and non-functional requirements for the application. Functional requirements are product features that must be implemented within the application for the user to accomplish tasks. Non-functional requirements are not needed for the application to work but instead define how the system should perform. For example, a functional requirement would be an alert sent to users after entering incorrect information. In contrast, a non-functional requirement would be that that alert appeared immediately rather than the user waiting a couple of seconds to see it. The user research section investigates what users want from this application and identifies the target audience. It's also carried out to identify any issues users would have with the application. The feasibility section will define how plausible it is to get this system running and the limitations associated with the project. To end this analysis, a test plan and a project plan for the developer will be discussed.

2.1 Requirements Investigation into Existing Applications

There are various different applications for face recognition available nowadays. This is because face recognition research has expanded over the past ten or so years. Many researchers and developers are finding new ways to use this technology to their advantage, and one of those is to create an automated attendance system using face recognition. There are many attendance systems made either for research papers or for industry, and many different methods are used with Convolutional Neural Networks being one of the top ones as it have extremely high accuracy.

To create a robust and accurate face recognition application, it is important to look at other existing systems as it can give insight into how other people have implemented similar applications. This can also show the problems that have arisen with certain methods. As similar applications have been created before it is important to look at them to see how they could be improved or what models have been used to create the more accurate applications. Through

this research I decided to create a neural network for my face recognition system as I realised other methods or pre-existing models weren't as accurate as I wanted. Many of the applications that were looked at are small and were used in research papers. These mainly took a photo of a class to put through a recognition model whereas I wanted it to work through a frontend application. Along with this the developer discovered that the application used for criminal investigation purposes are similar to those used for attendance. Below is a brief explanation of monitoring systems made for criminal investigations and attendance application.

In a face recognition paper for showing how face recognition can help, Virgil Petrescu mentioned how face recognition can help criminal investigations as these algorithms can provide additional information that a person is unable to decipher (Petrescu 2019). Advances in face recognition technology can allow for CCTV security systems to be used to identify faces at a crime scene using 3D face recognition. In addition to this a recent study was carried to detect street crime using a face recognition model called VGG-19, to see how accurate this system would be and how it would help in the prevention of street crime. The system ended up being 81% accurate and had 0.025 frames per second detection time.

In 2012 Baloch et al. discussed their method of implementing this system by using a hybrid approach involving neural networks and other methods of face recognition. This system proved to be somewhat accurate but there were issues brought up with people with facial hair or veils being harder to identify. On the other hand, this system is very easy to set up and was computationally inexpensive. Along with this a more recent study conducted by Chauhan and Pandey discusses using purely CNN which is a type of neural network for their system. The results of this system were very positive and allowed for quick and efficient attendance taking, the test set had 100% accuracy when all students were facing the camera. According to Chauhan and Pandey (2018) 'face recognition-based approach is found to be the best method for smart attendance system'.

An example of a commercial attendance system currently using face recognition is Alndra labs smart attendance system. This system takes pictures of a classroom and stores the information on a server.

The system will work on face recognition where each student in the class will be photographed, and their details will be stored in a server. The teacher can then record the attendance by just clicking some pictures of the classroom. The system will recognize the faces and verify the

presence or absence of each student. ("Face Recognition attendance system for schools and ...") This smart attendance system allows an admin to monitor the attendance of a full region on a smartphone.

The benefits of this system are:

- Efficient and saves time in a classroom
- Avoids errors from manual attendance
- Students can't fake attendance
- Becomes more accurate over time as it uses a machine learning model

This system is shown to be a very good one, but the issue is that it doesn't clearly show what technologies are used in the creation of the application. This makes it tough to find out how they went about making their system. Overall, this is a good application to look at to pick out features that would be useful. Features including saving the students information to a server and having a manual system where the teacher can verify the attendance manually. The research papers that were looked were very insightful and have given great ideas on features to add and others that will not be used. The developer didn't like how the images were taken and try to get close to but it's better to look at research papers and open-source applications to understand how to go about creating this from the bottom up.

2.2 Requirements Modelling

This section details all the requirements that would be needed for the ideal application. It is split between user requirements, technical requirements, functional requirements, and non-functional requirements. All of these are based on what I've read in other research projects about face recognition along with the aforementioned web application.

The user requirements are based on what existing applications have and what users of those applications liked. This application will not have many user requirements as most user requirements apply to the front end and this is primarily a recognition model. There will be a small user interface for very basic uses. The user should be able to view the attendance within the application and be able to simply upload or take a picture of themselves for the application.

2.2.1 Technical Requirements

Technical requirements refer to the technical issues that must be implemented to create the desired application. This includes the performance, availability, and reliability. In many software projects it refers to the programming languages that will be used to create the application and the standards or required features it requires.

The technical requirements for this application are as follows:

- High quality camera for accurate detection
- Efficient python system
- Anaconda with environment set up for the project
- Flask frontend
- NoSQL database or SQL database

2.2.2 Functional Requirements

Functional requirements are essential for a working application. They define the basic system behaviour that should occur when doing specific tasks. These requirements can be rated by priority as some items are more necessary than other, some will be high priority which will be counted as essential for the final product. Medium priority features are not essential for the user to use the application and may only be added down the line. Though most would make using the application better for the user. Low priority are items that would be ideal for the application to have but in the timeframe will most likely not get implemented and are not necessary for a prototype.

High Priority

- Camera to detect face
- Algorithm that first takes snapshots of face to be fed to model.
- Camera accurately recognises face
- Timestamp and unique id added to database
- If camera detects new face – added to the attendance report with new id
- Model can be constantly trained
- Images can be manually inputted to test model
- Model can manipulate images to help train model
- Model recognised faces fast and efficiently

Medium Priority

- Model can recognise multiple faces in on snapshot
- Front end app – ability to view list of people in attendance
- If unknown person a warning or alert is sent to user
- Application removes person once they leave room
- Alert sent when new face is detected
- New person can take a picture with their name and add to the database as new user

Low Priority

- Auto deleting of images once not needed
- Being able to delete images as they predict a face.
- Admin dashboard to edit attendance
- Live feed detecting attendance

2.2.3 Non-Functional Requirements

The following requirements are based how well a system behaves and if there are limits on its functionality. These requirements will not affect the base functionality of the application and are not needed for it to work. These include how quick the application responds to actions and how easy it is to use. These are listed below

- User-Friendly interface
- Get prediction quickly.
- Fast load time between pages
- Works on multiple browsers
- Doesn't slow down after multiple attendances are taken
- Easy for users to navigate

In future iterations of this applications more of these requirements would be more necessary as it would be a user-focused application in its final iteration. This is because of the current timeframe to complete this project. The primary focus will be on the model for the application rather than the user interface.

2.3 User research and personas

For the user research portion of requirements research on various types of users was undertaken during the requirements gathering phase of this project. Firstly, a survey was carried out. After the survey personas were created. These are characters created based on the ideal customer or user of the application. Personas can be created by talking to users and the survey that is conducted will aid in creating a persona. This will also be made using what the two main users of the application would be based on the existing applications section.

2.3.1 Survey

During the requirements gathering portion of this project, a survey was held with a small focus-group, to ascertain if they had any security concerns with the product. In this survey the age group that answered it were all between the ages of 20 and 35, the majority being college students in my course. Around 50% of the responses knew about face recognition and its applications whereas the other half were unsure about how these applications worked. Due to this the developer felt that the survey got a good variant of opinions due to the spread of ages and knowledge. The survey can be found along with the excel of the answers in Appendix A.

What age are you?
42 responses

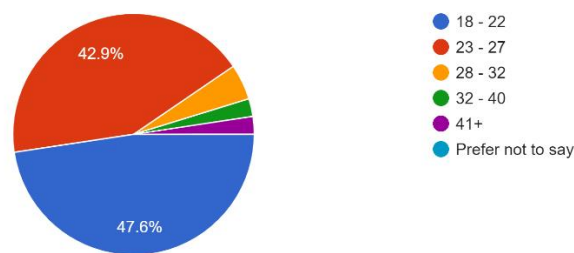


Figure 1 Ages of people taking the survey

To gain an understanding of how much the people taking the survey knew about face recognition the question shown in figure 2 was asked. This showed that about 50% of people knew a lot about the technology and the other half knew a little bit or nothing at all. It's important to know how knowledgeable the group was and to get others to take the survey if

the answers weren't diverse. As the developer wanted to make sure there was people answering that didn't know a lot a lot and people that did.

How much do you know about face detection and its uses?
42 responses

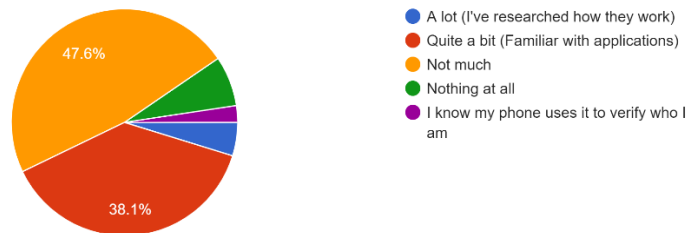


Figure 2 second survey Q

The image in figure 3 shows the question that was asked to get real world reference for what people think about face recognition technologies, As can be seen a third are concerned about the technologies and over 40% say it depends. Over 40% answered 'depends', for this the following question asked to elaborate on their answer as depends on can be taken as vague without context. From the extra information in the following question, it can be inferred that up to 70% of people surveyed feel it can be used for malicious reasons. Below is a list of the most common concerns from this question.

- Concerned about apps using the images for tracking purposes (by government or other organizations)
- Don't like the idea of it being necessary and would prefer it as an optional feature.
- The possibility of it being used in security cameras in future, feeling of no privacy.
- A lot of the participants state that it really depends on the morals around it and laws would have to come in to stop it being abused.

Are you concerned about the rise in use of face recognition technologies?

42 responses

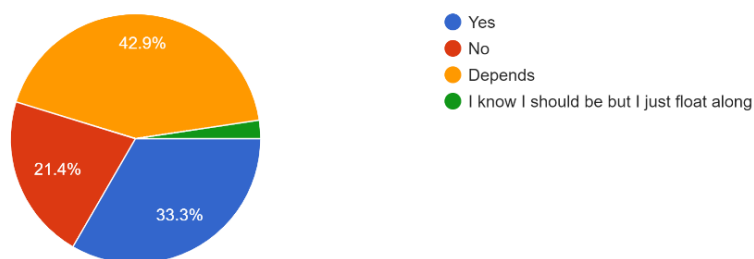


Figure 3 survey question 3

The following questions shown in figure 4 – 6 are about how comfortable people are with automatic attendance. These were asked from a scale of 1-5, 1 being very comfortable and 5 being very uncomfortable. The questions in figure 4 and 5 were the same except for in figure 4 the files would not be kept online and in figure 5, it asking about if the footage r images were kept on file. There is a clear difference in how comfortable someone is when the information is being kept and when the information is being deleted immediately. Over 60% of the group would be at least slightly comfortable with it if the footage were being deleted. This makes a large shift to 70% approx. not feeling comfortable when the footage is being kept. The information gathered here would be important for the application and will be considered during the implementation and design stages.

How comfortable would you feel about automatic attendance being taken using a video camera?

42 responses

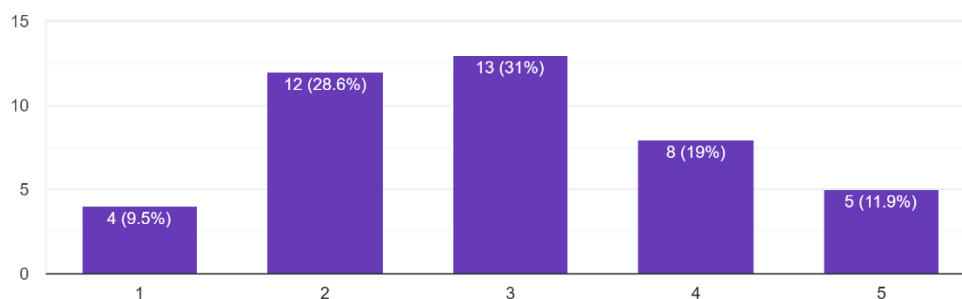


Figure 4 Question on comfort - info not kept of file

How comfortable would you feel about automatic attendance being taken using a video camera?

42 responses

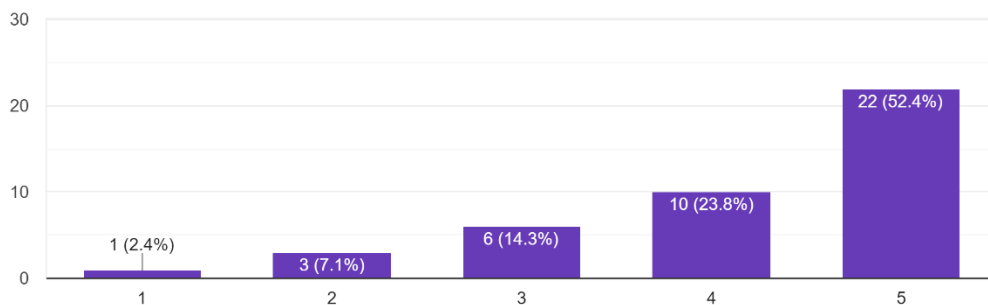


Figure 5 Question on comfort - info kept of file

The question in figure 7 concentrated on the general monitoring that would occur if it were a live video feed taking attendance. The overwhelming majority were extremely uncomfortable with this as well and in a question to elaborate on this people felt that it would be disconcerting to have a camera watching them all the time even if it wasn't recording.

How comfortable would you feel about a camera monitoring you?

42 responses

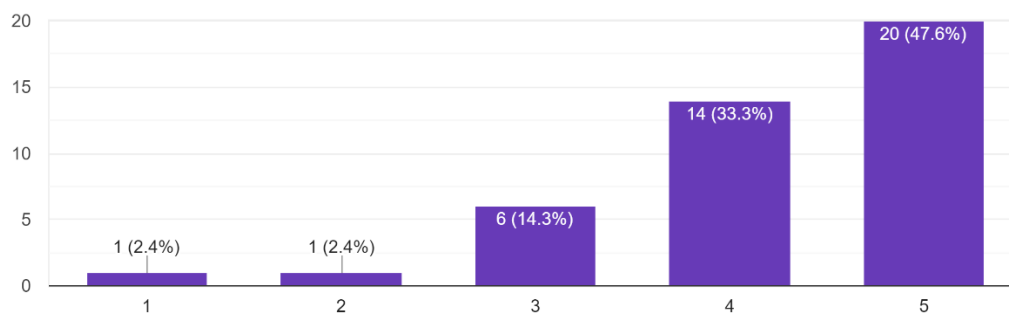


Figure 6 comfort being monitored

This section is based on the front end of this application. The front end will be a simple application that would show you the attendance in a certain room which will be captured by a live feed from a camera. There will also be options to search up people and see if they are present. Another feature would be that through the attendance gathering and the camera monitoring if someone leaves, the application will give predictions on when someone will return to a room or when a person is most likely not there.

Figure 7 Explanation for following section

In figure 7 there is a description, this is the scenario in which the following questions are related to. The description details what the application would be and that there would be a live attendance feed. The graph in figure 8 shows peoples hesitance with wanting this application to be used and that they would not use it themselves. Other extra comments mentioned that they do not see the use of it and feel like it is and unnecessary application as people generally don't need to know where other people are. The comments were mentioned in the following question to gain perspective on their answer.

Using the above description would you use this app?

42 responses

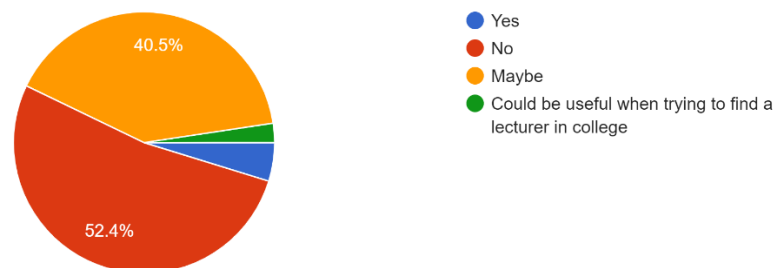


Figure 8 Pie chart

The image in figure 9 shows that people would be uncomfortable with having a camera connected to this application in the room. this is mainly due to security concerns around the app. If the wrong person was looking at it, or if the feed was being saved and their faces were kept on file. Some of the group also stated in an answer that it could affect their attendance badly if there was a camera in the room.

The final question asked was concerning security and any other concerns people have with this application. The main concerns people had were:

- Footage being misused by people stalking a person or the info getting into the wrong hands.
- The camera can cause undue stress on a class which could affect productivity and concentration.
- Location not being private anymore.
- Used for malicious purposes
- Identity theft
- Big concern was people feel that it's not necessary.

Would you feel comfortable in a room where there was a camera connected to this app?

42 responses

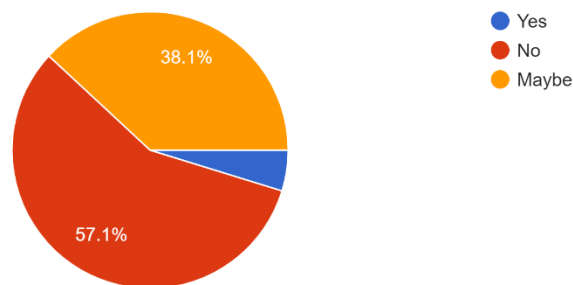


Figure 9 Question on application

2.3.2 User Personas

Personas are created as ideal or possible users of the software application. There are two different main types of users of this application: An employer or college lecturer and a student or worker. The lecturer would be the one using the admin side and keeping the attendance part of the system. The student would be using it by uploading or taking a snapshot of themselves for the application.

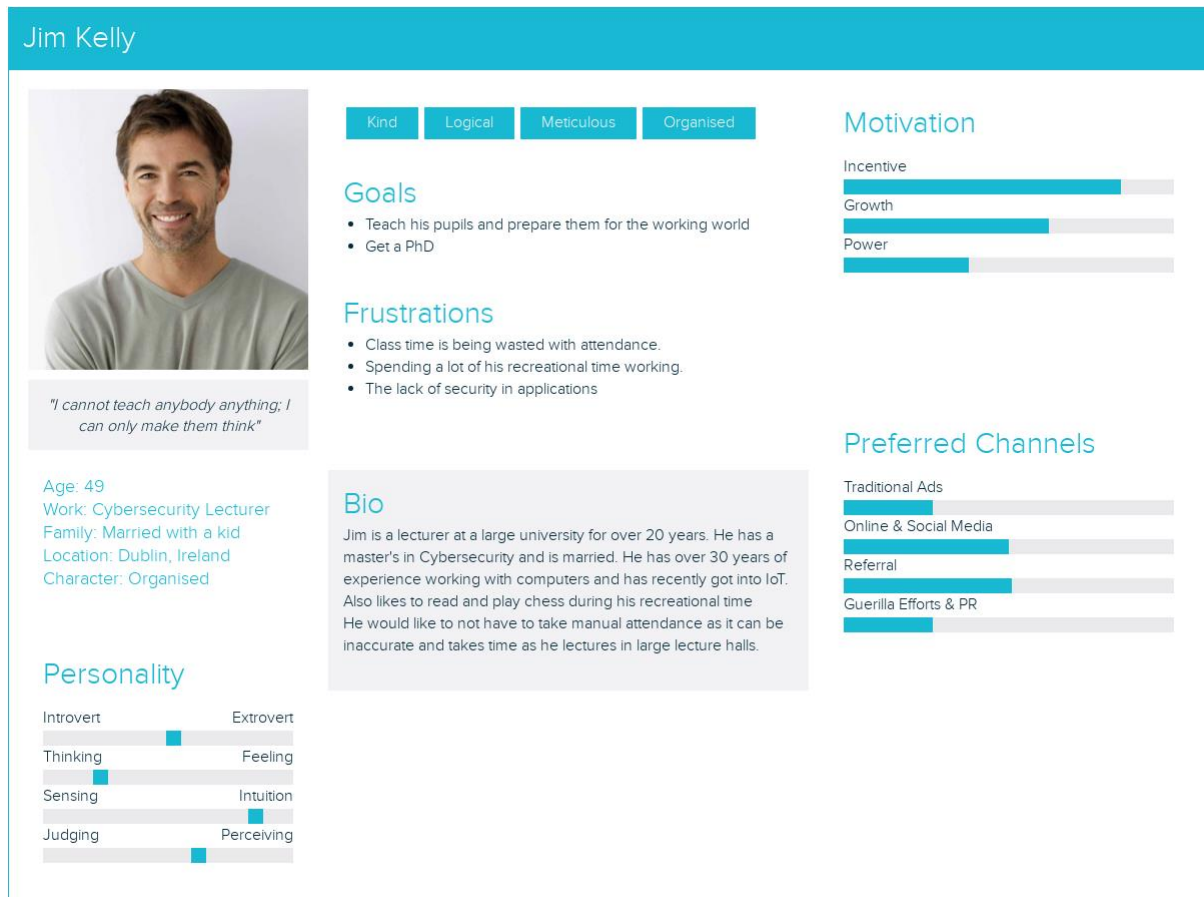


Figure 10 College Lecturer Persona

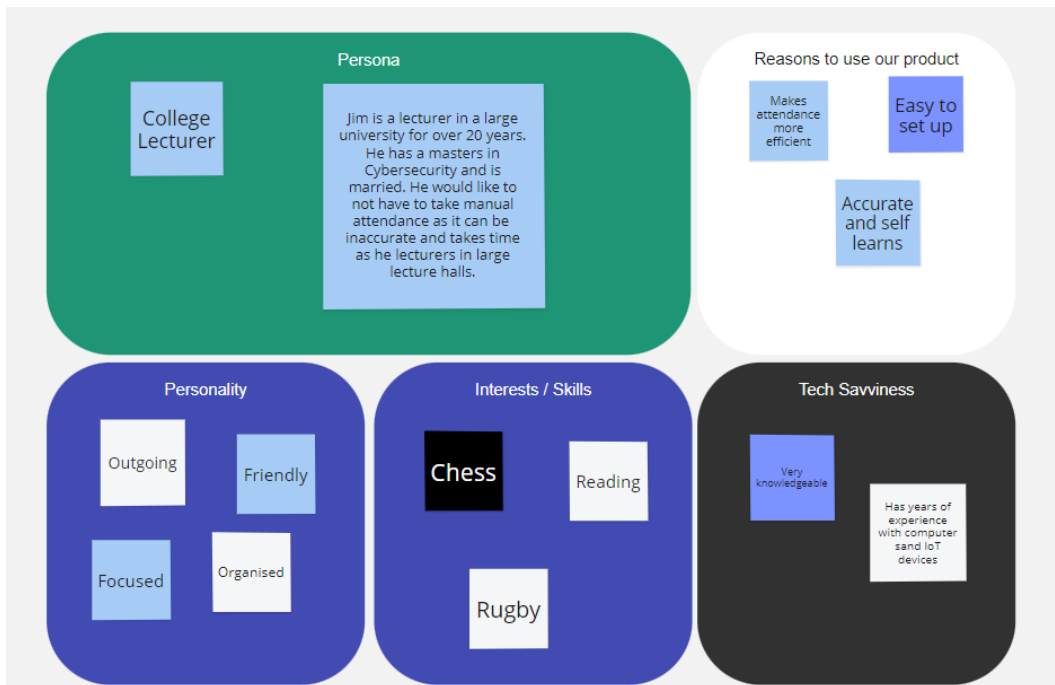


Figure 11 Lecturer Persona

The lecturer wants to be able to monitor attendance easier as they teach in large halls, they would be very interested in making attendance more accurate and having an application that does it quicker. This application would be used as the beginning of class while the lecturer is preparing the material. Each student would be told to take a snapshot of themselves. After class then the lecturer would download the attendance and if any new person or student that wasn't on the original attendance list uploaded their image it would be marked, and the lecturer would contact that student or admissions about training the dataset with the new person. On this side of the application the college lecturer may find it very useful as an accurate way to take attendance. This is backed up by the growing popularity of attendance systems used in classrooms.

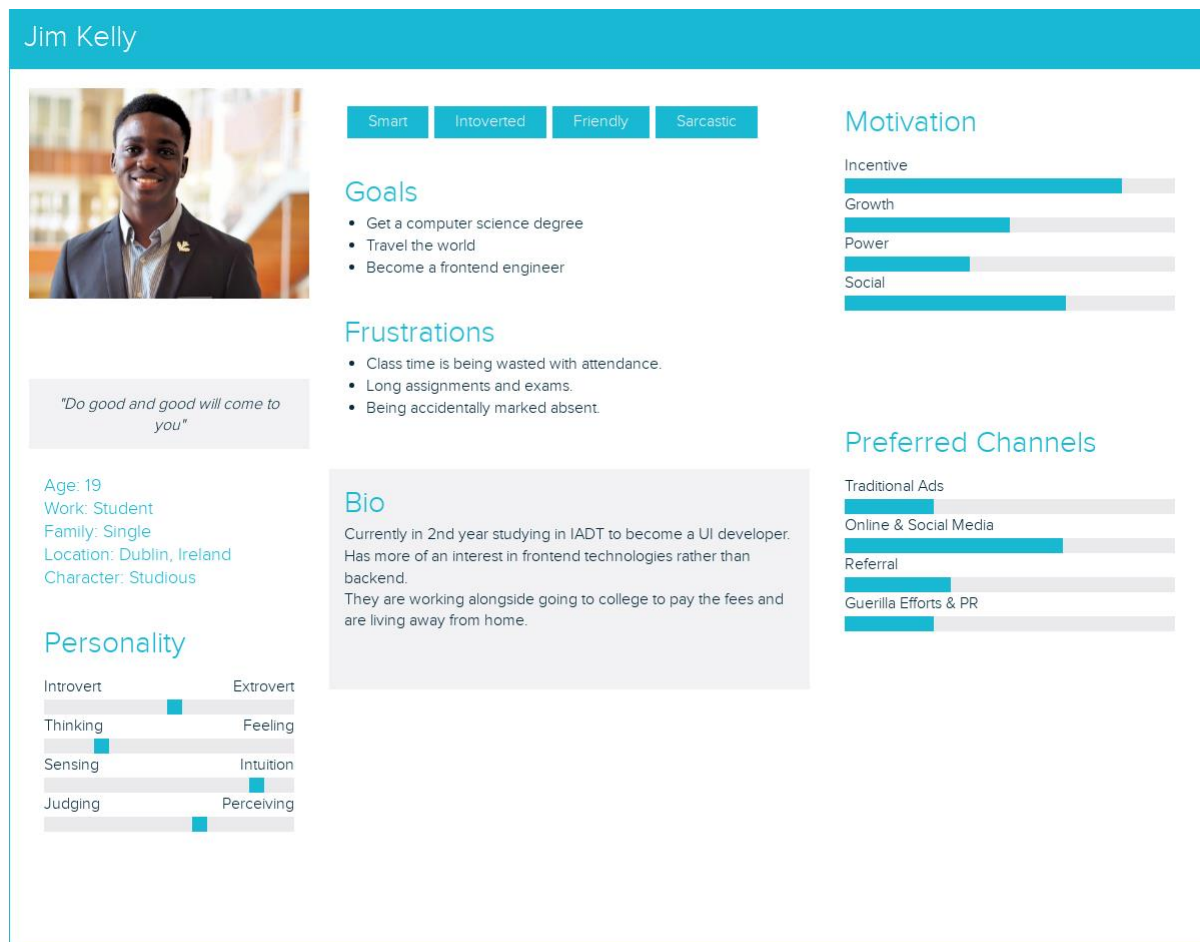


Figure 12 Student Persona

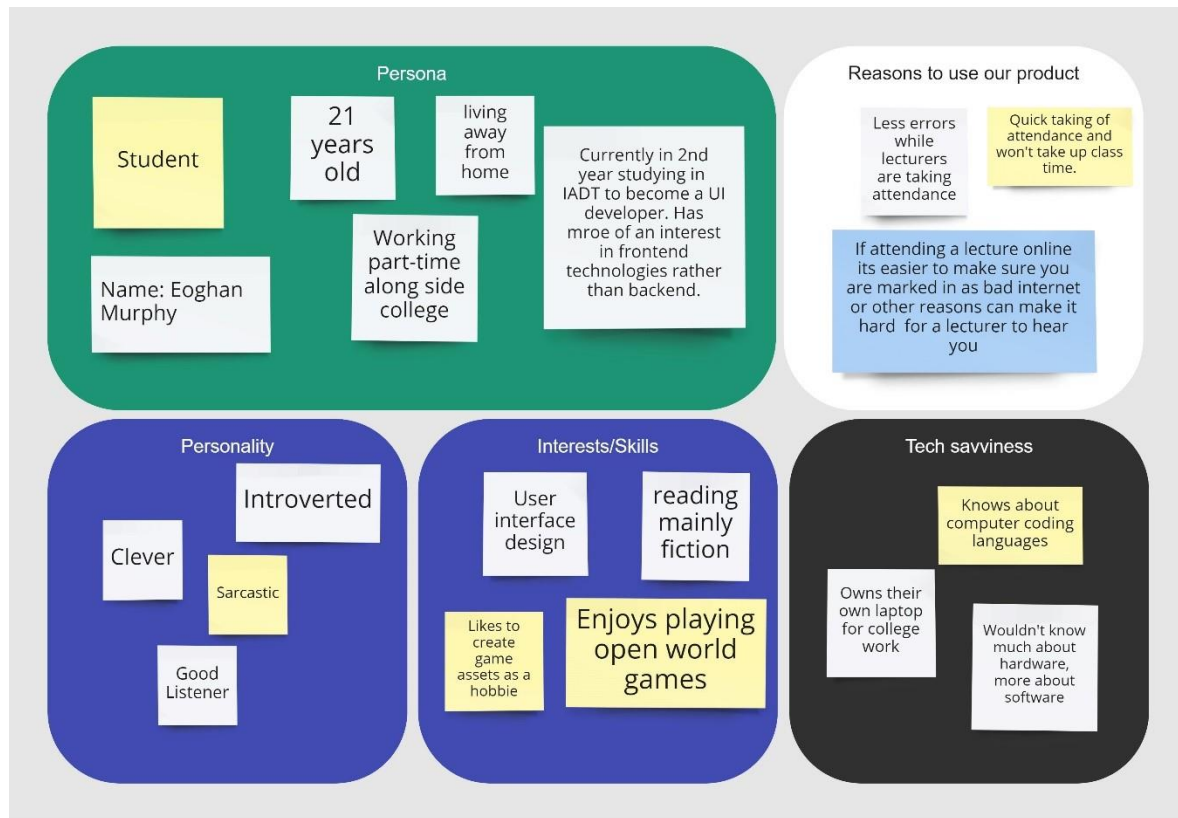


Figure 13 Student Persona

The second persona is based on a student that would use the application on the user side rather than the admin side. They would either upload their image or would take a screen capture of themselves. In another version of this application the application will take attendance from a live video feed. According to the survey undertaken as part of this project, the student is fine with the attendance taking as long as the images aren't saved online and also have many security concerns about what happens with the data. This information is taken from what was answered by the above survey. According to that survey about 50% of students wouldn't be a fan of this if the data is being stored. Even though there are problems this application would still come in handy for the student.

2.3.3 Use Case Diagrams

In figure 14 there is a simple use case diagram of what the desired front-end application will be able to do. The user and admin will have a few abilities and this app will be quite simplistic.

The user will be able to upload images, if they are a new student, they will be asked to name their file their full name. Once they are on file, they can either upload an image or capture their face using the webcam and they will be able to view the current attendance. The admin of the system will have the ability to upload new images for training as well as retraining the model.

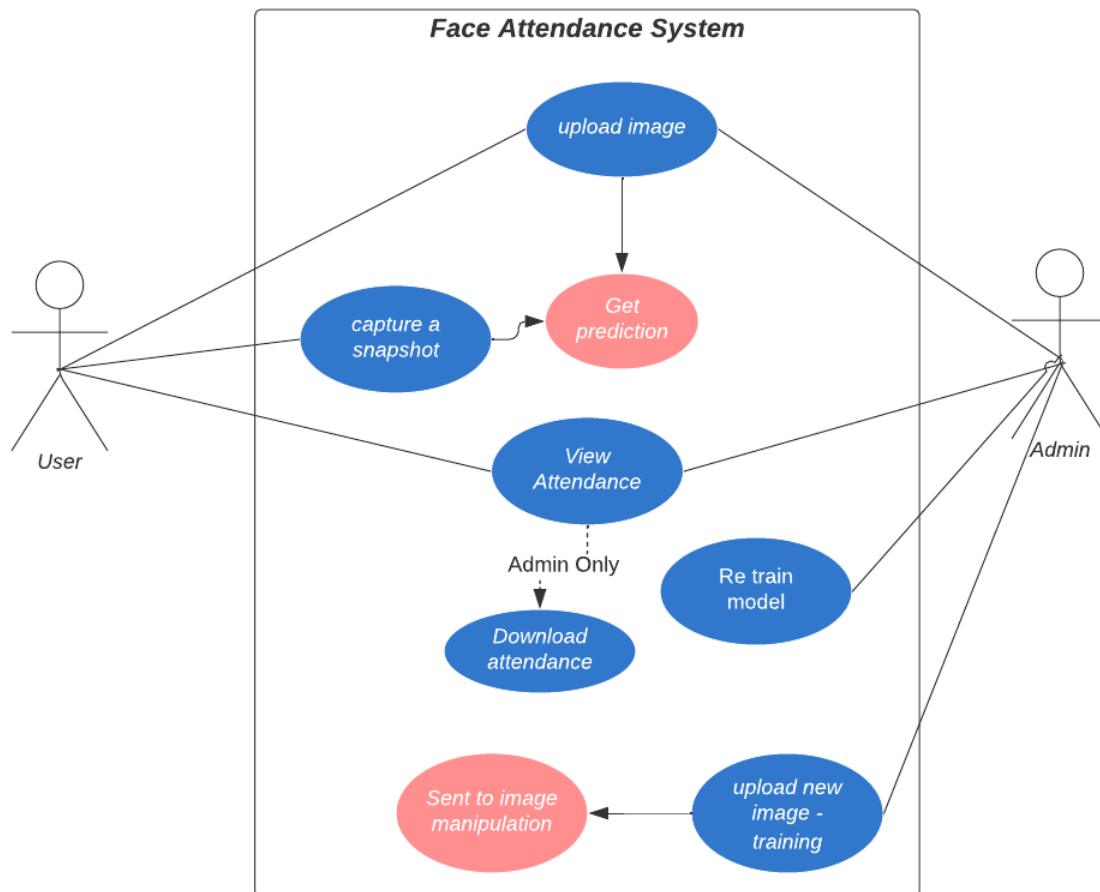


Figure 14 use case diagram

In figure 15 is the basic system model of how the model will work connected to a database. This model does not contain the front-end application. This is because the front end is described and shown in the design chapter. Below is what will happen when a person uploads or takes a snapshot of their face for the model and it successfully sends to the user.

If this fails, the user will be alerted that the file was not uploaded correctly. If the prediction itself fails as the user can be an unknown there is message sent out telling the user to add themselves to the system.

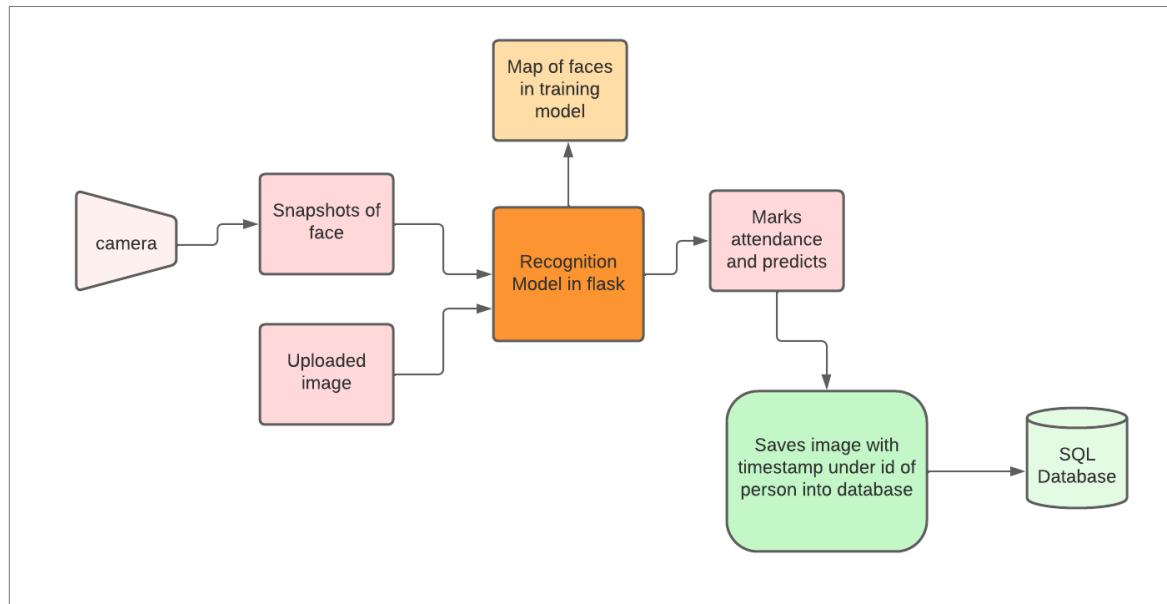


Figure 15 System Model

2.4 Feasibility Study

The main issues in terms of the feasibility of this project are; If the technologies work in the way they are meant to, and participants/users may have security concerns. The technologies in the following section will be used in this application, to create the desired application. The survey completed during requirements gathering highlighted additional security concerns within the project, and not all of these concerns can be effectively addressed within the scope of this project

2.4.1 Technologies

The technologies chosen for this project are Python, Flask and SQL. The chosen language was Python because it is very popular for the creating machine learning models and due to this has a lot of support and well-developed libraries to aid in the production of this application:

Python

The entire project will be coded using the python coding language. This was chosen as machine learning models are made very often using python. It offers readable and concise code allowing developers to concentrate on creating a model instead of needed to focus on the technical

issues with the language. It has also got a lot of libraries that are very helpful in creating very accurate and fast prediction models. Including Pandas, NumPy and scikit-learn. Using python folders can be created and files deleted easily within a method by importing the os module. This is useful as the participants of the survey mentioned they wouldn't want images saved. This gives the ability to delete the images once they have made a prediction. TensorFlow and Keras, Open CV and Flask will be the main modules used within Python.

Tensor flow or Keras

Keras is a deep learning API that runs on top of TensorFlow, which is a machine learning platform. These are used together to create API's for building and testing models like neural network models. Keras has a module for creating a sequential model which will be useful in creating the facial recognition model.

Open CV

Open-Source Computer Vision Library (OpenCV) is a computer vision and machine learning software library which can perform task such as face detection and image/video processing. It was developed by Intel and can be used in both Python and C++. It is used to process images and can also be used with the webcam in the browser. OpenCV can also save images to a specific location and edit the size of images so they fit the specific model.

Flask

Flask is a python web framework and using this framework the developer will make the API for the python model. It was created by Armin Ronacher, who runs an international organization of known as Pocco. The Flask template engine is built on the Werkzeug WSGI (Web Server Gateway Interface) toolkit and the Jinja2 template engine.

The WSGI is a standard for Python web application development and is a standard for developing interfaces between the web server and the web applications. Werkzeug is a WSGI toolkit, which implements requests, response objects, and other utility functions. This allows you to construct a web framework on top of it.

Flask has many extensions that can be imported to support various software. One that will be used within this application is an extension with will support SQLAlchemy this allows to use of

an ORM within the application and connect to an SQL database. During development, we will use a SQLite database and it will port over to SQL for production.

SQL and SQLite

SQLite will be used as the development database and SQL will be used at production level as SQLite is not supported at production level. SQL is a query language that is used with databases. This can work well within Flask and therefore will be used for this project.

2.4.2 The Benefits and Issues of the Application

For the feasibility of this application it's important to look at the benefits and the issues that are within it. These could affect the creation of the project or the ability to implement all necessary components in the time given. In this specific project the issues are more prevalent than the benefits as this is an invasive type of project. This was shown in the survey as people don't see the need or don't trust the application to be used in an ethical way.

The benefits of this is that the application can be used to speed up taking attendance and there would be no need for manual attendance. It removes the human error from taking manual attendance. This system can also help with being able to know who arrives exactly on time as timestamps can be checked.

In the period given and the datasets available may not allow for an accurate model to be created. As accurate models become more accurate over time with extensive testing and usage. There are ways to limit this issue but as there is a time constraint it depends on how well these solutions work. Another issue with datasets is the diversity of faces within them as you need a diverse range of expressions and people from different areas to guarantee the model works for everyone. A continuation from this if people will allow their faces to be used in the application as training images, this raises security and privacy concerns. Images used for training don't tend to be deleted as training is done continuously and the best way to improve training is to increase the image set size.

Security concerns that were raised in the survey may make it hard to use a webcam or live feed for this project. A live feed would be used down the line for proposed application and capturing of the face for the current planned prototype. This issue is large and nuanced and may cause discomfort around students.

One of the biggest issues that could affect feasibility and the ability create this application is that the front-end user interface won't connect properly to the model to show the user the necessary information i.e. the attendance and allow the user to download attendance report.

2.5 Project Plan

The development of this project was done using agile. This methodology is non-linear and allows for an iterative approach to creating a project. Instead of having to have all the design done in one iteration it can be changed while the project is being implemented. This suits the type of project being created as face recognition models need to be constantly trained and tweaked. Using Microsoft Planner, a digital Kanban was used which would be accessible to the supervisor at any time. This consisted of five columns To Do, Doing, Complete, Postponed and Stopped (No longer necessary). Each column holds a list of cards which represents a task that will need to be completed. Each task can be labelled with a due date and an importance. A document or link can be attached to these to, to be able to see the progress and for the supervisor to look at. Once a task on Microsoft Planner is completed it can be marked done and then it is added to the completed section.

The project will be Split into the following phases:

1. Understanding and comparing CNN models
2. Chosen model training
3. Connecting model to Flask application
4. Connecting database to Flask
5. Creating an attendance application
6. Finetuning model
7. Final application in Flask
8. Functionality and user testing

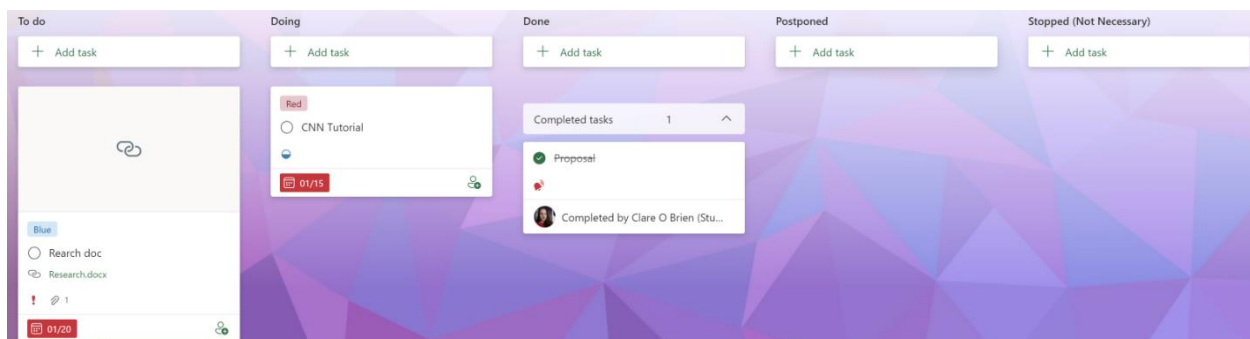


Figure 16 Microsoft planner

As Microsoft Planner allowed for due dates to be added and has its own interface for collating all the files and seeing when items are due it was a handy tool for everything to be accessible in one place. This was also handy as all of the documents were available through word online so they could be stored easily here. Along with this the developer used a journal on OneNote to keep notes while researching and coding as it can be kept in an organised manner with different sections and pages. Having all deadlines and tasks organised allowed for smooth project management

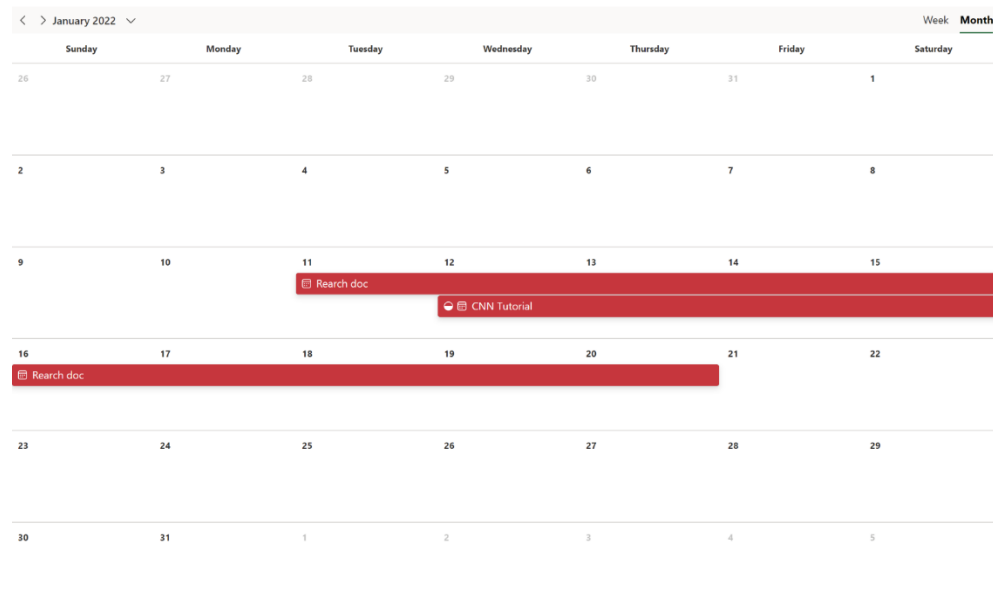


Figure 17 Calendar in Microsoft planner

2.6 Test Plan

The testing of this application will have multiple iterations as accuracy of this model is a necessity. These iterations will be to retrain the model continuously through the creation of the application. Therefore, the developer is planning to do some testing within as many sprints as possible. The preliminary model test layout is listed as follows:

1. For testing this application, the developer will need to collate images from a set of people to test on.
2. An image will be uploaded or captured within the application.

3. Timestamps will be printed out on the console and recorded to test how long it takes the recogniser to identify the face and insert it into the database.
4. The accuracy will be checked by having people that aren't in the database use the application and see if it correctly identifies them as a new/ unknown person.
5. After each round of evaluating the accuracy will be recorded and the model and initial dataset revisited.

There will also be a failure test to check what happens if a person captures an image containing no face to if the uploaded image doesn't contain a face. This will happen similarly to the layout above but instead of a prediction being given a message is printed saying that the image is not a face and will be deleted.

After these tests user testing will be run on the application, this will be to see if users like the layout of the site and if it is user friendly. In this testing there will also be a survey on what the users think of facial recognition models and if they would use this application.

2.7 Requirements Summary

This chapter provided the developer with a better understanding of our target market and what features are the best to implement so this model and application will function together. It also helped determine the functional and non-functional requirements for the web application. This requirements chapter has aided in defining exactly what needed to be implemented in the project.

The study of similar models and applications gave us a view on what features were used and what we needed to do to create our application. The survey aided in gaining an understanding of the users feelings towards this application. In the survey the mention of security and privacy was quite prevalent so that will be looked into while implementing the project and the feasibility section shows what will be used and the issues that may arise during development.

3 Research

The focus of this chapter is to investigate machine learning and deep learning and how it works with face recognition. From this analysis a lot of knowledge on how the inner working of these methods should be gained. There will be in depth analysis of machine learning types, there are three main types known as supervised, unsupervised and reinforcement learning. Once this is complete neural networks and deep learning will be investigated and why they are used more than machine learning methods in face recognition. Then how attendance systems can be created using these methods will be investigated followed by research into a popular library for creating deep learning models in Python.

3.1 Machine Learning

Machine learning can be used in many applications and its growing popularity is evident as a lot of applications and companies are making use of machine learning technology. Machine learning is a field within AI. It's known for giving computers the ability to learn without explicitly being programmed. There are three main subcategories of machine learning: supervised learning, unsupervised learning, and reinforcement learning.

3.1.1 Supervised learning

Supervised machine learning requires using labelled datasets to accurately train the machine learning algorithms to predict outcomes. It is also used to classify data accurately. Many companies use this as it is a very accurate form of machine learning.

This dataset used will include the inputs and their corresponding outputs so that the model can learn which the correct answers are over time. This is done through training and testing the model; the dataset is split into a training and test set. The model can then be trained on a certain amount of the dataset and tested on the test set. The more extensive and varied a dataset is, the better it is for supervised learning. A small dataset may only lead to accuracy for specific inputs and a significant error for others. The loss function is used to measure the accuracy of the chosen algorithm, adjusting until the error has been reduced as much as possible (IBM Cloud Education, 2020). A loss function measures how far an estimated value is from its actual value. The two main types of supervised learning problems are known as classification and regression.

Classification

This type uses an algorithm to classify test data into specific categories. It recognizes specific units within the dataset and attempts to draw some conclusions on how those units should be labelled or defined. Decision trees and support vector machines (SVM) are popular classifiers that are known for creating accurate classification algorithms.

An SVM is used in classification problems. In this algorithm each data item is plotted in n -dimensional space, ' n ' being the number of features. Each feature corresponds to the value of a coordinate. Classification is performed by finding the hyper-plane that clearly classifies the data items. This can be seen in the below figure. Hyperplanes are decision boundaries that aid in classification. The dimension of the hyperplane can change depending on the number of features. The figure below shows if the number of inputted features is 2 but if there were 3 features then it would become a two-dimensional plane instead of a line.

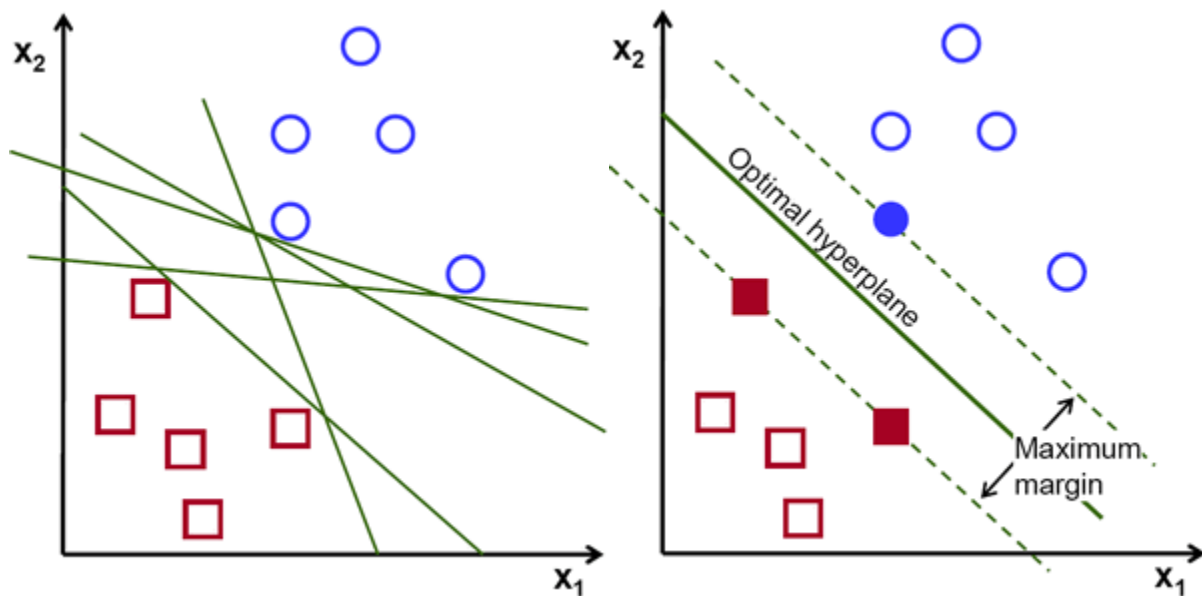


Figure 18 Possible hyperplanes (Gandhi, 2018).

The aim is to find the hyperplane with the biggest margin between each set of data points or features. The larger the margin allows for future features to be classified with high confidence. If there is only a small margin between the two classes, the model may not be very accurate. The support vectors within this model are the data items that fall close to the hyperplane, and they will be used to find the optimal margin for the classifier as shown in figure 19.

Another type of classification is a decision tree which looks like a flowchart-type tree structure. A decision tree is made up of nodes, the root node is the topmost node in the decision tree. It learns to partition on the basis of the attribute value. For example, if the decision node states ' $I = 10$ ', the two leaf nodes extending from this will represent yes and no and if in this case ' $I = 10$ ', the tree will partition at that leaf node. A decision will continue doing recursive partitioning which helps decision making.

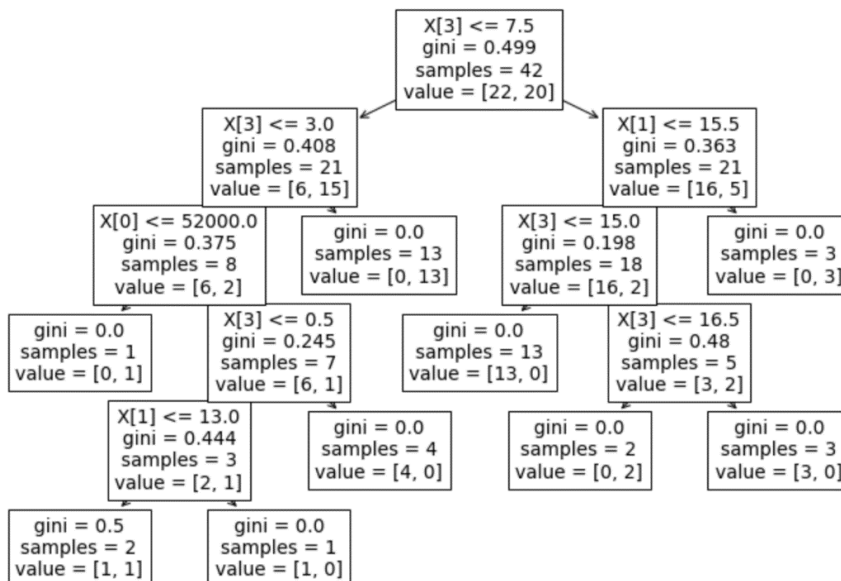


Figure 19 Example of decision tree

Above is an example of what a decision tree looks like when it is making decisions this was made using a plotting library in python and the decision tree is deciding whether to buy a pizza from one company or another. This was done using a pre-trained model that the developer made.

Regression

Regression models are used statistical method for determining the relationship between dependent and independent variables. It is widely used to produce sales and stock forecasts, such as those for a company's sales revenue. Popular regression algorithms include linear regression and logistical regression.

Linear regression is extremely popular and widely used to create regression algorithms and is also one of the simplest regression algorithms. Due to this they have many real-world applications in predicting house prices or future prices of stock in the stock market.

For example, let's presume that house size is the only determinant of house price. If we were to plot house size which is the 'predictor' variable or input as a function of house price (the outcome or output variable). From the assumption made there may be a linear relationship as shown in figure 20.

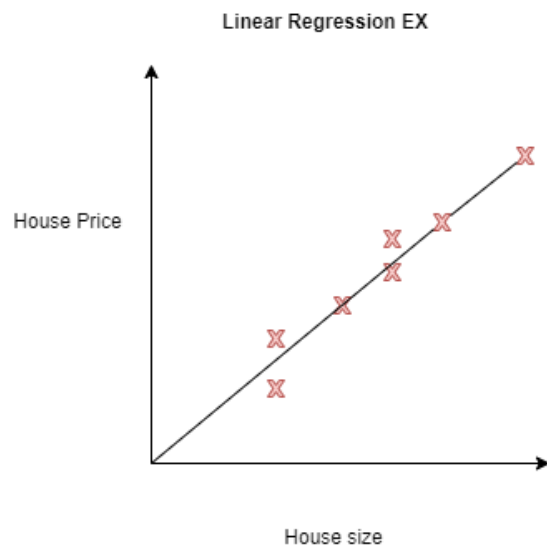


Figure 20 Linear regression Example - line graph

3.1.2 Unsupervised learning

For unsupervised models, the research must have at hand a dataset with some observations or pre trained data without the requirement of getting the labels/classes of the observations. Unsupervised learning studies how systems can get a function to explain a hidden structure from unlabelled data. The system doesn't predict the correct output, but instead, it explores the info and may draw inferences from datasets to explain hidden structures from unlabelled data. Unsupervised models can be grouped into clustering and association models.

Clustering

A clustering problem is where you wish to unveil the inherent groupings within the data, like grouping animals supporting some similar characteristics/features e.g., number of legs.

K-Means finds groups in data and the number of groups is represented by K. It is an iterative procedure where each item of data is assigned to a group based on the feature similarity. The way the algorithm starts is with an estimate of K centroids, these are randomly selected items from the dataset. The algorithm then iterates between assigning data items and updating the centroids. The graph in figure 21 shows simple clustering used in K means, each black dot represents a centroid, these centroids and clusters will change according to the testing.

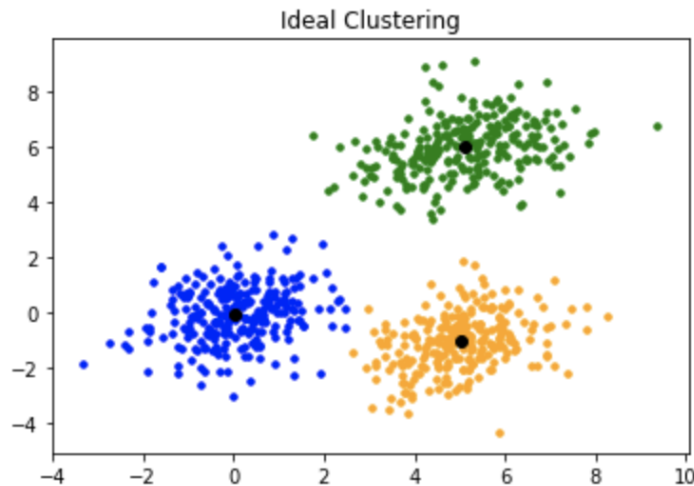


Figure 21 Graph of K-means clustering <https://media.geeksforgeeks.org/wp-content/uploads/20190812011831/Screenshot-2019-08-12-at-1.09.42-AM.png>

Each item of data is assigned to its closet centroid based on Euclidean distance(the length of a line segment between two points). The Cosine method can also be used for this algorithm which is measured by getting the cosine of the angle between two vector and seeing if they point in a relatively similar direction. Whichever of these give the best results will be used, and this can depend on the model or data itself. Centroids are updated by taking the mean of all items of data assigned to a particular cluster.

The elbow method is often used to calculate the value of K, shown in figure 22. In this method K-means clustering will be run for a range of values, depending on what seems best for your dataset e.g., K=1 to 10. Then the Sum of Squared Error (SSE) is calculated. This calculated as the mean distance between the cluster centroid and data points in that cluster. SSE is calculated as the mean distance between data points and their cluster centroid.

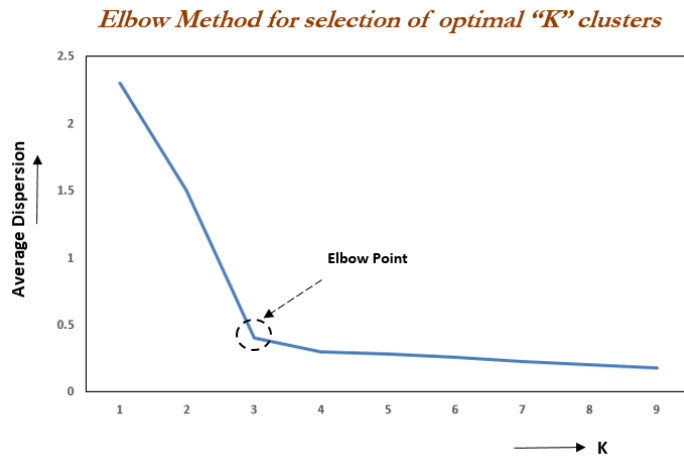


Figure 22 elbow function shown in line graph (The elbow method, 2022)

Association

An association rule learning is where you wish to get association rules like people who buy X also tend to shop for Y.

PCA as mentioned in a previous section can be used with Eigenfaces for feature extraction by using a dimensionality reduction technique. Within this technique, it tries to preserve the essential parts which tend to have more variation and remove the non-essential parts with fewer variations. The dimensions are features that represent the data, each pixel in an image are the dimensions or features that altogether represent an image.

There can also be semi- supervised machine learning algorithms, these just use a mixture of labelled and unlabelled data for training.

3.1.3 Reinforcement learning

Reinforcement machine learning is about giving an 'award' to the algorithm when it makes a correct prediction. These also use estimated errors if they large as a penalty.

The most important aspects of reinforcement learning are trial error search and delayed reward. This model family enables the automatic determination of the optimal behaviour inside a certain scenario in order to optimize the intended performance. For the model to learn which behaviour is optimal, it requires reward feedback, which is referred to as "the reinforcement signal."

Overall, each method can be highly effective while used within a face recognition system. Although through research it is known that supervised methods are the most accurate especially with a smaller dataset. Due to this the developer decided to use a supervised method for this application. The next section goes into detail about this method which is neural networks and then will go on to explain convolutional neural networks as that is the specific model to be used in this application.

3.2 Neural Networks

Neural networks and more specifically, artificial neural networks (ANNs) mimic the human brain through a set of algorithms. A neural network is made up of four components: inputs, weights, a bias or threshold, and an output. Like linear regression, which is mentioned above as a supervised learning method that's similar to neural networks, the algebraic formula would look something like this:

$$\sum_{i=1}^m w_i x_i + bias = w_1 x_1 + w_2 x_2 + w_3 x_3 + bias$$

Figure 23 Neural network algorithm

Neural network models are multiple linear regression models. Their algorithms are very similar too and the main change is that most neural networks have at least one hidden layer, but linear regression only has its single layer so therefore the algorithm is less complex. This does mean that linear regression is more useful for models that don't need any extra computation and only need the inputs and outputs like the example to the left of figure 24. To show how similar they are they are drawn beside the other in figure 24. Weights can also be added to regression models as different factors may have different importance in the example above on one factor was used so no weights were needed.

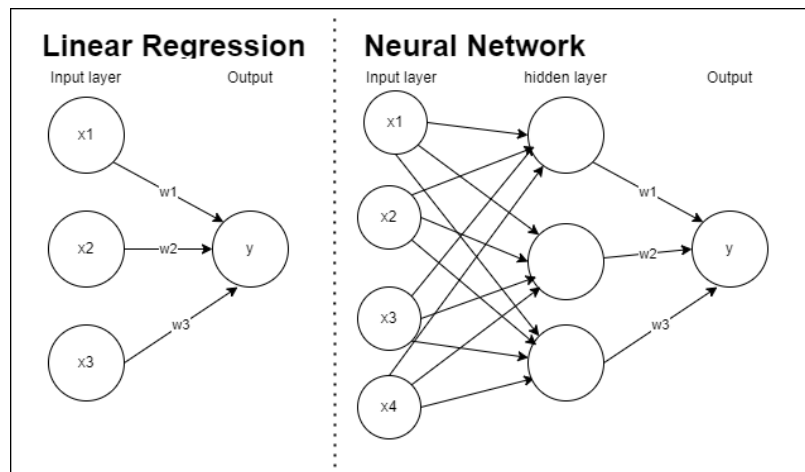


Figure 24 Linear regression vs neural network

3.2.1 Activation Functions

Within each hidden layer of a neural network there is an activation function, these can also be used in classification models that were described in the previous section. Though they are most known to be used in neural networks. An activation function is used to get the output of a node and is also known as a transfer function. It determines whether the output is a yes or a no and then maps the values between 0 and 1. There are linear activation function and non-linear activation functions. Linear activation functions don't add complexity and is similar to the linear regression model. Non-linear functions are the most popular as they help the model to adapt and improve and differentiate between the output. The activation functions that will be looked at are Sigmoid, SoftMax and ReLU. Multiple types of activation functions can be used within a neural network depending on the layer.

Sigmoid: This function simply takes a real value as input and outputs another value between 1 and 0. This is good for a classifier and has a smooth gradient. The output of the activation function is always going to be in range (0,1) compared to $(-\infty, \infty)$ of linear function. So we have our activations bound in a range. Sigmoid is used at the last layer of a neural network as it is good for predicting probabilities with two outcomes.

SoftMax: This activation function is also ,mainly used in the last layer of a neural network and is a more generalized logistic activation function for multi-class classification. Meaning that it can be used for solving a classification problem involving two or more classes.

ReLU: This function is mainly used for the hidden layers of a network and as it is less computationally expensive than sigmoid or SoftMax so can be used for more layers. It shouldn't be used for the output value as it can blow up the activation because every value under 0 becomes 0 and this is an issue because the range of ReLU is $[0, \infty]$.

3.2.2 Real World ANN Example

First you can pose a question like whether you should order take away for dinner. The predicted outcome will if you should or shouldn't order takeaway. For this example, assume that there are three main factors that influence your choice:

Saves time (Yes: 1; No: 0)

Have a healthy meal (Yes: 1; No: 0)

Saves money (Yes: 1; No: 0)

Then, let's give each of the above factors a value either 1 or 0 for simplicity purposes:

- $X1 = 1$
- $X2 = 0$
- $X3 = 0$

	Factors	1	0
X1	Saves time	yes	
X2	Have a healthy meal		no
X3	Saves money		no

This defines it as a perceptron which is a binary classifier. Most real-world issues are non-linear a binary classifier doesn't seem like the most useful. Therefore, values are required to reduce how much influence any single input can have on the outcome. This leads to the next step which is to add weights depending on how important each factor is in the overall equation as

one factor can be seen as more important than another. Below are the allocated weights based on the above factors:

- W1 = 5 – weight for X1
- W2 = 2 – weight for X2
- W3 = 3 – weight for X3

Finally, we will also assume a bias of -5 and therefore a threshold of 5. All of the values can now be inserted into the formula shown in figure 25.

$$\sum_{i=1}^m w_i x_i + bias = w_1 x_1 + w_2 x_2 + w_3 x_3 + bias$$

Figure 25 Neural network algorithm

Using the sigmoid activation function, the output can now be calculated and this will be our choice:

$$f(x) = \frac{1}{1 + e^{-(x)}}$$

Figure 26 Sigmoid Activation Function (Kishan Maladkar, 2018)

The answer is 2 and since it is above 0 that means we will order takeaway.

If the output of any particular node exceeds the given threshold value, that node is activated and begins transferring data to the network's next tier. Otherwise, no data is sent to the next network layer. As neural networks feature numerous "hidden" layers as part of deep learning algorithms, envision the above procedure being repeated multiple times for a single decision. Each hidden layer has its own activation function, which can potentially convey information from one layer to the next. Once all of the hidden layer outputs have been generated, they are used as inputs to calculate the neural network's final output. When it has hidden layers, it is a deep neural network which will be described in the next section.

3.3 Deep Learning

Deep learning is a subset of machine learning and works in a similar way to machine learning, but it has a few additional features. Machine learning models improve over time, but they still require supervision. Whereas deep learning models can detect if an algorithm can assess whether a prediction is correct using its own neural network, as was described in the previous section. The deep in deep learning refers to the depth of layers in a neural network, a neural network can be referred to as deep if it has more than three layers including the input layer and output layer. This is shown in figure 27.

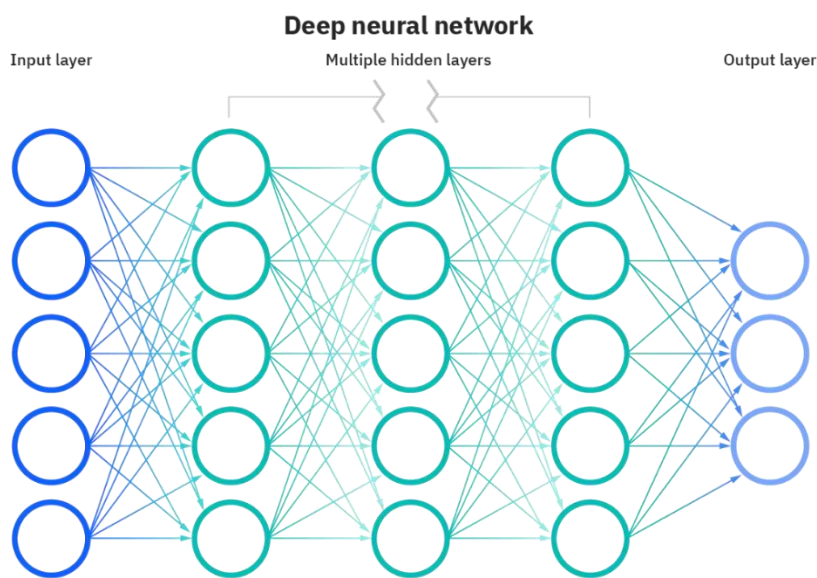


Figure 27 Deep neural network (IBM Cloud Education, 2020)

Most deep neural networks are feed-forward, this means that they flow only from input to output. However, you can also train your model through backpropagation; that is, move in opposite direction from output to input. Backpropagation is a training algorithm that has two steps first the values are fed forward and then the error is calculated, and it propagates (sends) the error back to the previous layers. Feedforward is part of back propagation, but it can also be used by itself. Having the error sent back and pushed forward again in batches will increase the accuracy overall.

As mentioned in the previous section neural networks are used in face recognition. The first convolutional neural networks were researched and used by Fukushima (1980). Fukushima took of approach of seeing if one could create a neural network model with the same ability of

pattern recognition as the human brain, this network was self-organised. Pattern recognition is very integral in many face-recognition models from the traditional to the deep learning methods.

3.4 Convolutional Neural Networks

As mentioned above CNN is a type of neural network model and is a supervised learning technique which is a technique that uses labelled data. Regarding deep learning this means it has a set of inputs and corresponding outputs $(x_t, y_t) \sim \rho$ (Alom et al., 2018). The neural network model's appeal stems in part from its remarkable achievements on face recognition in uncontrolled environments. (Guosheng Hu et al., 2015). Along with this CNN's are very powerful at analysing images and getting meaningful information from them with a very high efficiency. This is demonstrated by an example by Kulkarni and Harnoorakar (2020) that stated that the most basic of CNN models' accuracy is greater than 97%.

The basic CNN design has undergone numerous revisions, and other approaches have arisen over time. According to this author, the two main components of a CNN architecture are feature extraction, which is a convolutional layer that separates and identifies various features of an image, and a fully connected layer that uses the output of the convolution process to predict the class of the image based on the features extracted in previous stages (Gurucharan, 2020). CNNs are quite powerful for analysing and extracting useful information from images.

Typically, a CNN architecture looks like the image shown below in figure 28. This structure contains convolutional, pooling, activation functions (ReLU & SoftMax) and fully connected layers as detailed by Coskun et al. (2017). There can be multiple of each layer especially the convolutional layer and the pooling layer. As shown in figure 23 you can see multiple pooling and convolutional layers. Later in this section each layer will be briefly described followed by an example explaining how it all works together.

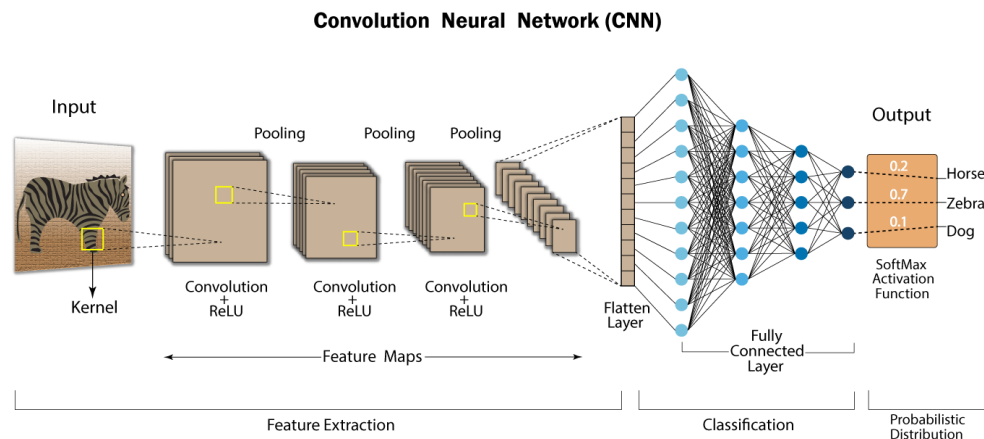


Figure 28 Basic Graph of CNN Architecture. (2019, August). [Illustration]. Research Gate.
https://www.researchgate.net/publication/335086346_Blind_Channel_Identification_Aided_Generalized_Automatic_Modulation_Recognition_Based_on_Deep_Learning

A computer doesn't see and almost automatically know what an image is like humans, it reads it as a collection of matrices. Depending on if the image is in colour it will be a 3D array with RGB channels ranging from 0 to 255 if we pass black and white image, then we will get a 2D array with binary values. This is demonstrated in the image below.

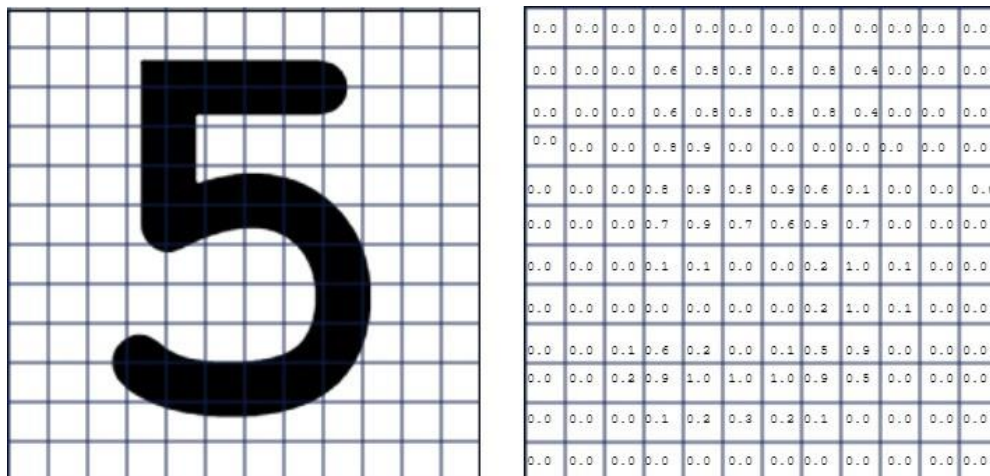


Figure 29 Image of a 5 with its binary map

The Convolution Layer

This layer handles most of the computational work according to M. Coşkun et al (2017). The convolution layer is used to extract features from the data that has been inputted, which happens to be a picture. Convolution maintains the spatial link between pixels by learning image attributes from small squares of the inputted image. A set of learnable neurons is used to

convolute the input image. As a result of this there will be an activation map or a feature map in the outputted picture/image, the next convolutional layer receives this image as input data.

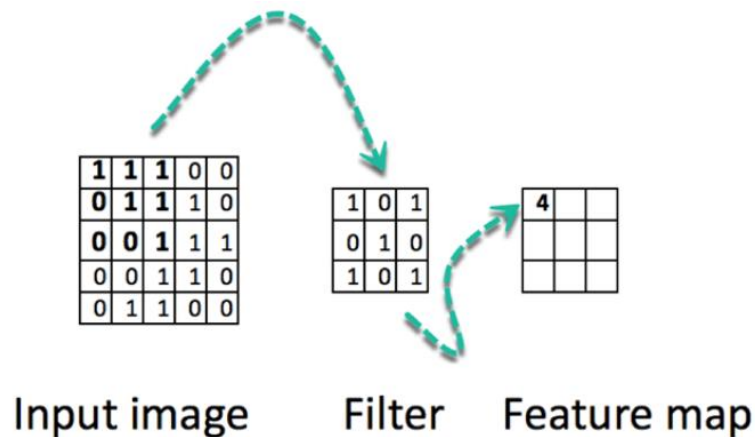


Figure 30 Convolution layer - feature map

Pooling Layer

In most architectures the pooling layer is the next layer after the convolutional layer. The main aim of this layer is to reduce the size of the feature map from the previous layer to lower the computational costs. It independently works on each feature map reducing the connections between layers (Gurucharan, 2020). Depending on the method, multiple types of Pooling operations that can be used. For instance, max pooling extracts only the maximum activation and average pooling weighs down the activation by combining the nonmaximal activations (Passricha & Aggarwal, 2019).

Activation Functions

The main feature of activation functions is mapping the input to the output in every kind of neural network. To compute the input value the weighted summation of the neuron input and its bias is used by creating the related output, the activation function can select whether to fire a neuron in response to a certain input (Alzubaidi et al., 2021).

One of the most common types of activation functions is ReLU. This activation function has been used in many applications of CNN, one of those being M. Coşkun et al (2017). Furthermore, it is a non-linear operation that includes units that use a rectifier. It is an element-wise operation, which means it is applied per pixel and replaces all negative values in the

feature map with zero. To understand how the ReLU works, we suppose that there is a neuron input supplied as x , and that the rectifier is defined as the following equation.

$$f(x)_{\text{ReLU}} = \max(0, x)$$

Fully Connected Layer

This layer specifies that all filters in the preceding layer are linked to all filters in the layer that follows. This layer is typically found at the end of any CNN architecture. Using the Fully Connected method, each neuron in this layer is connected to all neurons in the previous layer. This serves as the CNN classifier and employs the typical multiple-layer perceptron neural network approach. The previous pooling or convolutional layer provides input to the Fully Connected layer. This input is a vector that is formed after flattening the feature maps. (Alzubaidi et al., 2021).

As described above there are many layers that can be used within a CNN. There are also many reasons as to why you pick one layer over another layer in a certain position and in many neural networks have multiples of the same layer to maximise accuracy. The layers of a CNN are arranged so that first detect lines and curves and simple patterns and as the layers progress they can detect more complex patterns e.g. faces and objects. Below is an example of how these layers work together to detect a number. The section above described the layers in depth and the developer will use this to explain a network in easier language.

3.4.1 Example CNN

The way the CNN will work in this example is as follows:

- An image is inputted – in this case of the number '5'
- This image will be converted to an array
- A convolution layer.
- A max pooling layer.
- repeat both of the above layers
- Flatten the layers
- Fully Connected layers
- Get our output

So as was mentioned the image is inputted and converted to an array this is seen in the figure below.

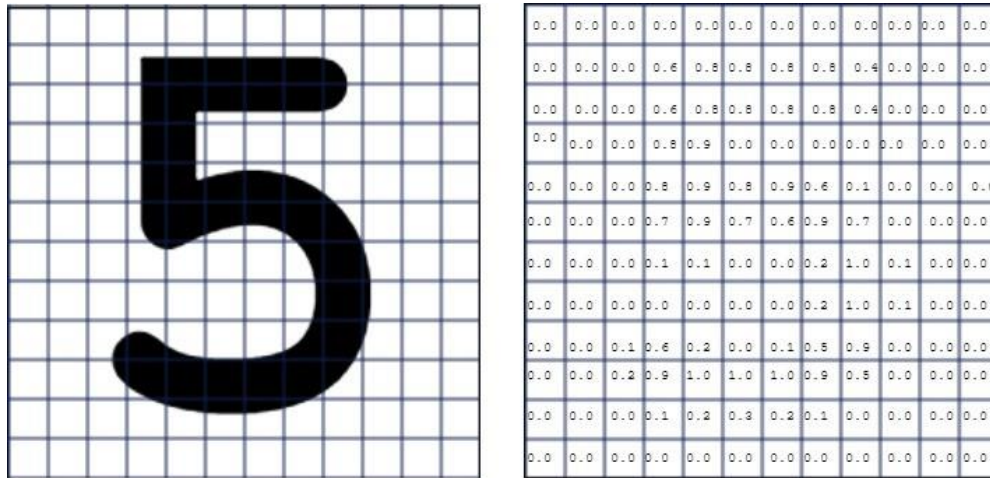


Figure 31 Image of a 5 with its binary map

Firstly, the convolution layer is needed to create an activation map to do this it performs a dot product between two matrices.

- Matrix 1 is a set of learnable parameters which will be referred to as the kernel.
- Matrix 2 is restricted portion of the receptive field

During what can be known as the forward pass the kernel (Matrix 1) slides across the height and width of the image. This will produce an image representation of the receptive region (Matrix 2). This will then become part of the activation map which is described prior. The kernel slides across the image in a way determined by the size, stride and padding allocated. This is described in figure 30.

$$n_{out} = \left\lfloor \frac{n_{in} + 2p - k}{s} \right\rfloor + 1$$

n_{in} : number of input features
 n_{out} : number of output features
 k : convolution kernel size
 p : convolution padding size
 s : convolution stride size

Figure 32 Convolution Layer formula

In figure 33 is the formula used in the convolution layer to calculate the number of output features in each dimension. The next diagram below is an overview of how the kernel and the current image create the activation map using the values in the kernel and images array map.

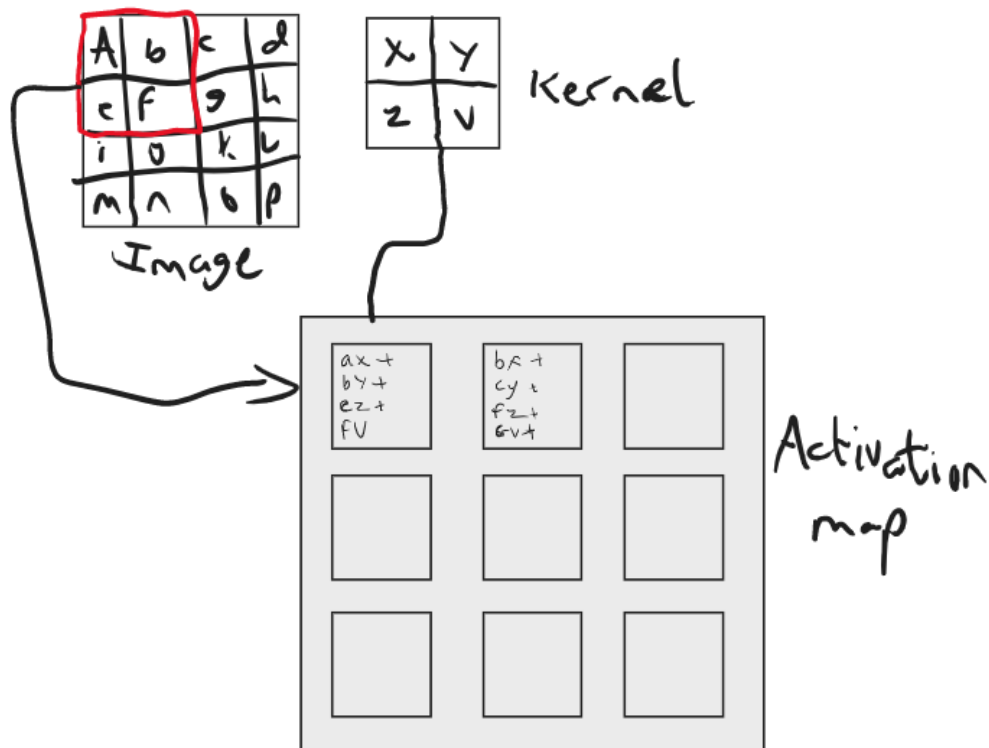


Figure 33 image to Activation Map

Second step from there is to send the activation map created in the first layer to the pooling layer. As mentioned above there is multiple types of pooling layers. The most popular being max pooling and that is the one that will be used in this example. Pooling layers replace the output with a derived summary of the nearby outputs this will help reduce spatial size which in turn reduces computational load. Pooling takes place in each slice of the activation layer separately.

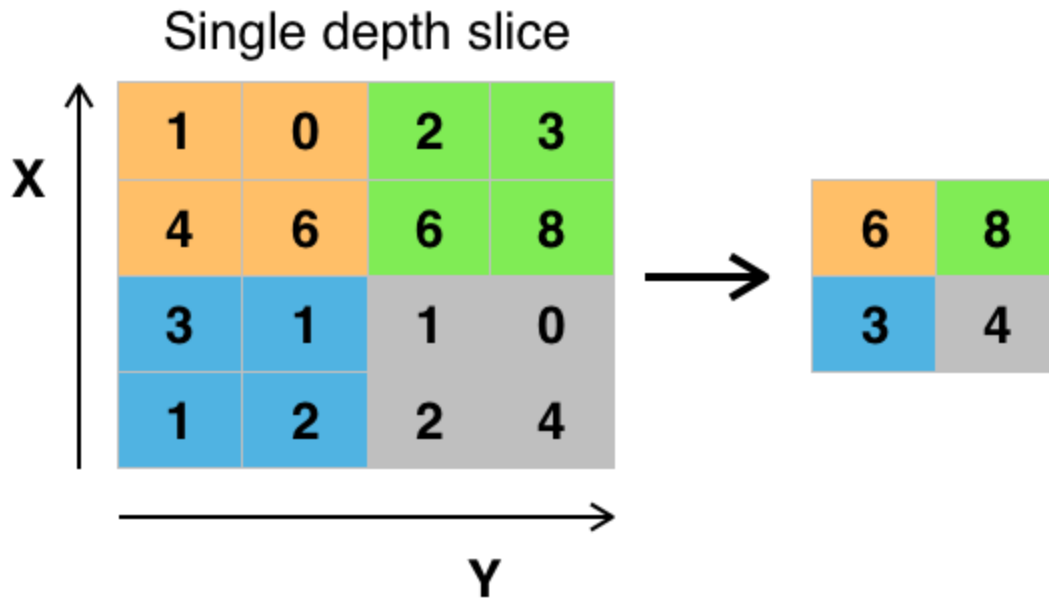


Figure 34 max pooling layer (DeepAI, 2019)

As can be seen above max pooling is taking place in a 2 x 2 square this can be defined and edited depending on what would work best for the image. pooling helps to make the representation become approximately invariant to small translations of the input. Invariance to translation means that if we translate the input by a small amount, the values of most of the pooled outputs do not change and thus the images integrity isn't lost.

In the original image of the CNN architecture in figure 28 two convolution and pooling layers are used, this is to get a small concise feature map for the fully connected layer and once again this can be tweaked, and even more layers added to gain the most accurate results. A thing to note however is the more layers you add the longer it may take to get the prediction.

As the fully connected layer can only read in 1 dimensional linear vector, the pooling layers 3-dimensional image must be flattened before being sent that layer. This is demonstrated in the figure 35 below. There is a 3x3 feature map of the image and this is flattened to a 9x1 layer which is then run through the fully connected layer.

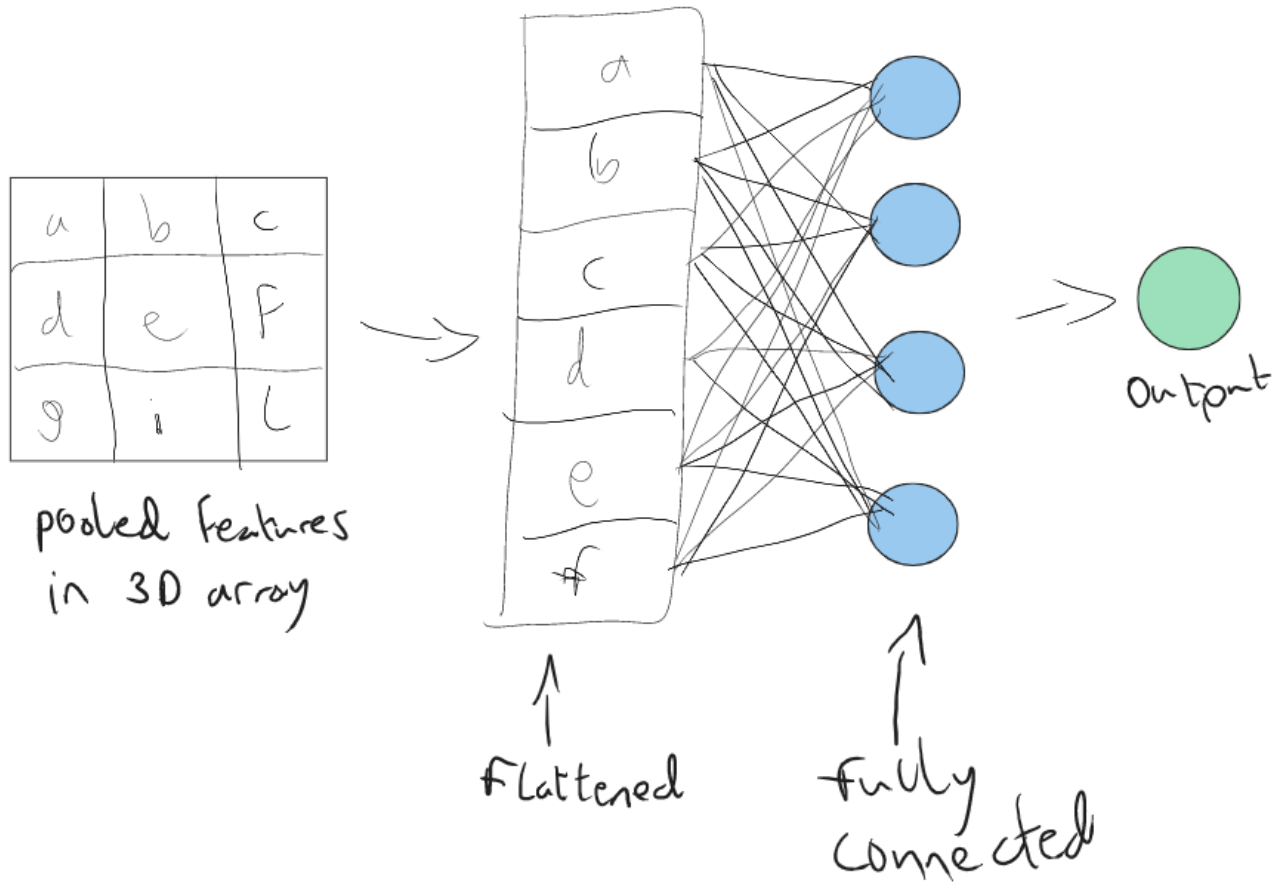


Figure 35 Flattening feature and getting output

The fully connected layer is a simple feed forward neural network. Generally, a CNN will have two or more fully connected layers the above example is just a simple version. A lot of networks can work well with one but having two can increase the accuracy of the network.

After this the SoftMax activation function is used and this will assign a probability to each class. The one with the highest probability is assumed to be the correct output. As this network started with an image of a 5 it should end with saying that the highest probability is that the image is a 5.

3.5 Face Recognition

Previously it has been explained how machine learning works and how CNN's are made. Now it is time to look at how this connects with face recognition and why it would be used by first explaining what face recognition is and the types that have existed over the years, after this the application of face recognition that use CNN will be looked at

3.6 Background

In 1964 and 1965, Bledsoe, Wolf, and Bisson were early adopters of automatic face recognition, working on utilizing a computer to recognize human faces. (Virgil Petrescu, 2019). Face recognition is a biometric method of identification, it differs from the more intrusive physiological methods i.e., placing an eye before a scanner or giving blood to verify DNA while also being highly accurate as stated by Lin (2000). During the seventies research and development started using specialised edge and contour detectors to locate parts of the face. These are also referred to as 'facial landmarks'. This method was discussed in comparison to the geometry-based approach which is faster and can be used in hybrid approaches (Daniyal, 2009).

Face recognition algorithm has the following main functions; a face image detector which finds the locations of human faces from a normal picture against simple or complex backgrounds and a face recognizer that determines who the person is. Each function has a feature extractor which is used to reduce the size of the dataset by creating new features from the existing features and a pattern recognizer. Both of which are very important in face recognition and these functions can be seen in many different face recognition techniques.

Types of face recognition

Facial recognition can be used for 2D or 3D images as faces are mainly having dimensions in 2D and 3D with different textures and facial expressions. All types of face recognition require the use of a machine learning model trained with either face locations or other methods as described below

Firstly, 2D face recognition was used and had four steps: detecting a face, face alignment, feature extraction, and feature matching from a database of existing users. There were various techniques used for the face detection including colour, intensity, and illumination. Furthermore, there were some limitations of 2D face recognition. In 2D face recognition system recognition rate and performance are dependent on image capture conditions like head orientation, image quality, lighting conditions, partial occlusion, facial expressions (Singh & Prasad, 2018).

The next progression of Face Recognition Technology was 2D-3D face recognition which was proposed by Andrea F. Abate et al. (2007) was a reliable technique for collective 2D visual

images and 3D model face recognition and was based on multiple parameters including recognition rate, number of addressed tasks and input size. Eigen faces (described below) and stereovision techniques were used to improve the performance of 2D face recognition system. Stereovision aided in this as it is the process of extracting 3D information from multiple 2D views of a scene, in which the face of a person at different positions was matched with the help of a scan lined-based neural network.

Eigen faces was introduced in 1991 by Turk and Pentland it uses dimensionality reduction and linear algebra to recognise faces. Eigen faces determines the variance in a collection of images of faces. The variances are then used to encode and decode a face without the full information which in turn reduces computation complexity. It is commonly used alongside principal component analysis (PCA) for feature extraction and recognition were effectively used for face recognition. This was introduced by Pearson first in 1901 and is a dimensionality reduction technique that use eigenvalues and eigenvectors to achieve this.

Sima Soltanpura et al. (2017) presented a survey based on local attributes for 3D face identification. Curves, key points, and surface were used to split the local descriptor. They used picture capturing techniques to compare different situations on a 3D face database. Generally, both face and head shape are represented as range data on a 3D dataset. An advantage of this approach is that the 3D model knows all the necessary information about the face geometry as stated by Abate et al. (2007). In addition to this, 3D face recognition also shows to be a further evolution of the 2D recognition problem, because a more accurate depiction of face features could lead to a stronger discriminating power. To show this a survey was undertaken by Zhou and Xiao (2018) on 3D recognition. They split 3D face recognition into three directions expression-invariant, occlusion-invariant and pose-invariant 3D face recognition. In this study the recognition method is done using a local method. According to the aforementioned survey, by using half face matching, a complete face model can be synthesized simply by using a statistical model to bypass the data missing issue.

3.7 Applications of Face Recognition Using CNN

Up to this point different methods of how deep learning CNN models work within face recognition and how the technology has developed over the years. There has been a wide variety of industries and companies using face recognition technology to aid their work, some of these include attendance systems, monitoring systems and criminal investigations. In

addition to this face recognition methods help save people time as it can scan the face without human intervention i.e., scanning a key card or fingerprint. Therefore, these systems can help different working professionals and the average in their day to day lives.

As mentioned, deep face recognition can be used to aid in criminal investigations for instance, as discussed in a review by Virgil Petrescu, in 2019. Within this review they mention how face recognition can help criminal investigations as these algorithms can provide additional information that a person is unable to decipher. These systems can save the faces of suspects, gangs, wanted criminals and those suspected of involvement in serious violent crimes.

Advances in face recognition technology can allow for CCTV security systems to be used to identify faces at a crime scene using 3D face recognition. In addition to this a recent study was carried to detect street crime using the VGG-19 CNN (Letchmunan et al., 2020), to see how accurate this system would be and how it would help in the prevention of street crime. The system ended up being 81% accurate and had 0.025 frames per second detection time.

3.8 TensorFlow and Keras

TensorFlow and Keras are two of the top deep learning libraries available in python. Both if these libraries can be used to create accurate and robust deep learning neural networks, that can in turn be used for face recognition. TensorFlow was developed by Google and is primarily used for deep learning applications. It is an end-to-end library that can be used on its own to make deep learning or machine learning models or it can be wrapped by a library that will simplify the process. One of these wrapper libraries is known as Keras and works seamlessly on TensorFlow.

TensorFlow was created by the Google Brain Team in November of 2015. It was creating to be primarily used in the python programming language, but there is C++ API that can also be accessed. TensorFlow is different from some other libraries as it was not only designed for use in deep learning systems but also for research in production systems. For example, it was used in the DeepDream project (get ref). How TensorFlow works is that it accepts data in the form of multi-dimensional arrays of higher dimensions referred to as Tensors. On these inputs a flowchart of operations can be performed using TensorFlow. As in many other systems the input goes in one end, and its output will be returned at the end.

As mentioned Keras is a machine learning library that's built on TensorFlow. Keras was created to be easy to extend, user friendly and to be compatible with Python. According to the Keras

As it was developed to focus on enabling fast experimentation. Keras is flexible and powerful and is used by big industries including NASA and YouTube as it provides an extremely high level of scalability and performance. All the different layers within a neural network like the convolution layers and the activation functions are all separate modules that can be combined to create new models. On top of this there's no need to have separate model configuration files and everything can be configured in the one file in Python code. Keras works with TensorFlow it can use CPU or GPU to run models, this allows for massive models to be trained with it using heavy duty GPUs.

3.9 Research Summary

This section went through how machine learning algorithms worked and where some would be used and how machine learning went on to be used as convolutional neural networks within face recognition.

It investigated how different models work and which were the most useful for this current scenario. It also went through and described in detail how neural networks work and why they have grown in popularity over the years. The examples show how these can be used in day to day life.

Supervised learning is the path that was chosen for this project as most CNN's are made through the supervised learning method, this is also currently the most accurate for smaller datasets. There is a clear progression from linear supervised learning to the non-linear models and understanding how all of this connects and how it progressed will help with choosing which model to use as each layer has been investigated and thoroughly researched. Once the CNN's were researched this chapter goes through a background of face recognition and how it has progressed and why CNN's became popular for it. Researching other applications and reading through research papers on different pretrained models gives insight into what to avoid and what to try to make a robust CNN model. This also shows that they are useful in attendance taking as they have become so accurate and so fast over the years.

4 Design

The following chapter will describe the functionality and system architecture of the face recognition system being developed. This system will be implemented into a web application to demonstrate its functionality. The application being developed for this project is a face recognition model and application. there are three main layers to this application. Firstly, there is a server layer which is the most integral for this application as it contains the recognition model.

Firstly, the chapter details the system architecture of the application. It shows how the different layers communicate with the other. This is followed by briefly explaining each of the technologies that will be used. Following this there will be an explanation of the inner workings of the model and sequence diagrams detailing how each function works within the application. After all the process design is demonstrated the user interface will be shown and how that will look and interact with the model.

4.1 System Architecture

This application will be created using the python Flask framework and using Jinja2 templates for the frontend. The recognition model will be written in Python using the Keras and TensorFlow Libraries and called within the flask application. To test against the existing faces in the training set a 'Result Map' of Faces will be used (as seen in the data layer). Requests will be sent through the templates to the Flask server and this will in turn call the model to send a prediction back to the user. Using the Flask-SQLAlchemy which is a SQL toolkit to allow the Flask server to connect to a SQL or SQLite database. The attendance will be stored in a table within the database. The system architecture can be seen in figure X.

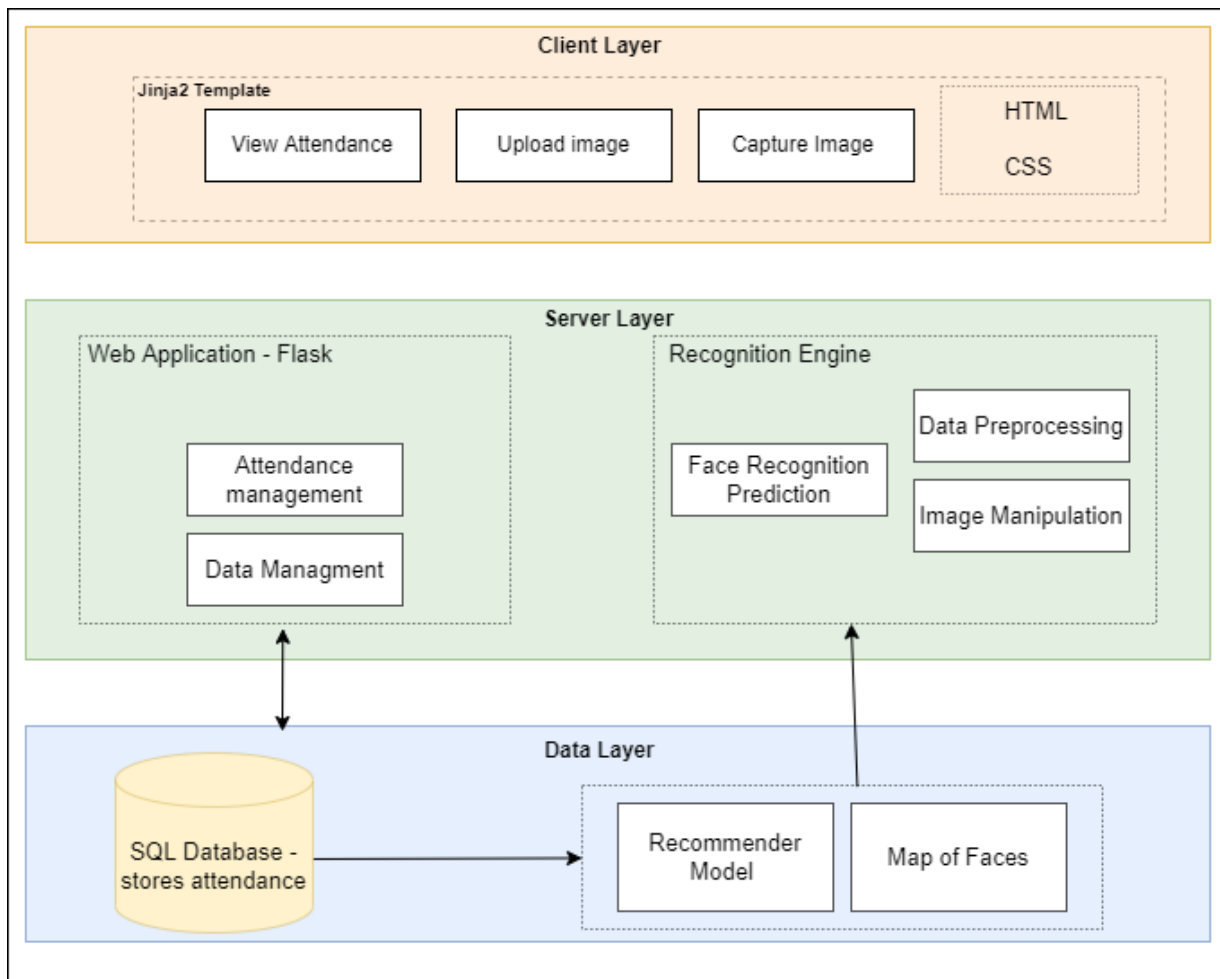


Figure 36 System Architecture

4.2 Sequence Diagram

The sequence diagram in Fig X shows how the various layers will interact with the other. This includes how the user interacts with the client layer and how the client layer interacts with the server and recognition layer and also how the server connects with the database. When a user uploads an image of themselves it is posted to the server, which will call the recognition model. The recognition model will then run and return a prediction to the server. This prediction will be displayed to the user along with their image. Then the name that was predicted is added to the database along with a timestamp to the attendance table. This will have the user marked as present.

The user can also capture an image of their face using the webcam and this is then posted to the server similarly to the uploaded image and the model predicts who it is and displays it to

the user. This feature can also be used to add new images to the training set if the user is new or just to add to existing users as to make the training set more robust. These images are also saved to the database.

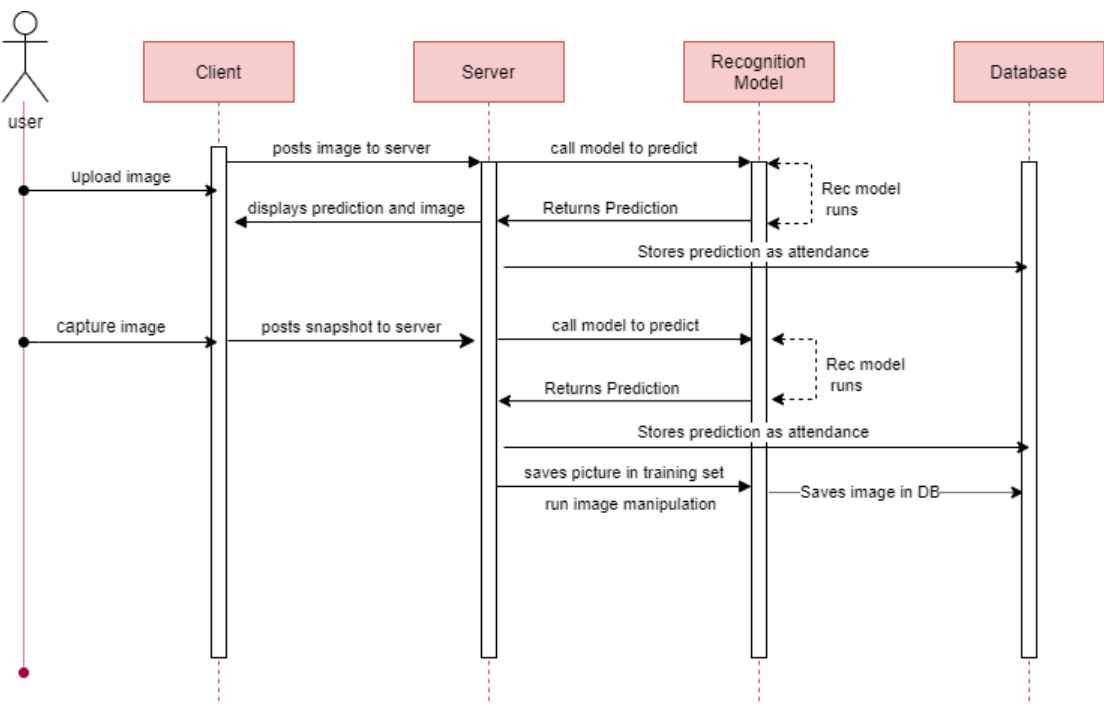


Figure 37 Capture Image

4.3 Database Design

The database will have two tables. One to store the attendance and another for storing images. One to store the faces and one to store the images or snapshots in the database. This will be used for training and testing the model as well as detecting if an inputted image of a face matches a person in the database.

Img	
PK	<u>id int NOT NULL</u>
	img text nullable=True
	name text nullable=True
	mimetype text nullable=True

Attendance	
PK	<u>id int NOT NULL</u>
	name text nullable=True
	timestamp text nullable=True

4.4 Process Design

This design stage was used to figure out how each step would work together to reach a goal. The main process that needed to be designed was how an inputted image would be seen to the model and the various response to that once it was predicted. In fig 8 a flowchart is shown demonstrating how this will work with in the system this is then followed by pseudocode describing this feature. Another small diagram is shown also to demonstrate how an image is sent for training.

Model Recognising Inputted Image

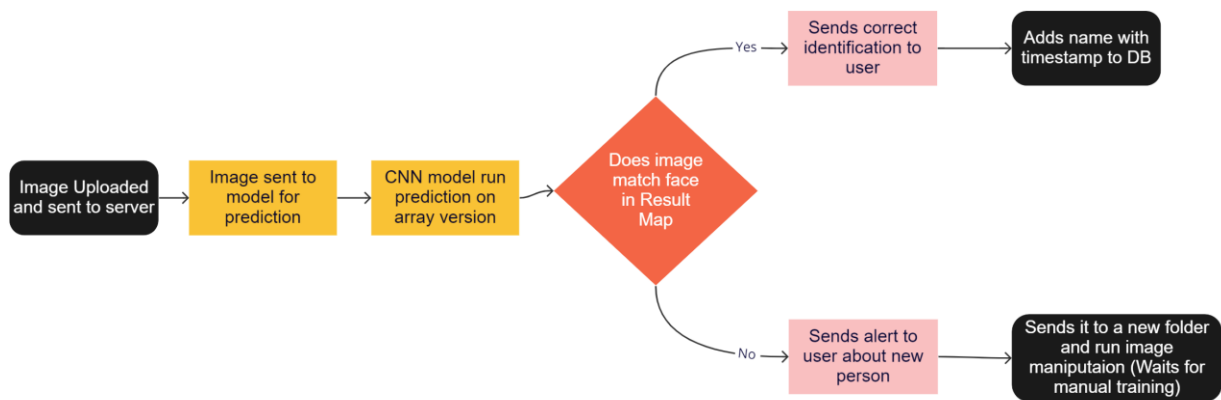


Figure 38 flow chart

1. In either insomnia or the frontend of the application you will upload an image.
2. This will be sent as a POST request to the API.
3. Once the API gets the image it will load it and set it to a certain size to fit the model. The image will be converted to a NumPy array.
4. The dimensions will be expanded to a 4-dimensional array.
5. Use the model to pre-process the array to prepare it for predictions.
6. Use a CNN model to predict the face against the faces in the current map of faces.
7. Output percentage chance it's a certain person in the database.
8. If it doesn't match any faces the face will be added to the training folder.
9. The predicted persons name will be added to the database along with the timestamp

Image captured for training

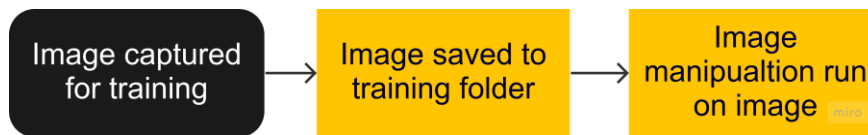


Figure 39 image manipulation example

This specific case is about when an image is captured for a user that isn't is being added to the site.

1. User takes image using web application
2. Image is sent to the server.
3. Image size is set and saved to the training folder in its own folder with a random name.
4. This image will then be identified by and admin and manipulation will be ran on it.
5. Either manually or within the server the image manipulation will be run.
6. The model will be retrained with the new images by the admin.

4.5 Model Design

The model that will be used in this application will be a convolution neural network (CNN). This was explained in the research chapter. It will take a lot of tweaking and editing to get the ideal model for this application as every model is different and what works for one model will not work for another. A larger dataset with more images in it will arguably be easier to train accurately than a small dataset. This will be a problem that will be addressed in the implementation of the model as the dataset has around 2000 images in it.

To help with understanding how these models work the developer plans to look at the MNIST and ImageNet datasets that are both accessible using Keras. ImageNet is a dataset of images from over 1000 categories and there were ImageNet challenges held to see which CNN was the most accurate for it each time. One year the VGG16 model won the challenge, this model will be looked at to gain understanding on how categorical CNN's work. The MNIST dataset is a dataset of images of digits and this will be used to help understand how Keras is used to create CNN's.

The different methods that the developer plans to look at are methods using the Keras and TensorFlow along with dlibs face_recognition library in Python. The steps in creating a recognition model are shown in figure 38.

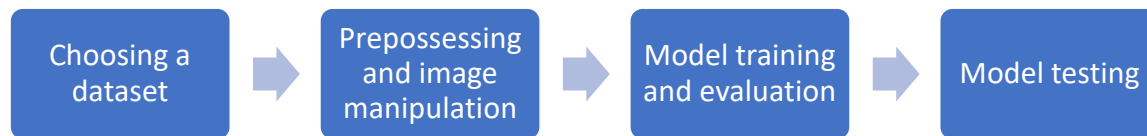


Figure 40 steps of creating recognition model

The dataset that is chosen is of random faces from multiple angles of people and celebrities on the internet. All these images are free to use for use in creating face recognition models. The developer also added images of themselves and friends to be used in the model. The images are separated into two folders, one for training and one for testing. Within each of these folders there are multiple folders with the name of each person that the model will be trained to recognise.

Next is to pre-process and manipulate the images. In this part images sizes will be changed to the same size and extra manipulations of each face will be added to each folder so that the dataset will be bigger. In this part each category will be labelled with a number and saved to a file, each person having a corresponding id and that will be used for training. In this part of the process OpenCV and Keras will be tested, and the best or most efficient method will be chosen.

For model training and evaluation part first the model needs to be created. This will either be designed using Keras sequential model which creates the model layer by layer and contains a module for each layer or OpenCV with dlib. Dlib is used within python to connect to the face_recognition library. This contains premade models that would be used instead of creating on line by line. An issue with this is there's less choice in how to make it where there is more options in Keras. Before the evaluation process the model is fit and compiled, then it is evaluated in epochs. An epoch refers to training all the data for one cycle, as in in one epoch all the data is used exactly once. Tweaking the amount of epochs can increase or decrease the accuracy, for a small dataset between 15 or 20 epochs should be enough to train the model to a high accuracy.

Once the model is evaluated the developer will have an idea of its accuracy and then it will be tested. Different methods of testing can be used. One of the methods that's in the Keras library is a simple `predict()` method this will accept an image and then predict who of the known faces it belongs to. Another method that will be tried is within the `face_recognition` library where two images are loaded, and it is tested whether both images are of the same person or not.

That is an overview of how the model is designed and in the following chapter these various methods will be tested, and a model chosen.

4.6 User Interface Design

This section describes the user interface design for the application. It will also go into what the application would look like with further developments also. The base features of the application are shown in the images and will be explained accordingly. These can also be viewed at Appendix B. There will be three main pages of the application along with a base page explaining what the application is.

The upload page will have two buttons to upload an image, one of which will show the user the image they uploaded with the name of the predicted person after upload. The other button is used for adding a new person and this is saved to the database. It will be specified that the filename must be the name of the person as that will be saved in the database alongside the image. In future, this would also have the option for the person to select or not the prediction is correct and to also select the correct person. If it doesn't predict that its anyone on the list the name will be left as unknown but still added to attendance.

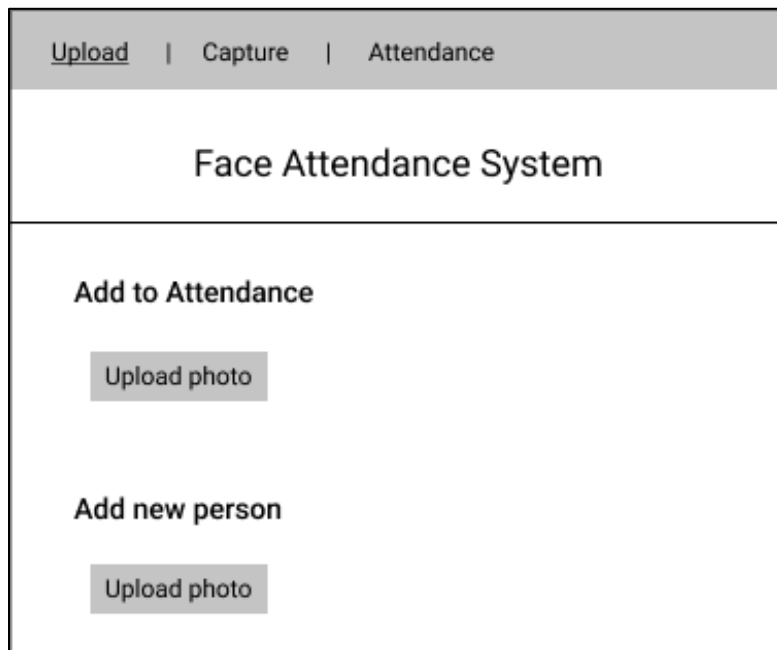


Figure 41 Upload image page

On the page shown in figure 40 there will be a feed from the webcam on a computer being shown. If there is no webcam on the computer, the webcam view won't show. There will also be two buttons one for basic capture which will send the image to the server to be uploaded and to add to person to the attendance. The other button is to start and stop the stream of video. The image that is captured will be sent for image manipulation and be added under a folder with the name person that it was predicted to be. This folder of new images will be stored in a folder outside the training folder to be added when the model is being retrained. The person that captured the image will still be added to the attendance.

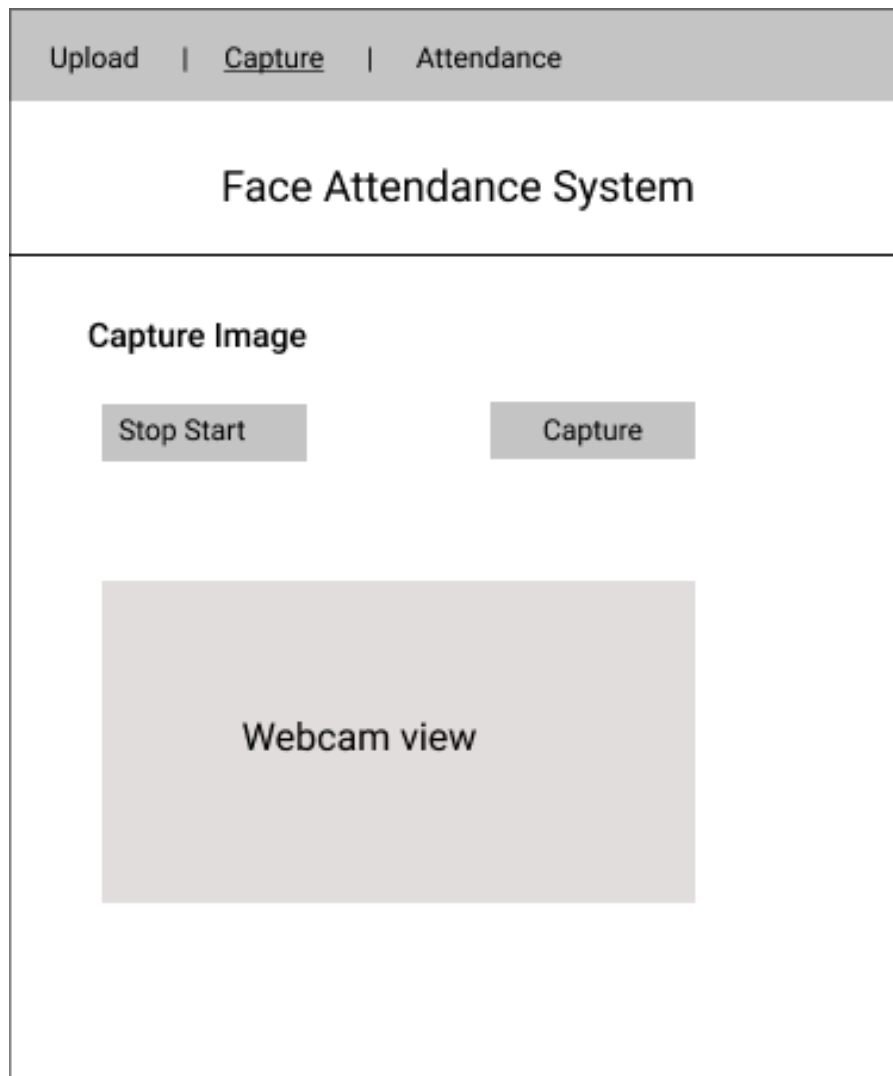


Figure 42 Capture image page

On the page shown in figure 41, the attendance list page is shown. This will show the current attendance for the specific day. This attendance is saved to a csv file along with into a database.

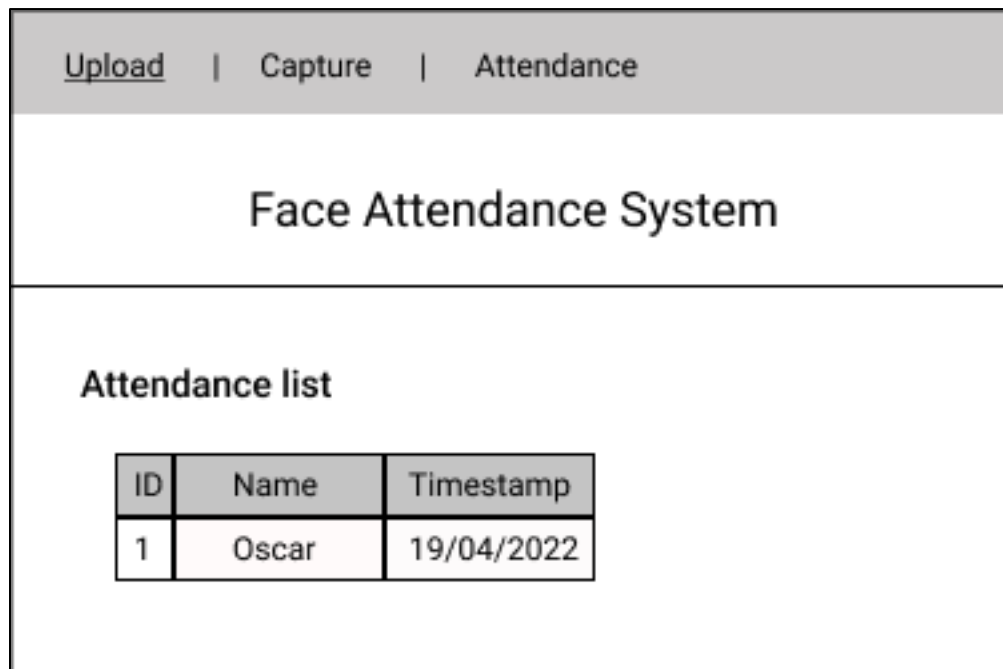


Figure 43 Attendance view page

4.7 Design Summary

Overall, this chapter has discussed how the application will be designed from end to end. This application aims to take attendance by use of a face recognition model. The system architecture described how each layer will communicate with the next and what technology and elements will be in each layer. Then there are sequence and flow diagrams showing how each request will be treated and the step-by-step response from the application.

After this the model design is described and the various options that will be looked into to choose the best one for this application, this also goes over how the dataset will be created. After this the database design is discussed using the ERD (entity-relationship diagram) to show how it will be structured. After this the user interface is discussed and shown via Figma created wireframes. This part explains what can be done on each page and the purpose of each feature.

5 Implementation

The following chapter details how the application and facial recognition model were chosen and implemented. This chapter goes through the implementation of this application. It details the various models tried and tested in the process of creating the face recognition model. Then how this model was also implemented into a web application using the Python Flask framework. While integrating these together a few test applications were created. Parts of these applications and scripts were then combined into the final face recognition application including the attendance feature.

Before delving into the implementation of the facial recognition model, the developer decided to take some time to analyse pre-existing image recognition models like VGG16. After some research, the developer decided the best way to start the implementation was to understand Keras using the MNIST dataset which is a dataset of handwritten digits from 1 to 9 which is used within machine learning and deep learning models. For the face recognition library, the developer decided to research OpenCV and Keras in Python. Using the knowledge gained with these tutorials a roadmap was created by the developer. The roadmap is shown in figure 42 and 43. This roadmap is an updated version of the project plan. It has been altered slightly to combine some of the phases as they have crossover.

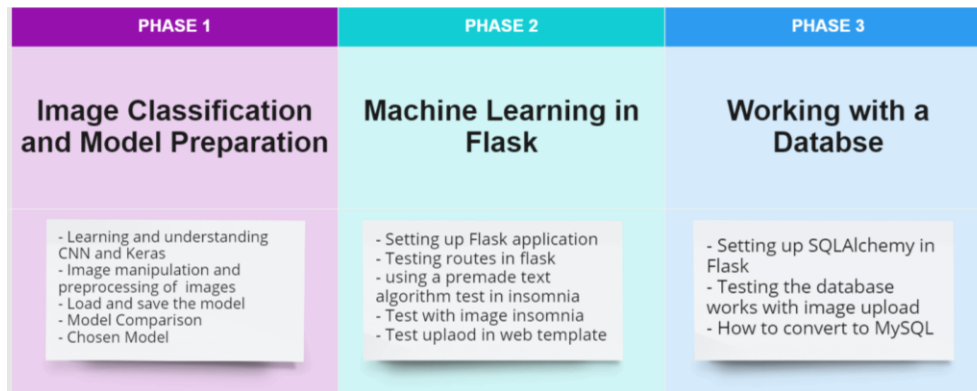


Figure 44 Phase 1-3

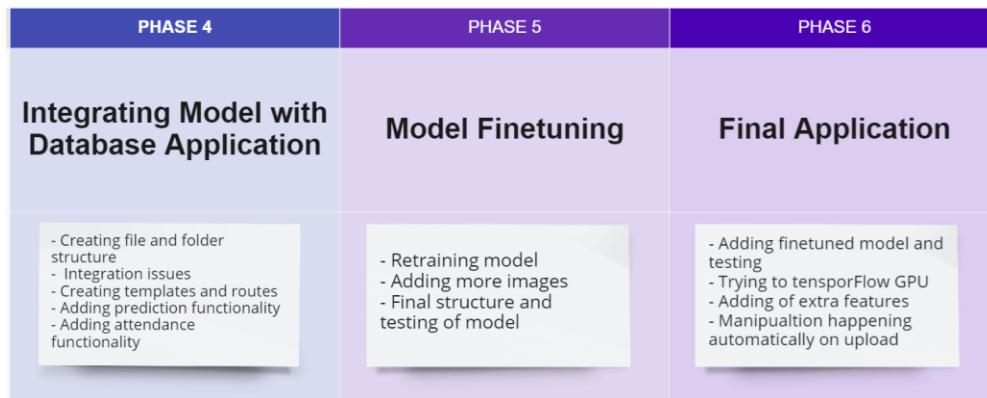


Figure 45 Phase 4-6

5.1 Development environment

To develop this project GitHub was used for source control management and Anaconda which is a distribution of Python packages was used as the development environment for the project. GitHub is useful for version control which is necessary when working in groups or using multiple machines for one project. The developer for this project used GitHub as they were working from home and college to make the project. It was useful for using in multiple places as its easy to push and pull code from a git environment. Anaconda is used mainly for data science. It allows for simple package management and deployment. It comes with over 250 packages automatically installed. Multiple environments can be set up using the different versions of python which can allow for easy management and reduced the amount of conflicts. Anaconda is also useful for working on multiple devices as environments can be exported and imported easily using the conda cmd within Anaconda. The developer installed Jupyter and VS Code to use within Anaconda. Jupyter is a notebook interface used for coding in Python and other languages. Within this environment visualisations can be created along with live code and questions. Visual Studio Code is an IDE and was mainly used for creating the flask application as it allows for coding full scripts easier and connecting to a database is easier within an IDE. Whereas Jupyter was used for creating the CNN model and the data pre-processing methods as it was easy to visualise how the model was being trained within Jupyter. There is also an extension to view Jupyter Notebooks in VS Code so using these alongside each other allowed for a smooth process.

5.2 Image Classification and Model Preparation

The goal of this was to learn how to create a CNN in Python using various libraries and to choose which of these to use in my application. This section will also analyse and explain how these models work and why one would be used over the other. Through this section, the developer analysed the VGG16 network which is a pre-trained network that was used with ImageNet. Another goal of this section is to show how to create an accurate model while only using a small training set using image manipulation. Also, this section will explain how to save and load a model from different libraries.

Pre-processing is a very common term to hear within the creation of neural networks and even machine learning models. It's important to know why this is used and how pre-processing helps in the creation and implementation of models. In image processing it is required to clean the data before it is inputted into the model. This is needed for when a model has a specified input size which will be seen in the models that will be implemented. Every image needs to be pre-processed to a certain size and then converted into an array as it cannot be read otherwise by the model. Using pre-processing can also decrease the time it takes a model to be trained and can make a model more reliable.

5.2.1 Understanding CNN's with Keras

To begin learning how to train a neural network using Keras and how the structure works the MNIST dataset was used. This dataset contains hand drawn digits, and it is used to train models to recognize inputted images of hand drawn digits. The developer decided to start here as this dataset can be loaded in with Keras very easily and it is an effective way to become comfortable with using Keras.

To start with the necessary libraries must be imported. The dataset is loaded from the `keras.datasets` library. Pre-processing tools are imported from the Keras image pre-processing library, these tools are for loading and converting the image to an array. The Keras `to_categorical` class is for separating the data into bins (a bin for each digit). The type of model that will be used is a sequential model this is because it allows the developer to create a model layer by layer which is the ideal way to create a CNN model.

First the data was loaded into NumPy arrays to get the shape of the arrays. The images are stored as 28x28 pixel images. To view this as an image format and not as an array the

matplotlib library is used. The developer specified the cmap (colourmap) as grey so that the picture was grayscale and not in colour, as the digits will be used as grayscale images in the model.

```
plt.imshow(X_train[5],cmap='gray')
```

<matplotlib.image.AxesImage at 0x20a07f6ca60>

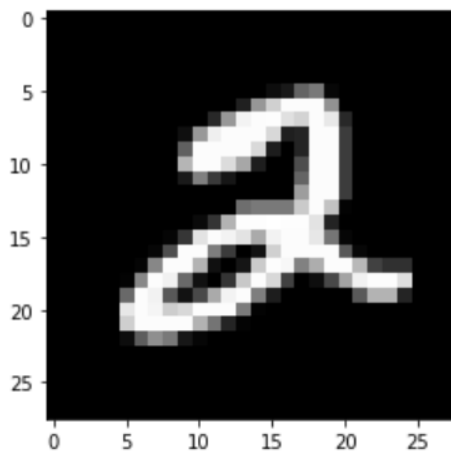


Figure 46 image from dataset

The depth of each image is 1 but this must be explicitly stated in the `expand dimensions` method when reshaping the images for the model. Change the type of each image to a float so that when they are divided by 255 all of the data is between 0 and 1. This will show the image as a binary map which can then be used with the CNN model. Final step before the model is built is to set a layer where the result can be sent, this will be 10 separate groups for each digit. This will be done in the `y_train` and `y_test` using the `to_categorical` method. the output from the model will go into the appropriate bin.

```
batch_size = 128
num_classes = 10
epochs = 20

X_train = X_train.reshape(60000,28,28,1)
X_test = X_test.reshape(10000,28,28,1)
X_train = X_train.astype('float32')
X_test = X_test.astype('float32')
X_train /= 255.0
X_test /= 255.0
y_train = to_categorical(y_train,num_classes)
y_test = to_categorical(y_test, num_classes)
```

Figure 47 pre-processing train and test sets

This CNN model will have two convolution layers, two max pooling layers and two Fully connected layers. The activation function that's used within the hidden layers is ReLU and the final layer used is SoftMax. There are many parameters that can be added to the convolutional layer to suit it to the specific model. In the first convolutional layer the input shape is specified (this is the shape of the image), as well as the kernel_size, padding, and activation function. This is repeated for the next convolutional layer, but the input shape does not need to be stated again. "padding='same'" is used when the output size will be the same as the input size and this will also ensure that the filter is applied to all the elements of the input. maxPooling2D is used for 2D spatial data and as the images have a depth of 1 it can be used for this model. After these layers run, there will be a feature map left and this map is then flattened making each feature map 1 layer of 784 nodes, each node is the value of a pixel within the feature map. All images will now be a single layer of 784 pixels also known as nodes. The next two layers are called Dense layers within Keras, which means that they are fully connected layers these are the final layers of the network. The second fully connected layer uses SoftMax instead of ReLU as it is the best at handling multiple classes which is needed for a categorical model like this.

```
digits_cnn = Sequential()
#first convolution Layer - specify input shape here
digits_cnn.add(Conv2D(32, kernel_size=(3,3),input_shape=(28,28,1), padding='same', activation='relu'))
#using MaxPooling to pool the feature maps
digits_cnn.add(MaxPooling2D())
#second convolution Layer - same as first expect leave out input shape
digits_cnn.add(Conv2D(32, kernel_size=(3,3),padding='same', activation='relu'))
digits_cnn.add(MaxPooling2D())
#flatten the feature map
digits_cnn.add(Flatten())
#Dense Layer is the fully connected Layer
digits_cnn.add(Dense(1024,activation='relu'))
#use softmax for predicting at the end
digits_cnn.add(Dense(10,activation='softmax'))
#compile the model
digits_cnn.compile(optimizer='adam',loss='categorical_crossentropy',metrics=['accuracy'])
print(digits_cnn.summary())
```

Figure 48 digits model layers and compiled

After this the model is fitted and accuracy is shown after each epoch. An epoch is a unit of time used to train a neural network with all the training data for a single cycle. We use all of the data exactly once in an epoch. A forward and backward pass are combined to make one pass. After the model is fit it is obvious that it is extremely accurate as the accuracy is above 99%.

```
Epoch 3/20
1875/1875 [=====] - 99s 53ms/step - loss: 0.0259 - accuracy: 0.9914 - val_loss: 0.0234 - val_accu
cy: 0.9921
Epoch 4/20
1875/1875 [=====] - 99s 53ms/step - loss: 0.0195 - accuracy: 0.9937 - val_loss: 0.0131 - val_accu
cy: 0.9956
Epoch 5/20
1618/1875 [=====>.....] - ETA: 12s - loss: 0.0144 - accuracy: 0.9952
```

Figure 49 Evaluation using epochs 99% accuracy

5.2.2 Understanding ImageNet

After using digits to understand Keras and how it works to create a CNN model with simple images the developer decided to move on and look at how it works for more complex images. This was done using ImageNet and VGG16. ImageNet is an image database with over a thousand categories varying from glaciers to flies. These are all categorised and has been used to test CNN models and see which is the best. VGG16 which was created by the Oxford Geometry Group was the most accurate model in 2014 when used to predict images in ImageNet.

VGG16 was used instead of creating a new CNN because there is a pretrained model using ImageNet and as ImageNet contains thousands of images it would take too long to train a new model. Along with this VGG16 won the ImageNet challenge, meaning that is an accurate and reputable model.

There is a VGG16 model within Keras making it quite simple to get access to, all that is needed is to import VGG16 and to image pre-processing libraries for preparing the images. The weights for imageNet will be imported into the VGG16 model as training would take hours. Then an image that is saved within a folder of images is loaded into the application using Keras pre-processing library for images, the target size must be set as 224 x 224 as that is the size of the image used within ImageNet. The image is then loaded in at this size and saved within the variable 'img'.

```
In [2]: model = vgg16.VGG16(weights='imagenet')

In [4]: #using target size we make sure the image is the correct size for the dataset
img = image.load_img('images/dog.jpg', target_size=(224,224))
img


Out[4]: 
```

Figure 50 loading in image into VGG

The image then needs to be converted to a Numpy array and then the dimensions must be expanded as the model will expect four dimensions. The `expand dimensions` method within the Numpy package will change the image from (224,224,3) to (1,224,224,3). It needs to be in four dimensions as it needs to be in the format (batch size, image width, image height, image depth).

To then pre-process the image before inputting the image, the developer used the `vgg16 preprocess_input` which is available with the package to pre-process the array.

The image must be sent to be predicted. There are 1000 categories of prediction in imageNet and it will be predicted as one of those, the closest prediction will be assumed as the correct one. In this example the top 5 predicted categories are displayed along with their score. The score is how similar the image is to others in a category.

```
# predictions for top 5 closest
# we must decode prediction
vgg16.decode_predictions(preds, top=5)

[[('n02099601', 'golden_retriever', 0.40887728),
 ('n02100583', 'vizsla', 0.30801427),
 ('n02101388', 'Brittany_spaniel', 0.08966445),
 ('n02099849', 'Chesapeake_Bay_retriever', 0.05151526),
 ('n02099267', 'flat-coated_retriever', 0.048557855)]]
```

Figure 51 VGG imageNet prediction

The value of using this and analysing the ImageNet dataset is to see how accurate a pretrained set is and how quickly it gets the prediction. Another thing that was valuable is that it increased the developer's knowledge of CNN's by using this model and that knowing a few of the top predictions rather than just one would be useful down the line.

5.2.3 Image Manipulation

To create an accurate face recognition model, it is important it is important to have a diverse range of images of each face in the dataset. Image manipulation works by rotating the image of the face slightly or distorting it so that the face can be recognized at different angles. Image manipulation can be done in many ways using python and there are libraries that be used to make the data augmentation of images quick. For this project OpenCV and Keras were looked at to do the manipulation. OpenCV has many techniques including rotation and flipping the

images though I found it was a bit tricky to do many small augmentations without having trouble below is some of the problems I came across. OpenCV is useful for cropping a large set of images together though which can be useful, in the case of this dataset the developer wanted pictures with faces far away and close to the screen so it wasn't used.

```
# Import packages

import cv2
import numpy as np

img = cv2.imread('clare.jpg')
print(img.shape) # Print image shape
cv2.imshow("original", img)

# Cropping an image
cropped_image = img[80:280, 150:330]

# Display cropped image
cv2.imshow("cropped", cropped_image)

# Save the cropped image
cv2.imwrite("Cropped Image.jpg", cropped_image)
```

Figure 52 cropping image using OpenCV

Keras on the other hand has a class that can do this with less lines of code and has more options for how to augment the image. This class is called the `keras.preprocessing.image.ImageDataGenerator` class. This is used to configure random transformations of images using various parameters including flips zooming and shifts.

Through testing different zooming and rotations, it was discovered that if certain rotations were chosen the image became unusable, shown in figure 34. This meant that testing had to be done with various rotations until a useable rotation was chosen. As seen in the unusable image the face is cropped out and a lot of the image is blue, this could cause bad testing as the image is a good image of a face. The useable image has a small bit of a blue border but because the full face is still in view it can be used. There are many parameters that can be used the ones that the developer used can be seen in the figure 33 with the descriptions of each parameter. After choosing the suitable parameters multiple manipulations of each image were created and saved into premade folders.


```
datagen = ImageDataGenerator(
    rotation_range=20,
    width_shift_range=0.02,
    height_shift_range=0.02,
    shear_range=0.1,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest')
```

Figure 53 Image data generator that was used



Figure 54 unusable image- manipulated



Figure 55 original image



Figure 56 usable image - manipulated



Figure 57 original image

5.2.4 Saving and Loading the Model

It is important to save the model created so it can be used for future projects and also so it can be loading into applications. This is quite a simple thing to do but it does work differently for different models. Joblib and pickle can be used to save a model, as well as this Keras has its own method to save a model.

Pickling a Python object structure means to serialize and deserialize the object. N this process the python object hierarchy is converted into a byte stream and when it is 'unpickled' the inverse is operated. Joblib runs the same operation but is more effective than pickling for large Numpy array, this is due to the fact it contains special handling for the array buffers of the data structure within Numpy.

```
from sklearn.externals import joblib

# Save to file in the current working directory
joblib_file = "joblib_model.pkl"
joblib.dump(model, joblib_file)

# Load from file
joblib_model = joblib.load(joblib_file)
```

Figure 58 save and load model - joblib

```
import pickle

#pickle save model
pickle.dump(classifier,open('./dnn_network/model_cnn_keras.pickle','wb'))

#pickle load model
model_cnn = pickle.load(open('./model/model_svm.pickle','rb'))
```

Figure 59 save and load model - pickle

A Keras model can be saved using pickle but it is recommended to use its own methods. Keras has its own internal libraries to save and load the model. There are two options the Saved Model option which is the newer version and saving the model in H5 format. In this application the developer decided to use the H5 format as it is more lightweight and saves everything as a single file rather than a folder of files with the saved model format. This saves the model's

architecture, weights values, and compile() information in a HDF5 format. The main reason to use Save Model over this is that it saves external loss metrics that aren't calculated within layers of the network. As the model I am creating contains only loss metrics within the layers of the network this isn't an issue.

```
import keras

# keras save model
model_cnn.save('my_h5_model.h5')

# keras load model
model_cnn = keras.models.load_model("my_h5_model.h5")
```

Figure 60 save and load model - Keras

5.2.5 Model Comparison

The next step was to start towards creating a model that would solely predict faces. This would be similar to the layout of ImageNet and its category system. The easiest way to begin this was to create a system that would predict whether an image was a specific face or if it was not that face. To avoid any issues regarding permissions to use a face the developer decided to use their own as the subject of this model. To test this multiple models were tested against each other. The chosen models are SVM (Support Vector Machine) with PCA(Principal Component Analysis), The face_recognition library in Python used with dlib, Keras similar to the one used in the MNIST example above and VGG16 as explained above. All models will be shown below with their accuracies.

SVM is not a convolution neural network and the main reason this was built is to study how these are built themselves and to test it against the CNN versions to prove that they are more accurate. PCA can be used within most prediction models and is used for dimensionality reduction as mentioned in the research chapter. As mentioned above images must be resized and changed to Numpy arrays to be read by the model. The faces in this model have been cropped and resized as necessary and categorised into two folders.

PCA is used to reduce the dimensionality to reduce the computational load and then create the eigen images uses the eigen vectors. Once thus is done the SVM model can be trained, the SVM model can be imported using sklearn and then trained with all the images that were pre categorised and tested. This is shown in figure 60.

```

▶ from sklearn.svm import SVC

▶ model = SVC(C=1.0,kernel='rbf',gamma=0.01,probability=True,)

▶ model.fit(x_train,y_train)
  print('model trained sucessfully')

▶ # score
  model.score(x_train,y_train)

▶ # score
  model.score(x_test,y_test)

```

Figure 61 SVM model setup

For a new image to be added to this and to make sure it matches the data in the dataset a pipeline model will need to be made to run some pre-processing to fit it to the model. This is made using the help of OpenCV. Through testing it was discovered that it Mainly works for frontal classification and not very good at anything else

The next model that was created was done using the python face_recognition library using dlib which is a toolkit for creating machine learning applications in C++ originally but contains Python bindings as well. There are two different versions of this library that will be used. The first is the detector which is created using HOG and SVM. The other being the CNN detector. These models were both first compared against the other and the CNN one was the best this also HOG which is Histogram of Oriented Gradients. After looking at how these models calculate the prediction it was decided to leave them out of the comparison as they weren't very malleable and the developer wanted to be able to have more control over the model. The figure below it shows how to make a face comparison script using dlib and face_recognition

```

#getting the face locations and face encodings fo face 1
faceLoc = face_recognition.face_locations(imgClare)[0]
encodeFace = face_recognition.face_encodings(imgClare)[0]
#creating a rectangle that will surround the face in the image
cv2.rectangle(imgClare,(faceLoc[3],faceLoc[0]),(faceLoc[1],faceLoc[2]),(255,0,255),2)

faceLocTest = face_recognition.face_locations(imgTest)[0]
encodeTest = face_recognition.face_encodings(imgTest)[0]
cv2.rectangle(imgTest,(faceLocTest[3],faceLocTest[0]),(faceLocTest[1],faceLocTest[2]),(255,0,255),2)

#compares the faces and gets the 'distance' - the face distance is the distance they are away from being the same
results = face_recognition.compare_faces([encodeFace],encodeTest)
faceDis = face_recognition.face_distance([encodeFace],encodeTest)
print(results,faceDis)
#puts the prediction on the faces bounding box
cv2.putText(imgTest,f'{results} {round(faceDis[0],2)}',(50,50),cv2.FONT_HERSHEY_COMPLEX,1,(0,0,255),2)

cv2.imshow('Clur',imgClare)
cv2.imshow('Test',imgTest)
cv2.waitKey(0)

```

Figure 62 dlib face comparison script

The two other models being used is the built in VGG16 model and a Keras model that is modelled similarly to the one described previously.

The comparison will be done with the VGG16 model within Keras, the SVM model and the Keras model that the developer made when using mnist but will be retrained for this comparison. These were all compiled and trained using the same dataset and then tested. Each model was tested on ten different images and the training accuracy was taken into account too. The SVM model was the poorest which was expected and the VGG16 and the custom Keras model predicted 9 out of 10 correct but the VGG had the highest accuracy overall.

The VGG16 model performed the best but only by a small margin due to this the model that would be chosen would be the custom one using some of the elements in the VGG16 model, including similar amount of convolution layers and same activation functions. Combining pre existing methods that are accurate with the ability to tune to exactly what is needed will speed up the training stage as the model being used already is over 90% accurate.

The model setup and test for the chosen model is shown in figure 62.

```

model= Sequential()

model.add(Convolution2D(32,
                        kernel_size=(5, 5),
                        strides=(1, 1),
                        input_shape=(64,64,3),
                        activation='relu')
            )

model.add(MaxPool2D(pool_size=(2,2)))

model.add(Convolution2D(64,
                        kernel_size=(5, 5),
                        strides=(1, 1),
                        activation='relu')
            )

model.add(MaxPool2D(pool_size=(2,2)))

model.add(Flatten())
|
model.add(Dense(64, activation='relu'))
model.add(Dense(OutputNeurons, activation='softmax'))

model.compile(loss='categorical_crossentropy', optimizer = 'adam', metrics=["accuracy"])

```

Figure 63 The models layers set up

```

import numpy as np
from keras.preprocessing import image

ImagePath='Face Images/Final Testing Images/Angela_Bassett/Angela_Bassett_0006.jpg'
test_image=image.load_img(ImagePath,target_size=(64, 64))
test_image=image.img_to_array(test_image)

test_image= np.expand_dims(test_image,axis=0)

result= model.predict(test_image,verbose=0)
#print(training_set.class_indices)

print('Prediction is: ',ResultMap[np.argmax(result)])

Prediction is:  Angela_Bassett

```

Figure 64 Testing the models accuracy

5.3 Machine Learning in Flask

In this phase the flask framework is shown and how it is possible to use a machine learning algorithm within its framework. This is first tested by using text inputs and then followed up

with image inputs. The goal for this phase is to create a Machine Learning API within flask that will accept an image and then run a prediction on it.

5.3.1 Setting up Flask

Firstly, the flask application must be created within Anaconda. Flask is also pre-installed with Anaconda. The IDE that the developer is working with is VS Code, to use Python in VS Code a Python interpreter must be chosen, this interpreter will be the environment in which Anaconda is running currently. All further modules that need to be installed for the application must be installed within that specific environment or they won't be accessible by the application. Flask comes with a light-weight web server that can be controlled by your Python code. The way to set up this server is shown in figure 37.

```
from flask import Flask, request
app = Flask(__name__)
@app.route('/')
def home():
    return 'Home route is running'
```

Figure 65 Set up flask server

The code states that first flask is imported from flask. Then an instance of the Flask class is created and the 'name' variable is passed. In the @app.route('/') the route() function is a decorator function this means it is a function that takes another function as its argument. This tells Flask that at that URL it must run the home() function. This home() function will receive a request and send the required output. In the case below it will just return 'Home route is running'.

To run the server, all that's needed is to run the command below and then navigate to the page on localhost:5000. This is the port that Flask automatically runs on.

```
PS C:\Users\clare\4th_Year\Major_Project\major_project_facerec> python main.py
```

The next step is to create a function for the model and load it in. The model that will be used is saved in a pickle file. This file can be opened and used within a flask application. To begin testing this the developer had to first install all necessary dependencies into the application including pickle and Keras. To avoid creating a webpage immediately the testing was done

inside insomnia which is a REST client for testing RESTful applications. The application being created is an API using HTTP requests so it can be testing within this interface.

5.3.2 Testing In insomnia

To begin the testing within insomnia the developer started with a pre-made model for testing whether or not a person would survive on the titanic. This was chosen instead of using an image model initially as the set up was easier and it was used to make sure all elements were working as planned. Another reason this was done is that posted images is more complex and the purpose of this test was to check if insomnia and flask were communicating and also that model was being opened correctly within flask. Below shows the request made in insomnia and the answer given in the console.

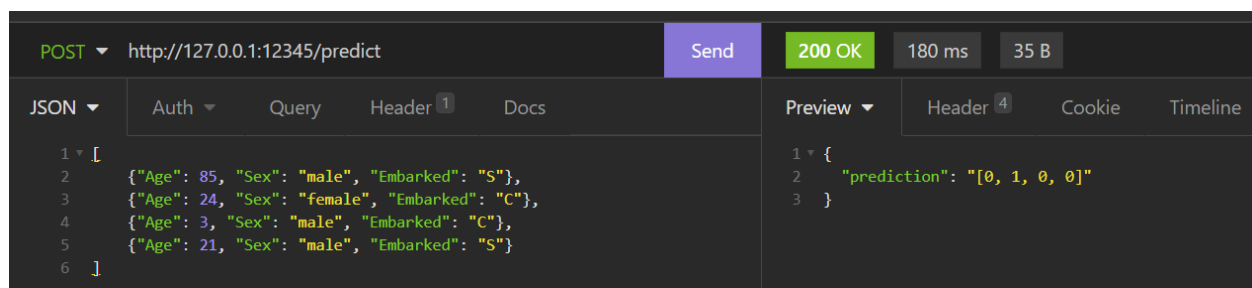


Figure 66 Example titanic dataset prediction

This was all repeated using the MNIST dataset as there are many available digit images online to test on it. As mentioned above to POST an image is more complex than just sending test. This is mainly because the file must be in certain formats and be of an exact size or the size will be changed within a method in flask. For the purpose of this only images that were the same size as the ones in the training set were used. This was a step-by-step approach to make sure each feature that was needed worked instead of just testing everything immediately. The way this worked was that you uploaded an image into insomnia and POST that to the 'upload' route. Flask then receives this request and gets the file from the request. This file is then sent to the pipeline model. The pipeline model converts the image into an array and changes it to greyscale if necessary (all digit images are greyscale by default). This model would also resize the image into the required size for the model before changing it to an array. Image is then sent to the model for classification, this is the model that is shown in fig X in section X. This model then returns a prediction of which digit it assumes the image to be of, this is shown in the console.

5.4 Working with a Database

The aim of this phase was to figure out how to connect Flask to a database. There are extensions for Flask to allow it to be connected to a database the one for connecting to an SQL or SQLite database is called Flask-SQLAlchemy this must be installed into the Anaconda environment and then imported in the Flask application. SQLAlchemy can be used within Flask without the Flask extension but the extension provides useful default to simplify using SQLAlchemy in Flask

The first step was to create a flask application object and set the database URI.

```
#database
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:/// ' + os.path.join(basedir, 'db.sqlite')
app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False
```

Figure 67 SQLite DB URI

Next step is to declare the models for each of the database tables that are being created, image and attendance. The base class for the models is db.Model, this is stored in the instance of SQLAlchemy that is made and shown in fig X. Names of columns are assigned using the db.Column class, within this you name the column and also give the column a type for example db.integer or db.string this is demonstrated in figure 66.

```
# image class/model
class Image(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String(100), unique=True)
    img = db.Column(db.Text, unique=True, nullable=False)
    mimetype = db.Column(db.Text, nullable=False)

    def __init__(self, name, description, img):
        self.name = name
        self.img = img
        self.mimetype = mimetype
```

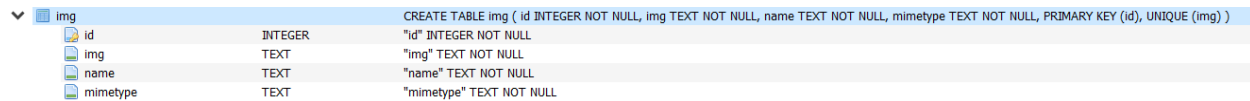
Figure 68 image model initialisation

To then create this database, open the python terminal and run the following commands.

```
PS C:\Users\clare\4th_Year\Major_Project\Tutorials\Ex_Files_Building_Deep_Learning_Apps\mlAPI> python
Python 3.8.12 (default, Oct 12 2021, 03:01:40) [MSC v.1916 64 bit (AMD64)] :: Anaconda, Inc. on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> db.create_all
```

Figure 69 Creating table in python terminal

To test that this created the table a browser extension must be downloaded to view the database in SQLite as VS Code cannot display it on its own. Once the extension is downloaded the database should show the table that was just created.



img	CREATE TABLE img (id INTEGER NOT NULL, img TEXT NOT NULL, name TEXT NOT NULL, mimetype TEXT NOT NULL, PRIMARY KEY (id), UNIQUE (img))
id	INTEGER "id" INTEGER NOT NULL
img	TEXT "img" TEXT NOT NULL
name	TEXT "name" TEXT NOT NULL
mimetype	TEXT "mimetype" TEXT NOT NULL

Figure 70 Table that was created from the model

Once this is done the method for uploading a file must be changed to allow the image to be added to the database. This means we have to send the data that matches the table data structure to the model and also open a database session to send the image and close it once its complete. Once this was successful a message was sent to the console and the image could also be seen in the database. Next the developer tested that the image was being saved accurately and there was no loss of quality to the image. This was done by creating a webpage to return the image.

5.5 Integrating Model with Database Application

The goal of this phase is to create an application where the image is uploaded to the database and saved as well as a prediction is run, and a console response is received. In addition to this the attendance functionality will be added along with all the necessary pages for the full application.

Firstly, the developer wanted to set up all the necessary folders and files in a readable format and make sure everything connects. This structure contains:

- A templates folder
- A folder with all the training images and the model
- A utils folder
- A model folder

5.5.1 Integration Issues

At the beginning of this process once there was a route to get a prediction and to add an image to the database. The developer decided to do a test run to see if everything integrated correctly, there was immediately errors with the building the application. This means nothing

would run and specifically scikit learn wasn't working which was used with the recognition model. This error was caused by Python version conflicts and while trying to reinstall and uninstall packages within the environment more errors with package versions started occurring until the main error was discovered. The model happened to be trained using Python 7 and therefore another version of scikit learn was used whereas the flask application that was in this environment was created using Python 8. The application needed to stay in Python 8 as it has the most widespread support along the various modules the developer was using. With the switch to Python 8 TensorFlow and Keras had to be reinstalled in a different version along with a few other modules including scikit learn which is integral to the creation of this project as it is one of the base libraries.

The developer decided to create a new environment in anaconda and all the dependencies were re installed using the anaconda cmd interface. This was the best way to make sure everything worked together instead of altering the other environments. These requirements were then exported to a requirements.txt file for future use in case the project needed to be downloaded elsewhere.

5.5.2 Creating Templates and Routes

Once the base route and model were working the templates were created. The templates were all created using the Jinja templating language which is basically a text fil that can generate and text-based format. All of the templates used a base template as the skeleton of the layout. This meant that the base contained the header, footer and any other common elements between pages. The rest of the created templates are extended from this base template and are called child templates. This saves on load time for the pages as the base is always loaded and just the internal elements of each page are altered.

The block tags '{% block %}' define four blocks that the child templates can fill in, they are the title, head, content and end blocks respectively. The only thing this block tag does it alert the engine that the extended template might override these portions. To allow the Jinja engine to know the template is extended from another it must contain the extends tag '{% extends *name_of_parent* %}' at the beginning of the page. The template in figure X is a child template of figure X. In the child it shows that its overridden the blocks expect for the end block.

Using these templates 'if' and 'for' statements can be used within the html, this used in the following image to show the uploaded image once the prediction is ran.

```

<div class="container">
  <br>
  <form action="#" method="post" enctype="multipart/form-data">
    <label for="upload">Upload to predict</label>
    <input type="file" id="upload" name="image" required="true">
    <input type="submit" value="upload & predict">
  </form>

  <br>
  <form action="http://localhost:5000/upload" enctype="multipart/form-data" method="POST">
    <input type="file" name="pic" />
    <input type="submit" value="Upload a file" />
  </form>
</div>

{% if fileupload %}

<div class="container">
  <div class="row">
    <div class="col-md-6" id="predict">
      
      <p>img_name</p>
    </div>
  </div>
</div>
</div>

```

Figure 71 prediction template - child template

When routes are being set up, the methods that will be used within the function in the route need to be stated. For example the attendance route needs the GET and POST methods, the get and post requests won't work unless these are explicitly mentioned.

```

# url
app.add_url_rule('/base', 'base', views.base)
app.add_url_rule('/', 'index', views.index)
app.add_url_rule('/snapshot', 'capture', views.capture)
app.add_url_rule('/attendance', 'attendance', views.attendance, methods=['GET', 'POST'])
app.add_url_rule('/predict', 'predict', views.predict, methods=['GET', 'POST'])
app.add_url_rule('/video_feed', 'predict', views.video_feed)
app.add_url_rule('/tasks', 'tasks', views.tasks, methods=['GET', 'POST'])

```

Figure 72 all URLs for application

5.5.3 Adding Upload Prediction Functionality

In order to add the prediction functionality the model must be loaded into the application and then a pipeline model must be created. The model is loaded in using Keras load_model method. While this is done the mapped_faces which is the current list of faces that the model has been trained with.

```

model_cnn = keras.models.load_model("./model/my_h5_model.h5")
mapped_faces = pickle.load(open("Mapped_Faces.pkl", 'rb'))

```

Figure 73 loading model and map

A pipeline contains the pre-processing necessary for the image to be sent to the model and that pipeline will also return the result and send to the webpage. This is the template that's shown above in figure 42. The image is loaded into the pipeline and the target size is set so that it fits in the model. The dimensions are then expanded to allow for batch size to be added to the dimensions when testing. This array of the image is then sent to the model for prediction and the prediction is save dint he result variable. This is then printed to the console using the mapped_faces to find the result as the result returns as a number and the faces map contains the names. The next few lines are describing the text that will be used to display this to the user and is done using OpenCV. Using OpenCV's put Text method the result text is placed on the image. The 'cv2.imwrite' saves the image in the specified location with the wanted format and this image can then be shown on the webpage with the result.

```
def pipeline_model(path,filename,color='bgr'):  
  
    ImagePath=path  
    test_image=image.load_img(ImagePath,target_size=(64, 64))  
    print(test_image)  
    test_image = image.img_to_array(test_image)  
  
    test_image=np.expand_dims(test_image,axis=0)  
    result=model_cnn.predict(test_image,verbose=0)  
    #print(result)  
    print('Prediction is: ',mapped_faces[np.argmax(result)])  
  
    font = cv2.FONT_HERSHEY_SIMPLEX  
  
    text = mapped_faces[np.argmax(result)]  
    cv2.putText(test_image,text,(1000,1000),font,5,(255,255,0),2)  
  
    cv2.imwrite('./static/predict/{}'.format(filename),test_image)
```

Figure 74 pipeline model

The functions shown in figure 45 are needed to predict and show the image in the webpage. The getwidth() function gets the path of the image and calculates it width, this will be needed to correctly view it in the webpage. The following method called if an image is uploaded it gets the filename and path, saves the path and then calls the pipeline model shown in figure 44 to get a prediction of who is in the image. This template is then rendered, and the prediction shown.

```

#width needed for displaying image
def getwidth(path):
    img = Image.open(path)
    size = img.size # width and height
    aspect = size[0]/size[1] # width / height
    w = 300 * aspect
    return int(w)

#receives the image, sends it to the pipeline model for prediction
def prediction():
    if request.method == "POST":
        f = request.files['image']
        filename= f.filename
        path = os.path.join(UPLOAD_FOLDER,filename)
        f.save(path)
        w = getwidth(path)
        # get prediction
        pipeline_model(path,filename,color='bgr')

        # show prediction with image
        return render_template('prediction.html',fileupload=True,img_name=filename, w=w)

    return render_template('prediction.html',fileupload=False)

```

Figure 75 methods for prediction webpage

5.5.4 Database integration

Adding the SQLAlchemy to the application and creating the database tables. Once the script in figure 76 is ran the database is created along with the models shown in figure 77. One of these will be used to store all the attendance while the other will be used to store images of unknown users of the application.

```

from flask_sqlalchemy import SQLAlchemy

db = SQLAlchemy()

# Function that initializes the db and creates the tables
def db_init(app):
    db.init_app(app)

    # Creates the logs tables if the db doesnt already exist
    with app.app_context():
        db.create_all()

```

Figure 76 Script to create database

```

from db import db

class Img(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    img = db.Column(db.Text, unique=True, nullable=False)
    name = db.Column(db.Text, nullable=False)
    mimetype = db.Column(db.Text, nullable=False)

class Attendance(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.Text, nullable=False)
    timestamp = db.Column(db.Text, nullable=False)

```

Figure 77 Database Models - tables

5.5.5 Adding Attendance

The function shown in figure 78 is used to mark attendance and save it to a csv. After the pipeline gives a prediction the markAttendance() function is called and the name for the prediction is sent to it. The csv file is then opened and sets myDataList as f.readlines(). This just returns the current list of attendees. Then it adds a comma separated line and write the name with dtString. This string is the date time which will be counted as the timestamp. The date time is imported from the date module in Python.

```

def markAttendance(name):
    with open('Attendance.csv','r+') as f:
        myDataList = f.readlines()

        for line in myDataList:
            entry = line.split(',')
            now = datetime.now()
            dtString = now.strftime('%H:%M:%S')
            f.writelines(f'\n{name},{dtString}')

```

Figure 78 Attendance saved to csv

Once this was working correctly, it was converted to a method that would save the name and timestamp to the database. This method is shown in figure 79. The datetime needed to be

converted to a certain format or it wouldn't save correctly and also saving it to this format will make it easier to display to the user as the change has already been done. The Attendance model is passed the parameters for name and timestamp, and this is saved to att. To check that this is happening it is printed to the console. Then to save it to the database a session must be started and then the item is added and then committed to the database. After this check the database if it added and once that's working the attendance functionality is created.

```
def markAttendance(name):  
    name = name  
    now = datetime.now()  
    timestamp = now.strftime('%H:%M:%S')  
  
    att = Attendance(name = name, timestamp = timestamp)  
    print('attendance',att)  
    db.session.add(att)  
    db.session.commit()  
  
    return 'added to db'
```

Figure 79 Attendance saved to database

The next step of this is to be able to view this within the application. A template was created for the attendance with a table for it to be viewed in. The route for attendance was set up. In this route a query must be sent to the database to retrieve the attendance in this current instance the developer wants to view all attendance ever gathered. This is done using the following code and is queried as the template is being rendered.

```
@app.route('/index')  
def index():  
    return render_template('index.html', attendance = Attendance.query.all())
```

Figure 80 Index route - shows attendance

5.5.6 Adding Basic Image Capturing Functionality

Instead of using a constant video feed for predictions, the developer decided to add an image capturing function to use instead of only having an upload feature for predictions. This was first implemented to send an image to be trained or to add a new user to the training set. The current way this works is that an image will be saved to a folder external from the training images folder and the admin will need to manually retrain the model with the new person. The

new person will still be added to the attendance regardless of this as they will be told to insert their name and then capture an image of themselves.

The first step of this process was to connect to the web cam using OpenCV to start the video feed, this done by using the VideoCapture(0) object to get the feed of the webcam, the default value of the webcam will be set to '0'. Then to read the frames of the camera frame by frame the .read() function is used. To show that the webcam is picking up this a route for the video feed is set up and this is linked to the capture template. It is displayed in an image object on the page using the url for the video_feed() function

```
<h2 class="mt-5">Take your picture for Attendance</h2>
<h4>Press the capture button to be added to the attendance and press stop </h4>
<form method="post" action="{{ url_for('tasks') }}">
  <input type="submit" value="Capture" name="click" />
  <input type="submit" value="Stop/Start" name="stop" />
</form>

```

Figure 81 Capture image template

Once the camera was set up a method to control the capturing of the image was set up. Post request sent to the method and the capture variable (a global variable) is set to '1', it originally was set to '0'. This also sends the captured frame to the function, as the global variable was switched the image capture will be saved in the capture_save function. This function is run on a loop through the video_feed function.


```

#capturing video from web cam
def capture_save():
    global out, capture
    #reads the camera while the on the capture page
    while True:
        success, frame = camera.read()
        if success:
            if(capture):
                capture = 0 #if an image is captured the 1 is set back to 0 to allow for more to be taken
                now = datetime.datetime.now() #sets the time of the image being taken
                #sets the path of the image
                path = os.path.sep.join(
                    ['./Face Images/new_additions', "shot_{}.png".format(str(now).replace(":", ''))])
                #saves image to specified path
                cv2.imwrite(path, frame)
                #gets filename by using the os module
                filename = os.path.basename(path)
                #sends it to pipeline model
                pipeline_model(path, filename, color='bgr')
                img_manipulate(frame)
            try:
                ret, buffer = cv2.imencode('.jpg', cv2.flip(frame, 1))
                frame = buffer.tobytes()
                yield (b'--frame\r\n'
                    + b'Content-Type: image/jpeg\r\n\r\n' + frame + b'\r\n')
            except Exception as e:
                pass

```

Figure 82 Capture image function

Within the capture_save() function the image frame is read in and saved in the image variable. Then the path for where the image will be saved is created using the OS module. The OS module in Python allows the developer to interact with the operating system for example making paths, creating, and deleting files and folders. Then using the 'cv2.imwrite()' function it is saved using the path. This feature at this point was used to add photos to a training folder and not for predictions.

5.6 Model Finetuning

For this phase, the chosen model is taken and finetuned and tested until it is as accurate as possible. To do this first the current training set is fit, and the accuracy recorded. Then other iterations are chosen, and all accuracy recorded until the model becomes as accurate as possible. Different iterations that were chosen were to use an extra convolution layer, change the kernel and padding size, use an extra fully connected layer in between the original two and changing the number of epochs used while fitting the model. During this phase the developer also looked at making the model more diverse as from looking through the images most of the faces were of western type faces and better models have faces from around the world.

5.6.1 Retraining the model

To start retraining the model the current model's accuracy was recorded by fitting and evaluating the model, the model was fitted 5 times and the average of the evaluation was taken. The average was 92.86%. Ideally this should be increased to 99% for the model. After the evaluation was recorded the model was tested by having it predict 10 random images from the test set, in this test the model got 8 out of 10 correct. This means there is plenty of room for the model to improve. In figure 84 the current summary of the model is shown with its current layers. There are 2 convolution layers with ReLu activation and two max pooling layers a flattening layer and two fully connected layers which are referred to as dense layers in Keras.

```
Epoch 15/18
51/51 [=====] - 11s 205ms/step - loss: 0.2427 - accuracy: 0.9343 - val_loss: 2.7561 - val_accuracy: 0.
6562
Epoch 16/18
51/51 [=====] - 11s 210ms/step - loss: 0.2502 - accuracy: 0.9245 - val_loss: 2.5036 - val_accuracy: 0.
6396
Epoch 17/18
51/51 [=====] - 11s 215ms/step - loss: 0.2033 - accuracy: 0.9368 - val_loss: 2.9828 - val_accuracy: 0.
6354
Epoch 18/18
51/51 [=====] - 11s 213ms/step - loss: 0.1961 - accuracy: 0.9392 - val_loss: 2.3459 - val_accuracy: 0.
6708
##### Total Time Taken: 3 Minutes #####
```

Figure 83 first model evaluated

```
model.summary()
Model: "sequential_1"
```

Layer (type)	Output Shape	Param #
conv2d_4 (Conv2D)	(None, 60, 60, 32)	2432
max_pooling2d_3 (MaxPooling 2D)	(None, 30, 30, 32)	0
conv2d_5 (Conv2D)	(None, 26, 26, 64)	51264
max_pooling2d_4 (MaxPooling 2D)	(None, 13, 13, 64)	0
flatten_1 (Flatten)	(None, 10816)	0
dense_3 (Dense)	(None, 64)	692288
dense_4 (Dense)	(None, 42)	2730

```
=====
Total params: 748,714
Trainable params: 748,714
.. . . .
```

Figure 84 first model summary

To retrain the model three main changes were tried and tested. These were adding one extra convolution layer after the first convolution layer, changing the size of the kernel from (5,5) to (4,4) and to add an extra dense layer. After completing these evaluations, it showed that all

three changes made the model accurate in each instance. Two extra evaluations were tried after this being all three changes used above added together and changing the kernel to (3,3). Then each of these models were tested with predicting 10 images. The results of these evaluations and tests are shown in the table below. This shows that that the highest accuracy is currently at 97%.

	Layers	Kernel Size	Accuracy
First model	2 conv, 2 pooling, 2 dense – SoftMax	5,5	0.9392
Second model	2 conv, 2 pooling, 2 dense – sigmoid	5,5	0.9234
Third model	3 conv, 2 pooling, 2 dense - SoftMax	3,3	0.9509
Fourth model	2 conv, 2 pooling, 3 dense - SoftMax	3,3	0.9556
Fifth model	3 conv, 2 pooling, 3 dense - SoftMax	3,3	0.9786

5.6.2 Adding more Manipulations and Images

Once the finetuning increased the accuracy to 97% the developer decided to add more manipulations and image of new people to the dataset to examine This was done in the hopes of making the model even more accurate as a larger dataset can increase accuracy. 50 extra images were created from manipulations for each person in the dataset. These manipulations were done using the Keras DataImageGenerator. After all of this was completed the model was retrained and evaluated using the updated dataset. More epochs and steps were added, because adding more images will allow the batch size to be increased which in turn allows more steps to be added.

After this evaluation was completed the accuracy that was shown was 98.5% on average. This model as then tested on multiple unknown images and every single one was calculated correctly. In figure 85 the summary of the model chosen is shown.

```
model.summary()
```

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 62, 62, 32)	896
max_pooling2d (MaxPooling2D)	(None, 31, 31, 32)	0
conv2d_1 (Conv2D)	(None, 29, 29, 64)	18496
conv2d_2 (Conv2D)	(None, 27, 27, 64)	36928
max_pooling2d_1 (MaxPooling2D)	(None, 13, 13, 64)	0
conv2d_3 (Conv2D)	(None, 11, 11, 128)	73856
max_pooling2d_2 (MaxPooling2D)	(None, 5, 5, 128)	0
flatten (Flatten)	(None, 3200)	0
dense (Dense)	(None, 128)	409728
dense_1 (Dense)	(None, 64)	8256
dense_2 (Dense)	(None, 42)	2730

=====
Total params: 550,890
Trainable params: 550,890
Non-trainable params: 0

Figure 85 Summary of chosen model - Finetuned

5.7 Final attendance app in Flask

This phase's goal was to put the finetuned model into an application that fit the requirements for the application. For this phase a few final changes were made and added to create a better application. Also, this focused on cleaning the application and making the structure of the files easy to follow. The features that were added here was to delete images once the person was predicted, get the prediction printed on the web page and not just in the console and to add prediction functionality to the image capture function.

5.7.1 Added Features

Adding Capturing image functionality to capture, this was relatively easy to implement. After the image is saved to the training folder it is sent to pipeline model to get a prediction and then the attendance is recorded. Also, now if an image of a person that isn't recognised is captured or uploaded the person is recorded as unknown on the attendance.

It was mentioned by the people that took the survey that they wouldn't feel comfortable with the images being saved. So, the developer decided to try and delete images after the prediction ran. This ended up being an easy endeavour as the os module that's used to create folders and files can also be used to delete them. All that needs to be done is to specify the path of the file that you want deleted.

5.7.2 Issues that Arose

There were a few issues throughout the development of the project, there were integration issues at first and issues with downloading attendance along with TensorFlow issues. The integration issues were explained above, and the problem was to do with conflicting versions of python.

In the jinja templates and even using bootstrap there wasn't an easy way to download the attendance within the application. If the developer had more time to spend on creating the interface this issue could've been resolved as the CSS framework could've been changed or the project could've added React to the frontend which would allow for the csv to be easily downloaded from a table.

TensorFlow works best with a GPU and prefers to work with it. This caused a few errors to appear as the application is being run every time it is being run. After researching why this occurs the developer discovered it wasn't an issue with the application but just a warning that TensorFlow. The application still works perfectly but if the model was larger or if the application needed to compute more, having it run on GPU would be much smoother

5.8 Implementation Summary

Throughout this chapter the application was pieced together, and an accurate model was built. Most of the goals of this application were reached bar the issues mention prior. The model was built and tested and tweaked until it was as accurate as it could be using this small of a dataset. The application had integration problems that were overcame in time to create a working prototype. The model predicts very accurately within the application and the attendance feature works as planned.

6 Testing

Testing the application was done in multiple ways throughout implementing the project, the testing that was undertaken can be labelled under three sections, these being functional testing, model testing and user testing.

- Functional testing is a type of software testing that validates the software system against the functional requirements. The purpose of functional tests is to test each function of the software application, by providing appropriate input, verifying the output against the functional requirements. This mainly involves a method called black box testing which means functions of the application is tested without knowledge of the internal code structure, implementation details and internal paths. This testing checks User Interface, Database, Security, Client/Server communication and other functionality of the Application Under Test. The testing can be done either manually or using automation.
- The model testing is the process of evaluating the performance of a fully trained model on a testing set. The testing set contains a collection testing samples which will be used to see if the model can accurately predict using unknown images. The test set should be a completely different set of images to the training set.
- The user testing was completing using surveys and asking users to test out the application. This was done in two ways one of which was to gather pictures of people's faces and test if the application makes the correct prediction on known faces. The other testing is to do it with new faces and see if it correctly tests if they are new and also add the new faces to the dataset. This will be done in conjunction with surveys that will contain some similar content to the survey in the requirements chapter and what their thoughts are of the application.

6.1 Functional Testing

The functional testing that was undertaken uses the Black Box Testing technique which means that the internal logic of the system being tested is not of interest to the tester. The tester is only interested in whether the actual output agrees with the expected output. This testing will be done on the links and form features within the application.

6.1.1 Links

All links were tested and all of them go to their desired location. All of these are within the app.py file and are rendering the correct template and running the correct functions with no issue.

6.1.2 Client/Server Communication

The forms in this site are all just buttons for various functions and the tests for these are shown in the table below.

Test No	Description	Input	Expected Output	Actual Output	Comment
1	Capture image button	Click button	Image captured and saved to folder	Image captured and saved to folder	Works as intended
2	Stop/Start camera feed	Click button	Webcam feed stopped or started	It works but is slow to do it	Barely works
3	Upload for prediction	Select image and click button	Correct prediction received	Correct prediction received	Works as intended
4	Upload to database	Select image and click button	Uploaded to db	Uploaded to db	Works as intended
5	Attendance table updates after uploaded prediction	Check attendance page after prediction	Updated attendance can be viewed	Updated attendance can be viewed	Works as intended

6.2 Model Testing

Testing the model was explained and demonstrated during the implementation chapter as it is an integral part of creating a model. In this section the developer will show what was involved in testing and which tests were done. Tests were done on the five models that are discussed in implementation. K-fold cross validation was done as well as testing each model on ten random images from the test set.

K-Fold cross validation is Using cross validation you do more than one split, in K-fold each of these splits is a fold. The general process for k-fold cross validation is:

1. The entire dataset is shuffled and split into k-folds without replacement.

2. k-1 folds are used for the model training and one-fold is used for performance evaluation.
3. This procedure is then repeated for as many times as is stated in k, for each iteration we get a performance score,
4. The final step is to get the mean of these scores.

After the mean scores are obtained, the five models can be compared and the one with the highest score chosen as the best and will be used. As seen in the previous chapter the fifth model was chosen as it had the highest score. For this test 10 folds were used, and the fifth model has the highest average at 98.4% accuracy which is very high, all of the models' averages were above 92% so they were all very accurate and robust models overall.

The ten-image test was less accurate as gauging which model was best as three different models got about the same score and even the fifth model with the highest accuracy got a few wrong here and there. For context there was multiples of this test done on each model until they had all been tested on about 50 images each to get a solid accuracy. Even with all that testing the third and fifth model were very close.

6.3 User Testing

There was very little user testing undertaken in this application as the user side of the application was very small. The main user testing was taking images of my friends and using them to train the model and getting them to take a picture in real time to see if it was correct. This was tested on three people; it was correct for all of them and they found the application interesting. Some feedback that was received is that they wouldn't want their information or images kept on file if it was a big organisation or school that was taking the images. It was interesting to hear that they only would use the application because they trusted the developer with their information. Another piece of feedback is that they would want to know that the application is very high security to limit data hacks.

7 Project Management

In the following chapter the management of the project will be discussed. A project plan was previously made to aid in reaching deadlines and working consistently over the months. This project took place from January to May with a project proposal being submitted in December before it. A lot of the deadlines were malleable as the developer was doing this project in more of an iterative process rather than getting specific items done at various times. These iterations and phases of the project are discussed in detail in the implementation and testing chapters.

7.1 Project Management Tools

The project was monitored and organised using Microsoft Planner. Every week this plan would be reevaluated depending on what was being done and what had to be put on hold. As long with this there was a meeting with a supervisor every week to go over progress or the project, in these meeting due dates may be altered depending on what was most important at the time. This project was quite research and learning heavy so that did take up quite a bit of time early on in development.

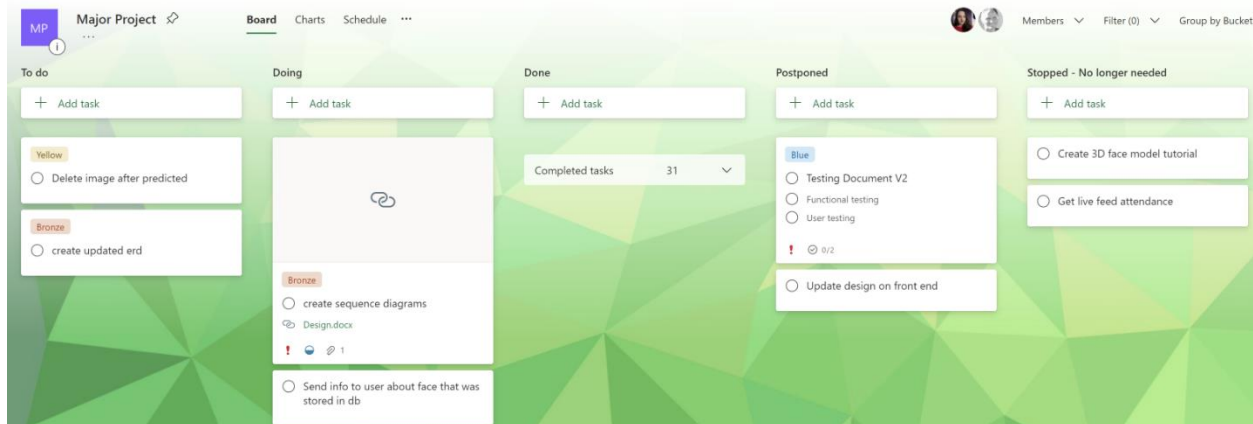


Figure 86 Microsoft Planner



Figure 87 Planner Graphs

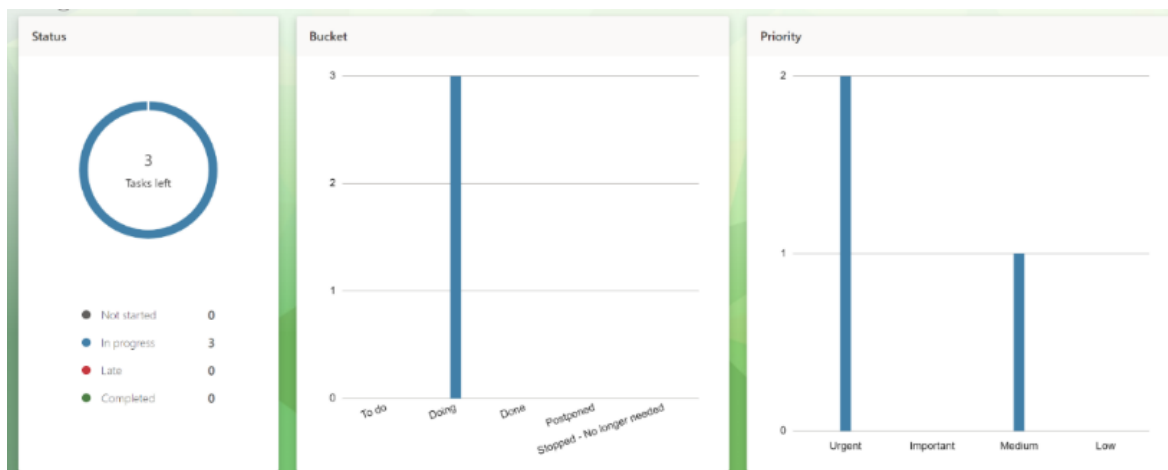


Figure 88 Microsoft planner graphs

Microsoft Planner has a calendar showing all the due dates and graphs to follow progress, these were very helpful to visualize how the project was progressing. As this was the developer's first time writing almost completely on their schedule a lot of effort was put into using this planner to keep on track and to consistently update and add new tasks from day to day. These mini day to day planners were created using Tape which is a not making software that can be sorted into collections. The collections were Daily To-do, Document To-do and Weekly To-do. Using this made it easy to follow what was immediately needed and writing down everything helped keep the developer organised.

To manage larger sections of notes the developer used OneNote for journaling important items from meetings and articles. As well as documenting what had been coded that day and what issues were occurring within the code. This helped for managing the document write up as it meant there was already notes written down during the process compiling everything would be simpler.

7.2 Difficulties and Reflexion

As this was the first major project the developer had to do by themselves it was difficult at time to keep up working at a steady pace. Due to conflicting projects, the developer fell quite behind at one point in implementing the project. This happened as too much was put on their plate and they weren't allocating the correct amount of time to each task. After a week or so they got back on track though. Another difficulty was to know how long a piece of functionality will take to implement, this ended up leading to the application not being as far along as planned. The application and model still got built to a level where the developer was satisfied but knowing how long certain elements would take would've aided the process.

If this project could be started again knowing what I know now, I would've looked further ahead into how this application would work on the web. This is because I spent a lot of time studying CNN's and deep learning and how to create a face recognition model when I could've also been learning Flask. This wasn't needed until a month or two in so the developer started learning it then. In hindsight learning all of them together would've made the project run smoother.

8 Conclusion

The main goal of this project was to create an accurate face recognition model that could take attendance of people and integrate this into a web application. The goal of this was to create an attendance system that could improve manual attendance taking systems. Another goal of this was to explore how people feel about face recognition software, and how likely they would be to use an application that employs facial recognition. The main goals of this application were reached within the timeframe.

8.1 Summary of Chapters

At the beginning of the project, the requirements analysis was undertaken. The goal of this was to discover the necessary feature of the application. This was done by finding similar applications and seeing how they worked, how the models connected to applications. Other requirements included a user survey and user research with personas. This section was integral as one of the aims was to see how people react to this kind of application and would be the ideal user. Requirements modeling was about describing in depth what functions the ideal application would have and what technologies would be used along with detailing a test plan for the project.

After the requirement's gathering research was carried out on machine learning and how convolutional neural networks (CNN) work. This research detailed the various layers of the network and how they would work together to create the network and the reasoning behind using this. Following this, research on face recognition and applications was carried out to understand how face recognition works and how it is implemented into CNN models. The way this can be used to aid in attendance gathering was also researched alongside the research into the applications.

The design chapter details how this application would function after figuring out the requirements and researching each element for the application. The system architecture, model architecture and user interface were designed, and it was shown how these would all communicate together to reach the requirements of the project.

Following this the implementation of the project began. The application was built using the Python Flask framework as the backend and using Jinja2 templates in the frontend client side of the application. The server side contained the logic of the CNN recognition model, which was

built using Keras, and the attendance system built with the help of SQLite. OpenCV was also used to access the webcam for use in the application.

The following chapter was the testing chapter, which has some overlap with the implementation chapter as a lot of the model testing takes place throughout creating the model. This chapter also contains functional testing and user testing. The project management chapter detailed how the project was managed and what tools were used to help this through the development period.

8.2 Future Improvements

There are many future improvements the developer would like to see in this project. If there was more time, the developer would implement a live feed into the application to constantly get attendance, this was mentioned in the survey but due to time pressures was not feasible. To achieve this a camera would either be set up in a room on its own connected to a laptop or a raspberry PI. This would be connected to the server and every time a face was detected the model would run on it.

Also, in future there would be an admin dashboard that would have the ability for the user to state whether or not the predictions were correct and to run the training automatically through the interface. In this dashboard the admin could monitor the attendance and check if the predictions were all correct. If some were wrong, then this would be recorded, and the model retrained.

Another feature that could be added would be that when a prediction is made the user confirms whether it is correct or not and if it's wrong, they can say who they are, and that image is sent to be trained to make the system more accurate. Also with this the user would be able to say who they are and the correct name would be recorded for attendance.

8.3 Personal Takeaways and Project Achievements

Overall, the project achieved the main goals set out for it. It is a functioning attendance-based face recognition system, that contains an accurate CNN model. Other achievements were that the developer received real-life insight into how users would feel about facial recognition technology, and how to make users more comfortable when using it. The developer also gained a deeper understanding of how to create CNN's. This was integral to the project but is also a big personal takeaway. The project was implemented as a web application and successfully

integrated all necessary elements within Flask. This was an achievement as it was the first time the developer had used a Python framework and didn't originally know how this would work.

As this project allowed for a lot of time for the developer to do work as this project was developer-driven, a schedule had to be created, to get work done in a timely manner, which is a crucial part of project management, and an invaluable skill learned through this project. This project also helped with learning new soft skills, such as; knowing when a feature should be withdrawn due to technical insufficiencies, knowing when to go back to the drawing board, and researching alternative options if plans aren't going as expected.

9 Bibliography

Abate, A. F., Nappi, M., Riccio, D., & Sabatino, G. (2007). 2D and 3D face recognition: A survey. *Pattern Recognition Letters*, 28(14), 1885–1906.

<https://doi.org/10.1016/j.patrec.2006.12.018>

Alom, M. Z., Taha, T. M., Yakopcic, C., Westberg, S., Sidike, P., Nasrin, M. S., ... & Asari, V. K. (2018). The history began from alexnet: A comprehensive survey on deep learning approaches. *arXiv preprint arXiv:1803.01164*.

Alzubaidi, L., Zhang, J., Humaidi, A. J., Al-Dujaili, A., Duan, Y., Al-Shamma, O., Santamaría, J., Fadhel, M. A., Al-Amidie, M., & Farhan, L. (2021). (Abate et al., 2007) 8(1).

<https://doi.org/10.1186/s40537-021-00444-8>

Balcoh, N. K., Yousaf, M. H., Ahmad, W., & Baig, M. I. (2012). Algorithm for efficient attendance management: Face recognition based approach. *International Journal of Computer Science Issues (IJCSI)*, 9(4), 146.

Bussey, D., Glandon, A., Vidyaratne, L., Alam, M., & Iftexharuddin, K. M. (2017, November). Convolutional neural network transfer learning for robust face recognition in NAO humanoid robot. In *2017 IEEE Symposium Series on Computational Intelligence (SSCI)* (pp. 1-7). IEEE.]

Chauhan, R. K., Pandey, V., & Lokanath, M. (2018). Smart attendance system using cnn. *International Journal of Pure and Applied Mathematics*, 119(15), 675-680.

Daniyal, F., Nair, P., & Cavallaro, A. (2009, September). Compact signatures for 3D face recognition under varying expressions. In *2009 sixth IEEE international conference on advanced video and signal based surveillance* (pp. 302-307). IEEE.

Foot, K. D. (2017, February 7). A Brief History of Deep Learning - DATAVERSITY. DATAVERSITY. <https://www.dataversity.net/brief-history-deep-learning/>

Fukushima, K. (1980). Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36(4), 193–202. <https://doi.org/10.1007/bf00344251>

Grossfeld, B. (2020, January 23). Deep learning vs. machine learning: a simple way to learn the difference. Zendesk; Zendesk. <https://www.zendesk.com/blog/machine-learning-and-deep-learning/>

Gu, J., Wang, Z., Kuen, J., Ma, L., Shahroudy, A., Shuai, B., Liu, T., Wang, X., Wang, G., Cai, J., & Chen, T. (2018). Recent advances in convolutional neural networks. *Pattern Recognition*, 77, 354–377. <https://doi.org/10.1016/j.patcog.2017.10.013>

Gurucharan, M. K. (2020, December 7). Basic CNN Architecture: Explaining 5 Layers of Convolutional Neural Network | upGrad blog. UpGrad Blog. <https://www.upgrad.com/blog/basic-cnn-architecture/>

Hu, G., Yang, Y., Yi, D., Kittler, J., Christmas, W., Li, S. Z., & Hospedales, T. (2015). When face recognition meets with deep learning: an evaluation of convolutional neural networks for face recognition. In *Proceedings of the IEEE international conference on computer vision workshops* (pp. 142-150).

Krizhevsky, A., Sutskever, I., & Hinton, G. E. ImageNet Classification with Deep Convolutional Neural Networks (AlexNet) ImageNet Classification with Deep Convolutional Neural Networks (AlexNet) ImageNet Classification with Deep Convolutional Neural Networks.

Kulkarni, S., & Harnoorkar, S. (2020). Comparative Analysis of CNN Architectures. vol, 7, 1459-1464.

Lin, S.-H. (2000). An introduction to face recognition technology. *Informing Science: The International Journal of an Emerging Transdiscipline*, 3, 1–8. <https://go.gale.com/ps/i.do?id=GALE%7CA205363359&sid=googleScholar&v=2.1&it=r&linkaccess=abs&issn=15479684&p=AONE&sw=w&userGroupName=anon%7E34d1c40c>

Letchmunan, S., Hassan, F. H., Zia, S., & Baqir, A. Detecting Video Surveillance Using VGG19 Convolutional Neural Networks

M. Coşkun, A. Uçar, Ö. Yildirim and Y. Demir, "Face recognition based on convolutional neural network," 2017 International Conference on Modern Electrical and Energy Systems (MEES), 2017, pp. 376-379, doi: 10.1109/MEES.2017.8248937.

Paray, M., Tanquiamco, D., & Jandayan, C. (2020). Analysis and Design of Employee Attendance Monitoring Using Face Recognition System for Archempres Fruit Corporation.

Passricha, V., & Aggarwal, R. K. (2019). End-to-End Acoustic Modeling Using Convolutional Neural Networks. *Intelligent Speech Signal Processing*, 5–37. <https://doi.org/10.1016/b978-0-12-818130-0.00002-7>

Prithiviraj R. (2020). Automated Attendance System based on Facial Recognition. *Journal of Advanced Research in Dynamical and Control Systems*, 24(4), 124–132.
https://www.academia.edu/26763120/Automated_Attendance_System_Based_on_Facial_Recognition

Soltanpour, S., Boufama, B., & Wu, Q. J. (2017). A survey of local feature methods for 3D face recognition. *Pattern Recognition*, 72, 391-406

Singh, S., & Prasad, S. V. A. V. (2018). Techniques and Challenges of Face Recognition: A Critical Review. *Procedia Computer Science*, 143, 536–543.
<https://doi.org/10.1016/j.procs.2018.10.427>

Virgil Petrescu, R. V. (2019). Face Recognition as a Biometric Application. *Journal of Mechatronics and Robotics*, 3(1), 237–257. <https://doi.org/10.3844/jmrsp.2019.237.257>

Wang, J., & Li, Z. (2018). Research on Face Recognition Based on CNN. *IOP Conference Series: Earth and Environmental Science*, 170, 032110. <https://doi.org/10.1088/1755-1315/170/3/032110>

Zhou, S., & Xiao, S. (2018). 3D face recognition: a survey. *Human-Centric Computing and Information Sciences*, 8(1). <https://doi.org/10.1186/s13673-018-0157-2>

DeepAI. “Max Pooling.” DeepAI, DeepAI, 17 May 2019, deepai.org/machine-learning-glossary-and-terms/max-pooling. Accessed 8 May 2022.

IBM Cloud Education. “What Are Neural Networks?” Ibm.com, 17 Aug. 2020, www.ibm.com/cloud/learn/neural-networks. Accessed 8 May 2022.---

“What Is Supervised Learning?” Ibm.com, 19 Aug. 2020, www.ibm.com/cloud/learn/supervised-learning. Accessed 8 May 2022.

(PDF) Disaster and Pandemic Management Using Machine
https://www.researchgate.net/publication/347449937_Disaster_and_Pandemic_Management_Using_Machine_Learning_A_Survey

K-Means Clustering with Math - Towards Data Science. <https://towardsdatascience.com/k-means-clustering-for-beginners-2dc7b2994a4>

K-Fold | K-fold Averaging on Deep Learning Classifier. (2021, September 16). Analytics Vidhya. <https://www.analyticsvidhya.com/blog/2021/09/how-to-apply-k-fold-averaging-on-deep-learning-classifier/>

The, H. (2017, April 5). A guide to receptive field arithmetic for Convolutional Neural Networks. Medium; ML Review. <https://blog.mlreview.com/a-guide-to-receptive-field-arithmetic-for-convolutional-neural-networks-e0f514068807>

Admin. (2021). Machine Learning and Facial Recognition. Pxl-Vision.com. <https://www.pxl-vision.com/en/blog/machine-learning-and-how-it-applies-to-facial-recognition-technology#:~:text=Facial%20recognition%20is%20a%20technology,in%20a%20pre-existing%20database>

Admin. (2021). Machine Learning and Facial Recognition. Pxl-Vision.com. <https://www.pxl-vision.com/en/blog/machine-learning-and-how-it-applies-to-facial-recognition-technology#:~:text=Facial%20recognition%20is%20a%20technology,in%20a%20pre-existing%20database>

Team, K. (2022). *Keras: the Python deep learning API*. Keras.io. <https://keras.io/>

Machine learning education | TensorFlow. (2022). TensorFlow. https://www.tensorflow.org/resources/learn-ml?gclid=Cj0KCQjw1N2TBhCOARIsAGVHQc4MC9ZLnlS65agtgaC0pZdH15QaDU7jzbNmzy7-RmZs-nL0g5lCFfcaAi3_EALw_wcB

AI. (2020). FACE RECOGNITION + ATTENDANCE PROJECT | OpenCV Python | Computer Vision [YouTube Video]. In YouTube. https://www.youtube.com/watch?v=sz25xxF_AVE

10 Appendices

10.1 Appendix A – survey for requirements and excel answers

https://docs.google.com/forms/d/1pt8NwLwSOZao25F5VunyDfOez66dlfl5Pecgi_9sPQw/edit#responses

Timestamp	By consenting to this you	What age are you?	How much do you know about	Are you concerned about	Additional comments based on	Do you agree with the following organisations using	Do you agree with the following
20/01/2022 12:52:15	I do consent	23 - 27	Quite a bit (Familiar with it)	Depends		Agree	Disagree
20/01/2022 12:56:20	I do consent, I do not consent	18 - 22	Not much	No		No opinion	Agree
20/01/2022 12:57:16	I do consent	23 - 27	Quite a bit (Familiar with it)	Yes		Agree	Strongly disagree
20/01/2022 12:59:20	I do consent	23 - 27	Not much	No		Strongly agree	Agree
20/01/2022 12:59:41	I do consent	18 - 22	Quite a bit (Familiar with it)	Yes		Disagree	Strongly disagree
20/01/2022 13:03:22	I do consent	23 - 27	Quite a bit (Familiar with it)	Yes	Photo gallery applications	Agree	Disagree
20/01/2022 13:03:27	I do consent	23 - 27	Not much	Depends		Strongly agree	Agree
20/01/2022 13:03:46	I do consent	23 - 27	Not much	Depends	I'm concerned with data from	Agree	Strongly disagree
20/01/2022 13:08:20	I do consent	18 - 22	Not much	Yes	Terrified of Technology and	Strongly agree	Strongly agree
20/01/2022 13:09:18	I do consent	23 - 27	Not much	Yes		Strongly agree	Disagree
20/01/2022 13:11:09	I do consent	18 - 22	Not much	Yes	Most of the time used for	Agree	Disagree
20/01/2022 13:12:44	I do consent	18 - 22	Not much	Depends		Strongly agree	Agree
20/01/2022 13:13:30	I do consent	18 - 22	Quite a bit (Familiar with it)	Depends		Strongly agree	No opinion
20/01/2022 13:15:24	I do consent	23 - 27	Not much	No		Agree	Disagree
20/01/2022 13:30:17	I do consent	23 - 27	Quite a bit (Familiar with it)	Yes		Strongly disagree	Strongly disagree
20/01/2022 14:00:01	I do consent	18 - 22	Not much	Depends	It can be useful and I'm not	Strongly agree	Agree
20/01/2022 14:13:06	I do consent	23 - 27	Quite a bit (Familiar with it)	No		Strongly agree	Strongly agree
20/01/2022 14:29:48	I do consent	32 - 40	Not much	No		Strongly agree	Strongly agree
20/01/2022 14:38:28	I do consent	41+	I know my phone uses it	Depends	If it's being used widely by	Agree	Disagree
20/01/2022 14:40:19	I do consent	28 - 32	Not much	Yes	Think facial recognition is	Strongly disagree	Strongly disagree
20/01/2022 14:56:39	I do consent	18 - 22	Not much	Depends	Like any technology there	Agree	Disagree
20/01/2022 15:19:30	I do consent	23 - 27	Quite a bit (Familiar with it)	Depends	Facial recognition used to	No opinion	Disagree
20/01/2022 15:23:05	I do consent	18 - 22	Not much	Depends		Agree	Agree
20/01/2022 15:33:40	I do consent	18 - 22	Not much	Depends		Agree	Agree
20/01/2022 15:38:08	I do consent	23 - 27	Nothing at all	No	Makes my life easier when	Strongly agree	Agree
20/01/2022 15:47:59	I do consent	18 - 22	A lot (I've researched how	Yes	As tempting and as exciting	Disagree	Strongly disagree
20/01/2022 15:54:10	I do consent	23 - 27	Quite a bit (Familiar with it)	Depends	Not concerned for convenience	Strongly agree	Strongly agree
20/01/2022 16:03:14	I do consent	23 - 27	Quite a bit (Familiar with it)	Depends	Can be used badly, but all	Agree	Disagree
20/01/2022 16:20:14	I do consent	23 - 27	Quite a bit (Familiar with it)	Yes		No opinion	Strongly disagree
20/01/2022 16:34:02	I do consent	18 - 22	Quite a bit (Familiar with it)	Depends	None	Agree	No opinion
20/01/2022 16:42:53	I do consent	18 - 22	Nothing at all	No		Agree	Disagree
20/01/2022 16:58:53	I do consent	23 - 27	Not much	Yes		No opinion	Strongly disagree
20/01/2022 17:13:03	I do consent	18 - 22	Not much	Yes		Strongly agree	Agree
20/01/2022 18:37:41	I do consent	28 - 32	Not much	No		Strongly agree	Agree
20/01/2022 20:17:51	I do consent	18 - 22	Quite a bit (Familiar with it)	Depends	1) It depends on what the	Agree	Strongly disagree
20/01/2022 20:34:41	I do consent	18 - 22	Not much	I know I should be but I	just float along	Agree	Agree
20/01/2022 20:43:54	I do consent	23 - 27	Quite a bit (Familiar with it)	Depends	China	Disagree	Disagree
20/01/2022 23:11:38	I do consent	18 - 22	Quite a bit (Familiar with it)	Depends	there's a place for it howe	Strongly agree	Agree
21/01/2022 14:41:53	I do consent	18 - 22	Nothing at all	No		Agree	Agree
21/01/2022 15:35:30	I do consent	18 - 22	Quite a bit (Familiar with it)	Yes	Privacy concerns, e.g. mil	Agree	Disagree
22/01/2022 09:02:35	I do consent	23 - 27	Not much	Yes		Disagree	Disagree
22/01/2022 18:16:19	I do consent	18 - 22	A lot (I've researched how	Depends	It would really depend on	Agree	Disagree

10.2 Appendix B – user interface and system model designs

<https://www.figma.com/file/MX2bXlqJwmN8SjAR8CfcJK/Untitled?node-id=0%3A1>

10.3 Appendix C – code repository

https://github.com/Clareob5/Major_Project_faceRec

10.4 Appendix D – Interim Presentations

Interim Presentations

10.5 Appendix E – Microsoft Planner

https://tasks.office.com/iadt.ie/Home/PlanViews/3ndrnd_u_ESUUcb5xwI6QZYAEfIB?Type=PlanLink&Channel=Link&CreatedTime=637876446629400000