# CSY3025: Artificial Intelligence Techniques

## Assignment 2: Face Scanning Attendance Register

### University of Northampton

Student Number: 18410839

Student Name: Emily Elizabeth Fletcher

Submission Date:

Word Count:

Contents

[Section 1: Links and Video Demo 3](#_Toc135521358)

[Dataset Link 3](#_Toc135521359)

[Demo Video Link 3](#_Toc135521360)

[Section 2: Introduction 3](#_Toc135521361)

[Section 3: Problem Analysis and Background Research 3](#_Toc135521362)

[Problem Analysis 3](#_Toc135521363)

[Domain Research 5](#_Toc135521364)

[Comparable Systems 5](#_Toc135521365)

[Deep Learning Frameworks Research 9](#_Toc135521366)

[Final Defined Objectives 11](#_Toc135521367)

[Ongoing Research 12](#_Toc135521368)

[Section 4: Program Design 13](#_Toc135521369)

[Software Architecture 13](#_Toc135521370)

[Wireframes 15](#_Toc135521371)

[Section 5: Building a Data Set and Deep Learning Network 16](#_Toc135521372)

[Data Set 16](#_Toc135521373)

[DL Network (Combined with Training and Evaluation 18](#_Toc135521374)

[Model 1 , Starting Model 18](#_Toc135521375)

[Model 2, Model Refinement 33](#_Toc135521376)

[Model 3, Model Refinement 40](#_Toc135521377)

[Section 6: Testing 41](#_Toc135521378)

[Test Plans 41](#_Toc135521379)

[Section 7: Feature Summary 41](#_Toc135521380)

[Section 8: Discussions and Conclusion 41](#_Toc135521381)

[References 42](#_Toc135521382)

[Appendix 1: Source Code 43](#_Toc135521383)

[1.1 Training Models (Complete) 43](#_Toc135521384)

[1.2 Application Code 83](#_Toc135521385)

[Appendix 2: Ongoing Commit Log 102](#_Toc135521386)

## 

## Section 1: Links and Video Demo

#### Dataset Link

The original DataSet was stored on Google Drive so it can be easily integrated with the Google Colab. However, so the marker can easily access this the author has uploaded it to FigShare. The public link for this is as follows.

<https://figshare.com/articles/dataset/Zip_Backup_of_DataSet/23006363>

#### Demo Video Link

## Section 2: Introduction

The following assignment aims to demonstrate the authors understanding of modern AI techniques. The author aims to show that they understand the basics of Artificial Intelligence and Machine Learning theory by outlining their problem-solving approach given a scenario based on a real-life application of AI.

The goals of this assignment are as follows.

1. Complete Problem Analysis on the given brief
2. Background Research on Existing Software and Deep Learning Networks
3. Creation of a Deep Learning Network including Dataset and Network Structure (Primary) and Companion Application (Secondary)
4. Complete product testing
5. Conduct a final discussion of findings.

## Section 3: Problem Analysis and Background Research

### Problem Analysis

The author was given a basic brief outlining of the assessed task. In order to show their understanding of this brief they have included the scenario and then annotated key points.

Provided Scenario:

“Your Task is to design and implement a deep-learning image classifier that that can recognise students in the classroom from close-up facial images to register their attendance.

The classifier should have the capability of handling at least five different people including yourself with a reasonable performance. You should discuss your choice of performance metrics. You are responsible to develop the image dataset for machine learning and testing. You must critically analyse any bias and ethical challenges related to your dataset and model design.

To achieve a higher grade, you are expected to design and implement additional features.

Examples of additional features are (But are not limited to):

* Handling more people (>5)
* Handling multiple people in the same shot
* Inclusion of different demographic groups
* An exceptional performance demonstrated through “in the wild” testing
* Implementation of user interfaces / user applications

Table 1: Scenario Analysis

|  |  |
| --- | --- |
| Point of Interest | Authors Observation |
| “Deep Learning Image Classifier” | Type of model needed, assigns labels to images based on characteristics or features present in them. |
| “Close-up facial images to register their attendance.” | Image recognition is based on facial recognition only, nothing relating to additional factors such as height and body language. |
| “Handling at least five different people including yourself with a reasonable performance.” | Minimum requirements are to recognise five people within a quick time with minimal mistakes made. |
| “Other students’ identities should be anonymised”. | Any training images used should not include identifiers such as name badges, university identifiable logos. Ethics should be considered when using external images. |
| “Discuss your performance metrics” | Ensure performance metrics are defined and included detail these in the testing phase. |
| “Responsible to develop the image dataset for machine learning and testing” | Cannot use a pre-made dataset sourced online, Author must adapt and develop their own. |
| “Critically analyse any bias and ethical challenges” | Ensure ethics are considered in the product creation, include ethical considerations in the final discussions and conclusions. |

The above table has allowed the author to start to generate an idea of what key features need to be included within the assignment. This acts as a good starting point for the assignment as it outlines the minimum expectations for the assignment.

From the following problem analysis, the author has used divide and conquer to establish the following basics requirements for the proposed software solution.

Table 2: Mandatory Features - Divide and Conquer

|  |  |
| --- | --- |
| Feature Needed | Steps Required |
| Recognise Facial Input | * Live Webcam Feed * Highlight Faces so the user can see facial input |
| Recognise specific faces | * Train a data sets on authorised users * Compare video frames to identify users |
| Program can be used in a classroom setting | * Easy and quick to use system * Optimised code so that the system does not run with delays and can be used efficiently in a classroom environment |
| Allow users to record their attendance | * Show visual output that allows the user to log their attendance if identified correctly, such as a name stamp of people identified and option to submit attendance. |
| When a user records their attendance there is a log made for this. | * A timestamp is saved to a log of the person logging their attendance with a session header. |
| Live Working Version | * The data set used needs to be based on real people that can be used to test the product, ruling out datasets that are solely focused on celebrities or synthetics. |

After the divide and conquer stage, the author now has an idea of the first steps they should make with the project. Now the author is going to move onto the domain research stage. For this stage the author is going to look at products that are already available on the market. This will allow the author to develop a better od idea of what they could include as additional features, or ideas they might have missed.

### Domain Research

For the domain research the author has decided to look at comparable systems as well as deep learning frameworks. The author will discuss comparable systems first.

### Comparable Systems

Comparable systems allow the author to look at systems that already exist commercially and solve similar problems as the ones outlined in the brief. This research allows the author to see similar solutions as well as inspire the author to include features they may not have thought of. This could help them extend their work into the higher-grade boundaries with their additional features.

Initially the author noted there was a struggle in finding previous software that has been used for this purpose. After further researching into blog posts and possible news sources the author noted that the assignment topic was a big trend during covid. Especially as educational settings were trying to go contactless, as well as the challenges of facial recognition in relation to masks.

The author widened their search for comparable products by looking at facial recognition systems in a business setting instead.

#### Product 1: Jibble

Jibble is a free to use product for an unlimited number of users. It uses whatever product the user would like, giving the option to register attendance on the users phone or on any terminal of choice. The application boasts that it reliable and quickly registers attendance with ease.

Users provide Jibble with multiple angles of their face which allows the ai to generate a complete face map. The model used by Jibble develops over time, it adds to the data set every time the user clocks in with a photo. Allowing continuous update of the data.

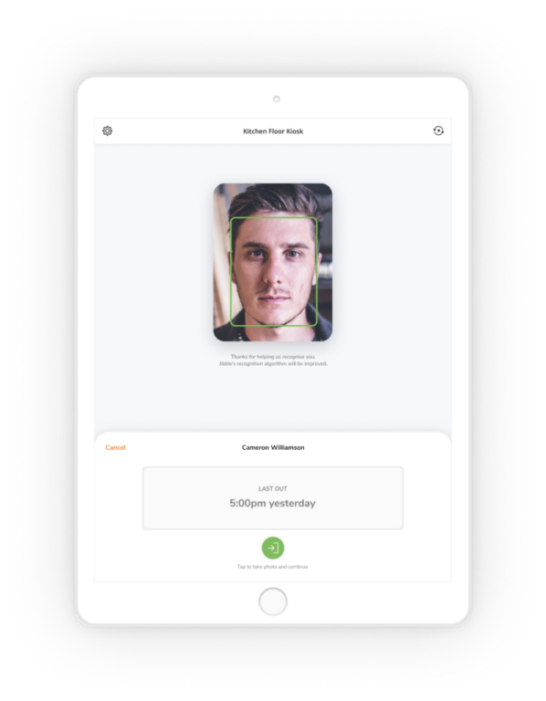
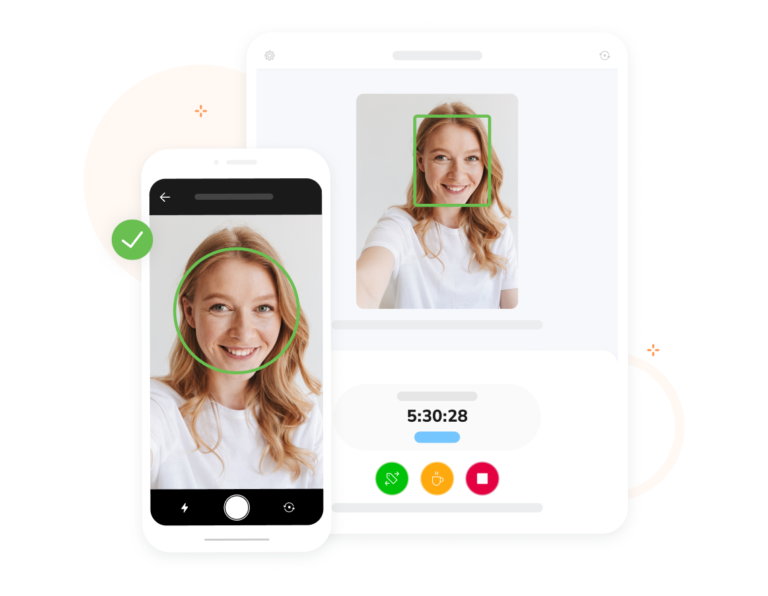


Figure 1:JIbble Screen Captures

Table 3: Jibble Advantages and Disadvantages

|  |  |
| --- | --- |
| **Advantages** | **Disadvantages** |
| Saves a photo as well as records attendance, allows checks of any false flags. | Use of own hardware can result in a mix of performance results. |
| Continuously learning AI, the more someone clocks in the more data that the model can be trained on. | Dependent on Wi-Fi as face data is stored on the cloud. |
| Works on users’ own hardware. | Extra features and storage can un-nerve employees as they may feel like they are always being watched. |
| Can product a log of last attended. |  |
| Face spoofing protection, using depth estimation technology |  |
| Has offline sync using employees GPS if Wi-Fi connection is temporarily not available |  |
| In built data analytics |  |
| Push notification for employees if they need to check attendance in a specific place. |  |

Overall, the author is on the fence about this software solution. The software solution prides itself on an easy to use piece of software that allows employers to track the location of their employees as well as assign them specific areas where they should be. This extra layer of tracking employees does bring up some ethical concerns for the author. While the author agrees that employers need to know if their employees are in the right place there are some times where the employee has a right to privacy. The author is not sure whether this can be maintained with the tracking features of this solution.

The author agrees with the point that interface design is important and an important consideration, as if the solution is hard to use or has limited performance then it is unlikely to be used by users.

The author was impressed by the continuous learning the model uses and the fact that it gathers more information about users to improve the identification results in the future. This may be something that the author looks into should they have time to implement any additional features.

#### Product 2: AccuTime Face Recognition – Pro

This product is a terminal based commercial product that allows employees to check into work using facial recognition as well as finger biometrics. The product details do not discuss much in terms of how the facial recognition of this product works so it’s hard for the author to assess the methods used.

Compared to the previous product this product is a more professional based workplace product, however this costs significantly more than the previous product which was free. There is also an employee limit on this solution, where you can have 50 or 100 employees. Anything larger and the requester has to get in touch with the manufacturer.

This product has significantly less features than the product before, however the author does not think the ethical concerns that the previous product has are present in this solution.



Figure 2: AccuTime Terminal

|  |  |
| --- | --- |
| **Advantages** | **Disadvantages** |
| Easy to use | Bespoke terminal ups the cost of the device |
| Performance is consistent as all terminals have the same hardware | Expensive solution |
| Offers other features such as a calendar and biometrics information. | Limited number of users |
|  | No explanation of AI technology. |
|  | No sources on performance outside of user reviews. |

Most products the author found invoke a similar system to this one. This is where the company provide all the hardware units as well as all the prebuilt software and AI models. While this does not present the ethical concerns that the previous product does due to its much more limited staff tracking, there is other concerns that the author has thought of. The bespoke unit means that it may be hard for the purchaser to upgrade or maintain the system themselves. This can result in hidden charges that may not have been thought about when the system was first implemented. There is also a limited number of users for this system. The purchaser can enquire to buy more, however this could be a turn away factor for businesses that plan on quick expansion in the future.

#### Product 3: ClockRite Facial Recognition Biometric Clocking System

This is another system that has the main terminal as well as the software bundled together. Unlike the previous product this product has a much smaller terminal. This may make it a better choice where space is limited. This product can also support a maximum of 800 staff members so this can be used by small and large businesses alike. The product advertises that it uses a *“Biometric Facial Recognition”* system. Implying that this system uses measurements on a persons face to create and save patterns. The system also offers that the system can be used with both wired and a WiFi set up which increases its versatility. There are also additional features advertised such as a printable attendance register as well integrated holiday and absence plans. There is a slight ethical concern from the author, the terminal also has the feature to download facial clocking data. The author would want to be reassured that this cannot be performed by just anyone as this could present a security risk.



Figure 3:ClockRite Terminal

|  |  |
| --- | --- |
| **Advantages** | **Disadvantages** |
| Smaller Terminal | Could be some concerns over downloading of facial data |
| Supports both WiFi and Cabled use | Small lighter units could be stolen |
| Supports up to 800 users | No mention of whether downloaded data is encrypted or protected, this could be an issue if the computer the data is later transferred to is not secure. |
| Initial registration can be completed at the unit | No mention of specifics Data sets or pr-trained models, not advantageous to the author. |
| Employees can clock in up to 16 times in one day. |  |
| Can also be accompanied by a security pin for extra security. |  |
| Facial profiles are stored in a mathematical algorithm that would require de-encryption. |  |
| Each clocking in takes no more than a couple of seconds in normal operation |  |

This is the second terminal system the author has analysed. This is because the author struggled to find systems that did not use terminals. Overall, the author prefers this option to the second product. This is because this product has more details around it that the author can analyse, as well as more flexibility in the systems it offers. Such as offering both wired and WiFi and offering multiple staff limits. The author also likes additional features such as multiple attendances by uses and additional features such as security pins.

Overall, the author has gathered that the provided brief is fairly unique in terms of off the shelf software. However, it should be considered that this does not mean the product does not exist. Instead, it should be considered that there could be multiple bespoke systems that are not advertised.

The author has learnt that all of these comparable products offer much more than facial recognition and attendance recording. They also offer reports as well as most boasting about extra security. These could be considered by the author as additional features they could try to imitate in their project to try and obtain higher grades.

Research has also helped the author understand the logistics of the how they could make their solution. This has helped the author feel less lost with the assignment. All the products researched talk about facial data and this has helped the author realise they need to develop a system than can first identify a face, save a temporary image as a face and then compare this to an author-built dataset of users for their own system.

### Deep Learning Frameworks Research

Part of this project involves the use of Deep Learning Frameworks. In order to show the authors understanding of what a DLN (Deep Leaning Network) is they conducted additional research on what DLN’s are. As well as highlighting a choice of what DLN they will use in the final project.

Nvidia (2023) define a DLN as *“building blocks for designing, training, and validating deep neural networks through a high-level programming interface.”*

This suggests that DLN’s are pre-built way to train user developed neural networks. This allows developers to focus on their data sets and product creation rather than spending resources on developing an algorithm to interpret the users Dataset.

Another source from ExxactCorp (2021) discusses the benefits of using Deep Learning Frameworks, this has helped the author develop a deeper understanding of how they work as well as the benefits. The article discusses benefits such as *“readily available libraries for defining layers, network types and common model architectures”* as well as *“supporting computer vision techniques”* This has helped the author understand what they need to look at next, including what options are favoured and relevant to the authors project, and what to look at next. Such as layers and architectures.

From introductory research the author has learnt there are many different ways to implement deep learning networks. To help condense this learning down, the author has presented it in a table form.

Table 4: Deep Learning Networks Comparisons

|  |  |
| --- | --- |
| Deep Learning Network | Points |
| TensorFlow | Open Source by Google  Can support multiple languages (Python is most popular)  Best suited for experimenting  Can implement graphs  As supported by Google is supported and has quick updates  Can be slow compared to other competitors  Cost Free Software Library  One of the most popular deep learning frameworks |
| PyTorch | Open Source by Facebook  Aimed to be used for research and the progress up to a full product deployment  C++ Front End on top of a Python interface  Operates dynamically so you can make changes to the model during the building process. (Computational work is distributed among multiple CPU or GPU Cores.  Has a relatively short learning curve |
| Keras | Runs on top of TensorFlow, Acts as an interface for the TensorFlow Library  Supports parallelism, leads to accelerated training time  Low level computations are not a strength of Keras  Excellent for beginners  Allows for fast experimentations  Has a collection of pre-trained models  Often used for rapid prototyping  Errors are not always easily identifiable  also, Open Source  Priorities developer experience |
| Sonnet | Built on top of TensorFlow  Allow custom building  Can build modules that declare other submodules |
| MXNet | High scalable  Fast model training  Multiple language support  Less open-source community support compared to others  Lack of community support means bug fixes and feature updates take a longer time due to a shortage of community support. |

There are many frameworks available, however the author has chosen five main ones to analyse. They have noted the key points to help them decide as well as being led by the work taught in class so far. The resources for this were Kechit Goyal (2022), Vidushi Meel (2023) and Project Pro (2023).

The author has chosen to use a combination of TensorFlow as well as Keras. The use of Tensor Flow is because the this is open sourced and has the most amount of community support available. It is also suited to experimenting which is suitable for the author as they have a lot of experimenting to do within the project as they are new to completing Machine Learning Tasks. It is also what they are most familiar with as it’s what has been used so far in class. They have chosen to combine this with Keras as is acts a beginner friendly interface for Tensor Flow.

### Final Defined Objectives

The final objectives are a list that the author can follow to achieve the final project outcomes. The author has split these into functional requirements and performance requirements.

Functional requirements are requirements that relate to specific features and identified features. Performance requirements are related to the final performance of the proposed software solution. These are also the final features that will be included with the system. The author has included just what they consider core to their first thoughts and design. So, there may be additional features in the final product that are not listed here.

Functional Requirements

* Input of a live video feed with moving frames
* The system recognises a face and gives the user an option to log attendance when a face is detected
* The user can easily navigate the software with a simple GUI
* When the user registers their attendance, a live frame is captured and temporary stored
* The captured frame it passed through an author created data set and searched for a match
* If there is a match the user can register their attendance and they added to a list.

Performance Requirements

* The system can recognise at least five individuals
* Face recognition is correct 8 / 10 times consistently
* Face identification is correct 8 / 10 times consistently
* There is no visible delay between frame data being shown and the user viewing it
* There is minimal latency (3 seconds at longest) between the user clicking search and getting a result from the data set
* The application has efficient code, so the final program executes on most standard PC’s (i3 Processor or Above)

### Ongoing Research

In order to demonstrate that the author has continuously developed their skills this table has been included. The table included below demonstrates this as it shows what the author has been researching around the task, even if not explicitly used and referenced.

Table 5: Ongoing Research Table

|  |  |  |
| --- | --- | --- |
| **Resource (And Short Citation)** | **Topic Covered** | **Author Findings** |
| Face Recognition Python: Popular Techniques and Libraries  Bonani Bose (2023) | Python Face Recognition Libraries | OpenCV is one of the most popular libraries for facial recognition. The resource also discusses details about how OpenCV’s facial recognition works.  The author has used OpenCV Previously for a different computer vision project. |
| Everything you need to know about VGG16  Rohini G (2021) | How the VGG16 Model Works | VGG16 us a type of CNN, often considered one of the best computer vision models.  Object detection algorithm and classification algorithm.  Resource includes layers and model layout. |
| Beginner’s Guide to VGG16 Implementation in Keras | VGG16 Introduction | How to utilise the VGG16 Model, including importation, passing data initialising, and compiling the model. |
| Deep Learning Fundamentals – Classic Edition  Deep Lizard(2017) | Different Layer Types Within Neural Networks | The reasoning for different layers, common layer types including Dense, Convolutional, Pooling, Recurrent and Normalisation Layers as well as introduction to defining these. |
| 7 Types of layers you need to know in Deep Learning and how to use them  Tom Keldenich(2021) | Downloading Models on Colab | More details about layer types with a simpler explanation of what they are best used for. |
| How can I download this model from Colaboratory  Korakot(2018) | Solution to a specific code question on Stack Overflow | How to download a complete model from Google Colab. |
| What is Overfitting? AWS(2023) | Common Overfitting Problems | Overfitting occurs when the model does not generalise and fits too closely to the training data. There can be several reasons for this, training data size is too small, large amounts of noisy data, the model trains too long, the model complexity is high, so it learns noise. |
| What is Overfitting in Deep Learning  Pragati Baheti (2021) | Common Overfitting Problems | Discusses another way to look at Overfitting, such as “These models fail to generalize and perform well in the case of unseen data scenarios, defeating the model’s purpose”  Discusses comparisons to underfitting. As well as methods to detect and remove overfitting. |
| Regularization in Machine Learning  Alind Gupta (2023) | Regularization in Machine Learning | Explanation of Bias and Variance within Machine Learning. Explains difference types of Regularization as well as mathematical equations behind them. |
| Regularization in Machine Learning (With Code Examples)  Dataquest(2022) | Regularization in Machine Learning | Discusses how regularization restricts a model to avoid overfitting by shrinking the coefficient estimates to zero, by adding penalties to the model. Unlike the above resource this resource was used to help understand code through its code examples. |
| What is early stopping>?  Anusheh Zohair Mustafeez (2023) | Early Stopping in Machine Learning | How Early stopping can be implemented to stop overfitting and types of early stopping. |
| Python – GUI Programming (Tkinter) TutorialsPoint (2023) | Tkinter Guis | Introduction to GUIS including windows, frames and common widgets that can be used in Tkinter. Also covers geometry management. |
| Python Tkinter Spinbox  JavaTPoint (2023) | Tkinter Spinners | Gives the author an example of how to use Spinboxes and how to implement this into code within Tkinter. Used for the first frame so the author can select a time frame. |
| Reading and Writing to text files in Python  GeeksForGeeks (2023) | File Reading and Writing in Python | Details how to access, read and write to files in Python. Used for logging students’ names and late status. Includes code examples. |
| Datetime – Basic date and time types  Python Software Foundation (2023) | Date Times in Python | Docs detailing all the ways the author can use Datetime within Python. Including different types of DateTime objects and formats for it. |
| Python – Tkinter Frame | Understanding Frames in Tkinter | Understanding how to set up frames in Tkinter, research needed as the author needed a replacement for frames to stop the user navigating out of the app loop. |

## Section 4: Program Design

### Software Architecture

Software Architecture details how the author predicts that the user can interact with the software and the actions that the software will take. They are useful before coding as it gives the developer smaller goals to hit rather than just developing an entire process, which can be overwhelming.

The author has developed a simple chart with only the core requirements included. This is to divert the author away from the risk of developing all the extra features rather than the core. It also helps simplify the diagram. A busy software architecture diagram may highlight many features but becomes hard to read. Hard to read architectures can result in development mistakes and project burnout.

The author has considered small errors in this design by including what would happen if no input were found or a match is not found. However, the author plans to include more error casing in the code itself. (Such as if there is no video input found). However, to keep the diagram simple the author has chosen not to include all their error cases on the diagram as they just wanted to show the core.

The diagram works by taking a live feed input and a pre-trained dataset. When a person’s face is detected a frame from the live video is temporarily saved. This frame is then fed through the data set to find a possible match. If a match is found, then the students name is returned, and they are given the option to register their attendance. When the attendance process is ongoing the live feed is paused. This is to prevent user error with the software and reduce the chances of un-caught errors.

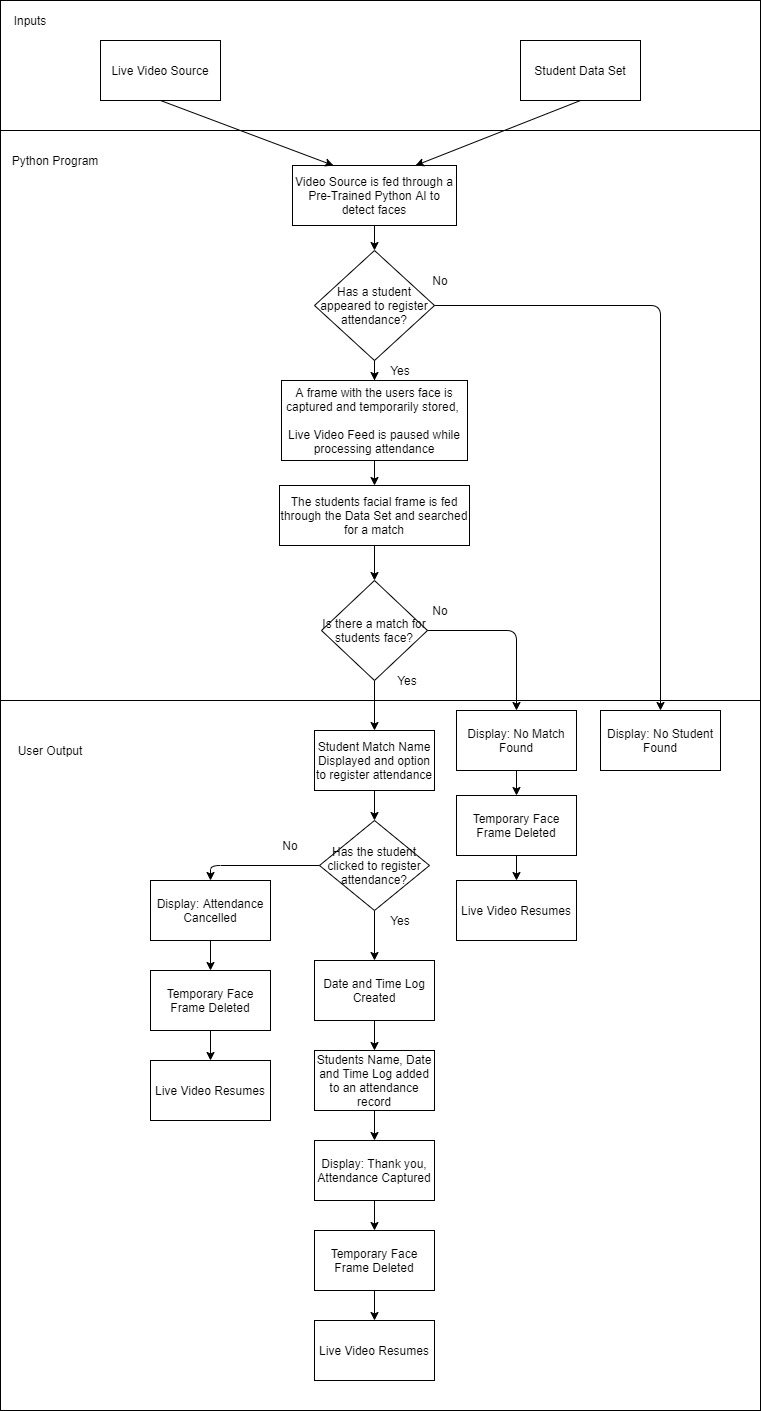


Figure 4: Core Structure Architecture

### Wireframes

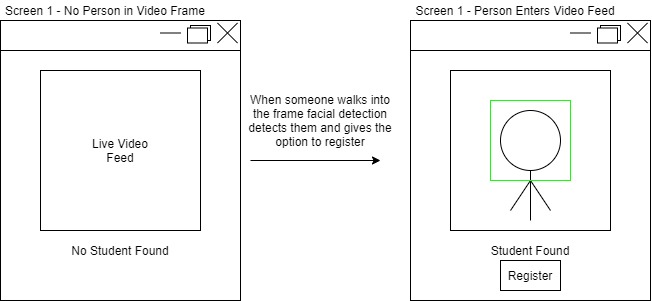
The author has made their wireframes based on the architecture model designed previously. They chose only to make the simplest wireframes as they did not want to spend a lot of time on this stage. This is because the GUI aspect of the assignment is optional. However, the author thinks designing a GUI is important as the users need to be able to interact with the final product and a console output is not very user friendly.

Figure 5: Screen One Wire Frame

In the first set of wireframes there is a demonstration of the first screen. When the user is not in the screen, they cannot register their attendance. This is to prevent false registration and registration spam which could break the system and cause it to crash.

This is planned to be implemented using OpenCV’s face detection library. When a student is in the frame the user is told that a student is found, and they can press register. When they click register this takes them to a second screen.

There are no tabs in the program, this is to prevent the user from breaking the expected order in the application and re-registering while a frame is being processed. This could also break the system.

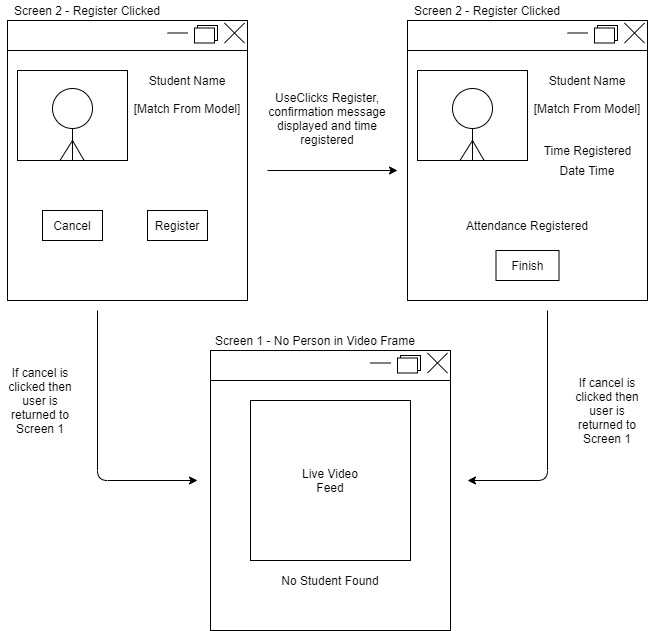


Figure 6: Screen Two Wire Frames

The second set of wireframes occurs when the user presses register. It shows the captured frame and a prediction of the person. Is there the option to cancel if the person identified is wrong so a new frame can be captured. There is also a cancel just in case the person chooses against attending. If cancelled the app returns to the first screen, this allows the process to start again for the next user.

If register is clicked, then a time stamp with the time the user registered is created. This allows for a log of all the times a student logs in. This in the future can be used to see if students register late. Then when the user clicks finish it returns them to the first screen.

## Section 5: Building a Data Set and Deep Learning Network

### Data Set

In part section three the author analyses the brief and notes that the dataset needs to include five people including the author. This tells the author that the dataset has to be a custom dataset and the author cannot use one solely sourced online.

The author has decided to stick with the base size of five people. This is because creating a custom dataset can be very time consuming, as the author has to collect all the data to be used themselves. There is a very fast approaching deadline for the student, so it is important for them that as much time as possible is spent on training the dataset.

The other concern with the author creating their own dataset is ethical concerns. In order to use real people, it is important to get consent and ensure that their images are stored securely. This is to keep in align with the Data Protection act 2018 (Gov.Uk, 2018) which details how data should be stored and used. This includes the eventual deletion of the project.   
  
The author has chosen to anonymise the users as part of their project ethics. This can protect those who give their data as it reduces their trackability. The project will also not be hosted publicly online as part of this. The authors anonymisation process will involve ensuring that student’s name is not used in the main application. Another step of the anonymisation process would be to ensure that there is no way to identify where the picture was taken. Such as removing university id lanyards or name badges before taking the pictures.

For the number of images per person the author has requested at least a hundred This is so the author can do an even split of images and account for any images that may not be up to a good standard and to account for people taking less than a hundred. The author used seventy images in total. This was because one of the five people did not provide enough images.

It is always important to ensure there is an even number of images as if you have more images in a folder you accidently create a bias for that person. This means the model is more likely to detect the person with the most images and therefore the model will train mainly off that person.

I have aimed for a split of fifty images in the training folder, ten in the validation folder, and ten in the test folder. This is to have a majority of roughly 70% in the training folder. Having a larger testing set allows for more patterns to be detected and allows the chances of coming in to contact with more differences, such as lighting changes and differing shadows. It also prevents overfitting, which is where the test set becomes too good at recognising specific patterns but does not respond well to new data.

Ideally having more photos would produce better results however this would take a long time to collect. Therefore, the author plans to use Data Augmentation to artificially inflate the data set. This will apply transformations to the images used such as rotations, zooms and flips. This results in more images to train from.

The author considered using online datasets to help train the model on common facial expressions.

However, this can cause ethical and licensing issues. Most online face recognition datasets are built from photos that are publicly available on the internet and are often of famous people. This is because there are multiple pictures of famous individuals often with different lighting and outfits. However, this can cause issues as there is no way to know if the photo is consensual due to poor paparazzi ethics, as well as who to credit. You would have to credit every photo within your dataset if you used professional imagery.

Another consideration was using a Synthetic Dataset. A data set by Microsoft called DigiFace-1M. This Data Set doesn’t have possible ethical issues like other online ones as all the faces in this set are generated randomly. However, this can cause issues in the training. For example, if I trained more than five synthetic faces alongside my others then the model would develop a bias for synthetics and may skip key features that appear in humans but not in synths. The other issue is the quality of the Synths. The Synths have to closely resemble humans otherwise the model will start to learn patterns that don’t exist in real humans like odd facial expressions. This can reduce the overall accuracy of the model.

<https://www.microsoft.com/en-us/research/publication/digiface-1m-1-million-digital-face-images-for-face-recognition/>

The author experimented with Synths but decided against using them as they reduced the accuracy of the overall model.

### DL Network (Combined with Training and Evaluation

Rather than over explain a final model, for this section the Author has chosen to detail how their first developed working model works. This is to show they had a starting base. Then they would like to demonstrate their growth by showing what they changed to improve their model. They hope that this demonstrates to the marker that they have experimented with the model and took a trial-and-error approach to learning. Rather than random chance of coming across a model that works. Experimentation is very important in model generation as each model is made to purpose and unique so it’s training will be different each time.

### Model 1 , Starting Model

Model one marks the first model that the author created that worked and predicted a set of values that could be worked with. This model was rudimentary with only a few layers. The author’s idea was that changing multiple parts of the model without a working model would make a much harder development as the author would not be able to deduce what parameters were actually negatively or positively inflecting on the model.

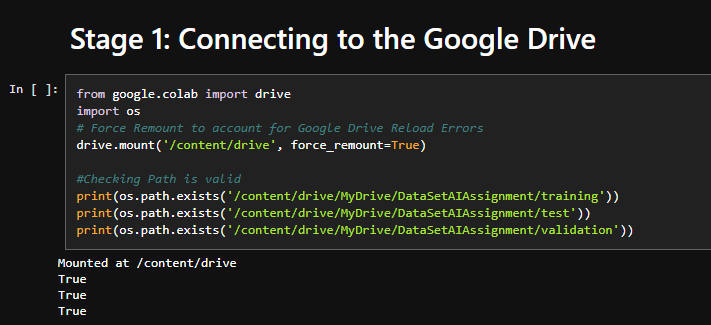


Figure 7: Model 1, NoteBook Screenshot 1

The first part of the model connects the model notebook to a google drive. Once mounted there are three print statements. These check that there are the required parent folders within the Google Drive. These parent folders contain all the images for the dataset.

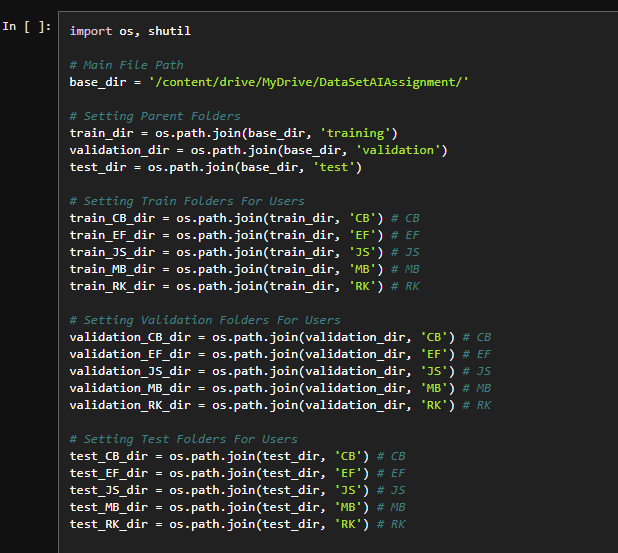


Figure 8: Model 1, NoteBook Screenshot 2

Next the Notebook declares the pathway to find the images in the dataset, then there are three blocks of code, each defining the paths for each of the five people’s images. Initials are used to help remove identities but still assign meaningful labels.

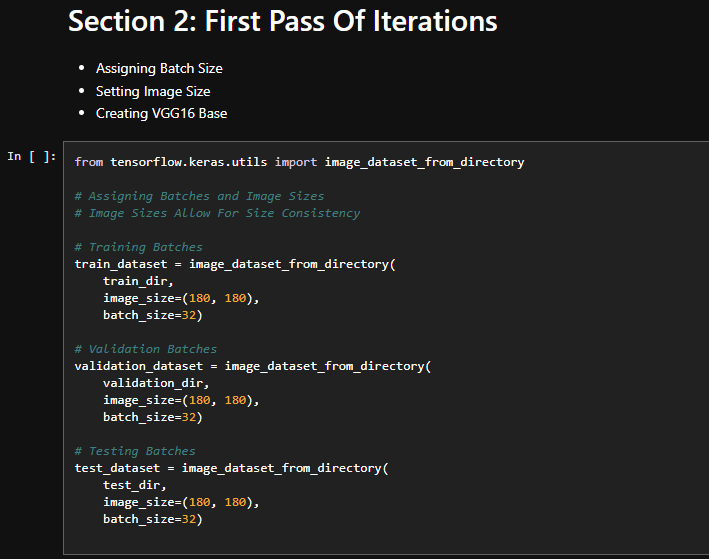


Figure 9:Model 1, NoteBook Screenshot 3

Next the author defines the batches, this splits the dataset into batches of 32 for each iteration of learning. The images are also set as all the same size. This is for consistency and prevents issues from training on image parameters and issues processing large images.

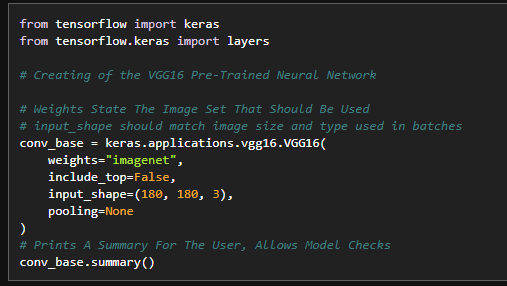


Figure 10: Model 1, NoteBook Screenshot 4

Next the author implements VGG16, A pretrained Neural Network, the network is told to be retrained using image net as it’s weights. Image net is a dataset of images that will help train the model.

The top layer is not included, this allows the author to continue adding their own layers to the model and make it unique for their dataset. Then the VGG16 model is told what to expect as input. The input should match the same as the image parameters defined earlier.  
  
Then pooling is set to none., this was to stop pooling happening within the VGG16 Network. This could cause the output at this stage to differ which could make the first iteration of the model hard to experiment and fine tune for the author. Therefore, it was turned off to make the first part of the model more consistent and easier to work with.  
  
Then a model summary is printed. This allows the author to check the VGG16 model, however it is not needed for code execution and can be deleted if not needed.

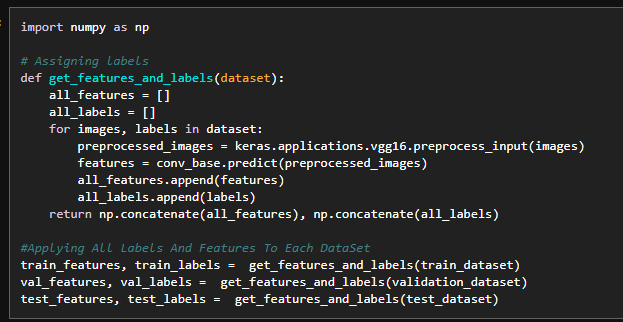


Figure 11:Model 1, NoteBook Screenshot 5

The next part of the code assigns labels to all the images, this matches the google drive names and allows the model to separate subjects as it learns.

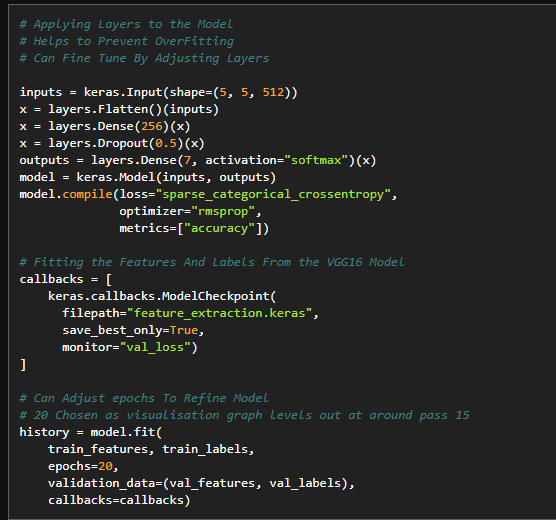


Figure 12: Model 1, NoteBook Screenshot 6

This part of the code defines the layers defined by the author as well as what file path to save this pass too. The first part of the model is called Feature Extraction as this is the aim of the first path.   
  
It is important to note the Dense layer called outputs is set up wrong, as it is set to 7. Instead, it should be set to 5. The reason this is set to 5 is for each class that the final output can be in, one for each person.   
  
It is important to note here that there is only one Dense layer and one Drop Out Layer. This means that there will not be much learnt for this first model. However the author created as a base to work on rather than perfection so a poor output was expected.

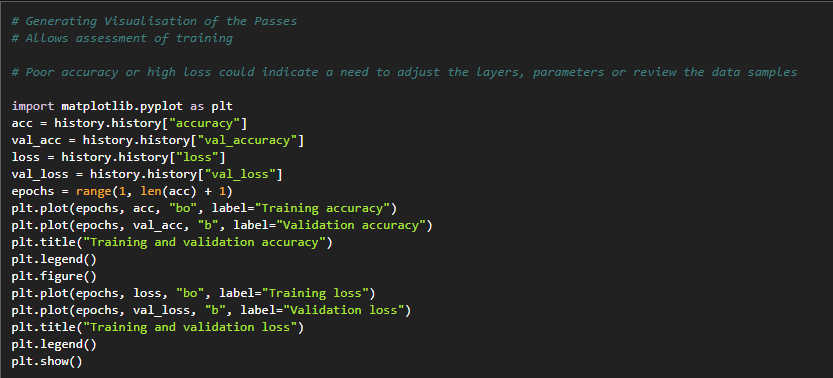


Figure 13: Model 1, NoteBook Screenshot 7

This part of the code plots two graphs. These are the graphs that are included in the evaluations part of this report. They include the accuracy and the loss of the model at current time. This allows the author deeper analysis outside of the EPOCHS feedback.

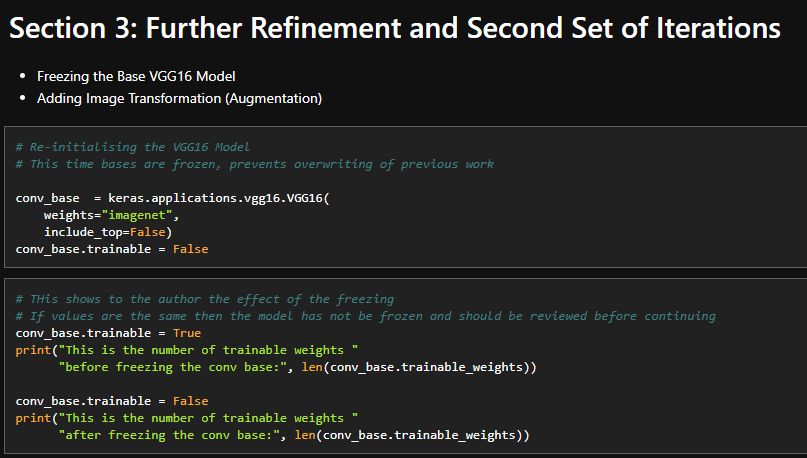


Figure 14: Model 1, NoteBook Screenshot 8

Section three moves onto the next set of iterations. After completing the first set the author freezes the base layer of the VGG16 Model. This is so what the model has previously learnt is not forgotten.

Then the author prints what is trainable before and after freezing the base. This is so the author can see if the freezing of the base has correctly taken place.

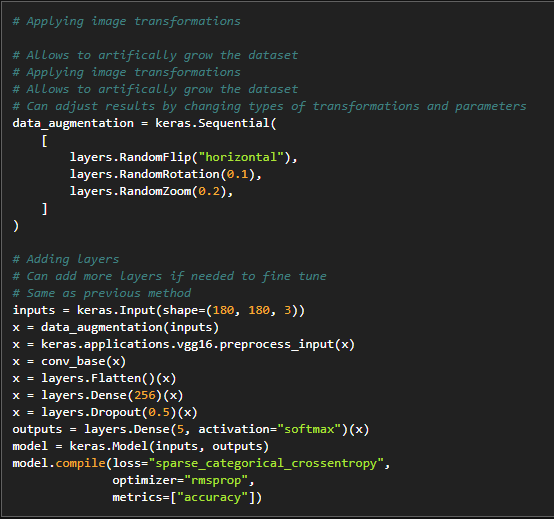


Figure 15:Model 1, NoteBook Screenshot 9

As this is the next iteration there is augmentation at this stage. This involves applying mild transformations to the images which allow the author to artificially inflate the dataset with transformed images.

The next part of the code applies the same layer types as earlier, this involves one dense layer and one drop out later. Then below that the dense with 5 categories. Interestingly in the previous run there was 7 layers, which the author commented was a mistake. However, this was corrected in this run.



Figure 16: Model 1, NoteBook Screenshot 10

The author then printed a model summary. Like earlier this can be omitted, or it can be used to assess what the current model is going to do.

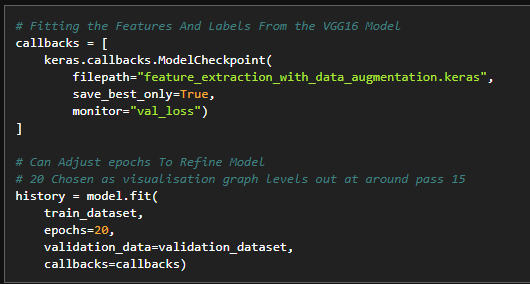


Figure 17:Model 1, NoteBook Screenshot 11

Next the author assigns the next model to a new file path. This one as with data augmentation to mark it different than the other file saved earlier. The best result Is saved and then the model is trained. The author assigned 20 epochs to this. This was to allow significant growth and monitor the results given. Too little would make the models beginning performance hard to assess. Too many would take a lot longer to process and result in over fitting.

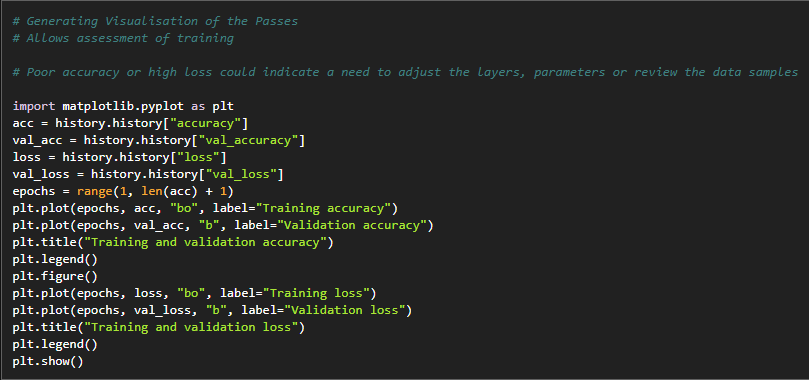


Figure 18: Model 1, NoteBook Screenshot 12

Then the author used the same code as before to draw up the graphs to allow them to assess their model.

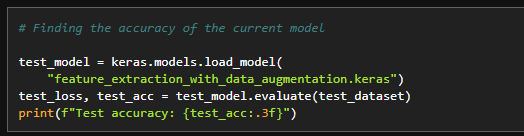


Figure 19: Model 1, NoteBook Screenshot 13

This time the author printed a total accuracy for the model at this stage, this number will give the author a rough idea on whether the model has a good accuracy. However, this was initially deceptive to the author as a high number is not always indictive of a model that is accurate at detecting images.

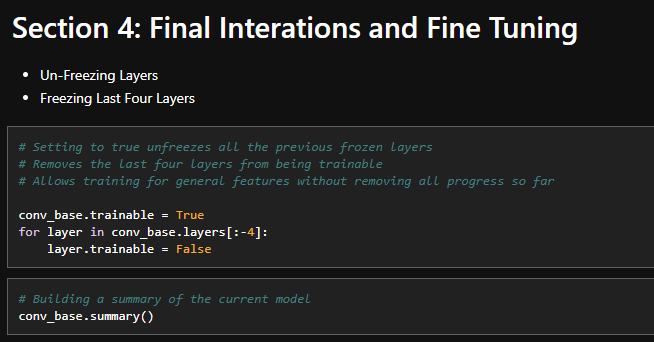


Figure 20: Model 1, NoteBook Screenshot 14

In this piece of code, the author freezes all layers apart from the last four. This allows continued training on the last four layers without the risk of overfitting or by losing all previous progress. The author notes their original comments detail the opposite. This shows at the time the authors limited understanding of the model. The authors knowledge improved the more they worked with the model.

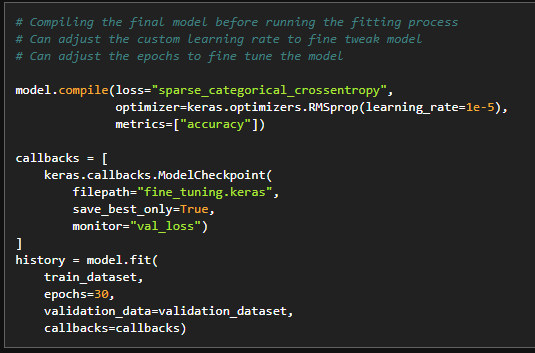


Figure 21: Figure 20: Model 1, NoteBook Screenshot 15

The final pass has different metrics than the first, with a much-reduced learning rate. This is because this final step is for fine tuning the model so there should not be large jumps in learning. There should be small steps to fine tune, rather than new patterns being low.  
  
This time the author set the Epochs to 30 rather than 20 as testing 20 resulted there was still growth being made and that 30 was a better fit.

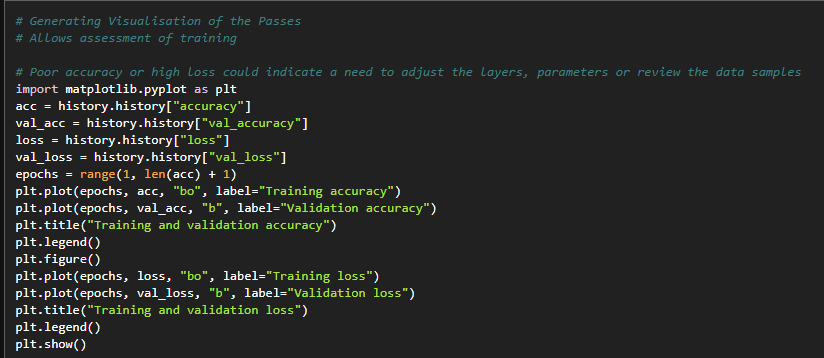


Figure 22:Figure 20: Model 1, NoteBook Screenshot 16

The author then plotted the graph again so they could see what the final model looked like.

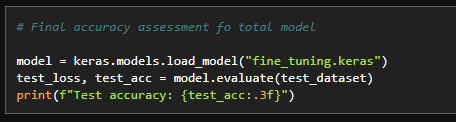


Figure 23: Figure 20: Model 1, NoteBook Screenshot 17

The author then printed the final model accuracy as a percentage.

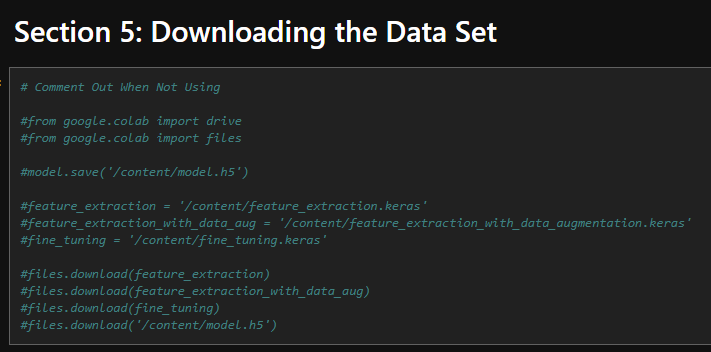


Figure 24: Figure 20: Model 1, NoteBook Screenshot 18

Lastly the author included code to download the final model. This can be injected into the authors application code and allow the quick substitution of models as they train more accurate ones.  
  
The code in the screenshot was commented out to stop the user accidentally executing the long download process while they were testing.

#### Testing And Evaluation

Before running the model inside of the authors application, the author assessed the model performance. Every time they iterated within the model, they plotted the iterations on a chart so they could assess the progress made.

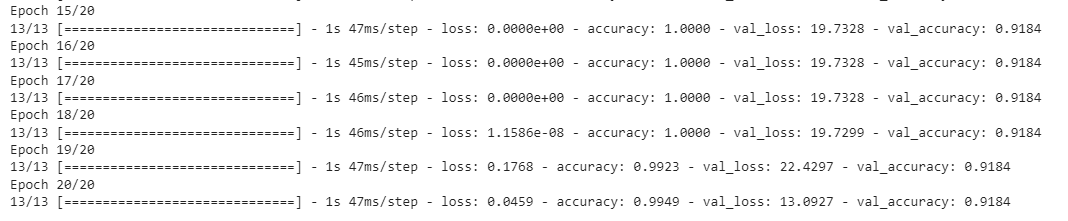
There are three separate sets of iterations within my model structure. As discussed earlier. This allows for close monitoring of the model when iterations are made allowing near continuous assessment as well as highlighting any issues in sections rather than the overall model.

Figure 25: First Pass Model 1

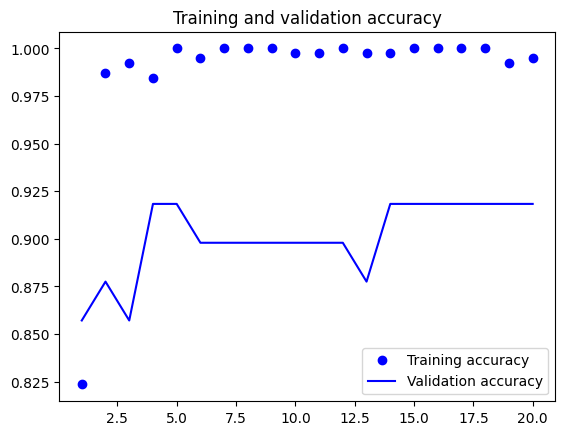
Initially the author was misguided and was looking just at the accuracy values for accuracy. This led to them being pleased with the model as they were close to one. However, these epochs imply overfitting. This is due to the abnormal loss values and the accuracy reaching one. However, while a poor result this is a good starting point to note.

Figure 26: First Pass Model 1, Training and Validation Accuracy

There is a large gap between training and validation. This implies that the data is not responding to validation images, instead it trains and learns specific images rather than patterns. The straight line of the graph also implies that the model is not as good as it can be. The mostly straight line implies that the model is not really progressing in anyway and staying consistent in its accuracies.

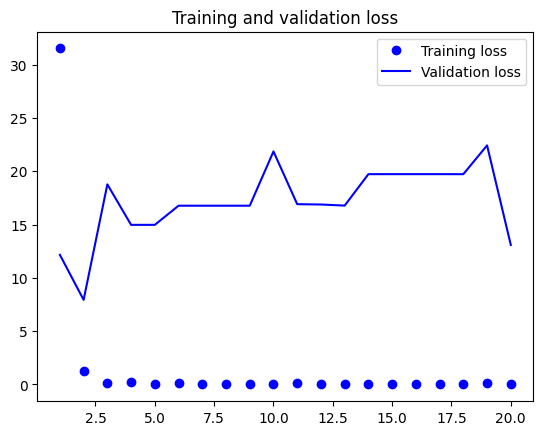


Figure 27: First Pass Model 1, Training and Validation Loss

The loss graph implies a mix. The author is happy with the training loss as this has a slight curve to it implying the loss is decreasing. However, the validation is not following a similar pattern. Instead, the mixed peaks imply there is no consistency to the validation and not much learning is occurring. Ideally the author should aim to bring the two lines closer together.

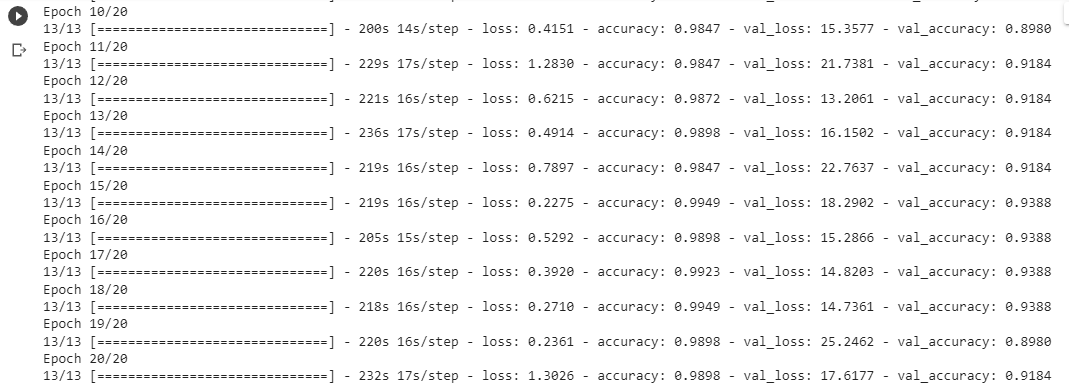


Figure 28: Second Pass Model 1

The author then moved onto the second pass. The second path contained the augmentation layer. So, as well as the previous layer this layer applies transformations to the images to increase the dataset. The accuracy has reduced however is still very high.

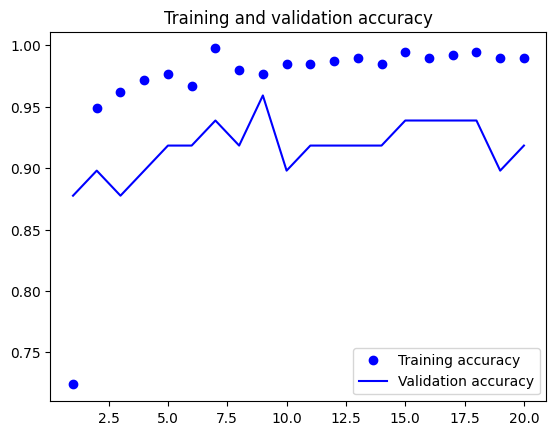


Figure 29: Second Pass Model 1, Training and Validation Accuracy

Compared to previous the author sees the second pass as an improvement. The training and validation are a lot closer together, implying the recognition of patterns rather than image learning. However, the lack of a curve is not good as it implies the model is not learning, rather it is stagnating.

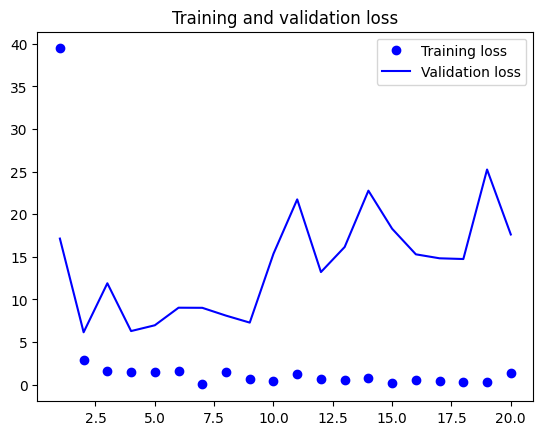


Figure 30: Second Pass Model 1, Training and Validation Loss

The training loss is slowly reducing in general, which is a good thing, however there is still the large gap between the two, implying the same disconnect noted earlier. The loss is slowly getting worse for validation rather than improving.

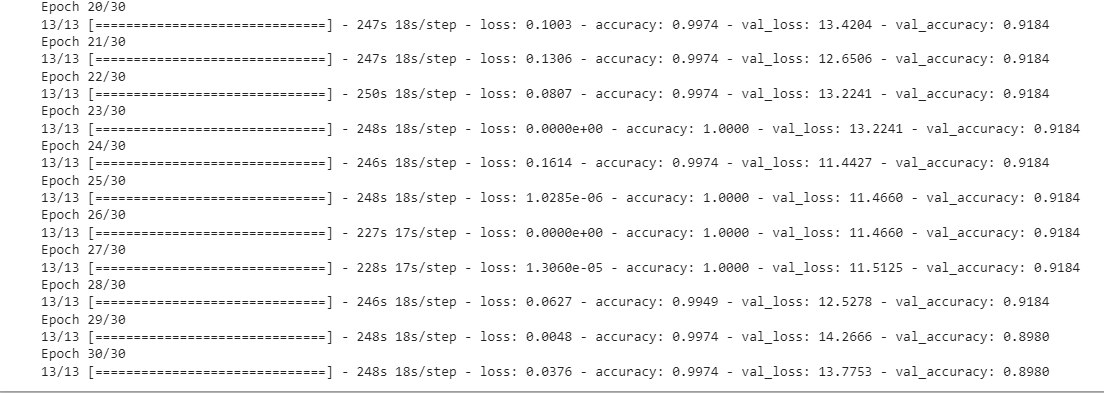
Then on the final section the fine tuning is implied. The previous problem that was spotted earlier has returned. With the high validation score. Implying a serious overfit.

Figure 31: Model 1, Third Pass Epochs

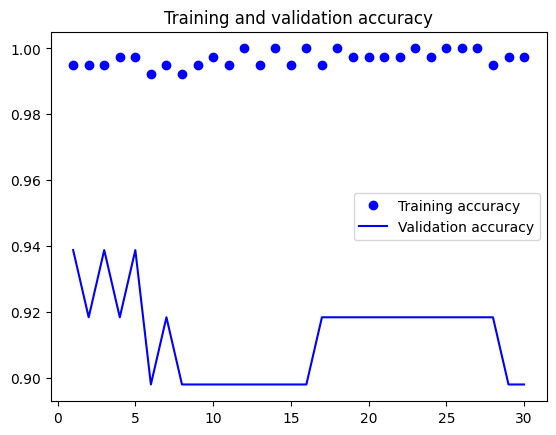


Figure 32: Model 1, Third Pass Training and Validation Accuracy

The final set of graphs for the fine tuning are particularly poor with not much growth occurring as well as a data disconnect due to the wide gap between the two. The lack of growth is consistent in this model which implies there could be a problem with the number of layers the author has presented.

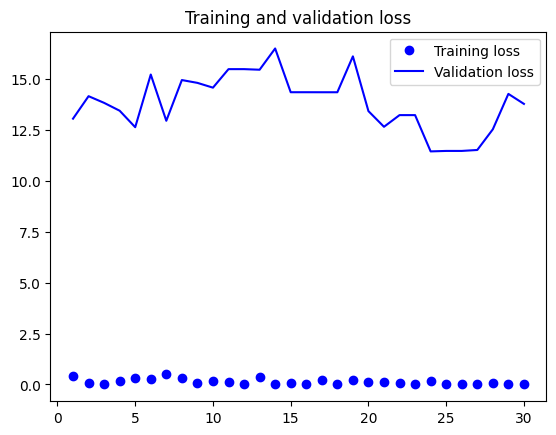


Figure 33: Model 1, Third Pass, Training and Validation Loss

The last graph is also not a good indicator of a good model, this is again due to the disconnect and poor growth.

Overall while this is a good starting point and the model is generated there is a limited amount of learning, the author thinks this is because of the limited number of layers, on the next model they will start experimenting with layers, mainly the number of layers first before experimenting with types.

The most important aim of the next model is to ensure the overfitting issue is combatted, with a secondary aim to reduce the gap between training and validation.

After running the model within my application, I noticed that my starting model had a high accuracy rating but not good at predicting identities when presenting it with new images. It suggested that every person entered into the data set was person ‘DELTA’.

The author then decided to do extra research into their model, especially into the topic of overfitting as their model frequently was returning values close to one.

### Model 2, Model Refinement

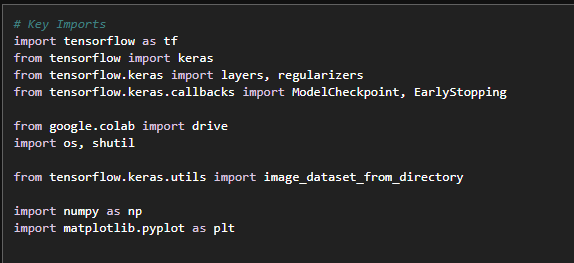


Figure 34: First Set of Changes; Model 2

The first change the author made was moving all their import statements. While this does not directly affect the performance of the model it does make the code easier to work with and improves its quality. There were some layer conflicts with imports which was resolved when all the imports were moved.

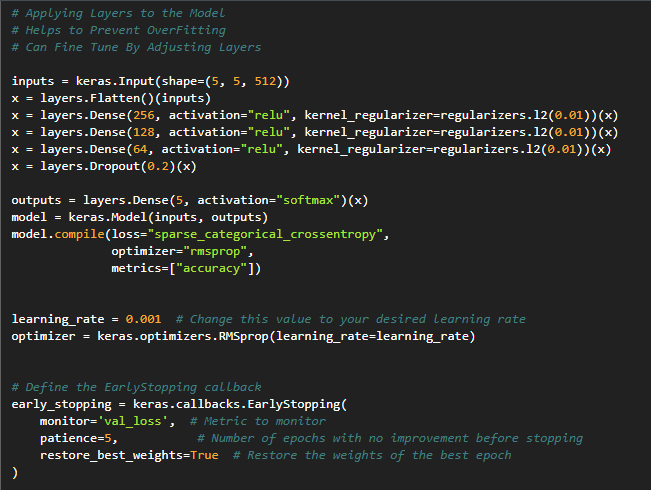


Figure 35: Changes to layers; Model 2

As a starting point for improvements Early Stopping was implemented. This stops the Epochs early and was seen as a crucial step in stopping overfitting, I started with a patience of five and kept this the same. This allows me to save time as the author doesn’t have to experiment with exact epoch measurements to find a perfect value.

There was also a custom learning rate applied to allow that author to experiment with learning values, however upon changing this they did not notice any major changes to the model, so they left this alone after initial experimentation.   
  
The next change they made was adding more layers. They decided to experiment with the number of layers to see if adding more dense layers brought the model out of its previous plateau. They author experimented with three new layers. They also added a regularizer on these layers. This was to encourage the model to learning new features and not rely on majority features, which the straight lines on the previous graph could have been implying. It also prevents high data sensitivity which stops the data massively reacting to small changes.

The author then copied this set of changes over to the other set with augmentation. They decided to keep this consistent for this model to assess improvements and save parameter experimentation for the next model.

#### Testing And Evaluation

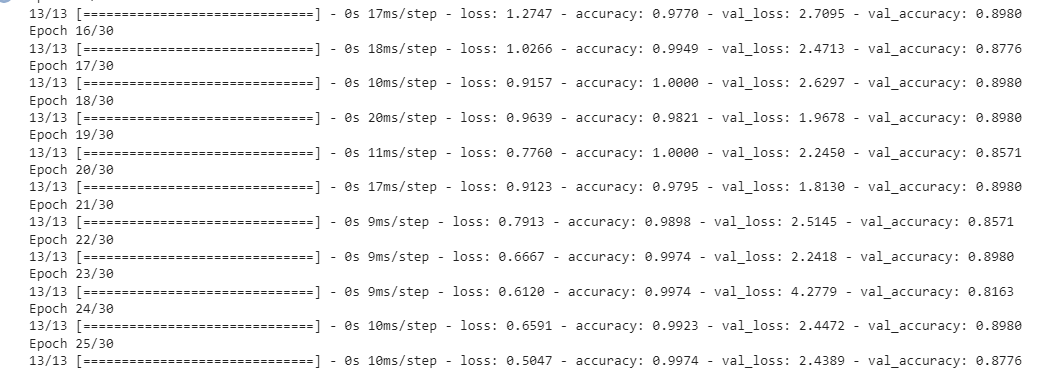
Before running the model inside of the authors application, the author assessed the model performance. Every time they iterated within the model, they plotted the iterations on a chart so they could assess the progress made, much like the testing of the first model.

Figure 36: Model 2, First Pass EPOCHS

Upon the first epoch it is important to notice that while close to the previous model, the additional layers have reduced the overfitting issue that was present previously. This can be assessed due to the much more accurate loss value that is no longer logs, as well as the lack of consistent 1.0000’s in the final epochs. However, it’s hard to draw this conclusion based on jut the Epochs.

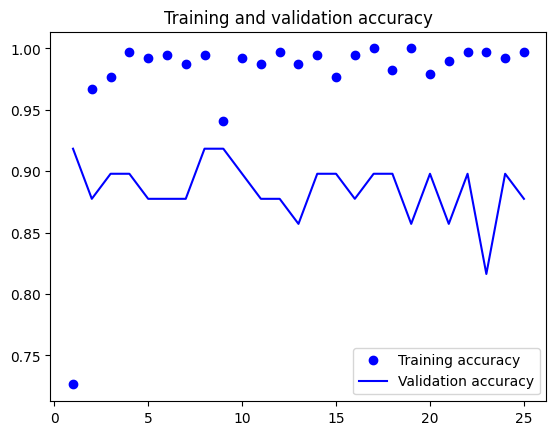


Figure 37: Model 2 First Pass Training and Validation Accuracy

There is a slight improvement in accuracy compared to Model One on the first pass. This can be seen as the Training and Validation Accuracy are closer together. This tells us that the sets are performing better with new data rather than learning key features from the images and replicating exact images. However, there is still a high concern over fitting since there is still a high amount of training data that is at the topmost value. Growth is still remaining mostly as a solid line rather than a progression, although the author is starting to see more obvious rises and falls which indicates the start of a less linear process and an attempt at better feature matching.

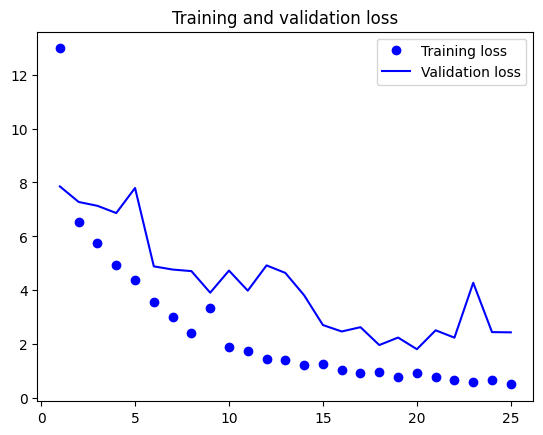


Figure 38: Model 2, Pass 1 Training and Validation Loss

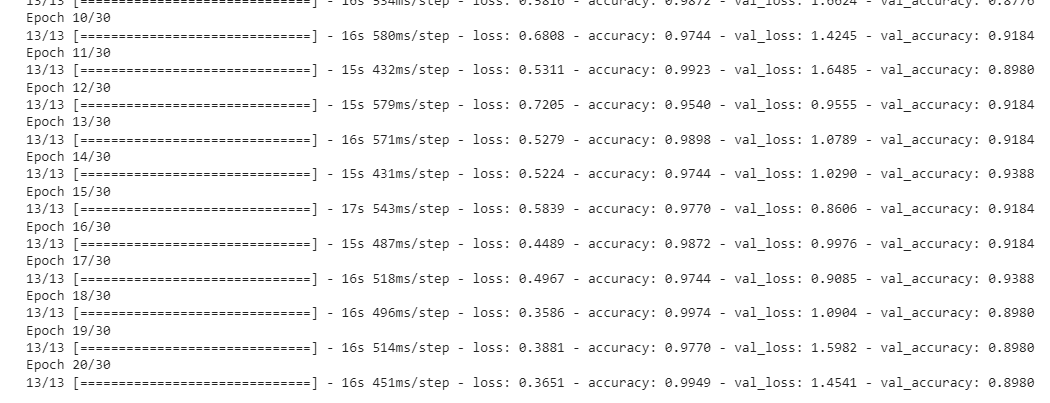
This loss graph is starting to align more closely to what the author finds as acceptable. There is an obvious downwards curve within the graph. Implying that as each Epoch progresses the loss is reducing closer to 0. Their training loss and validation loss are following mostly the same pattern and are much closer together, even though there are a few peaks where the two loss factors spread back out again. However, this can be refined with later experimentation. Overall stage one was much more of an improvement over the first model.

Figure 39: Model 2, Second Pass EPOCJS

Starting stage two the author notices that the loss is similar in some places but the validation loss is much more reduced so the validation loss may now be closer to the loss. However, the accuracy looks like it may have drifted apart from the Validation accuracy so there is a slight spread expected. The other thing to now consider is if the tweaks to the model mean an improvement between each set. If not, this could imply the need for the author to switch to a unique structure between sets.

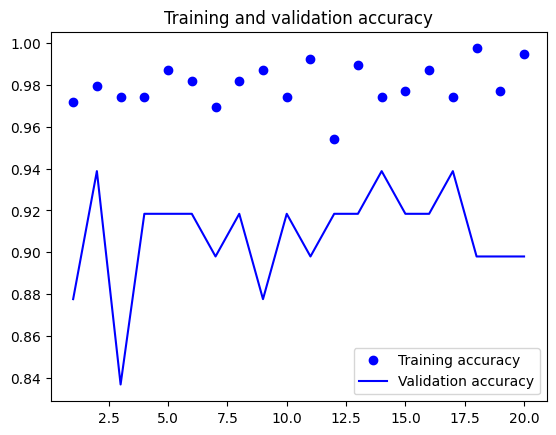


Figure 40: Model Two Training and Validation Accuracy Pass 2

Compared to the first model we can see an improvement in the second pass, the Y-Axis is now in much smaller increments before, allow the author to see that validation and training are now much closer linked than before, however this is only because there was an anomaly on the first model that is not present. In reality they are very similar. In reality the similarity means that regarding the training and validation accuracy there is still area for improvement using a third model.

In terms of within model improvements we can see that the second set of Epochs has had a slight improvement over the first, now the training accuracy is further away from the 1.00 value, implying that we are now less likely to suffer from over fitting. We can also see the dynamic stopping has come into play correctly in this set, as the data stops being project when we start to see a consistent flat line. This stops unnecessary epochs and a faster model generation as well as integrity protection.

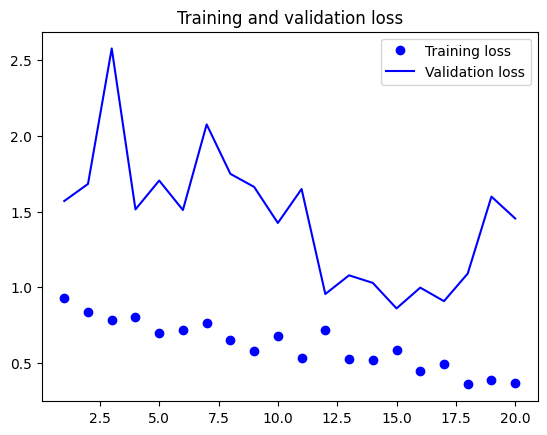


Figure 41: Model Two Training and Validation Loss Pass 2

Compared to the first model we can see the loss has improved on the second pass, this is because in model one the graph was very stagnated implying not much progression had been made, whereas this graph shows a steady decrease in loss as expected.

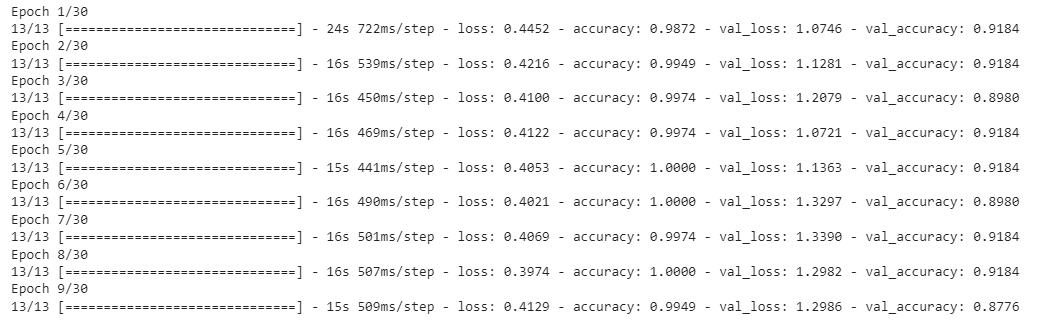
However, comparing to the previous set of Epochs within the same model the author notes that they have lost their close curve they had before. This implies that while adding more layers has had a benefit, on the second pass this has been a negative. In turn this implies that while a layer change is needed. The author should investigate different layer types rather than just duplications. This will be considered in the third model.

Figure 42: Model Two Pass 3 Epochs

For the fine tuning we can see slight improvements have occurred. The author has more aligned val\_loss than before and has reduced the accuracy overfits. This has been consistent with the rest of the second model compared to the first.

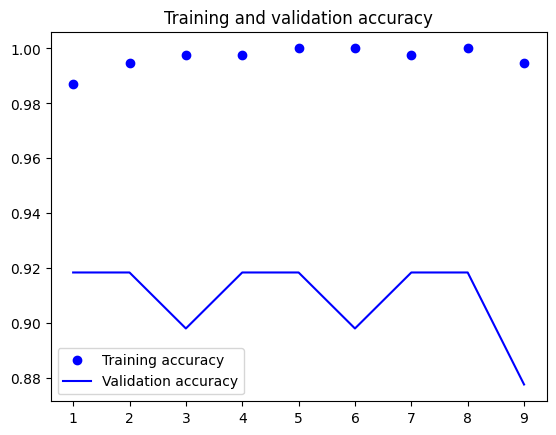


Figure 43: Model 2, Training and Validation Accuracy Pass 3

The fine-tuning chart hasn’t changed much between version one and two, the early stopping results in the data looking much more spread out, however there is now only 9 Epochs rather than the twenty plus from before. There is not much to comment on here as not much improvement has occurred. However, this is because not many changes to the code were made.

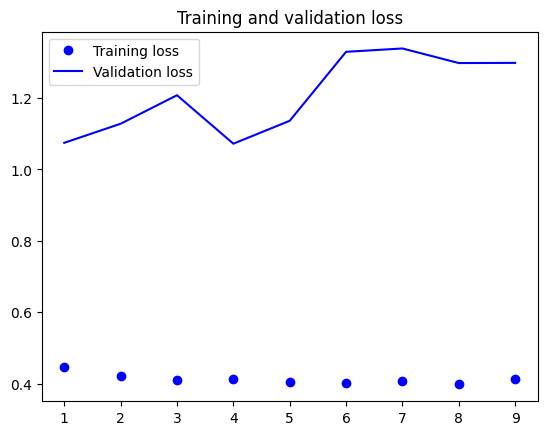


Figure 44: Model 2, Training and Validation Loss Pass 3

There is a lot less loss than in the previous model. Before the top value was above 15.0 for validation loss, now the author is working with 1.5 as the highest. Therefore, the author has managed to get the loss values consistently closer to each other. This is a great improvement to the model overall.

The author can see how changing the number of dense layers has resulted in the loss values becoming closer to each other.   
However, to improve further the author has identified they should experiment with different types of layers rather than just duplicating ones they know already work.

When fed into the final model there was still a bias in selection as the Model was still selecting ‘DELTA’ for all images. However, there was noticeable percentage indicators this time for all face options provided. And it was noticeable that ‘DELTA’ was no longer a 1.00 match, instead they were a 0.8. Indicating that the overfitting issue had been reduced from model 1.

### Model 3, Model Refinement

Although an improvement, the author has not yet generated a model that produced accurate results. The authors aim with the next model was to start experimenting with the layer types they had as well as regularizers.

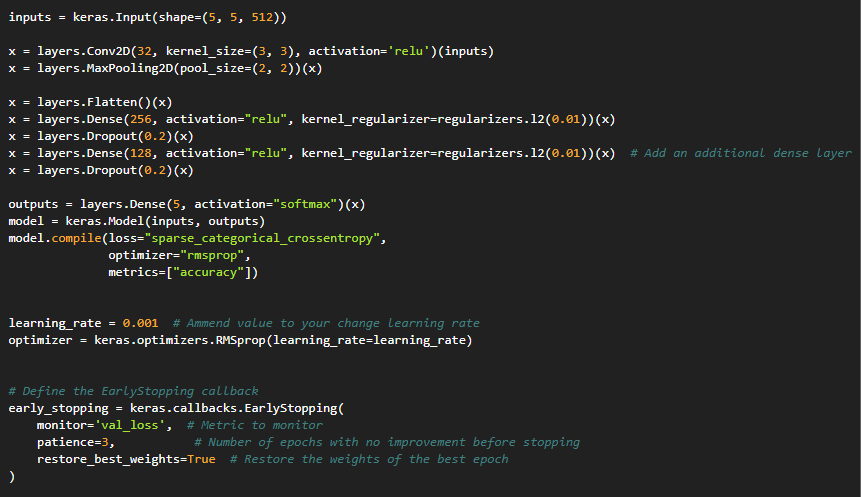


Figure 45: Model 3 Layer Changes Pass 1 and Pass 2

In this model there are now three dense layers, these alternate with a dropout layer. The author reduces the number with each dense layer to help minimise channels. In this instance we apply Ridge Regularizers. These add penalties to the layer and encourages more of an ever weight distribution and reduce highly impactful single weights.

The author also manually applied the learning rate. This allowed them to experiment with the jumps in learning the model made and see if any changes in learning rate would affect the model’s effectiveness.

The author has also applied early stopping. This reduces the need for manual epoch adjustments as if the value returned is the same for the same number of iterations as defined in the patience variable, the Epochs are halted. This reduces overfitting and stops resources being wasted on Epochs that aren’t important.

These layer changes as well as early stopping was also implemented into the second pass. However, to save space the author has omitted the screenshot as they are the same code.

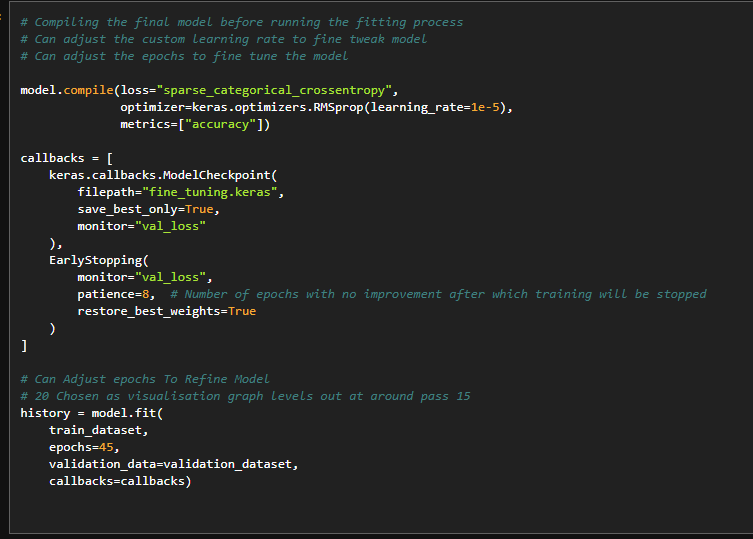


Figure 46: Model 3 Layer Changes Pass 3

On the final pass the author has made a few changes, the main one adding early stopping. However, unlike this one the author has made the patience a lot higher that the others. The author has also increased the Epochs a large amount. This is to account for the increase in patience tolerance.

The reason for this is that the aim at this stage is fine tuning. There is an extremely small learning rate, lots of small learning rate steps can help fine tune the model. However, this requires a lot of Epochs. Originally the author did not take this approach, and this resulted in large jumps with overfitting rather than good fits. This approach was much more successful than large jumps.

Moving onto the training and analysis of this model. Overall, the author noted some improvements in the model. However, there was still limits, which can be seen in the graphs, which the author will now discuss.

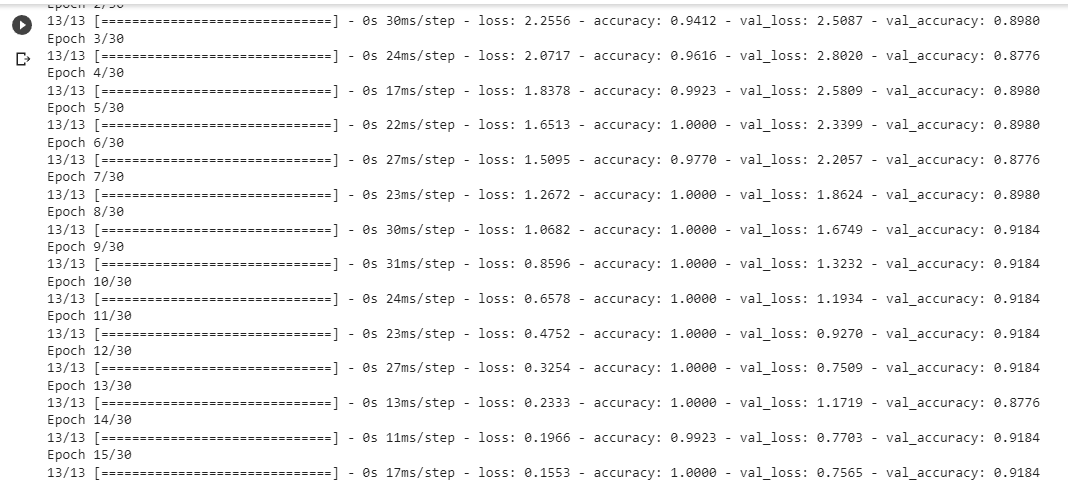


Figure 47: Model 3 Pass 1 Epochs

Assessing just the Epochs on its own can be difficult when there is a large amount. However, the author can see how adding early stopping has benefitted the model. Instead of running for 20 Epochs and possibly overfitting, the author can see that it stopped early at 15 Epochs. It also appears that numerically the losses and accuracy pairs seem a lot closer together than previous.

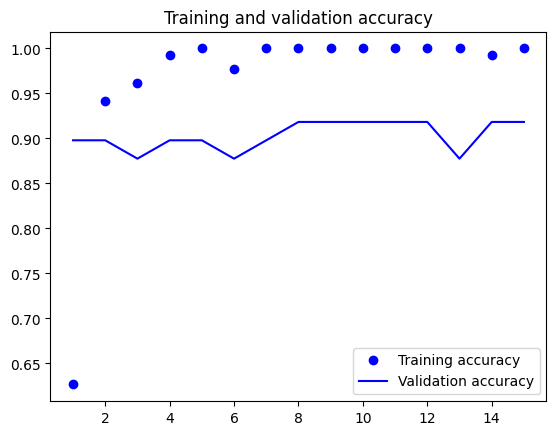


Figure 48: Model 3 Pass 1 Training and Validation Accuracy

While overall improvements have been successful the author notes that there is still work to be done. There is now a clear progression with the training accuracy which the author is pleased with. However, there is not really an improvement in validation accuracy as the graph progresses. This limited rise could imply that the model is unlikely to respond well to new data. The author notes that is an area that still needs to be improved, even though the growth of the training accuracy is much more consistent than previous.

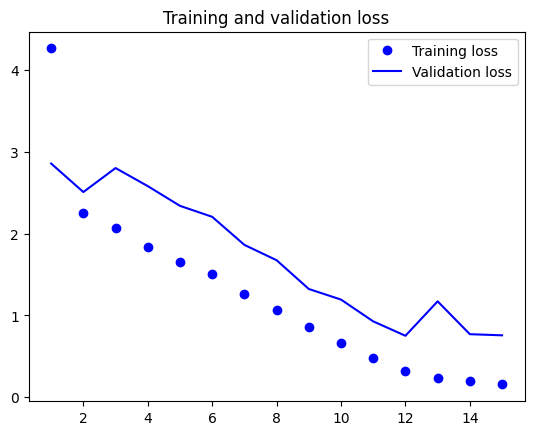


Figure 49: : Model 3 Pass 1 Training and Validation Loss

The author is very pleased with the training and validation graph on the model at this pass. The two values are very close together implying there is great consistency in the model, is also notes that the loss consistently falls as the Epochs continue before levelling out and stopping. This is the behaviour the author was aiming for.

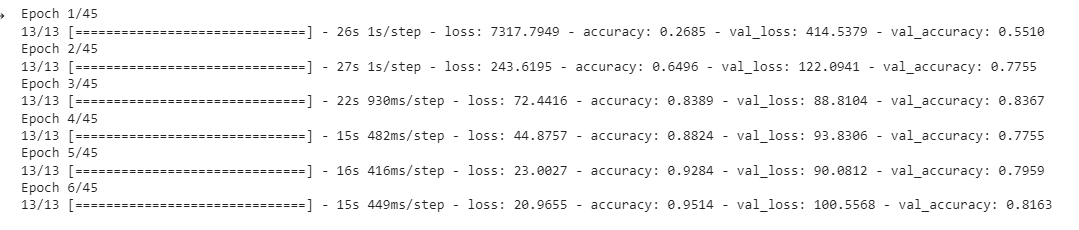


Figure 50: Model 3 Pass 2 Epochs

The next set of Epochs implies that the authors changes were definitely accurate. This time only 6 Epochs out of a possible 45 were executed. Therefore, the authors previous hypothesis of overfitting was highly likely to be true.

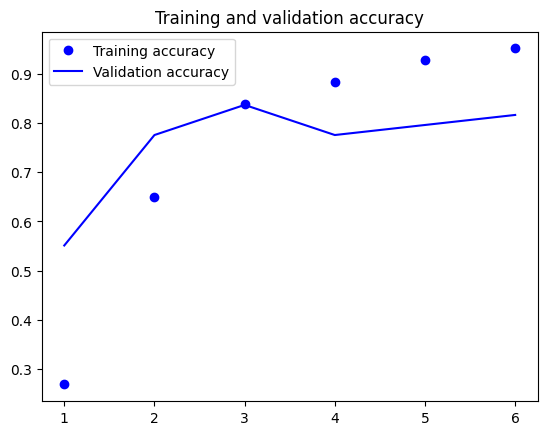


Figure 51: Model 3 Pass 2 Training and Validation Accuracy

The author found this graph quite interesting as it does not follow the usual patterns in graphs. Previously the author has had issues where the accuracy was spread. However, in this pass we can see that the two accuracies have swapped over around the third Epoch. There is a consistent growth with the training accuracy. However, the validation accuracy has a slight fluctuation. Although it should be noted that there was still an overall improvement.

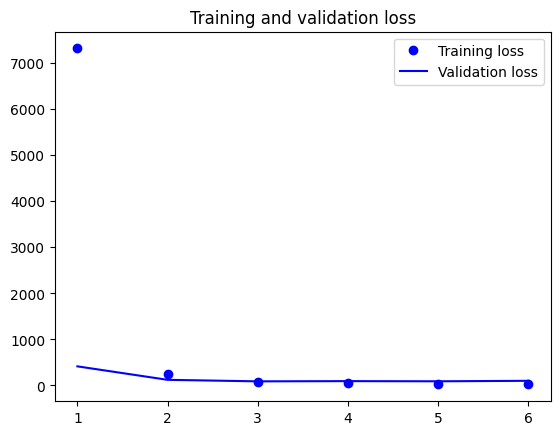


Figure 52: : Model 3 Pass 1 Training and Validation Loss

This graph is quite deceptive. This is because of an anomaly training loss value. The graph looks stagnant and like the losses aren’t decreasing. However, if the anomaly was ignored there would be a clear decrease in loss. Besides the anomaly the author is happy with this graph.

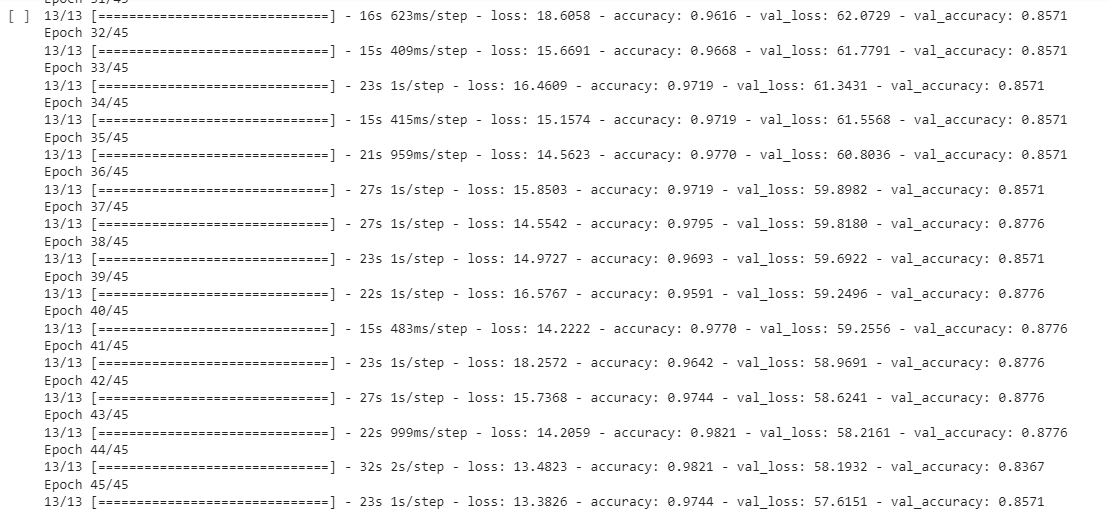


Figure 53:Model 3 Pass 3 Epochs

The author was happy with the changes to pass three. The number of Epochs was drastically changed by the new tolerances and reduced learning rates. This resulted in many Epochs. However, it is important to note that the early stopping did not take place here. Therefore, there could still be learning progress that could be made and this stage could require further tweaking.

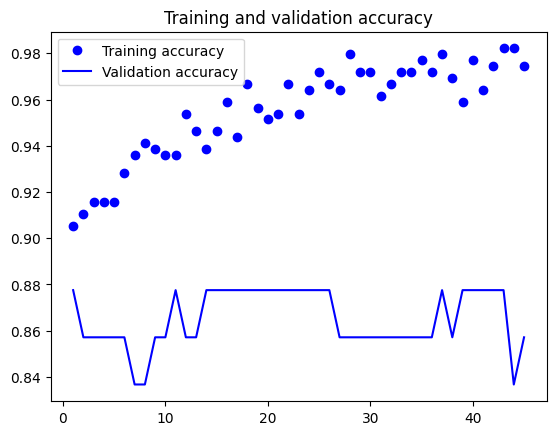


Figure 54: Model 3 Pass 3 Training and Validation Accuracy

The author was half please with this graph. In terms of training accuracy, they were happy with the growth made and the consistency in the pattern. However, they were disappointed in the validation accuracy. The validation accuracy seems to greatly fluctuate and not really improve or decrease overtime. This is very disappointing as it means that the model is not responding to new data very well, Instead it means that it may be relying too heavily on the training data and learning replication still rather than patterns.

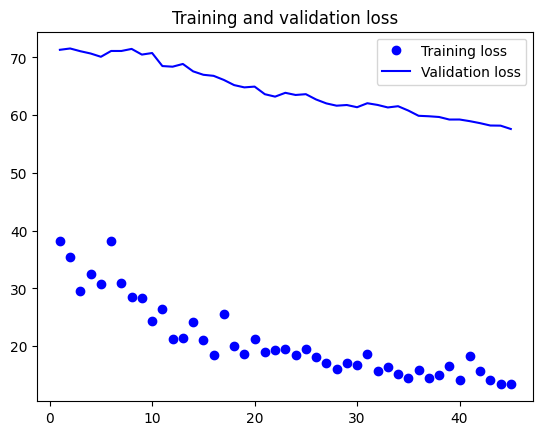


Figure 55: Model 3 Pass 3 Training and Validation Loss

However ,in contrast the author is please with this graph. While there is a large looking gap between the two values, it is important to note how small the Y-Axis is so the gap is not as large as it appears. The author is please with how the chart looks as losses are slowly falling and these look to be around equal in gradients, implying that the training and validation loss are fairly accurate towards each other.

Final Output Model 3  
A picture containing text, screenshot, font

Description automatically generated

Figure 56: Sample of Model 3's final output

In terms of output the author is now a little closer, the weights are looking for a lot more accurate than previous and the bias seems to have dissipated. However, the model is currently only detecting two out of five people and flicks back and forth between Alpha and Delta for the student names rather than detecting accurately.

Due to time constraints the author may not get time to look in detail at drawing up another model, this they find disappointing as they now believe they are on the right track with this third model. They believe that looking at the validation in the first pass and focusing time into the third pass may be the secret to unlocking the successful model.

## Section 6: Testing

### Test Plans

Black Box Testing

Performance Testing

Live Testing

## Section 7: Feature Summary

The author challenged themselves in this project. They wanted to show their progression across the third year and wanted to aim for a application that did much more than expected. This significantly reduced time on the model which in retrospect may have not been a good idea. However, the author acknowledges that this drive can from being personally dissatisfied with what they handed in for a Media Technology Module.

To demonstrate these features, the author has included this in table format. Which has the feature, what it does and the evidence of said feature.

|  |  |  |
| --- | --- | --- |
| Feature | Explanation | Evidence |
| GUI | There is an interactive GUI that the user can interact with to use the program. This works concurrently with the program and requires knowledge of Tkinter | Not Applicable as shown throughout table |
| Lecturer Session Definition | There is a spinner that is linked to time values, the lecturer can pick a session to record attendance for upon start. |  |
| Face Detection | The author has included facial detection for the application. This was based on previous knowledge gained in a different module and uses OpenCV. This stops the user from spam clicking register as the register button only appears when a face is on screen. This improves the integrity of the application as it stops users crashing the system by repeatedly saving frames. |  |
| Frame Saving | When the user clicks register a frame inside the motion detection box is saved. This is saved in a folder called temporary and over written when a new frame is saved. This frame is then passed into the model made by the author. |  |
| Name Detection | When the user clicked register a possible name match pops up on the screen, the user is then given the option to register their attendance or cancel. Cancel takes them back to the open camera |  |
| Registration Log | After registering the students name and late status is saved to a file with the session start date and time. This allows the lecturer to have an ongoing log of who has registered and whether they were on time or within a 15 minute late window. |  |

## Section 8: Discussions and Conclusion

#### Summary

Overall, the author has built a system that recognises people and allows them to register their attendance as the person an AI model recognises. This system also saves the time registered and notes whether the student was late or on time. There is lots of possible easy expansions for the system, such as adding an extra tab that allows users to see what time people registered or allowing detection of multiple people.   
  
In terms of the overall application the author is pleased with their work, however this is on just building the application and not the model.

In terms of the model the author struggled significantly to get the model started. There is a great challenge within Machine Learning projects especially object recognition. This can be seen by the lack of commercial products that preform this service. However, this was a great challenge for the author which pushed them to the limits of their capabilities as an Undergraduate developer. There was a lot of external reading and lecture content analysed to develop a better understanding of Machine Learning and Neural Networks. Even so the authors model is a little disappointing, in their opinion, they are pleased with their experimentation and the learning they conducted through out the project.

In terms of report the author is pleased, the report was extended to be a lot longer than they expected and it was very tempting to cut corners to obtain more development time. However, they did not do this as they wanted to showcase their entire development process. The authors aim with this was to show that they have been engaged in the module by being able to demonstrate the entire process from having minimal knowledge to their current knowledge.

Although detailed and lengthy the author is especially proud of noting the experimental process of developing the model. The author believes including significant jumps in models shows a continuing development of understanding the topic. They believe that this shows author transparency as the author is not afraid to show development mistakes to show overall improvement. They also believe showing model development showcases their understanding in much better depth than being able to explain the final model. This is because it showcases the author knows how each tiny change can affect the model rather than just explaining a complete model.

#### Learning Points

Before starting this process, the author had not worked on any Machine Learning projects before. Therefore, the largest learning point for the author was Machine Learning theory and applying this theory to a practical model.   
  
However, the author should not let this overshadow their other work in the project. This was the second time that the author has created a large project in Python. Therefore, there has been significant growth in the authors understanding of Python, especially in comparisons in other languages the author knows.

The author has also learnt an important lesson about the time experimentation takes. Especially when having to apply a trial-and-error approach. The author hasn’t worked on a project like this before, therefore they greatly underestimated the time needed on the model and did not expect the large amount of experimentation.

#### Improvements

One of the largest improvements the author has noticed is with their datasets. Expansion of the authors dataset could result in more accurate results and better outcomes for the final software solution. If given more time the author could collect more data and a large range of people. This would make the project more complete and could results to better fits due to having more to train models with. The reason this was not actioned in the project was because collecting data can take a large amount of time, and instead the author would have rather focused on model and application development.

Regarding the application the authors next steps would be to implement more of a management side of the application. Allowing lecturers to see who is signed in in the GUI rather than in a text documents. The author has also considered larger scale improvements. Such as implication of Databases within application. This could allow storage of user’s attendance and then allow analytics for each student’s attendance. This could result in a more fleshed out application that could be used.   
  
Relating to ongoing issues the authors file saving system needs reworking. Currently the last file saved is selected. This is on assumed logic that the last saved will be the latest lecture. However currently the AM and PM alter the storage location and results in AM files always being after PM files, meaning names are not saved in the correct files. This was highlighted in the feature testing stage.

#### Suitability of Product

After considering all the above, the author thinks the project could be scaled to become more suitable for actual use. There is good groundwork for this project. However, the model will require much more work to suit the final needs of the brief.

The application itself is suitable and works as intended and the author is pleased with their overall logic and what they have developed.

If the model was improved the author thinks it could be easily scalable and many more students could be added to create a complete system. However this would require a lot more experimentation on the authors part.

#### Final Thoughts

While not an expected outcome, with the product not meeting the minimum five people expectation. The author is still proud of their work. They have done their best to complete the objectives and has completed an extensive report on how they have developed. If the author did not have the depth of experimentation they currently have they would consider the project a failure with the same output. The experimentation aspect of the project is the authors saving grace as they believe this has showed a deep level of effort and engagement with the project. They only hope the marker agrees.

## References

AlindGupta (2023) Regularization in Machine Learning.  *GeeksForGeeks* [Online]. Available from: <https://www.geeksforgeeks.org/regularization-in-machine-learning/> [Accessed 12/05/2023].

Amazon (2023) What is Overfitting. *AWS Amazon* [Online]. Available from: <https://aws.amazon.com/what-is/overfitting/#:~:text=Overfitting%20occurs%20when%20the%20model,to%20several%20reasons%2C%20such%20as%3A&text=The%20training%20data%20size%20is,all%20possible%20input%20data%20values>. [Accessed 15/05/2023].

Baheti, P (2021) What is Overfitting in Deep Learning [+10 Ways to Avoid it]. *V7Labs* [Online]. Available from: <https://www.v7labs.com/blog/overfitting> [Accessed 15/05/2023].

Binns, R. Bouhali, M (2022) The Best Facial Recognition Attendance Systems. *Expert Market* [Online]. Available from: <https://www.expertmarket.com/uk/time-and-attendance-systems/face-recognition-time-attendance-management-system-reviews> [Accessed 29/04/2023].

Bose, B. (2023) Face Recognition Python: Popular Techniques and Libraries. *Digital Vidya* [Online]. Available from: <https://www.digitalvidya.com/blog/face-recognition-python/> [Accessed 06/05/2023].

ClockRite (2023) Facial Recognition Biometric Clocking System. *ClockRite* [Online]. Available from: <https://www.clockrite.co.uk/attendance-clocking-systems/facial-recognition-biometric-clocking-system/> [Accessed 05/05/2023].

DataQuest (2022) Regularization in Machine Learning (with Code Examples). *DataQuest* [Online]. Available from: <https://www.dataquest.io/blog/regularization-in-machine-learning/> [Accessed 12/05/2023].

DeepLizard (2017) Deep Learning Fundamental – Classic Edition. *DeepLizard* [Online]. Available from: <https://deeplizard.com/learn/video/FK77zZxaBoI> [Accessed 14/05/2023].

Exxact (2021) A Breakdown of Deep Learning Frameworks. *ExxactCorp* [Online]. Available from: <https://www.exxactcorp.com/blog/Deep-Learning/a-breakdown-of-deep-learning-frameworks> [Accessed 08/05/2023].

GeeksForGeeks (2023) Reading and Writing to Text Files in Python. *GeeksForGeeks* [Online]. Available from: <https://www.geeksforgeeks.org/reading-writing-text-files-python/> [Accessed 11/05/2023].

Goyal, K. (2022) Top 10 Deep Learning Frameworks in 2023 You Can’t Ignore. *upGrad* [Online]. Available from: <https://www.upgrad.com/blog/top-deep-learning-frameworks/> [Accessed 07/05/2023].

Jibble (2023) 100% FREE Face Recognition Attendance. *Jibble* [Online]. Available from: <https://www.jibble.io/face-recognition-attendance-system> [Accessed 29/04/2023].

Keldenich, T. (2021) 7 Types of Layers you need to know in Deep Learning and how to use them. *Inside Machine Learning* [Online]. Available from: <https://inside-machinelearning.com/en/7-types-of-layers-you-need-to-know-in-deep-learning-and-how-to-use-them/> [Accessed 14/05/2023].

Keshavdas, M (2022) Contactless Attendance Using Facial Recognition For Students Wearing Masks. *Fleetroot* [Online]. Available from: <https://www.fleetroot.com/blog/contactless-attendance-using-facial-recognition-for-students-wearing-masks/> [Accessed 29/04/2023].

Korakot (2018) How can I download this model from colaboratory. *Stack Overflow* [Online]. Available from: <https://stackoverflow.com/questions/51078996/how-can-i-download-this-model-from-colaboratory> [Accessed 16/05/2023].

Meel, V. (2023) Top 10 Deep Learning Frameworks in 2023. *Viso.ai* [Online]. Available from: <https://viso.ai/deep-learning/deep-learning-frameworks/> [Accessed 07/05/2023].

Mustafeez, A. (2023) What is early stopping?. *Educative* [Online]. Available from: <https://www.educative.io/answers/what-is-early-stopping> [Accessed 17/05/2023].

Nvidia (2023) Deep Learning Frameworks. *Nvidia Developer* [Online]. Available from: <https://developer.nvidia.com/deep-learning-frameworks> [Accessed 05/05/2023].

ProjectPro (2023) 15 Popular Machine Learning Frameworks for Model Training. *ProjectPro* [Online]. Available from: <https://www.projectpro.io/article/machine-learning-frameworks/509> [Accessed 07/05/2023].

Python Software Foundation (2023) datetime – Basic date and time types. *Doc.python* [Online]. Available from: <https://docs.python.org/3/library/datetime.html> [Accessed 11/05/2023].

Rohini, G. (2021) Everything you need to know about VGG16. *Medium.Com* [Online]. Available from: <https://medium.com/@mygreatlearning/everything-you-need-to-know-about-vgg16-7315defb5918> [Accessed 10/05/2023].

Thakur, R. (2023) Beginner’s Guide to VGG16 Implementation in Keras. *builtIn* [Online]. Available from: <https://builtin.com/machine-learning/vgg16> [Accessed 10/05/2023].

Tutorials Point (2023) Python Tkinter Spinbox. *TutorialsPoint* [Online]. Available from: <https://www.javatpoint.com/python-tkinter-spinbox> [Accessed 11/05/2023].

TutorialsPoint (2023) Python – GUI Programming (TKinter). *TutorialsPoint* [Online]. Available from: <https://www.tutorialspoint.com/python/python_gui_programming.htm> [Accessed 07/05/2023].

TutorialsPoint (2023) Python – Tiknter Frame. *TutorialsPoint* [Online]. Available from: <https://www.tutorialspoint.com/python/tk_frame.htm> [Accessed 07/05/2023].

## Appendix 1: Source Code

### Training Models (Complete)

#### Model 1

# -\*- coding: utf-8 -\*-

"""DataSet (1) (1).ipynb

Automatically generated by Colaboratory.

Original file is located at

https://colab.research.google.com/drive/1-Zqml9sfnCnaAB\_EFR8Rqd-YGxuTZhM0

# File Header

\*\*File Author\*\*: Emily Fletcher

\*\*Student Number\*\*: 18410839

\*\*File Purpose\*\*: Passing a set of images into a Data Set, Aim is to train a model

on recognition of the 5 people sampled.

\*\*Last Changed\*\*: 16/05/2023

\*\*Last Model Download\*\*: 16/05/2023

\*\*Version\*\*: 1.0

\*\*Instructions\*\*

Ensure model is corrected to Google Drive with correct directories before proceeding through Stage 2 Onwards

# Stage 1: Connecting to the Google Drive

"""

from google.colab import drive

import os

# Force Remount to account for Google Drive Reload Errors

drive.mount('/content/drive', force\_remount=True)

#Checking Path is valid

print(os.path.exists('/content/drive/MyDrive/DataSetAIAssignment/training'))

print(os.path.exists('/content/drive/MyDrive/DataSetAIAssignment/test'))

print(os.path.exists('/content/drive/MyDrive/DataSetAIAssignment/validation'))

import os, shutil

# Main File Path

base\_dir = '/content/drive/MyDrive/DataSetAIAssignment/'

# Setting Parent Folders

train\_dir = os.path.join(base\_dir, 'training')

validation\_dir = os.path.join(base\_dir, 'validation')

test\_dir = os.path.join(base\_dir, 'test')

# Setting Train Folders For Users

train\_CB\_dir = os.path.join(train\_dir, 'CB') # CB

train\_EF\_dir = os.path.join(train\_dir, 'EF') # EF

train\_JS\_dir = os.path.join(train\_dir, 'JS') # JS

train\_MB\_dir = os.path.join(train\_dir, 'MB') # MB

train\_RK\_dir = os.path.join(train\_dir, 'RK') # RK

# Setting Validation Folders For Users

validation\_CB\_dir = os.path.join(validation\_dir, 'CB') # CB

validation\_EF\_dir = os.path.join(validation\_dir, 'EF') # EF

validation\_JS\_dir = os.path.join(validation\_dir, 'JS') # JS

validation\_MB\_dir = os.path.join(validation\_dir, 'MB') # MB

validation\_RK\_dir = os.path.join(validation\_dir, 'RK') # RK

# Setting Test Folders For Users

test\_CB\_dir = os.path.join(test\_dir, 'CB') # CB

test\_EF\_dir = os.path.join(test\_dir, 'EF') # EF

test\_JS\_dir = os.path.join(test\_dir, 'JS') # JS

test\_MB\_dir = os.path.join(test\_dir, 'MB') # MB

test\_RK\_dir = os.path.join(test\_dir, 'RK') # RK

"""# Section 2: First Pass Of Iterations

\* Assigning Batch Size

\* Setting Image Size

\* Creating VGG16 Base

"""

from tensorflow.keras.utils import image\_dataset\_from\_directory

# Assigning Batches and Image Sizes

# Image Sizes Allow For Size Consistency

# Training Batches

train\_dataset = image\_dataset\_from\_directory(

train\_dir,

image\_size=(180, 180),

batch\_size=32)

# Validation Batches

validation\_dataset = image\_dataset\_from\_directory(

validation\_dir,

image\_size=(180, 180),

batch\_size=32)

# Testing Batches

test\_dataset = image\_dataset\_from\_directory(

test\_dir,

image\_size=(180, 180),

batch\_size=32)

from tensorflow import keras

from tensorflow.keras import layers

# Creating of the VGG16 Pre-Trained Neural Network

# Weights State The Image Set That Should Be Used

# input\_shape should match image size and type used in batches

conv\_base = keras.applications.vgg16.VGG16(

weights="imagenet",

include\_top=False,

input\_shape=(180, 180, 3),

pooling=None

)

# Prints A Summary For The User, Allows Model Checks

conv\_base.summary()

import numpy as np

# Assigning labels

def get\_features\_and\_labels(dataset):

all\_features = []

all\_labels = []

for images, labels in dataset:

preprocessed\_images = keras.applications.vgg16.preprocess\_input(images)

features = conv\_base.predict(preprocessed\_images)

all\_features.append(features)

all\_labels.append(labels)

return np.concatenate(all\_features), np.concatenate(all\_labels)

#Applying All Labels And Features To Each DataSet

train\_features, train\_labels = get\_features\_and\_labels(train\_dataset)

val\_features, val\_labels = get\_features\_and\_labels(validation\_dataset)

test\_features, test\_labels = get\_features\_and\_labels(test\_dataset)

# Applying Layers to the Model

# Helps to Prevent OverFitting

# Can Fine Tune By Adjusting Layers

inputs = keras.Input(shape=(5, 5, 512))

x = layers.Flatten()(inputs)

x = layers.Dense(256)(x)

x = layers.Dropout(0.5)(x)

outputs = layers.Dense(7, activation="softmax")(x)

model = keras.Model(inputs, outputs)

model.compile(loss="sparse\_categorical\_crossentropy",

optimizer="rmsprop",

metrics=["accuracy"])

# Fitting the Features And Labels From the VGG16 Model

callbacks = [

keras.callbacks.ModelCheckpoint(

filepath="feature\_extraction.keras",

save\_best\_only=True,

monitor="val\_loss")

]

# Can Adjust epochs To Refine Model

# 20 Chosen as visualisation graph levels out at around pass 15

history = model.fit(

train\_features, train\_labels,

epochs=20,

validation\_data=(val\_features, val\_labels),

callbacks=callbacks)

# Generating Visualisation of the Passes

# Allows assessment of training

# Poor accuracy or high loss could indicate a need to adjust the layers, parameters or review the data samples

import matplotlib.pyplot as plt

acc = history.history["accuracy"]

val\_acc = history.history["val\_accuracy"]

loss = history.history["loss"]

val\_loss = history.history["val\_loss"]

epochs = range(1, len(acc) + 1)

plt.plot(epochs, acc, "bo", label="Training accuracy")

plt.plot(epochs, val\_acc, "b", label="Validation accuracy")

plt.title("Training and validation accuracy")

plt.legend()

plt.figure()

plt.plot(epochs, loss, "bo", label="Training loss")

plt.plot(epochs, val\_loss, "b", label="Validation loss")

plt.title("Training and validation loss")

plt.legend()

plt.show()

"""# Section 3: Further Refinement and Second Set of Iterations

\* Freezing the Base VGG16 Model

\* Adding Image Transformation (Augmentation)

"""

# Re-initialising the VGG16 Model

# This time bases are frozen, prevents overwriting of previous work

conv\_base = keras.applications.vgg16.VGG16(

weights="imagenet",

include\_top=False)

conv\_base.trainable = False

# THis shows to the author the effect of the freezing

# If values are the same then the model has not be frozen and should be reviewed before continuing

conv\_base.trainable = True

print("This is the number of trainable weights "

"before freezing the conv base:", len(conv\_base.trainable\_weights))

conv\_base.trainable = False

print("This is the number of trainable weights "

"after freezing the conv base:", len(conv\_base.trainable\_weights))

# Applying image transformations

# Allows to artifically grow the dataset

# Applying image transformations

# Allows to artifically grow the dataset

# Can adjust results by changing types of transformations and parameters

data\_augmentation = keras.Sequential(

[

layers.RandomFlip("horizontal"),

layers.RandomRotation(0.1),

layers.RandomZoom(0.2),

]

)

# Adding layers

# Can add more layers if needed to fine tune

# Same as previous method

inputs = keras.Input(shape=(180, 180, 3))

x = data\_augmentation(inputs)

x = keras.applications.vgg16.preprocess\_input(x)

x = conv\_base(x)

x = layers.Flatten()(x)

x = layers.Dense(256)(x)

x = layers.Dropout(0.5)(x)

outputs = layers.Dense(5, activation="softmax")(x)

model = keras.Model(inputs, outputs)

model.compile(loss="sparse\_categorical\_crossentropy",

optimizer="rmsprop",

metrics=["accuracy"])

# Prints a model summary, allows for user to check accuracy

model.summary()

# Fitting the Features And Labels From the VGG16 Model

callbacks = [

keras.callbacks.ModelCheckpoint(

filepath="feature\_extraction\_with\_data\_augmentation.keras",

save\_best\_only=True,

monitor="val\_loss")

]

# Can Adjust epochs To Refine Model

# 20 Chosen as visualisation graph levels out at around pass 15

history = model.fit(

train\_dataset,

epochs=20,

validation\_data=validation\_dataset,

callbacks=callbacks)

# Generating Visualisation of the Passes

# Allows assessment of training

# Poor accuracy or high loss could indicate a need to adjust the layers, parameters or review the data samples

import matplotlib.pyplot as plt

acc = history.history["accuracy"]

val\_acc = history.history["val\_accuracy"]

loss = history.history["loss"]

val\_loss = history.history["val\_loss"]

epochs = range(1, len(acc) + 1)

plt.plot(epochs, acc, "bo", label="Training accuracy")

plt.plot(epochs, val\_acc, "b", label="Validation accuracy")

plt.title("Training and validation accuracy")

plt.legend()

plt.figure()

plt.plot(epochs, loss, "bo", label="Training loss")

plt.plot(epochs, val\_loss, "b", label="Validation loss")

plt.title("Training and validation loss")

plt.legend()

plt.show()

# Finding the accuracy of the current model

test\_model = keras.models.load\_model(

"feature\_extraction\_with\_data\_augmentation.keras")

test\_loss, test\_acc = test\_model.evaluate(test\_dataset)

print(f"Test accuracy: {test\_acc:.3f}")

"""# Section 4: Final Interations and Fine Tuning

\* Un-Freezing Layers

\* Freezing Last Four Layers

"""

# Setting to true unfreezes all the previous frozen layers

# Removes the last four layers from being trainable

# Allows training for general features without removing all progress so far

conv\_base.trainable = True

for layer in conv\_base.layers[:-4]:

layer.trainable = False

# Building a summary of the current model

conv\_base.summary()

# Compiling the final model before running the fitting process

# Can adjust the custom learning rate to fine tweak model

# Can adjust the epochs to fine tune the model

model.compile(loss="sparse\_categorical\_crossentropy",

optimizer=keras.optimizers.RMSprop(learning\_rate=1e-5),

metrics=["accuracy"])

callbacks = [

keras.callbacks.ModelCheckpoint(

filepath="fine\_tuning.keras",

save\_best\_only=True,

monitor="val\_loss")

]

history = model.fit(

train\_dataset,

epochs=30,

validation\_data=validation\_dataset,

callbacks=callbacks)

# Generating Visualisation of the Passes

# Allows assessment of training

# Poor accuracy or high loss could indicate a need to adjust the layers, parameters or review the data samples

import matplotlib.pyplot as plt

acc = history.history["accuracy"]

val\_acc = history.history["val\_accuracy"]

loss = history.history["loss"]

val\_loss = history.history["val\_loss"]

epochs = range(1, len(acc) + 1)

plt.plot(epochs, acc, "bo", label="Training accuracy")

plt.plot(epochs, val\_acc, "b", label="Validation accuracy")

plt.title("Training and validation accuracy")

plt.legend()

plt.figure()

plt.plot(epochs, loss, "bo", label="Training loss")

plt.plot(epochs, val\_loss, "b", label="Validation loss")

plt.title("Training and validation loss")

plt.legend()

plt.show()

# Final accuracy assessment fo total model

model = keras.models.load\_model("fine\_tuning.keras")

test\_loss, test\_acc = model.evaluate(test\_dataset)

print(f"Test accuracy: {test\_acc:.3f}")

"""# Section 5: Downloading the Data Set"""

# Comment Out When Not Using

#from google.colab import drive

#from google.colab import files

#model.save('/content/model.h5')

#feature\_extraction = '/content/feature\_extraction.keras'

#feature\_extraction\_with\_data\_aug = '/content/feature\_extraction\_with\_data\_augmentation.keras'

#fine\_tuning = '/content/fine\_tuning.keras'

#files.download(feature\_extraction)

#files.download(feature\_extraction\_with\_data\_aug)

#files.download(fine\_tuning)

#files.download('/content/model.h5')

#### Model 2

# -\*- coding: utf-8 -\*-

"""DataSetV2 (1).ipynb

Automatically generated by Colaboratory.

Original file is located at

https://colab.research.google.com/drive/1RvL14cPxrPYyS9xwAUs9eo5qk3iEkID0

# File Header

\*\*File Author\*\*: Emily Fletcher

\*\*Student Number\*\*: 18410839

\*\*File Purpose\*\*: Passing a set of images into a Data Set, Aim is to train a model

on recognition of the 5 people sampled.

\*\*Last Changed\*\*: 16/05/2023

\*\*Last Model Download\*\*: 16/05/2023

\*\*Version\*\*: 2.0

\*\*Instructions\*\*

Ensure model is corrected to Google Drive with correct directories before proceeding through Stage 2 Onwards

"""

# Key Imports

import tensorflow as tf

from tensorflow import keras

from tensorflow.keras import layers, regularizers

from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping

from google.colab import drive

import os, shutil

from tensorflow.keras.utils import image\_dataset\_from\_directory

import numpy as np

import matplotlib.pyplot as plt

"""# Stage 1: Connecting to the Google Drive"""

# Force Remount to account for Google Drive Reload Errors

drive.mount('/content/drive', force\_remount=True)

#Checking Path is valid

print(os.path.exists('/content/drive/MyDrive/DataSetAIAssignment/training'))

print(os.path.exists('/content/drive/MyDrive/DataSetAIAssignment/test'))

print(os.path.exists('/content/drive/MyDrive/DataSetAIAssignment/validation'))

# Main File Path

base\_dir = '/content/drive/MyDrive/DataSetAIAssignment/'

# Setting Parent Folders

train\_dir = os.path.join(base\_dir, 'training')

validation\_dir = os.path.join(base\_dir, 'validation')

test\_dir = os.path.join(base\_dir, 'test')

# Setting Train Folders For Users

train\_CB\_dir = os.path.join(train\_dir, 'CB') # CB

train\_EF\_dir = os.path.join(train\_dir, 'EF') # EF

train\_JS\_dir = os.path.join(train\_dir, 'JS') # JS

train\_MB\_dir = os.path.join(train\_dir, 'MB') # MB

train\_RK\_dir = os.path.join(train\_dir, 'RK') # RK

# Setting Validation Folders For Users

validation\_CB\_dir = os.path.join(validation\_dir, 'CB') # CB

validation\_EF\_dir = os.path.join(validation\_dir, 'EF') # EF

validation\_JS\_dir = os.path.join(validation\_dir, 'JS') # JS

validation\_MB\_dir = os.path.join(validation\_dir, 'MB') # MB

validation\_RK\_dir = os.path.join(validation\_dir, 'RK') # RK

# Setting Test Folders For Users

test\_CB\_dir = os.path.join(test\_dir, 'CB') # CB

test\_EF\_dir = os.path.join(test\_dir, 'EF') # EF

test\_JS\_dir = os.path.join(test\_dir, 'JS') # JS

test\_MB\_dir = os.path.join(test\_dir, 'MB') # MB

test\_RK\_dir = os.path.join(test\_dir, 'RK') # RK

"""# Section 2: First Pass Of Iterations

\* Assigning Batch Size

\* Setting Image Size

\* Creating VGG16 Base

"""

# Assigning Batches and Image Sizes

# Image Sizes Allow For Size Consistency

# Training Batches

train\_dataset = image\_dataset\_from\_directory(

train\_dir,

image\_size=(180, 180),

batch\_size=32)

# Validation Batches

validation\_dataset = image\_dataset\_from\_directory(

validation\_dir,

image\_size=(180, 180),

batch\_size=32)

# Testing Batches

test\_dataset = image\_dataset\_from\_directory(

test\_dir,

image\_size=(180, 180),

batch\_size=32)

# Creating of the VGG16 Pre-Trained Neural Network

# Weights State The Image Set That Should Be Used

# input\_shape should match image size and type used in batches

conv\_base = keras.applications.vgg16.VGG16(

weights="imagenet",

include\_top=False,

input\_shape=(180, 180, 3),

pooling=None

)

# Prints A Summary For The User, Allows Model Checks

conv\_base.summary()

# Assigning labels

def get\_features\_and\_labels(dataset):

all\_features = []

all\_labels = []

for images, labels in dataset:

preprocessed\_images = keras.applications.vgg16.preprocess\_input(images)

features = conv\_base.predict(preprocessed\_images)

all\_features.append(features)

all\_labels.append(labels)

return np.concatenate(all\_features), np.concatenate(all\_labels)

#Applying All Labels And Features To Each DataSet

train\_features, train\_labels = get\_features\_and\_labels(train\_dataset)

val\_features, val\_labels = get\_features\_and\_labels(validation\_dataset)

test\_features, test\_labels = get\_features\_and\_labels(test\_dataset)

# Applying Layers to the Model

# Helps to Prevent OverFitting

# Can Fine Tune By Adjusting Layers

inputs = keras.Input(shape=(5, 5, 512))

x = layers.Flatten()(inputs)

x = layers.Dense(256, activation="relu", kernel\_regularizer=regularizers.l2(0.01))(x)

x = layers.Dense(128, activation="relu", kernel\_regularizer=regularizers.l2(0.01))(x)

x = layers.Dense(64, activation="relu", kernel\_regularizer=regularizers.l2(0.01))(x)

x = layers.Dropout(0.2)(x)

outputs = layers.Dense(5, activation="softmax")(x)

model = keras.Model(inputs, outputs)

model.compile(loss="sparse\_categorical\_crossentropy",

optimizer="rmsprop",

metrics=["accuracy"])

learning\_rate = 0.001 # Change this value to your desired learning rate

optimizer = keras.optimizers.RMSprop(learning\_rate=learning\_rate)

# Define the EarlyStopping callback

early\_stopping = keras.callbacks.EarlyStopping(

monitor='val\_loss', # Metric to monitor

patience=5, # Number of epochs with no improvement before stopping

restore\_best\_weights=True # Restore the weights of the best epoch

)

# Fitting the Features And Labels From the VGG16 Model

callbacks = [

keras.callbacks.ModelCheckpoint(

filepath="feature\_extraction.keras",

save\_best\_only=True,

monitor="val\_loss"),

early\_stopping # Add the early stopping callback

]

# Can Adjust epochs To Refine Model

# 20 Chosen as visualisation graph levels out at around pass 15

history = model.fit(

train\_features, train\_labels,

epochs=30,

validation\_data=(val\_features, val\_labels),

callbacks=callbacks)

# Generating Visualisation of the Passes

# Allows assessment of training

# Poor accuracy or high loss could indicate a need to adjust the layers, parameters or review the data samples

acc = history.history["accuracy"]

val\_acc = history.history["val\_accuracy"]

loss = history.history["loss"]

val\_loss = history.history["val\_loss"]

epochs = range(1, len(acc) + 1)

plt.plot(epochs, acc, "bo", label="Training accuracy")

plt.plot(epochs, val\_acc, "b", label="Validation accuracy")

plt.title("Training and validation accuracy")

plt.legend()

plt.figure()

plt.plot(epochs, loss, "bo", label="Training loss")

plt.plot(epochs, val\_loss, "b", label="Validation loss")

plt.title("Training and validation loss")

plt.legend()

plt.show()

"""# Section 3: Further Refinement and Second Set of Iterations

\* Freezing the Base VGG16 Model

\* Adding Image Transformation (Augmentation)

"""

# Re-initialising the VGG16 Model

# This time bases are frozen, prevents overwriting of previous work

conv\_base = keras.applications.vgg16.VGG16(

weights="imagenet",

include\_top=False)

conv\_base.trainable = False

# THis shows to the author the effect of the freezing

# If values are the same then the model has not be frozen and should be reviewed before continuing

conv\_base.trainable = True

print("This is the number of trainable weights "

"before freezing the conv base:", len(conv\_base.trainable\_weights))

conv\_base.trainable = False

print("This is the number of trainable weights "

"after freezing the conv base:", len(conv\_base.trainable\_weights))

# Applying image transformations

# Allows to artifically grow the dataset

# Applying image transformations

# Allows to artifically grow the dataset

# Can adjust results by changing types of transformations and parameters

data\_augmentation = keras.Sequential(

[

layers.RandomFlip("horizontal"),

layers.RandomRotation(0.1),

layers.RandomZoom(0.2),

]

)

# Adding layers

# Can add more layers if needed to fine tune

# Same as previous method

inputs = keras.Input(shape=(180, 180, 3))

x = data\_augmentation(inputs)

x = keras.applications.vgg16.preprocess\_input(x)

x = conv\_base(x)

x = layers.Flatten()(x)

x = layers.Dense(256, activation="relu", kernel\_regularizer=regularizers.l2(0.01))(x)

x = layers.Dense(128, activation="relu", kernel\_regularizer=regularizers.l2(0.01))(x)

x = layers.Dense(64, activation="relu", kernel\_regularizer=regularizers.l2(0.01))(x)

x = layers.Dropout(0.5)(x)

outputs = layers.Dense(5, activation="softmax")(x)

model = keras.Model(inputs, outputs)

model.compile(loss="sparse\_categorical\_crossentropy",

optimizer="rmsprop",

metrics=["accuracy"])

# Prints a model summary, allows for user to check accuracy

model.summary()

# Fitting the Features And Labels From the VGG16 Model

callbacks = [

keras.callbacks.ModelCheckpoint(

filepath="feature\_extraction\_with\_data\_augmentation.keras",

save\_best\_only=True,

monitor="val\_loss"

),

EarlyStopping(

monitor="val\_loss",

patience=5, # Number of epochs with no improvement after which training will be stopped

restore\_best\_weights=True

)

]

# Can Adjust epochs To Refine Model

# 20 Chosen as visualisation graph levels out at around pass 15

history = model.fit(

train\_dataset,

epochs=30,

validation\_data=validation\_dataset,

callbacks=callbacks)

# Generating Visualisation of the Passes

# Allows assessment of training

# Poor accuracy or high loss could indicate a need to adjust the layers, parameters or review the data samples

import matplotlib.pyplot as plt

acc = history.history["accuracy"]

val\_acc = history.history["val\_accuracy"]

loss = history.history["loss"]

val\_loss = history.history["val\_loss"]

epochs = range(1, len(acc) + 1)

plt.plot(epochs, acc, "bo", label="Training accuracy")

plt.plot(epochs, val\_acc, "b", label="Validation accuracy")

plt.title("Training and validation accuracy")

plt.legend()

plt.figure()

plt.plot(epochs, loss, "bo", label="Training loss")

plt.plot(epochs, val\_loss, "b", label="Validation loss")

plt.title("Training and validation loss")

plt.legend()

plt.show()

# Finding the accuracy of the current model

test\_model = keras.models.load\_model(

"feature\_extraction\_with\_data\_augmentation.keras")

test\_loss, test\_acc = test\_model.evaluate(test\_dataset)

print(f"Test accuracy: {test\_acc:.3f}")

"""# Section 4: Final Interations and Fine Tuning

\* Un-Freezing Layers

\* Freezing Last Four Layers

"""

# Setting to true unfreezes all the previous frozen layers

# Removes the last four layers from being trainable

# Allows training for general features without removing all progress so far

conv\_base.trainable = True

for layer in conv\_base.layers[:-4]:

layer.trainable = False

# Building a summary of the current model

conv\_base.summary()

# Compiling the final model before running the fitting process

# Can adjust the custom learning rate to fine tweak model

# Can adjust the epochs to fine tune the model

model.compile(loss="sparse\_categorical\_crossentropy",

optimizer=keras.optimizers.RMSprop(learning\_rate=1e-5),

metrics=["accuracy"])

callbacks = [

keras.callbacks.ModelCheckpoint(

filepath="fine\_tuning.keras",

save\_best\_only=True,

monitor="val\_loss"

),

EarlyStopping(

monitor="val\_loss",

patience=5, # Number of epochs with no improvement after which training will be stopped

restore\_best\_weights=True

)

]

# Can Adjust epochs To Refine Model

# 20 Chosen as visualisation graph levels out at around pass 15

history = model.fit(

train\_dataset,

epochs=30,

validation\_data=validation\_dataset,

callbacks=callbacks)

# Generating Visualisation of the Passes

# Allows assessment of training

# Poor accuracy or high loss could indicate a need to adjust the layers, parameters or review the data samples

import matplotlib.pyplot as plt

acc = history.history["accuracy"]

val\_acc = history.history["val\_accuracy"]

loss = history.history["loss"]

val\_loss = history.history["val\_loss"]

epochs = range(1, len(acc) + 1)

plt.plot(epochs, acc, "bo", label="Training accuracy")

plt.plot(epochs, val\_acc, "b", label="Validation accuracy")

plt.title("Training and validation accuracy")

plt.legend()

plt.figure()

plt.plot(epochs, loss, "bo", label="Training loss")

plt.plot(epochs, val\_loss, "b", label="Validation loss")

plt.title("Training and validation loss")

plt.legend()

plt.show()

# Final accuracy assessment fo total model

model = keras.models.load\_model("fine\_tuning.keras")

test\_loss, test\_acc = model.evaluate(test\_dataset)

print(f"Test accuracy: {test\_acc:.3f}")

"""# Section 5: Downloading the Data Set"""

# Comment Out When Not Using

#from google.colab import drive

#from google.colab import files

#model.save('/content/model.h5')

#feature\_extraction = '/content/feature\_extraction.keras'

#feature\_extraction\_with\_data\_aug = '/content/feature\_extraction\_with\_data\_augmentation.keras'

#fine\_tuning = '/content/fine\_tuning.keras'

#files.download(feature\_extraction)

#files.download(feature\_extraction\_with\_data\_aug)

#files.download(fine\_tuning)

#files.download('/content/model.h5')

#### Model 3

# -\*- coding: utf-8 -\*-

"""DataSetV3.ipynb

Automatically generated by Colaboratory.

Original file is located at

https://colab.research.google.com/drive/1ZQy7xUw1Oy7oRPP3h0wyyuxgtto1PfY0

# File Header

\*\*File Author\*\*: Emily Fletcher

\*\*Student Number\*\*: 18410839

\*\*File Purpose\*\*: Passing a set of images into a Data Set, Aim is to train a model

on recognition of the 5 people sampled.

\*\*Last Changed\*\*: 17/05/2023

\*\*Last Model Download\*\*: 17/05/2023

\*\*Version\*\*: 3.0

\*\*Instructions\*\*

Ensure model is corrected to Google Drive with correct directories before proceeding through Stage 2 Onwards

"""

# Key Imports

import tensorflow as tf

from tensorflow import keras

from tensorflow.keras import layers, regularizers

from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping

from google.colab import drive

import os, shutil

from tensorflow.keras.utils import image\_dataset\_from\_directory

import numpy as np

import matplotlib.pyplot as plt

"""# Stage 1: Connecting to the Google Drive"""

# Force Remount to account for Google Drive Reload Errors

drive.mount('/content/drive', force\_remount=True)

#Checking Path is valid

print(os.path.exists('/content/drive/MyDrive/DataSetAIAssignment/training'))

print(os.path.exists('/content/drive/MyDrive/DataSetAIAssignment/test'))

print(os.path.exists('/content/drive/MyDrive/DataSetAIAssignment/validation'))

# Main File Path

base\_dir = '/content/drive/MyDrive/DataSetAIAssignment/'

# Setting Parent Folders

train\_dir = os.path.join(base\_dir, 'training')

validation\_dir = os.path.join(base\_dir, 'validation')

test\_dir = os.path.join(base\_dir, 'test')

# Setting Train Folders For Users

train\_CB\_dir = os.path.join(train\_dir, 'CB') # CB

train\_EF\_dir = os.path.join(train\_dir, 'EF') # EF

train\_JS\_dir = os.path.join(train\_dir, 'JS') # JS

train\_MB\_dir = os.path.join(train\_dir, 'MB') # MB

train\_RK\_dir = os.path.join(train\_dir, 'RK') # RK

# Setting Validation Folders For Users

validation\_CB\_dir = os.path.join(validation\_dir, 'CB') # CB

validation\_EF\_dir = os.path.join(validation\_dir, 'EF') # EF

validation\_JS\_dir = os.path.join(validation\_dir, 'JS') # JS

validation\_MB\_dir = os.path.join(validation\_dir, 'MB') # MB

validation\_RK\_dir = os.path.join(validation\_dir, 'RK') # RK

# Setting Test Folders For Users

test\_CB\_dir = os.path.join(test\_dir, 'CB') # CB

test\_EF\_dir = os.path.join(test\_dir, 'EF') # EF

test\_JS\_dir = os.path.join(test\_dir, 'JS') # JS

test\_MB\_dir = os.path.join(test\_dir, 'MB') # MB

test\_RK\_dir = os.path.join(test\_dir, 'RK') # RK

"""# Section 2: First Pass Of Iterations

\* Assigning Batch Size

\* Setting Image Size

\* Creating VGG16 Base

"""

# Assigning Batches and Image Sizes

# Image Sizes Allow For Size Consistency

# Training Batches

train\_dataset = image\_dataset\_from\_directory(

train\_dir,

image\_size=(180, 180),

batch\_size=32)

# Validation Batches

validation\_dataset = image\_dataset\_from\_directory(

validation\_dir,

image\_size=(180, 180),

batch\_size=32)

# Testing Batches

test\_dataset = image\_dataset\_from\_directory(

test\_dir,

image\_size=(180, 180),

batch\_size=32)

# Creating of the VGG16 Pre-Trained Neural Network

# Weights State The Image Set That Should Be Used

# input\_shape should match image size and type used in batches

conv\_base = keras.applications.vgg16.VGG16(

weights="imagenet",

include\_top=False,

input\_shape=(180, 180, 3),

pooling=None

)

# Prints A Summary For The User, Allows Model Checks

conv\_base.summary()

# Assigning labels

def get\_features\_and\_labels(dataset):

all\_features = []

all\_labels = []

for images, labels in dataset:

preprocessed\_images = keras.applications.vgg16.preprocess\_input(images)

features = conv\_base.predict(preprocessed\_images)

all\_features.append(features)

all\_labels.append(labels)

return np.concatenate(all\_features), np.concatenate(all\_labels)

#Applying All Labels And Features To Each DataSet

train\_features, train\_labels = get\_features\_and\_labels(train\_dataset)

val\_features, val\_labels = get\_features\_and\_labels(validation\_dataset)

test\_features, test\_labels = get\_features\_and\_labels(test\_dataset)

# Applying Layers to the Model

# Helps to Prevent OverFitting

# Can Fine Tune By Adjusting Layers

inputs = keras.Input(shape=(5, 5, 512))

x = layers.Conv2D(32, kernel\_size=(3, 3), activation='relu')(inputs)

x = layers.MaxPooling2D(pool\_size=(2, 2))(x)

x = layers.Flatten()(x)

x = layers.Dense(256, activation="relu", kernel\_regularizer=regularizers.l2(0.01))(x)

x = layers.Dropout(0.2)(x)

x = layers.Dense(128, activation="relu", kernel\_regularizer=regularizers.l2(0.01))(x) # Add an additional dense layer

x = layers.Dropout(0.2)(x)

outputs = layers.Dense(5, activation="softmax")(x)

model = keras.Model(inputs, outputs)

model.compile(loss="sparse\_categorical\_crossentropy",

optimizer="rmsprop",

metrics=["accuracy"])

learning\_rate = 0.001 # Ammend value to your change learning rate

optimizer = keras.optimizers.RMSprop(learning\_rate=learning\_rate)

# Define the EarlyStopping callback

early\_stopping = keras.callbacks.EarlyStopping(

monitor='val\_loss', # Metric to monitor

patience=3, # Number of epochs with no improvement before stopping

restore\_best\_weights=True # Restore the weights of the best epoch

)

# Fitting the Features And Labels From the VGG16 Model

callbacks = [

keras.callbacks.ModelCheckpoint(

filepath="feature\_extraction.keras",

save\_best\_only=True,

monitor="val\_loss"),

early\_stopping # Add the early stopping callback

]

# Can Adjust epochs To Refine Model

# 20 Chosen as visualisation graph levels out at around pass 15

history = model.fit(

train\_features, train\_labels,

epochs=30,

validation\_data=(val\_features, val\_labels),

callbacks=callbacks)

# Generating Visualisation of the Passes

# Allows assessment of training

# Poor accuracy or high loss could indicate a need to adjust the layers, parameters or review the data samples

acc = history.history["accuracy"]

val\_acc = history.history["val\_accuracy"]

loss = history.history["loss"]

val\_loss = history.history["val\_loss"]

epochs = range(1, len(acc) + 1)

plt.plot(epochs, acc, "bo", label="Training accuracy")

plt.plot(epochs, val\_acc, "b", label="Validation accuracy")

plt.title("Training and validation accuracy")

plt.legend()

plt.figure()

plt.plot(epochs, loss, "bo", label="Training loss")

plt.plot(epochs, val\_loss, "b", label="Validation loss")

plt.title("Training and validation loss")

plt.legend()

plt.show()

"""# Section 3: Further Refinement and Second Set of Iterations

\* Freezing the Base VGG16 Model

\* Adding Image Transformation (Augmentation)

"""

# Re-initialising the VGG16 Model

# This time bases are frozen, prevents overwriting of previous work

conv\_base = keras.applications.vgg16.VGG16(

weights="imagenet",

include\_top=False)

conv\_base.trainable = False

# THis shows to the author the effect of the freezing

# If values are the same then the model has not be frozen and should be reviewed before continuing

conv\_base.trainable = True

print("This is the number of trainable weights "

"before freezing the conv base:", len(conv\_base.trainable\_weights))

conv\_base.trainable = False

print("This is the number of trainable weights "

"after freezing the conv base:", len(conv\_base.trainable\_weights))

# Applying image transformations

# Allows to artifically grow the dataset

# Applying image transformations

# Allows to artifically grow the dataset

# Can adjust results by changing types of transformations and parameters

data\_augmentation = keras.Sequential(

[

layers.RandomFlip("horizontal"),

layers.RandomRotation(0.1),

layers.RandomZoom(0.2),

]

)

# Adding layers

# Can add more layers if needed to fine tune

# Same as previous method

inputs = keras.Input(shape=(180, 180, 3))

x = data\_augmentation(inputs)

x = keras.applications.vgg16.preprocess\_input(x)

x = conv\_base(x)

x = layers.Conv2D(64, kernel\_size=(3, 3), activation='relu')(inputs)

x = layers.MaxPooling2D(pool\_size=(2, 2))(x)

x = layers.Flatten()(x)

x = layers.Dense(256, activation="relu", kernel\_regularizer=regularizers.l2(0.01))(x)

x = layers.Dropout(0.2)(x)

x = layers.Dense(128, activation="relu", kernel\_regularizer=regularizers.l2(0.01))(x) # Add an additional dense layer

x = layers.Dropout(0.2)(x)

outputs = layers.Dense(5, activation="softmax")(x)

model = keras.Model(inputs, outputs)

model.compile(loss="sparse\_categorical\_crossentropy",

optimizer="rmsprop",

metrics=["accuracy"])

learning\_rate = 0.001 # Ammend value to your change learning rate

optimizer = keras.optimizers.RMSprop(learning\_rate=learning\_rate)

# Prints a model summary, allows for user to check accuracy

model.summary()

# Fitting the Features And Labels From the VGG16 Model

callbacks = [

keras.callbacks.ModelCheckpoint(

filepath="feature\_extraction\_with\_data\_augmentation.keras",

save\_best\_only=True,

monitor="val\_loss"

),

EarlyStopping(

monitor="val\_loss",

patience=3, # Number of epochs with no improvement after which training will be stopped

restore\_best\_weights=True

)

]

# Can Adjust epochs To Refine Model

# 20 Chosen as visualisation graph levels out at around pass 15

history = model.fit(

train\_dataset,

epochs=45,

validation\_data=validation\_dataset,

callbacks=callbacks)

# Generating Visualisation of the Passes

# Allows assessment of training

# Poor accuracy or high loss could indicate a need to adjust the layers, parameters or review the data samples

import matplotlib.pyplot as plt

acc = history.history["accuracy"]

val\_acc = history.history["val\_accuracy"]

loss = history.history["loss"]

val\_loss = history.history["val\_loss"]

epochs = range(1, len(acc) + 1)

plt.plot(epochs, acc, "bo", label="Training accuracy")

plt.plot(epochs, val\_acc, "b", label="Validation accuracy")

plt.title("Training and validation accuracy")

plt.legend()

plt.figure()

plt.plot(epochs, loss, "bo", label="Training loss")

plt.plot(epochs, val\_loss, "b", label="Validation loss")

plt.title("Training and validation loss")

plt.legend()

plt.show()

# Finding the accuracy of the current model

test\_model = keras.models.load\_model(

"feature\_extraction\_with\_data\_augmentation.keras")

test\_loss, test\_acc = test\_model.evaluate(test\_dataset)

print(f"Test accuracy: {test\_acc:.3f}")

"""# Section 4: Final Interations and Fine Tuning

\* Un-Freezing Layers

\* Freezing Last Four Layers

"""

# Setting to true unfreezes all the previous frozen layers

# Removes the last four layers from being trainable

# Allows training for general features without removing all progress so far

conv\_base.trainable = True

for layer in conv\_base.layers[:-4]:

layer.trainable = False

# Building a summary of the current model

conv\_base.summary()

# Compiling the final model before running the fitting process

# Can adjust the custom learning rate to fine tweak model

# Can adjust the epochs to fine tune the model

model.compile(loss="sparse\_categorical\_crossentropy",

optimizer=keras.optimizers.RMSprop(learning\_rate=1e-5),

metrics=["accuracy"])

callbacks = [

keras.callbacks.ModelCheckpoint(

filepath="fine\_tuning.keras",

save\_best\_only=True,

monitor="val\_loss"

),

EarlyStopping(

monitor="val\_loss",

patience=8, # Number of epochs with no improvement after which training will be stopped

restore\_best\_weights=True

)

]

# Can Adjust epochs To Refine Model

# 20 Chosen as visualisation graph levels out at around pass 15

history = model.fit(

train\_dataset,

epochs=45,

validation\_data=validation\_dataset,

callbacks=callbacks)

# Generating Visualisation of the Passes

# Allows assessment of training

# Poor accuracy or high loss could indicate a need to adjust the layers, parameters or review the data samples

import matplotlib.pyplot as plt

acc = history.history["accuracy"]

val\_acc = history.history["val\_accuracy"]

loss = history.history["loss"]

val\_loss = history.history["val\_loss"]

epochs = range(1, len(acc) + 1)

plt.plot(epochs, acc, "bo", label="Training accuracy")

plt.plot(epochs, val\_acc, "b", label="Validation accuracy")

plt.title("Training and validation accuracy")

plt.legend()

plt.figure()

plt.plot(epochs, loss, "bo", label="Training loss")

plt.plot(epochs, val\_loss, "b", label="Validation loss")

plt.title("Training and validation loss")

plt.legend()

plt.show()

# Final accuracy assessment fo total model

model = keras.models.load\_model("fine\_tuning.keras")

test\_loss, test\_acc = model.evaluate(test\_dataset)

print(f"Test accuracy: {test\_acc:.3f}")

"""# Section 5: Downloading the Data Set"""

# Comment Out When Not Using

from google.colab import drive

from google.colab import files

model.save('/content/model.h5')

#feature\_extraction = '/content/feature\_extraction.keras'

#feature\_extraction\_with\_data\_aug = '/content/feature\_extraction\_with\_data\_augmentation.keras'

#fine\_tuning = '/content/fine\_tuning.keras'

#files.download(feature\_extraction)

#files.download(feature\_extraction\_with\_data\_aug)

#files.download(fine\_tuning)

files.download('/content/model.h5')

### Application Code

#### cameraFeed.py

# AI - Attendance System - Emily Fletcher 18410839

# Project Title: AiAssignment2

# File Title: cameraFeed.py

# File Purpose: Loads camera frames as well as assigns computer vision techniques

# Author: Emily Fletcher

# Student Number: 18410839

# Version: 1.0

# Import Statements

import cv2

# Load Face Recognition From Open CV

face\_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade\_frontalface\_default.xml')

# Global variables

labelStatus = "No Face Detected"

def detect\_and\_draw\_faces(frame):

# Fetch Global Label

global labelStatus

# Changing the frame to Grey Scale, Makes Face Recognition Easier

gray = cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY)

faces = face\_cascade.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=5, minSize=(30, 30))

# If a face has been detected, update global label

if len(faces) > 0:

labelStatus = "Face Detected"

else:

labelStatus = "No Face Detected"

# Code to Draw a Green Box around any faces found

for (x, y, w, h) in faces:

# Draw a rectangle around the face

cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)

# Returning the current video frame

# Returns if a face has been detected

# Returns Face Boolean

return frame, labelStatus

#### dateTime.py

# AI - Attendance System - Emily Fletcher 18410839

# Project Title: AiAssignment2

# File Title: dateTime.py

# File Purpose: Contains functions for setting times and dates

# Author: Emily Fletcher

# Student Number: 18410839

# Version: 1.0

# Import Statements

from datetime import datetime, date

# Receives time from spinner

# Converts it to a time that can be used in other functions

def getTime(time):

# print("Selected Time:", time) Testing Statement

timeConvert = datetime.strptime(time, "%H:%M")

formattedTime = timeConvert.strftime("%I:%M %p")

# print("Converted Time:", formattedTime) Testing Statement

return formattedTime

# Gets the current date from the applications host machine

def getDate():

currentDate = date.today()

formattedDate = currentDate.strftime("%d-%m-%Y")

# print("Formatted Date:", formattedDate) Testing Statement

return formattedDate

def getCurrentTime():

currentTime = datetime.now()

formattedTime = currentTime.strftime("%H:%M")

# print("Formatted Time:", formattedTime) Testing Statement

return formattedTime

#### files.py

# AI - Attendance System - Emily Fletcher 18410839

# Project Title: AiAssignment2

# File Title: files.py

# File Purpose: Creates the Session Log and Parent Folder

# Author: Emily Fletcher

# Student Number: 18410839

# Version: 1.0

# Import Statements

import os

import dateTime

def createSessionLog(time):

# Create Folder if it does not exist

folder\_name = "sessions"

if not os.path.exists(folder\_name):

os.makedirs(folder\_name)

# print("Folder Created") Testing Statement

# Getting date values for the session

todaysDate = dateTime.getDate()

# : causes issues with the file name, so replaced with underscore

time = time.replace(":", "\_")

fileName = "Session\_Date\_" + todaysDate + "\_Session\_Time\_" + time + ".txt"

filePath = os.path.join(folder\_name, fileName)

startingText = "Start of Attendance Log\n"

# Writes the starting text and marks the file as readable

with open(filePath, "w") as file:

file.write(startingText)

# Used to edit and save to a file

def writeToFile(fileName, fileContent, studentName, folderPath = "sessions"):

if folderPath:

fileName = os.path.join(folderPath, fileName)

totalContent = "Student: " + studentName + " Late Status: " + fileContent

with open(fileName, 'w') as file:

file.write(totalContent)

#### lateChecker.py

# AI - Attendance System - Emily Fletcher 18410839

# Project Title: AiAssignment2

# File Title: lateChecker.py

# File Purpose: Takes date objects and finds the difference, used to report if the student is late or on time.

# Author: Emily Fletcher

# Student Number: 18410839

# Version: 1.0

# Import Statements

from datetime import datetime, time

def check(currentTimeStr, convertedTimeStr):

# Convert Time To AM/PM Layout

convertedTime = datetime.strptime(convertedTimeStr, "%I:%M %p").time()

print(convertedTime)

# Convert Time to Hours Minute Layout

currentTime = datetime.strptime(currentTimeStr, "%H:%M").time()

print(currentTime)

# Combine the times and subtract from each other to find difference

timeDifference = datetime.combine(datetime.min, convertedTime) - datetime.combine(datetime.min, currentTime)

print(timeDifference)

# Statements account for 15 minute leeway if late or if the student is very early.

if timeDifference.total\_seconds() < -15 \* 60:

print("The student is early.")

lateStatus = "The student is early"

elif -15 \* 60 <= timeDifference.total\_seconds() <= 15 \* 60:

print("Registered On Time")

lateStatus = "Registered On Time"

else:

print("The student is late.")

lateStatus = "The student is late"

return lateStatus

#### machineLearning.py

# AI - Attendance System - Emily Fletcher 18410839

# Project Title: AiAssignment2

# File Title: machineLearning.py

# File Purpose: Connects to the model and inserts the frame into the model

# Author: Emily Fletcher

# Student Number: 18410839

# Version: 1.0

# Import Statements

import cv2

from tensorflow import keras

import os

import shutil

# from tensorflow.keras.utils import image\_dataset\_from\_directory

from tensorflow import keras

# from tensorflow.keras import layers

import numpy as np

import matplotlib.pyplot as plt

# from google.colab import drive

# from google.colab import files

# Load the saved model

model\_dir = "model"

model\_file = "model.h5"

model\_path = os.path.join(model\_dir, model\_file)

# Check if the model file exists

if os.path.exists(model\_path):

try:

# Load the model from the file

model = keras.models.load\_model(model\_path)

except Exception as e:

print("Error loading the model")

else:

print("Model file not found!")

def frameSetup(frame):

imagePath = "imagesForTest\RK.jpg"

frame = cv2.imread(imagePath)

# Preprocess the frame (resize, normalize, etc.) to match the model's requirements

resizeFrame = cv2.resize(frame, (180, 180))

resizeFrame = resizeFrame / 255.0 # Normalize the pixel values

resizeFrame = np.expand\_dims(resizeFrame, axis=0)

print("Frame Processed")

# List of Students

students = {

0: "Alpha",

1: "Bravo",

2: "Charlie",

3: "Delta",

4: "Echo"

}

prediction = model.predict(resizeFrame)

predicted = np.argmax(prediction)

# Map the predicted class to a student name using the class\_mapping dictionary

student\_found = students.get(predicted, "Unknown")

print("Prediction Weights:", prediction)

print("Predicted student:", student\_found)

return student\_found

#### main.py

# AI - Attendance System - Emily Fletcher 18410839

import glob

# Project Title: AiAssignment2

# File Title: main.py

# File Purpose: Main Controller File, also contains the GUI

# Author: Emily Fletcher

# Student Number: 18410839

# Version: 1.0

# Import Statements

import os

import cv2

import tkinter as tk

import dateTime

from PIL import Image, ImageTk

import files

import lateChecker

import machineLearning

import saveFrame

from cameraFeed import detect\_and\_draw\_faces

# Global Variables

videoPause = False

convertedTime = None

lateStatus = None

studentName = None

# Function that forgets current tab and loads the next.

def changeTab(tab):

setUp.pack\_forget()

attendance.pack\_forget()

register.pack\_forget()

tab.pack()

# Register Attendance Button Action (3rd Tab)

def registerClicked():

global convertedTime

global studentName

print("Register Clicked")

# Parent Folder = sessions

folder\_path = "sessions"

allSessions = os.listdir(folder\_path)

lastFile = None

for currentFile in allSessions:

if lastFile is None or currentFile > lastFile:

lastFile = currentFile

print("Last file in folder:", lastFile)

time = dateTime.getCurrentTime()

lateStatusCheck = lateChecker.check(time, convertedTime)

lateStatus.config(text=lateStatusCheck)

files.writeToFile(lastFile, lateStatusCheck, studentName)

changeTab(attendance)

# Start Session Button Action (1st Tab)

# Gets Spinner Time and Converts to Time Object

# Creates Session Log

# Changes Tab to Attendance

def startButtonClick():

global convertedTime

# print("Start Button Clicked") Testing Print

selectedTime = var.get()

convertedTime = dateTime.getTime(selectedTime)

files.createSessionLog(convertedTime)

changeTab(attendance)

# Register Attendance Button Action (2nd Tab)

# Gets the Frame and saves it as a png

def buttonClick():

global studentName

# print("Button Clicked") Testing Print

frame = fetchFrame.frame

saveFrame.saveFrame(frame)

image\_path = "temp/frame.png"

# If the image path exists open the image

# (Used for the 3rd Tab)

if os.path.exists(image\_path):

image = Image.open(image\_path)

resized\_image = image.resize((250, 200))

photo = ImageTk.PhotoImage(resized\_image)

frameLabel.config(image=photo)

frameLabel.image = photo

studentName = machineLearning.frameSetup(frame)

print("Predicted Student Name:", studentName)

studentMatch.config(text=studentName, font=("Helvetica", 16, "bold"), padx=10, pady=10)

changeTab(register)

# If cancelled, takes the user back to the 2nd Tab

def cancelButtonClicked():

print("Cancel Button Clicked")

changeTab(attendance)

# Create the Tkinter window

root = tk.Tk()

root.title("Attendance System")

# Frame Set Up

setUp = tk.Frame(root)

attendance = tk.Frame(root)

register = tk.Frame(root)

# Setting Application Window Dimensions

wWidth = 800

wHeight = 600

root.geometry(f"{wWidth}x{wHeight}")

# Tab Set One

descriptLabel = tk.Label(setUp, text="Select a Session Start Time and Press Button to Start Recording Attendance")

descriptLabel.pack()

spinnerLabel = tk.Label(setUp, text="Select Time")

spinnerLabel.pack()

spinnerValues = ["9:00", "9:30", "10:00", "10:30", "11:00", "11:30", "12:00", "12:30", "13:00", "13:30",

"14:00", "14:30", "15:00", "15:30", "16:00", "16:30", "17:00"]

var = tk.StringVar(value=spinnerValues[0])

dropdown = tk.OptionMenu(setUp, var, \*spinnerValues)

dropdown.pack()

startButton = tk.Button(setUp, text="Start Session", command=startButtonClick)

startButton.pack()

# Tab Set Two

# Setting WebCam Dimensions

vWidth = int(wWidth \* 0.8)

vHeight = int(wHeight \* 0.8)

videoFrame = tk.Frame(attendance, width=vWidth, height=vHeight)

videoFrame.pack()

# Adding Video Frame and Labels to the App

videoSource = tk.Label(videoFrame)

videoSource.pack()

VSLabel = tk.Label(attendance, text="")

VSLabel.pack()

faceDetectionLabel = tk.Label(attendance)

faceDetectionLabel.pack()

button = tk.Button(attendance, text="Search for Student", command=buttonClick)

button.pack()

button.grid\_remove() # Hide the button initially

startSessionLabel = tk.Label(setUp, text="Start Session")

startSessionButton = tk.Label()

# Tab Set Three

# Setting Two Frames so Image is displayed side by side with the text values

savedFrame = tk.Label(register)

savedFrame.grid(row=0, column=0, padx=10, pady=10)

optionsFrame = tk.Frame(register)

optionsFrame.grid(row=0, column=1, padx=10, pady=10)

registerLabel = tk.Label(optionsFrame, text="If a Match is Found then Please Register Your Attendance")

registerLabel.pack()

frameLabel = tk.Label(savedFrame)

frameLabel.pack()

frameLabelText = tk.Label(savedFrame, text="Student Image")

frameLabelText.pack()

studentMatch = tk.Label(optionsFrame, text="No Match Found")

studentMatch.pack()

registerButton = tk.Button(optionsFrame, text="Register", command=registerClicked)

registerButton.pack()

timeRegistered = tk.Label(optionsFrame, text="Time Registered")

timeRegistered.pack()

lateStatus = tk.Label(optionsFrame, text="")

lateStatus.pack()

cancelButton = tk.Button(optionsFrame, text="Cancel Registration", command=cancelButtonClicked)

cancelButton.pack()

# Function to capture frames from the webcam and update GUI

def fetchFrame():

if not videoPause:

ret, frame = video.read()

if ret:

frame = cv2.cvtColor(frame, cv2.COLOR\_BGR2RGB)

frame = cv2.resize(frame, (vWidth, vHeight))

frame\_with\_faces, labelStatus = detect\_and\_draw\_faces(frame)

image = Image.fromarray(frame\_with\_faces)

photo = ImageTk.PhotoImage(image)

videoSource.config(image=photo)

videoSource.image = photo

faceDetectionLabel.config(text=labelStatus) # Update the label text

fetchFrame.frame = frame

# Only allows user to register attendance if there is a face in the frame

if labelStatus == "Face Detected":

button.pack() # Show the button

else:

button.pack\_forget() # Hide the button

# Schedule the next iteration of the capture\_frame function after 10 milliseconds

root.after(10, fetchFrame)

# Capture frames from webcam

video = cv2.VideoCapture(0)

fetchFrame()

# Ensure the correct tab is loaded when the application starts

changeTab(setUp)

root.mainloop()

#### saveFrame.py

# AI - Attendance System - Emily Fletcher 18410839

# Project Title: AiAssignment2

# File Title: saveFrame.py

# File Purpose: Controls the current frames and saving them as PNGS

# Author: Emily Fletcher

# Student Number: 18410839

# Version: 1.0

# Import Statements

import os

import cv2

from cameraFeed import face\_cascade

# Saves the current frame as a PNG

def saveFrame(frame):

# Create a folder if it does not exist.

folder\_name = "temp"

if not os.path.exists(folder\_name):

os.makedirs(folder\_name)

gray = cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY)

faces = face\_cascade.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=5, minSize=(30, 30))

if len(faces) > 0:

(x, y, w, h) = faces[0]

cropped\_frame = frame[y:y + h, x:x + w]

else:

cropped\_frame = frame

# Save the cropped frame as an image in the folder

file\_name = os.path.join(folder\_name, "frame.png")

# Converts the frame to a colour frame for ease of use

# Originally greyscale to help computer vision

cv2.imwrite(file\_name, cv2.cvtColor(cropped\_frame, cv2.COLOR\_RGB2BGR))

# print("Frame saved successfully!") Test Statements

# Deletes the frames in the folder

# Not currently used

def deleteFrame():

# Specify the folder to be cleared

folder\_name = "temp"

# Check if the folder exists

if os.path.exists(folder\_name):

# Get a list of all files in the folder

file\_list = os.listdir(folder\_name)

# Delete each file in the folder

for file in file\_list:

file\_path = os.path.join(folder\_name, file)

os.remove(file\_path)

print("Folder cleared")

else:

print("Folder does not exist.")

#### saveFrame.py

# AI - Attendance System - Emily Fletcher 18410839

# Project Title: AiAssignment2

# File Title: saveFrame.py

# File Purpose: Controls the current frames and saving them as PNGS

# Author: Emily Fletcher

# Student Number: 18410839

# Version: 1.0

# Import Statements

import os

import cv2

from cameraFeed import face\_cascade

# Saves the current frame as a PNG

def saveFrame(frame):

# Create a folder if it does not exist.

folder\_name = "temp"

if not os.path.exists(folder\_name):

os.makedirs(folder\_name)

gray = cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY)

faces = face\_cascade.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=5, minSize=(30, 30))

if len(faces) > 0:

(x, y, w, h) = faces[0]

cropped\_frame = frame[y:y + h, x:x + w]

else:

cropped\_frame = frame

# Save the cropped frame as an image in the folder

file\_name = os.path.join(folder\_name, "frame.png")

# Converts the frame to a colour frame for ease of use

# Originally greyscale to help computer vision

cv2.imwrite(file\_name, cv2.cvtColor(cropped\_frame, cv2.COLOR\_RGB2BGR))

# print("Frame saved successfully!") Test Statements

# Deletes the frames in the folder

# Not currently used

def deleteFrame():

# Specify the folder to be cleared

folder\_name = "temp"

# Check if the folder exists

if os.path.exists(folder\_name):

# Get a list of all files in the folder

file\_list = os.listdir(folder\_name)

# Delete each file in the folder

for file in file\_list:

file\_path = os.path.join(folder\_name, file)

os.remove(file\_path)

print("Folder cleared")

else:

print("Folder does not exist.")

#### File Structure

#### A screenshot of a computer Description automatically generated

## Appendix 2: Ongoing Commit Log