



# **Final Report**

## **Computer Vision and Object Detection for Pedestrian and Vehicle Traffic Optimisation Xavier Sanchis**

3<sup>rd</sup> Year Individual Project

I certify that all material in this thesis that is not my own work has been identified and that no material has been included for which a degree has previously been conferred on me.

**Signed** Xavier Sanchis

College of Engineering, Mathematics, and Physical Sciences  
University of Exeter



# Final Report

ECM3175/ECM3149

Title: Computer Vision and Object Detection for  
Pedestrian and Vehicle Traffic Optimisation

Date of submission: 02/05/2023

Student Name: Xavier Sanchis  
Programme: Electronic Engineering Beng  
Student number: 700051124  
Candidate number: 247437

Supervisor: Dr Anna Baldycheva

## Abstract

Inefficient traffic signal control methods have resulted in significant economic losses worldwide. The main reason for this is that the current pre-programmed signal control methods lack consideration for dynamic traffic conditions, leading to delays for both vehicles and pedestrians at junctions, which in turn leads to traffic jams and even accidents.

New projects have emerged that investigate the use of machine learning (ML) agents to control the phase decisions of traffic lights based on real-time data, to optimize traffic flow. The experimental results have shown a significant reduction in vehicle waiting times under this new model.

This project aims to contribute to this new concept by taking the initial steps to expand this idea into pedestrian crossings, by using the pre-trained ML object detection model “Yolov5” to fully map pedestrian and vehicle traffic conditions at these junctions. Furthermore, it aims to use convolutional neural networks programmed with the “tensorflow” library to classify pedestrians into age and disability groups to provide each one with a customized minimum safety crossing time.

This model detects people with more than 90% accuracy, it classifies them in the correct age group with an average accuracy of 87% and it classifies them in the correct disability group with an average accuracy of over 98%. Furthermore, it detects and classifies each subject within milliseconds. This shows positive initial results, however, the lack of generalizability of the model indicates the need for further training with a wider, more diverse dataset that is specific to this context before real life implementation.

*Keywords: Traffic, Pedestrian Crossing, Machine Learning, Object Detection, Computer Vision*

# Table of contents

1. Introduction and Background.....	1
2. Literature Review.....	2
2.1. Brief Background.....	2
2.2. Current Vehicle Traffic Control Methods.....	2
2.3. Current Pedestrian Traffic Control Methods.....	3
2.4. Machine Learning for Traffic Control .....	3
2.5. Image Classification Algorithms and Object Detection Model .....	5
3. Methodology and Theory.....	6
3.1. Artificial Neural Network Architecture and Theory .....	6
3.2. Model Overtraining Prevention and Datasets .....	8
3.3. Convolutional Neural Networks .....	9
3.4. Evaluation of Networks .....	11
3.5. Object Detection Models .....	11
4. Design and Analytical Investigation.....	12
4.1. Image Classification.....	12
4.2. Object Detection .....	17
4.3. Object Detection and Image Classification Integration .....	18
5. Presentation of Experimental Results .....	19
5.1. Age Classification Final Network Results-Accuracy .....	19
5.2. Age Classification Final Network Results-Time Taken .....	21
5.3. Disability Classification Final Network Results-Accuracy .....	22
5.4. Disability Classification Final Network Results-Time Taken .....	23
5.2. Integration with Object Detecction System .....	24
6. Discussion and Conclusions.....	26
6.1. Datasets.....	26
6.1. Convolutional Neural Networks .....	26
6.1. Future Work.....	26
7. Project Management .....	27
References.....	29

# 1. Introduction and Background

With today's renovated vision of what ideal cities should look like and motivated by the climate crisis, city councils throughout Europe have found in transportation their focal point for reform [1]. Many cities around Europe are now repudiating and transforming conventional vehicle spaces in the pursuit of greener, more walkable cities. Hence, pedestrian crossings (this paper refers to pelican, toucan and any other pedestrian crossing junction controlled by traffic lights as pedestrian crossings) are becoming an increasingly big part of the commuting routines for many.

However, the basis on which traffic lights in these junctions work today, are ineffective [2]. All traffic lights in the United Kingdom today operate with manually designed phase plans (for example pre-timed phase plans) that give very limited or no consideration to the fluctuations in the flow of traffic, leaving both pedestrians and vehicles having to cope with unnecessarily long waiting times at these junctions. The immediate consequence of this problem is traffic jams, which in 2019, was reported to cost the United Kingdom £6.9 billion in fuel, extra cost of transportation and lost productivity of workers [3]. That amount increased for the United States to more than \$81 billion last year, according to a report by Government Technology [4].

While much harder to quantify, time delays for pedestrians in pedestrian crossings can also account for an economical loss due to lost productivity of workers. Furthermore, it constitutes a safety hazard for impatient pedestrians who do not respect the traffic lights. In 2021, it is reported that 361 pedestrians were killed, over 5,000 were severely injured and over 11,000 were slightly injured in the UK [5].

This project goes through the initial steps to tackle this issue. It employs an object detection model (Yolov5 [6]) that can detect both pedestrians and vehicles at a junction, and therefore is able to fully map the state of traffic in real time. This is the first step towards programming a deep reinforcement learning agent to control the phase decisions of traffic lights at a junction as it requires this object detection capability to base its decisions on.

Furthermore, minimum safety crossing times are established to guarantee pedestrian safety in the event of the AI agent making an erroneous phase decision that could endanger pedestrians. This aims to limit the frequency of the policy changes in the phase plans by the agent, as policies which are too short, for example switching from a green light to a red light in a pedestrian traffic light too quickly, could be very unsafe.

Given that both age and disability play an important role in the mobility of people, the optimal minimum safety crossing time could be different for different age and disability characteristics. Therefore, in order to provide optimal minimum crossing times that minimize delay while also being safe, the system also counts with convolutional neural networks programmed with tensorflow which allocate the identified pedestrians into different age and disability classes, and based on that, provides a customized minimum safety crossing time to each pedestrian.

Therefore, the aim of this project is to develop the first steps towards a solution for the optimization of traffic flow, by creating a full mapping of the state of traffic, and the design of customised minimum safety crossing times for pedestrians, and hence the reduction of pedestrian accidents.

The objectives that must be achieved are therefore:

- An object detection capability of detecting more than 95% of pedestrians and cars.

- A convolutional neural network that classifies pedestrians into different age groups with an accuracy of more than 90%.
- A convolutional neural network that classifies pedestrians based on disability with an accuracy of more than 95%.
- A system quick enough to operate at a video rate of 25 frames per second (can detect and classify all the subjects in an image within 40 milliseconds).

## 2. Literature Review

### 2.1. *Brief Background*

The invention of the traffic light is attributed to the British railway engineer John Peake Knight, who invented them in 1868, inspired by the design of railway signaling system [7]. It served as a method for controlling the increasing horse-drawn traffic and its effects on pedestrians and was composed of three semaphore arms with green and red gas lamps for nighttime operation [7]. It was operated manually by a police officer with a lever [7]. In 1912 Lester Wire created the electrically powered automatic red-green timed traffic light [7]. The invention propagated quickly around the United States and Europe [7].

### 2.2. *Current Vehicle Traffic Control Methods*

Fast forward to today, in the UK, the most common method of traffic control is still extremely similar to that invented in 1912. This method is based on traffic lights which operate on fixed time phase plans, known as the webster method [2]. These fixed time intervals that occur before a change in the traffic light output are generally determined by considering the historical traffic patterns at a given junction. This method takes no consideration on the real-time state of traffic in its output decision (whether to change phase or not) as it simply works by toggling its output at a pre-set time interval, therefore proving highly inefficient, as it ignores the real time traffic flow demands which causes unnecessary delays.

The second most common method uses inductive loops underneath the roads as a sensor to detect cars passing by [2]. This method takes in the number of cars passing through this sensor as an input to determine the phase decision of the traffic light (red or green light), so it does partially take into consideration the changing conditions of traffic in determining the output of the traffic light [8]. However, this approach still has many flaws.

Firstly, it has severe traffic mapping limitations. The concept inherently is unable to map pedestrians which provides it with a limited ability to fully map the state of traffic and therefore can only be used for vehicle junctions. Within vehicle mapping it can also struggle with certain traffic scenarios [8]. For example, in the case of a traffic jam, in the moment the car queue reaches the location of the sensor, cars will no longer be passing over it, causing the system to map the state of traffic in that lane only up to the point where the sensor is placed. This can prevent the system from making the right decision as it doesn't have all the information on traffic, which makes it common practice for traffic operators to be forced to manually overturn phase decisions made by the system to avoid congestions [8].

Secondly, most of these systems still rely on timed signal phase plans that take over once the sensor has actioned the output change into a green light [8]. In other words, this system only marginally considers the changing state of traffic in its output phase.

Lastly, inductive loops must be placed underneath the roads, making maintenance severely disruptive

and costly, even more so considering their short effective lifespan of 3-7 years [8].

### **2.3. *Current Pedestrian Traffic Control Methods***

Pedestrian crossing control in the United Kingdom is mostly done with two main methods [9]. The first one, like in vehicle junctions, is the webster method with pre-timed phase plans, which has the forementioned limitations. The operating process, as explained before is that the output light is toggled after a pre-set time passes, and this process is repeated continually. Therefore, the main limitation is that no consideration for the real time state of traffic is given, which can lead to pedestrians waiting at a red light when no vehicles are going through the junction, or vice versa, causing unnecessary delays.

The second method is known as the pedestrian button method, whereby pedestrians press a button located at the junction to indicate their will to cross [9]. This method has the advantage over the previous one that, at least in theory, it has some consideration over the traffic needs at the junction, as it can base its phase decisions on the known presence of pedestrians.

However, this method has some problems. Firstly, it has been demonstrated that this button has no effect on many of the pedestrian crossings during daytime, and the webster method is used to regulate traffic during this period [9]. Only at nighttime in the UK (from 0000 to 0700) and in standalone pedestrian crossings unconnected to junctions, pressing the button will guarantee to reduce the amount of time taken to show a red light for incoming traffic [9]. Therefore, this method simply relegates most of the pedestrian crossings to webster method designs when outside of this narrow time window.

Secondly, within this operational window, this system struggles to fully map the real time state of traffic. It takes in the will of a pedestrian to cross, but it cannot record the number of pedestrians or vehicles at the junction, which significantly limits its judgement for prioritising the traffic flow of either class in its phase decisions. Therefore, this system also often leads to unnecessary delays.

In conclusion, all currently used traffic control methods are suboptimal, as none of them fully consider the real time traffic state in their output decisions, and this leads to severe delays at junctions that could otherwise be avoided.

### **2.4. *Machine Learning for Traffic Control***

There is a new and exciting concept in its experimental phase which involves using deep reinforcement learning (DRL) along with machine vision to control traffic, by using cameras to detect vehicles in junctions and training a machine learning agent to make phase decisions to optimize traffic flow in real time [2,10].

Using cameras to map the traffic state in real time has the advantage of creating a more detailed mapping than any other method. The video footage produced by the camera can then be sampled in real time at a given frequency and be inputted into a convolutional neural network, which will extract the state of the environment, which consists of the traffic state (number of vehicles and pedestrians, queue length, waiting times etc.) and current traffic light phase [10]. The agent then uses this state of the environment as input to make the decision of whether to keep this traffic light phase or change it.

The AI agent is trained and tested in a computer traffic simulation in the optimization of traffic flow [2,10]. In order ensure this agent is trained to achieve optimal decision making in real life, many of these projects use *domain randomization* to make the traffic and weather situations on which the agent



is trained on in the simulation completely unpredictable [2]. This improves the generalizability of the agent as it learns to assess randomized traffic and weather scenarios, making it able to successfully adapt to new and different situations. Domain Randomization has reportedly caused a significant boost in performance in the transition from training the agent in the simulation and bringing it into real life cases [2].

Deep reinforcement learning (DRL) is mostly used for the decision making of these new experimental projects as it doesn't require hand engineering of any parameters, instead it is able to formulate an output based solely in its raw sensory inputs. It uses a reward system to evaluate the policy used for each traffic situation [2,10]. For example, it obtains a positive reward (e.g., +1) for every vehicle going through the junction and a negative reward (e.g., -1) for every vehicle waiting at a junction. Reward can also serve as a way to prioritize certain desired vehicles (such as emergency vehicles or public transport vehicles) in a junction by using machine vision to identify these objects from the images and giving them higher rewards (e.g., +3). Other reports recommend more complex reward functions that take in more parameters to create a more complete representation, (such as waiting time, average speed of vehicles as a ratio of maximum lane speed etc.) and perform a weighted sum of them [10]. This reward will be saved into the memory alongside the state and policy implemented, and the agent will learn from this to obtain more optimal policies that will maximize reward. Periodically this memory will serve to update the network, old memory samples will be deleted as new ones are uploaded. Figure 1 shows a simplified framework of the system.



Figure 1. Framework of Deep Reinforcement Learning For Traffic Control [10]

The evaluation of the policy used by the agent is also important for optimal decision making, as different policies can sometimes lead to equal rewards, however some policies are more appropriate and optimal than others [10]. Generally, the phase plans with the least phase change frequency are more suitable as they are less unpredictable for humans and hence safer. This also shows that reward cannot be the exclusive metric to evaluate the performance of the model, and the policies used should also be assessed [10].

For the real-life implementation of the model, reports indicate the usefulness of using a concept similar to “memory palace” to update the model [10]. The idea is to store reward, environment state, and agent action of similar situations together in a memory location different to the ones used to store other rewards states and phases that occur in other different situations [10]. A block diagram of this structure is shown in part 1 of the appendix.

This is useful as there would be no domain randomization in real life. Using a memory palace structure could enable to substitute the older elements of a given situation by another similar but updated version of that situation. This would solve the main disadvantage of saving every reward, state and phase in the

same memory location, which is that memory would eventually be mostly overwritten by the most common scenarios so it will learn only to manage those cases and will fail to manage less common situations [10]. This learning bias would reduce the generalizability of the agent [10].

## **2.5. *Image Classification Algorithms and Object Detection Model***

Recently, a comparative study on recognition algorithms for facial recognition was performed by Paul Sanmoy and Acharya Sameer for the NMIMS University [11]. This study compared the accuracy of PCA, Viola Jones, support vector machine, convolutional neural network and k-nearest neighbour algorithms for the task of face recognition. The results showed that the algorithm with the highest accuracy rate was the convolutional neural network, with a 3% higher accuracy than any other method [11]. Compared to the other algorithms, convolutional neural networks require less image pre-processing, making it “the most effective learning algorithm for comprehending picture material” [12]. Hence it is the method used by all of the leading professionals in computer vision competitions [12].

Another benefit from neural networks is its versatility, as this method has very positive results for a vast array of image classification problems. It has been used to classify handwritten letters and numbers with great success, yielding accuracy rates of more than 98% while at the same time taking just seconds to train and test [13].

It has also been used for drowsiness detection for drivers in autonomous vehicles [14]. An article published in 2020 describes a project implementing four already existing convolutional neural network models; AlexNet, VGG-FaceNet, FlowImageNet and ResNet to analyse colour video of the behaviour of the drivers to detect drowsiness. Each convolutional neural network model oversaw the extraction of different information from the video footage, to have a more complete mapping of the situation. Different factors like environmental changes, hand gestures, facial expressions or head movements were all extracted by the combination of the models [14]. This project yielded an average accuracy of 85% in the detection of drowsiness [14].

For this reason, the convolutional neural network was the method investigated in this project for image classification.

Another Drowsiness detection programme was designed with the use of the pre-trained single stage object detection model YOLOv5 [6]. This model is trained on the COCO dataset which is “a large-scale object detection, segmentation, and captioning dataset” [15]. Therefore, this model is pre-trained to detect a large number of classes such as “person” or “dog” within images.

This project used LabelImg [16] to train the model to detect custom classes for drowsiness detection- “awake” and “drowsy” and then used a dataset of images of somebody having awake and drowsy facial expressions to train the YOLOv5 model [17].

The output of this model are the same images that are inputted but with boxes bounding the objects that have been classified by the model which also include the name of the class that that bounded object belongs to [17]. Some example outputs from this project are shown in figures 2a and 2b below.



Figure 2a. Subject Detected as Awake [17]    Figure 2b. Subject Detected as Drowsy [17]

### 3. Methodology and theory

#### 3.1. Artificial Neural Network Architecture and Theory

Artificial neural networks can be used for a vast array of different applications which all use different architectures. This project uses the feedforward multi-layer network architecture, as it is the most used for classification problems, and it enables the use of backpropagation as a learning algorithm [18]. This architecture and context uses *supervised learning*, which is when the desired classification result of each datapoint is known, and the model learns by trying to match its classification outputs to the known correct ones [18]. Figure 3 below shows an example of a feedforward multi-layer architecture.

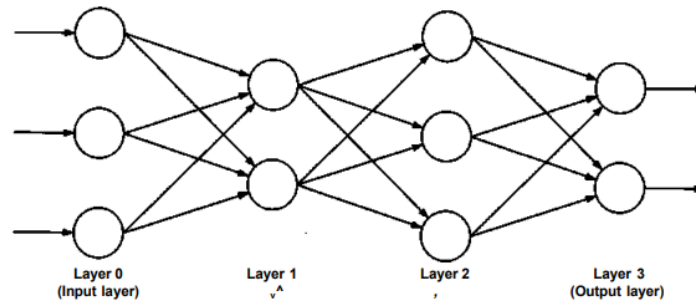


Figure 3. Example of a 3-2-3-2 Feedforward Network [18]

This network architecture is characterized by the nodes forming *layers* where *links* forward-connect the nodes from one layer to all the nodes in next one, which makes the layers *fully connected*. Each node can be thought of as a cell that takes up a value. This value is determined by a function taking in the weighted sum of all the values of the preceding nodes linking up to it [18].

In the input layer of this network, the nodes contain the raw information of the inputted object. In the case of image classification, each node could take up the normalised grayscale brightness of a different pixel in the image [19].

In the output layer, each node represents one of the classes that the neural network is trying to allocate each image into. For each image that is inputted, the node in the output layer that, after all the weighted sums are performed, takes up the highest value, or in other words, has the highest activation, is the class that the network allocates the image into [19].

The layers between the input and output layers are the hidden layers. Each node in each hidden layer can be thought of as representing a distinguishing pattern or a property that the objects in each class

take up. In the case of an image, this could be a group of pixels lighting up, which form a reoccurring outline [19]. Ideally, the weighted sum of the values of the preceding layer into the incoming nodes activate the correct nodes that represent the properties that its class contains. *Biases* are also used, which are values added to the weighted sums to determine a minimum threshold for node activation. This process is repeated for all the hidden layers, where nodes from higher layers ideally “abstract higher level features from preceding layers” [18].

For this process to occur successfully, the weighted sums must map the inputs into its correct features or properties throughout the whole process. To determine the optimal weights and biases for the correct mappings to occur throughout the layers, the backpropagation learning algorithm is used, which is based on gradient descent [18]. Generally, the initial weights and biases inside the network are initially set randomly. As the network trains, the error of the network is recorded after each iteration (each time the model goes through all of the training dataset) and the weights and biases are altered with a gradient descent method that makes the new weights and biases go in the direction of a negative error gradient in loss functions, thus, reducing the error rate after each iteration [18]. Similar methods such as least mean squared error can be used, but backpropagation is used in this project as it is more useful in mapping how errors in higher layers track back into the weights and biases of lower inner layers, hence yielding better results [18].

Activation functions are used to map the weighted sum obtained into the desired activation at a node [19]. They take as an input the weighted sum of all the values of the nodes in the preceding layer and the bias, and its output is the value that the successive nodes take on. Generally, one activation function is selected to govern all the nodes in each layer of the network [19].

For the hidden layers of the networks implemented in this project, the rectified linear unit (ReLU) function was used due to its simplicity, and due to its great results in classification problems [19]. This is one of the most common activation functions in convolutional neural networks. Mathematically it is defined as [20]:

$$f(x) = \begin{cases} x, & \text{if } x > 0 \\ 0, & \text{otherwise} \end{cases}$$

For the final output layer, the SoftMax function was used. This function takes in a vector of K values and produces a probability distribution with those values, with the larger values mapping into higher probabilities [21]. In this context, it takes the activations of the nodes in the output layer prior to the application of the activation function and maps them into a value in the range from 0 to 1, with all the activations summing up to 1 [21]. This can be described mathematically as shown below [21].

$$\theta(z)_i = \frac{e^{z_i}}{\sum_{j=1}^K e^{z_j}}$$

Where  $z$  is the vector containing the values and  $K$  is the number of values in the vector [21]. This function is chosen for the output layer as it enables the output of each node to be read as the probability of that input image belonging to that respective class [21]. Hence the percentage confidence of the model in classifying an input can be extracted from these results.

Summing it all up, the values of the nodes, or the activations for each layer can be described mathematically as [19]:

$$a^{n+1} = \sigma(wa^n + b)$$

Where  $\sigma$  is the chosen activation function for that layer,  $a^n$  is a matrix containing the  $m$  number of activations of the preceeding layer, and  $a^{n+1}$  is a matrix that contains the  $k$  number of activations of the next layer which can be described mathematically as [19]:

$$a^n = \begin{bmatrix} a_1^n \\ a_2^n \\ \vdots \\ a_m^n \end{bmatrix}, \quad a^{n+1} = \begin{bmatrix} a_1^{n+1} \\ a_2^{n+1} \\ \vdots \\ a_k^{n+1} \end{bmatrix}$$

$a_m^n$  is the activation of the  $m^{th}$  node of the preceding layer, and  $a_k^{n+1}$  is the activation of the  $k^{th}$  node of the next layer.

$w$  is the matrix that contains all the weights and  $b$  is the matrix that contains the  $k$  biases of the layer activations which can be described mathematically as [19]:

$$w = \begin{bmatrix} w_{11} & w_{12} & w_{13} & \dots & w_{1m} \\ w_{21} & w_{22} & w_{23} & \dots & w_{2m} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ w_{k1} & w_{k2} & w_{k3} & \dots & w_{km} \end{bmatrix}, \quad b = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_k \end{bmatrix}$$

$w_{km}$  is the weight of the  $m$  input node going into the  $k$  output node.  $b_k$  is the bias of the  $k^{th}$  node of the next layer. These parameters can be visualized in the network diagram shown in figure 4 below.

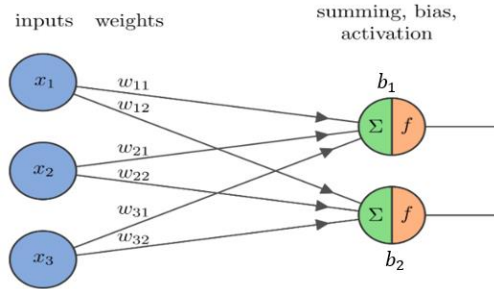


Figure 4. Network Architecture Showing Weights, Biases and Activation Function [22]

### 3.2. Model Overtraining Prevention and Datasets

If a model has too much “capacity” or trains for too many iterations, the model will find weights and biases that will work extremely well only for the classification of that specific dataset and will perform poorly for the classification of other examples outside this dataset. This is called overtraining; it is analogous to human beings learning how to add by memorizing thousands of additions as opposed to learning the concept of summation. This yields poor generalizability- a large discrepancy between training error and errors produced when the model is tested with outside data [18].

The typical curves for error in training and testing datasets as the capacity of the model in the training is increased, is shown in figure 5 below. Figure 5 shows that the optimal capacity is found when the error rate for the test set is generally the lowest. To prevent overtraining in this project, both training and testing accuracies are evaluated at the same time and training is stopped when the test accuracy stops improving. This is termed *early stopping*.

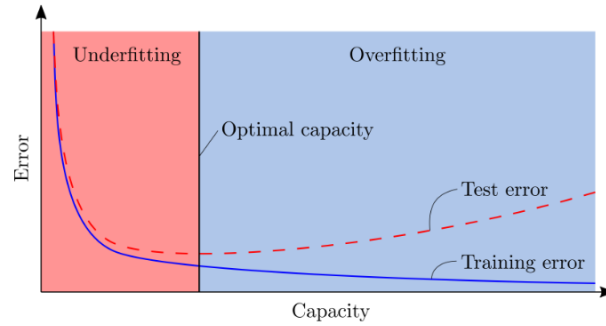


Figure 5. Sketch of Testing and Training Errors with Respect to Capacity [23]

Obtaining a complete dataset to train the network is also essential to achieving high testing accuracies and good generalizability. Having a largely sized training dataset with a high diversity in images improves generalizability as it represents each class with a wide variety of examples for the neural network to train on. Furthermore, for optimal results, class sizes should be balanced. If a certain class is over-represented in the training phase, the neural network can establish a bias towards that class, preventing it from accurately classifying its inputs.

### 3.3. *Convolutional neural Networks*

Convolutional neural networks are expanded versions of artificial neural networks which employ image pre-processing tools for computer vision. Figure 6 below shows the block diagram of a convolutional neural network (CNN).

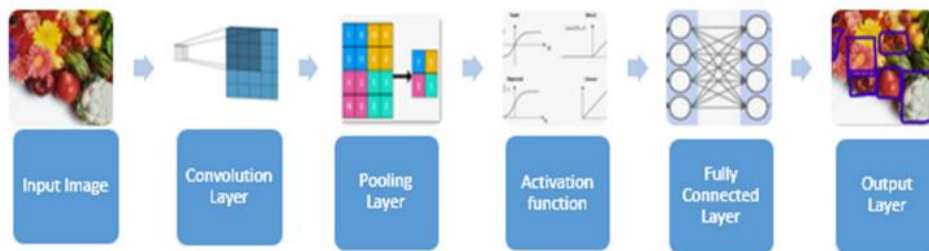


Figure 6. Block Diagram of a Convolutional Neural Network [12]

#### 3.3.1. *Convolution Layer*

This project uses convolution layers. Convolution is a linear filtering method used to extract information from the inputted images.

A grayscale image can be thought of a matrix in which each term represents the brightness of a different pixel. In convolution, there is a filter matrix, which must be smaller in size than the image matrix. This filter matrix slides over the image matrix, starting from the top left corner and multiplying its values with its overlapping values, and performing a sum of all the multiplications before sliding one pixel to the right. This process is repeated until the filter has slid over the whole image matrix, and the output of each of these operations is then stored in a new matrix [12]. This process can be visualized in the example shown in figure 7a and 7b below.

*No padding* was used to reduce the computation requirements of the model, which allowed the reducing of the size of the images after convolution (see part 2 of the appendix).

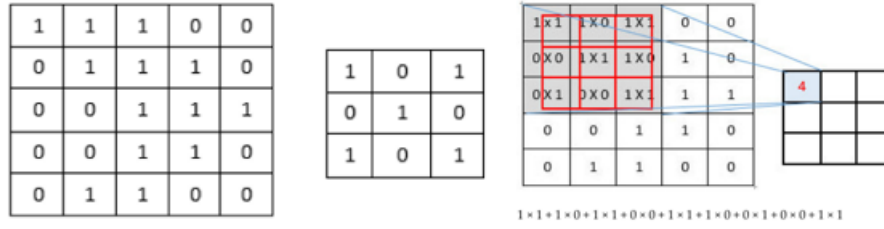


Figure 7a. Image and Filter Matrices [12] Figure 7b. First Step of Convolution [12]

Figure 8 shows a visualization of how convolution extracts information from an image that makes it easier for a neural network to extract patterns and classify correctly.



Figure 8. Original Image (far left), Image after Convolution with Edge Detection Filters are Applied (right) [23]. (See Part 3 of the Appendix for Convolution Filters)

### 3.3.2. Pooling Layer

The pooling layer is used in this project for the main task of shrinking the sizes of the images before inputting them into the network, so that the processing requirements of the neural network is reduced [11]. Having the model have to iterate thousands of times through more than 200,000 input pixels would be too computationally demanding. There are two main forms of pooling: maximum pooling and average pooling. Both were considered.

In maximum pooling a kernel of size  $m \times m$  is selected. The inputted image array is then grouped into  $m \times m$  regions and the maximum value from each region is selected to represent the group in the output array, which has the same size as the pooling kernel [12]. Average pooling is similar, but it extracts the mean of all the values in the region to represent the region in the output array [12]. This can be visualised in figure 9.

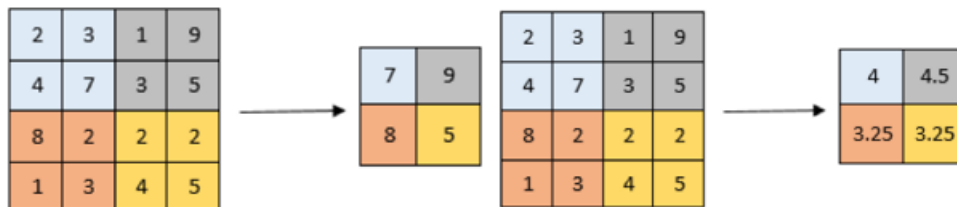


Figure 9. Examples of Maximum Pooling (Left) and Average Pooling (Right) for a 2x2 Kernel Size [12]

Pooling also helps in the additional extraction of features in the images and helps to prevent over-fitting [12].

### 3.3.3. Hyperparameter Tuning



Hyperparameter Tuning refers to the process of selecting the optimal hyperparameters for the network. These parameters include number of hidden layers and nodes, number of convolution filters, kernel sizes etc. For this, a combination of manual tuning and grid search tuning was used. Grid search is the method by which combinations of different hyperparameters are looped together and tested in the network, and the combination that yields the highest accuracy is selected [25]. This was used to select the number of hidden layers and hidden units in each network. Manual tuning is the process whereby parameters are selected by hand. This method was used in combination with trial and error to select all other parameters.

### 3.4. *Evaluation of Networks*

The performance of an artificial neural networks can be assessed in three different ways [18]: The first one is how accurately the neural network classifies the data on which it is trained, or in other words, the accuracy rate of the neural network in the training dataset.

Secondly, how accurately the neural network classifies unseen data or what is the accuracy of the neural network in the training set. This is a measure of how well it extrapolates the patterns detected in the training set into other data, the most important result in network evaluation.

Finally, how many resources does it take up. In the case of a neural network this is a measure of how computationally efficient it is, especially in its training.

### 3.5. *Object Detection Models*

These are convolutional neural network-based models which combine object classification and object localisation to be able to classify more than one object within an image. There are 2 main architecture types for object detection. This project focuses on single-stage detectors, but multi-stage detector implementation can be seen in part 4 of the appendix.

Single-stage detectors are trained in a single stage. This method consists of feature extractors which extract different regions which can potentially contain objects for classification (feature maps), which are bounded, extracted, and decoded for classification [24]. If more than one box is pointing at the same object, the box that has the best fit is chosen [24]. The block diagram of the operation of this method can be seen in figure 10 below.

Single-stage architectures are generally much faster than multi-stage networks at the expense of having slightly lower accuracies [24]. For that reason, this project employs the YOLOv5 [6] model single-stage network, which has an average image detection speed of 15 milliseconds, which is almost ten times faster than multi-stage networks and has an average accuracy of 67.9% (10% less than multi-stage networks) [24]. This speed is critical for the real-time mapping of the state of traffic.

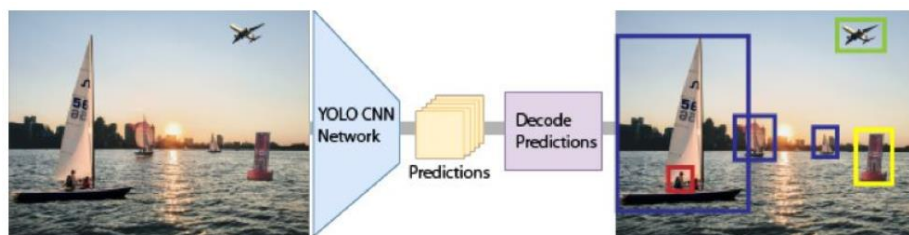


Figure 10. Single-Stage Detector Block Diagram [24]



## 4. Design and Analytical Investigation

Both the object detection functionality for the real-time mapping of the state of traffic and the minimum safety crossing time functionality of the system were implemented as one integrated system. Figure 11 shows the block diagram design for the implementation of this system.

The system as shown in figure 11 below, would take in an image, perform object detection of that image to extract the vehicles and the pedestrians to map the state of traffic. For each image, a variable “timeMain” would be initialized with an initial value of 0. Then it would look through the cropped images of the pedestrians detected and would classify each of them into age and disability classes. A minimum safety crossing time variable is allocated to each of the classes. Once looped through all the classified pedestrians, the one that has the highest allocated minimum safety crossing time allocated to it, dictates what the minimum safety crossing time is for that phase. This process is repeated for each image sampled from the video feed.

This system can be separated into two integrated subsystems- object detection and image classification. Each of them will be looked at individually.

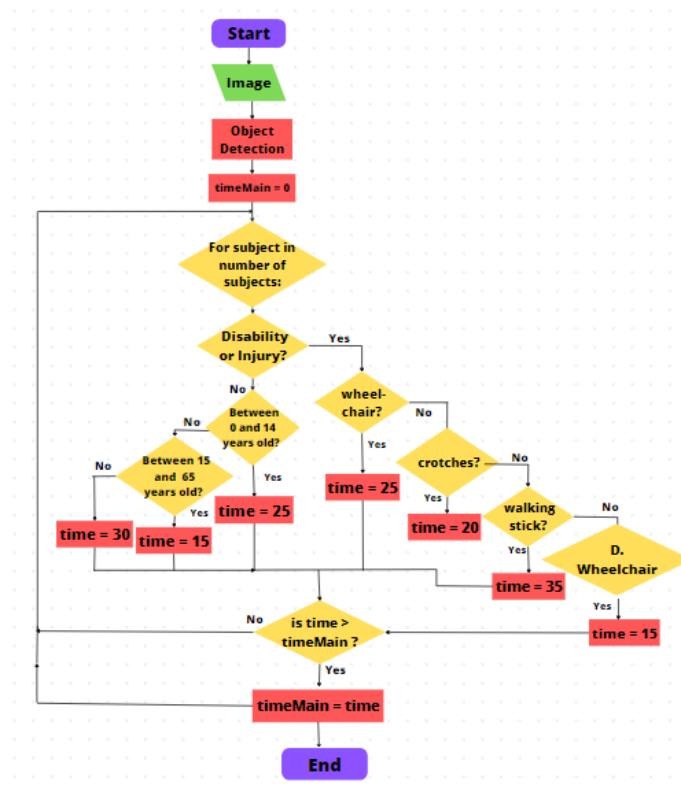


Figure 11. Block Diagram of the System

### 4.1. Image Classification

For image classification, two convolutional neural networks were created, one oversaw age classification while the other was in charge of disability classification of pedestrians. For this, one dataset for each of the convolutional neural networks had to be created, to train and test them.

#### 4.1.1. Dataset Design for Age Classification

For the task of age detection, a pre-made dataset composed of 9778 images of the faces of differently aged people was selected [25]. These images had a size of 200x200 pixels. This dataset was selected as it had a large size and contained clear, close-up images of the faces of people and had a wide diversity of ethnicities and lightings for the subjects in the images for good generalization.

Given that, as shown in figure 14, the system looked to classify in three age categories (0-14, 15-65, 66 onwards), three folders were created for the labelling of the images into their respective designed output classes. The first folder stored the images that were in the class of people aged from 0 to 14 years old. The second one stored images in the class of people aged from 15 to 65, and finally the last folder stored images in the class of people aged 66 onwards. Finally, the dataset had to be edited to erase irrelevant images, and class sizes were adjusted to obtain a balance between conserving a big dataset and obtaining similar sized classes to obtain good accuracy and generalizability. Within each class, the number of images per age was also adjusted so that there was a relatively equal representation of people of different ages in each age group to also prevent biases within classes.

The resulting dataset is composed of 9448 images, in which 3466 images belong to the 0-14 age class, 5024 belong to the 15-65 age class and only 994 images belong to the 65 onwards age class. The third class is clearly underrepresented but the dataset didn't have enough images from that age group. This is an intrinsic problem to this task as it is harder to find images of faces of people in these age groups. Further readjusting the other two classes to balance this would result in a dataset size too small for this purpose.

#### **4.1.2.      *Dataset Design for Disability Classification***

For the task of disability classification, a pre-made dataset composed of more than 10,000 images of people with different mobility impairments was selected [26]. These images had a size of 960x540 pixels. It was extremely hard to find a dataset for this application, and the one selected was created for a very different context of “deep 3-D perception of people with mobility aids” for hospitals [26]. Therefore, a lot of effort was required to manually adapt this dataset for its application in the context of this investigation. However, this dataset was selected as it was the largest dataset that could be found that contained high-resolution images.

For this process, five different classes were created to classify injury or disability. These classes were “wheelchair”, “crotches”, “driven wheelchair”, “walking stick” and “none”. All the relevant images were manually extracted from the pre-made dataset and allocated into its corresponding labelling folder. Finally, the folder sizes were adjusted to obtain a better balance between dataset size and obtaining similarly sized classes to prevent biases.

The resulting dataset is composed of 5203 images, of which 694 belong to the “none” class, 793 belong to the “crotches” class, 1249 belong to the “walking stick” class, 1551 belong to the “wheelchair” class and 916 belong to the driven wheelchair class. This also shows that the dataset is not balanced but the further reduction of the dataset size would again reduce the accuracy of the network.

#### **4.1.3.      *Image Extraction***

For the extraction of the images of the designed datasets, a function was created, shown in part 5 of the appendix. This function loads images from a provided directory. This function assumes that each of the folders inside this provided directory contain the images for one class. The images from all the class

folders are loaded in grayscale and can be resized if needed. The function returns an array containing the image data of all the loaded images, and another array containing the label of each of the images, so that the network can train by finding patterns that better map its prediction for each image label into the real label.

This process was too computationally heavy for the loading the dataset for the disability classification because of its large image sizes. This is why the resizing functionality was implemented, so that the images from this dataset were first shrunk to a smaller pixel size before saving them into the array and memory. The images in this case were shrunk to a smaller size of 622x350 but keeping the aspect ratio of 16:9 as to not distort them. Furthermore, a dynamic memory allocation was used to save the images into the memory in batches to prevent the memory from crashing. The code for this can be seen in part 6 of the appendix.

Finally, once the image data was loaded into the programme, it had to be normalised to make it compatible with the networks. To do this, the array containing all the image data was divided by 255, which is the brightness range for pixels, as show in part 7 of the appendix.

#### **4.1.4. *Overtraining Prevention and Hyperparameter Tuning***

To avoid overtraining, both datasets were split into three, (which is known as the train-test split) as shown in part 8 of the appendix. This is so that data could be divided into training, validation and testing data. The test-train split was performed to allocate 50% of the images of each dataset to their respective training sets, 40% to the test sets and 10% to the validation sets respectively. The training data was used to train the convolutional neural network and adjust its weights and biases. The validation data was the data that the network was evaluated on to prevent overfitting and adjust hyperparameters, and therefore, must also be seen by the dataset at each iteration. Finally, the test data served as the final proof of functionality. It evaluated the network accuracy for a completely unseen dataset. This is done to improve generalizability and overall accuracy.

For hyperparameter tuning, a grid search method was used to select the number of hidden layers and units in both network architectures (hidden units being the number of nodes in each layer). 1,2,4 and 5 hidden layers, with each of them containing 64, 128,256 and 512 hidden units, were considered. The code for this is shown in part 9 of the appendix.

Other parameters such as the number of convolutional filters or the size of said filters were selected manually through a process of trial and error to identify the parameters that yield the best balance between accuracy and training duration. The final network obtained from these methods are found below.

#### **4.1.5. *Network Architecture for Disability Classification***

For the disability classification network, two convolutional layers were manually selected. Different numbers of convolution filters and filter kernel sizes for these filters were tested, and the output accuracy of the model and the training time under each variation were recorded and compared to obtain the most suitable architecture. The resulting accuracy and training time under each variation is shown in table 1 below. It was decided that the architecture with eight filters with a 3x3 kernel size was most optimal as it provided with an optimal balance between high accuracies and training times. This decision

was also influenced by reliability, as the programme would often crash when training the model with more than eight convolution filters in each layer as it was too computationally expensive.

<b>Number of Filters</b>	6	8	10	12
<b>Test Accuracy (%)</b>	97.3	98.5	98.9	99.1
<b>Training Time (s)</b>	521	617	678	746

<b>Size of Kernels</b>	3x3	4x4	5x5	6x6
<b>Test Accuracy (%)</b>	98.5	97.8	96.9	96.1
<b>Training Time (s)</b>	617	583	555	528

Table 1. Investigation on Number and Sizes of Convolution Filters for Disability Classification

Each convolutional filter was followed by its subsequent pooling layer, which reduced the size of the images into a 2x2 output array. This small size was also chosen to reduce the memory requirements. Maximum pooling was used because it yielded slightly larger accuracies.

After that, a flattening layer was implemented to flatten the 3D output of the convolutional layers into a 1D array, so that the data could be processed by the neural network.

The final neural network architecture selected, contained a single hidden layer with 64 hidden units. For this hidden layer, the activation function selected was the ReLU, as mentioned in the methodology section. Early stopping was used to prevent overtraining. This network stopped the training when the results in the validation set had not improved for two successive iterations and returned to the best performing weights and biases. This is a relatively low patience which was designed to limit the memory constraints of the system and reduce training time. Furthermore, it was observed that increasing the patience only marginally increased accuracy.

Given that this neural network has the task of classifying the image into five classes, the output layer contained five nodes. The SoftMax function was used for this final layer as mentioned in the methodology section.

The code for the final, full network architecture of the disability classification convolutional neural network can be seen in part 10 of the appendix. The block diagram can be visualized in figure 12 below.

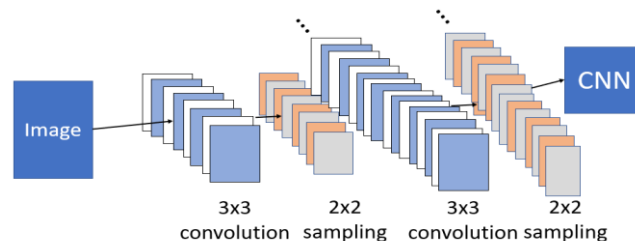


Figure 12. Block Diagram of CNN Architecture for Disability Classification

The neural network in charge of the classification, had the final architecture design as shown in figure 13 below.

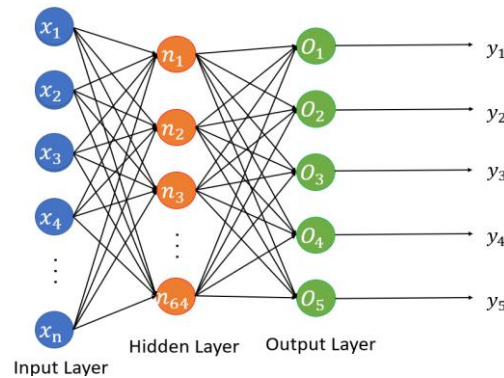


Figure 13. Neural Network Architecture for Disability Classification

#### 4.1.6. Network Architecture for Age Classification

For the age classification network, two convolutional layers were manually selected to each have twelve convolutional filters, with a filter kernel sized 3x3. In this case a larger number of 12 filters was chosen because the smaller size of the dataset for the age classification was not as restrictive on the computation capabilities of the network. Larger filters were tested which reduced the computational cost but at the expense of lower accuracies. The results of testing the number and sizes of filters are shown in table 2.

Number of Filters	10	12	14	16
Test Accuracy (%)	86.0	86.7	87.2	87.6
Training Time (s)	332	369	398	423

Size of Filters	3x3	4x4	5x5	6x6
Test Accuracy (%)	86.7	86.2	85.7	85.1
Training Time (s)	369	347	323	299

Table 2. Investigation on Number and Sizes of Convolution Filters for Age Classification

Each convolutional filter was followed by its subsequent pooling layer, which was designed to reduce the size of the images into a 2x2 output array. This small size was also chosen to reduce the memory requirements. In this case, the choice of maximum or average pooling yielded negligible differences.

After that, a flattening layer was implemented to flatten the 3D output of the convolutional layers into a 1D array, so that the data can be processed by the neural network.

The final neural network architecture selected, contained two hidden layers with 128 hidden units on each one. For the hidden layers, the activation function selected was the ReLU. The output layer consisted of 3 nodes, one for each output class. Again, the SoftMax activation function was selected for the final output layer. Early stopping was again used to prevent over-training. This time the model was programmed to stop the training process after 5 successive iterations without improvement in the

validation set. After that, the weights and biases would return to the values that yielded the greatest accuracy in the validation set. A patience of 5 epochs generally guarantees that the network has found the optimal capacity.

The code for the final, full network architecture of the age classification convolutional neural network can be seen in part 10 of the appendix. The block diagram for this convolutional neural network is very similar to that shown in figure 12, except for the number of convolutional filters.

The neural network in charge of the classification, had the final architecture design as shown in figure 14 below. This model has three nodes in the output layer because this model is designed to classify each image into one of the three designed classes mentioned before.

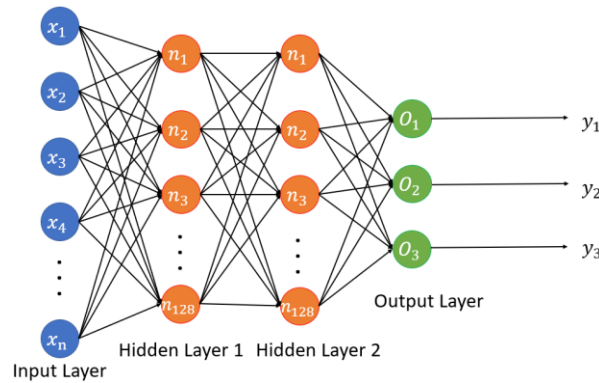


Figure 14. Neural Network Architecture for Age Classification

## 4.2. Object Detection

The YOLOv5 pre-made model for object detection has an architecture of more than 200 layers and 7 million parameters. It is trained on the COCO dataset to detect some in-built classes such as people, dogs, cars or boats. With the detection of vehicles and people, the system can map the real-time state of traffic. The system detects different objects in an image and returns the number of objects of each class detected. The *render* command outputs the original image with the objects detected bounded in boxes and labelled. An example of this functionality is shown in figure 15 below.



Figure 15. Input Image (Left) and Output Image with Objects Bounded by Boxes and Labelled (Right)

The other purpose of the YOLOv5 model for this project was to extract each pedestrian from the image to input it into the age and disability classification models to obtain a minimum safety crossing time.

To do this, the command `xyxy` of the YOLOv5 model was used. This command returns a tensor that stores the coordinates of each of the bounding box edges of each of the detected objects. Therefore, this contains the information of where each detected object is located within the image.

A function named `image_cropper2` was designed that took an image as an input. It then used the YOLOv5 model to process this image. The `xyxy` command was then used to extract the coordinates of the bounding boxes of the detected objects. Finally, the function would loop through all the objects, the arrays containing the image information contained in the bounding box coordinates would be extracted and appended into a previously empty array. This way the information of the images “cropped” by the bounding boxes would be stored, ready to be inputted into the convolutional neural networks for classification (see part 11 in the appendix).

Another function named `image_identifyer` was designed to offer a better visual method of representing the functioning of `image_cropper2`. It functioned very similarly, but instead of returning an array containing the individual arrays of the cropped objects, it plotted the original image and all the cropped images of the detected objects that are stored in the output array in `image_cropper2`. This function was created so that the output of this system could be better interpreted by humans. The code for this can be found in part 12 of the appendix.

### 4.3. *Object Detection and Image Classification Integration*

The code for the integration of the computer vision and object detection models can be seen in part 13 of the appendix. It takes in the output of the function `image_cropper2`, and inputs it into the disability classification convolutional neural network for it to evaluate the disability class that each detected pedestrian belongs to. It does this by resizing the arrays of the cropped objects, outputted by `image_cropper2`, into a suitable format for inputting into convolutional neural network. Furthermore, the image sizes are edited to equal the image size of the dataset on which the classifier was trained on (622x350), as it is the only image size that the model is trained to receive. It then normalizes the image arrays, and it is fed into the disability classifier.

The code prints the activation of the output nodes of the network, or in other words, the probability of each of the pedestrians detected of being a part of each class. It then also prints the class the network has predicted each image into, and the percentage of confidence of the model in its decision. Figure 16 shows a block diagram summarizing the process.

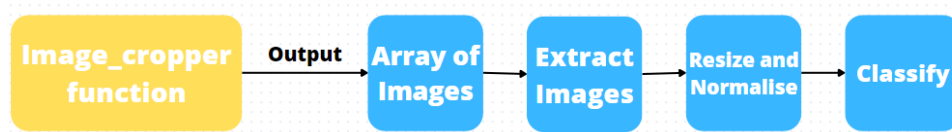


Figure 16. Block Diagram of System Integration (See Part 14 of Appendix for Detailed Block Diagram for Code Design)

Unfortunately, this could only be done for disability classification, as this system took advantage of the pre-trained YOLOv5 model ability of detecting and labelling people. The model boxes the full body of the people when detecting them. This is positive for the disability classification network because it is trained on full-body images, however, the age detection network is trained on face images, making the cropped images of pedestrians unsuitable for the age classification model.

To overcome this, a custom class of object detection for faces was attempted to be added to the YOLOv5 [6] model with LabelImg [16] as discussed in the literature review. However, the incompatibility of this programme with the computer employed for this project made it impossible for the implementation of this feature into the model. Furthermore, the implementation of this feature would be extremely time consuming as it would require manually bounding hundreds or thousands of face images to train the network. This could have been unsuitable for the timeframe of this project.

## 5. Presentation of experimental results

The final design for the age classification convolutional neural network was evaluated on four key metrics- training accuracy, testing accuracy and time taken to train, and time taken to predict test images.

### 5.1. *Age Classification Final Network Results- Accuracy*

The training accuracy of the model was of 92.5%, the validation accuracy was of 86.9% and finally the testing accuracy was of 87.5%. The initial results are relatively high, however, testing accuracies are below what was aimed for. To fully evaluate the performance of this model further results must be extracted. Figure 17 shows a confusion matrix of this model where all images in the dataset are used.

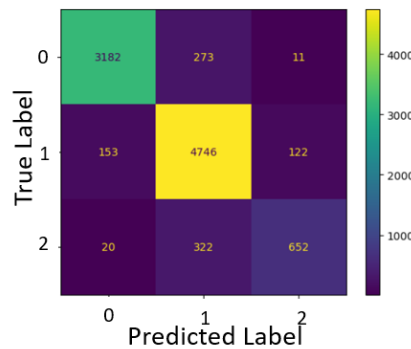


Figure 17. Confusion Matrix for Age Classification Network

Confusion matrices are a way of visualizing the performance of the classification model. For this, the model predicts the class or label of each of the images of the dataset. Those results are visualized in the x-axis, where each cell represents a class and the number inside it represents the number of images classified into that class. The y-axis shows the number of images which really belong to each class. This shows the discrepancies between the true results and the results predicted by the network. Therefore, the cells going diagonal from left to right in the matrix, represent the number of instances where the model has predicted the class correctly as, in those cells, the predicted labels and true labels align. In this diagram the label 0 represents the class of people aged from 0 to 14, the label 1 represents the class of ages 15-65 and the label 2 represents the class of ages 66 onwards.

From this it can be seen that 96.5% of the times where the network wrongly predicts an image label, it is because it is predicting it to be in an adjacent class. It can also be seen that the biggest point of error is the classification of people in the 66 onwards age category into the 35-65 age category (accounts for 32% of the error in that class), which may be caused by the bias of the model created by the uneven



data sizes of the classes. Given that the 35-65 class is more than five times bigger than the 66 onwards class, the model may assume that there is a higher probability of images pertaining to the former class.

To further understand the shortcomings of the model, the dataset was spilt into narrower age groups and fed back into the network to evaluate the accuracy of the model with pictures of each age group. For this, the dataset was split in groups of ages of 0-10, 11-14, 15-19, 20-35, 36-50, 51-60, 61-65, 66-70 and 71 onwards. Each group was allocated its corresponding true label and classification accuracy was recorded for each age group. This is shown in the table 3 below. The code for this can be seen in part 15 of the appendix.

<b>Ages</b>	0-10	11-14	15-19	20-35	36-50	51-60	61-64	65-70	71-80	81...
<b>Accuracy (%)</b>	97.8	84.5	65.2	86.3	92.1	85.6	74.9	35.9	57.1	70.9

Table 3. Table of Accuracies of each Age Group

The results shown in this table can be found in bar chart form in figure 18. The lowest accuracies occur when classifying the groups pertaining at the edges or boundaries of each class, while the images of the age groups which are far from the boundaries are generally classified with greater accuracy.

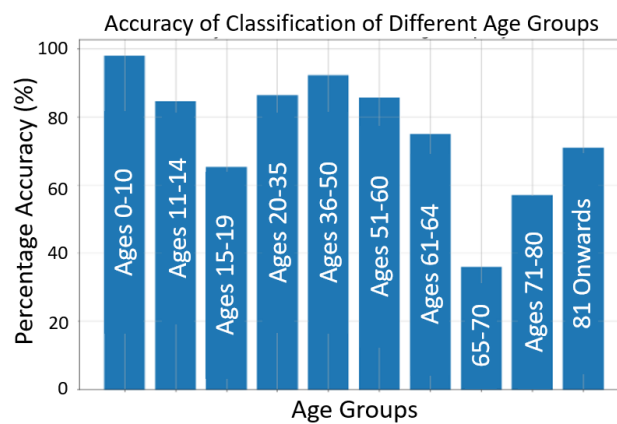


Figure 18. Accuracy of the Age Classification Network for each Age Group

Given that this is used for the allocation of a minimum safety crossing time, the accuracy of the model in classifying pedestrians of age groups which are far from the boundaries take up a greater priority. This is because safety crossing times for each class are designed to especially protect people far from the boundaries of each class. For example, it is much more important that a 90-year-old person is correctly classified than a 66-year-old, even though they belong to the same class. Generally, the 90-year-old person will require that extra time to cross, while the 66-year-old will generally be able to cross without struggles if allocated to its adjacent corresponding class. For this it is important that safety crossing times are well designed so that pedestrians aged at the boundaries of classes can easily cross even if allocated to an adjacent class, as the task of age classification inherently becomes harder the closer to the boundary one gets.

The accuracy of this network is below of what was aimed for but serves as a proof of concept. From figure 18 and table 4, it can be seen that the age groups that have the highest number of images (groups 0-10,20-35 and 36-50) have the greatest accuracies. A greater training dataset and more balanced classes could significantly improve this imbalance. Furthermore, higher quality images in the dataset could also

improve accuracy. The 200x200 images in the dataset might have had too low of a resolution for an optimal extraction of features by the convolutional neural network.

## 5.2. Age Classification Final Network Results- Time Taken

The training time for the network was recorded to be of 6 minutes and 9 seconds. This time was comprised of 13 epochs or iterations of average time of 28.4 seconds, with a range of times between 28 and 31 seconds per epoch, which is a relatively short time.

For the evaluation of the testing time, the time taken to obtain the results in figure 18 were recorded. This also evaluated whether the network would take different times in processing images belonging to different classes or age groups. Table 4 shows the time taken for all the images of each of the age groups shown in table 3 and figure 18 to be classified.

Ages & images	0-10 3085 images	11-14 381 images	15-19 592 images	20-35 1931 images	36-50 1213 images	51-60 970 images	61-65 315 images	66-70 256 images	71-80 394 images	81... 344 images
Time Taken (ms)	4074	492	798	2623	1596	1302	450	368	572	462

Table 4. Time Taken for all Images in each Age Group to be Classified

Part 16 of the appendix shows these results plotted in a bar chart. Given that each group has such a diverse number of images, figure 19 shows a bar chart with the normalised results, which were obtained by dividing the time for each group by the number of images, to obtain an average time per image for each group.

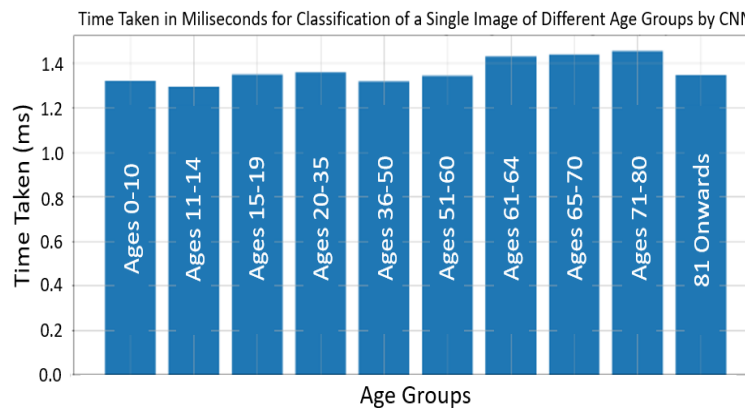


Figure 19. Bar Chart Showing Average Time for Single Image Classification for each Group

From these results it can be seen that the network processes and classifies each image relatively fast, with an average time taken of 1.34 milliseconds, which would allow this system to classify images at an average frequency of 741 images per second which is more than sufficient for this application, as the requirement is of the whole system to operate at a rate of 25 frames per second. It can also be seen that the difference between the time taken to process images from different age groups are negligible, with a range of average time between all the groups being only of 0.158 milliseconds.

## 5.3. Disability Classification Final Network Results- Accuracy

The training accuracy of the model was of 100%, the validation accuracy was of 98.2% and finally the testing accuracy was of 98.4%. The confusion matrix for this model is shown in figure 20 below. This contrasts with the accuracies obtained in the age classification network. The difference may be caused by having a more balanced dataset and having higher quality pictures in the dataset.

Labels 0, 1, 2, 3, 4 in figure 20 represent the classes “none”, “crotches”, “wheelchair”, “walking stick” and “driven wheelchair” respectively. It can therefore be seen from this confusion matrix that the greatest error point, although relatively small, is in classifying images of people being driven in wheelchairs as people self-driving wheelchairs. More images for that class could be inserted for future further training to improve this.

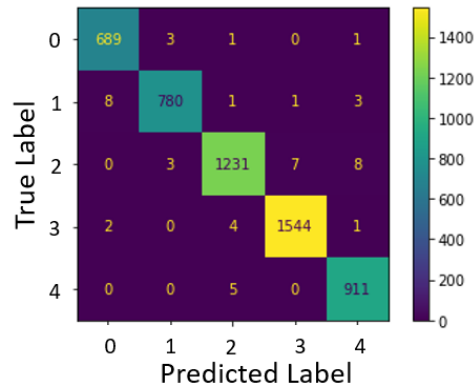


Figure 20. Confusion Matrix for Disability Network

To further test this model, 200 images of each class were selected randomly and inputted into the network to evaluate the classification accuracy. The results for each class are shown in the table in table 5 below.

Disability	none	crotches	wheelchair	walking stick	Driven wheelchair
Accuracy (%)	95.5	98.5	100	95.6	98.6

Table 5. Average Accuracy of Classification for each Class

These results can be further visualized in the bar chart in figure 21 below.

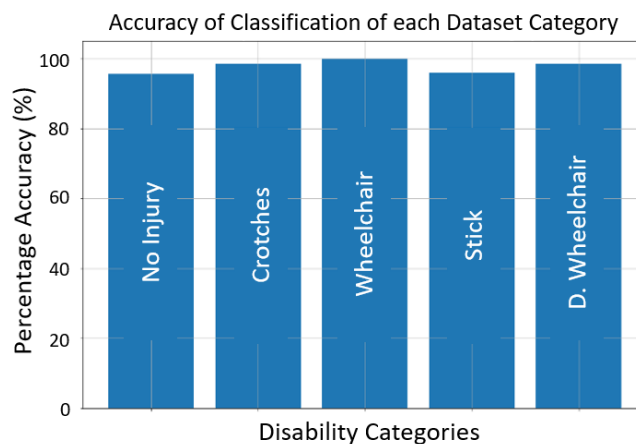


Figure 21. Bar Chart Showing the Average Accuracy of Classification for each Class

It can be seen that the accuracies for all classes are very similar, but the model is especially best at classifying people on wheelchairs while it struggles the most in classifying people with no mobility impairments, which are the most and least represented classes respectively in the training dataset. A more balanced dataset with more images might solve that imbalance.

#### 5.4. *Disability Classification Final Network Results- Time Taken*

The training time for the network was recorded to be of 10 minutes and 17 seconds. This time was comprised of 9 epochs or iterations of average time of 68.6 seconds, with a range of times between 61 and 71 seconds per epoch. This is longer than the previous case, but still considered relatively short.

For the evaluation of the testing time, the time taken to obtain the results in figure 21 were recorded. This also evaluated whether the network would take different times in processing images belonging to different classes or age groups. Table 6 shows the time taken for all the 200 test images of each of the disability groups shown in table 5 and figure 21 to be classified.

Disability	none	crotches	wheelchair	Walking stick	Driven wheelchair
Time taken (ms)	1078	1043	978	1050	1029

Table 6. Time Taken to Classify 200 Images from each Class

Figure 22a below shows a bar chart with the information shown in table 6. The normalized results, to show the average time taken for a single image of each class to be classified are shown in figure 22b.

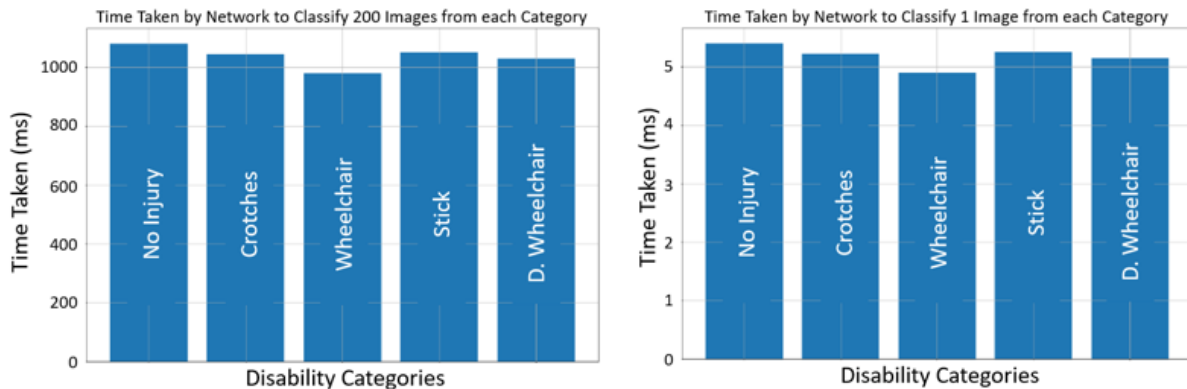


Figure 22a. Bar Chart Showing the Time Taken for 200 Images of each Class to be Labelled (Left)

Figure 22b. Bar Chart Showing the Average Time Taken for 1 Image of Each Class to be Labelled (Right)

From figure 22b it can be seen that the network processes and classifies each image relatively fast, with an average time taken of 5.1 milliseconds, which, even though would be slower than the previous network, it would allow this system to classify images at an average frequency of 196 images per second, which again would suffice the requirements of the system. It can also be seen that the difference between the time taken to process images from different age groups are negligible, with a range of average time between all the groups being only of 0.5 milliseconds, which shows that the model takes the same time in classifying images from different classes.

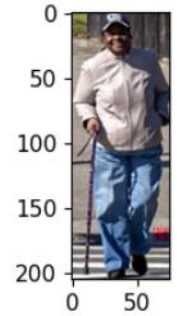
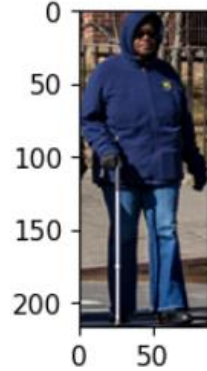
### 5.5. Integration With Object Detection System

To test the integration of the disability classification network with the object detection model, an image portraying pedestrians crossing the road was fed into the image\_cropper2 function and then the output of that was edited correspondingly and inputted into the disability classification model. The code for this can be seen in part 13 of the appendix. The image used for this is shown in figure 23 below.



Figure 23. Image used to Test Integration between Object Detection and Computer Vision

The result of the disability classification of the pedestrians are shown in the table 7 below.

Object Detected	Label	Percentage Confidence	Classification Veredict
	Walking Stick	99.9%	Correct
	Walking Stick	99.99%	Correct

	Crotches	97.6%	Incorrect
	Crotches	98.6%	Incorrect
	Crotches	99.9%	Incorrect
	Crotches	99.5%	Incorrect

Table 7. Classification of the Detected Pedestrians

From table 7 it can be seen that the model for classification of pedestrians for this image has a classification accuracy of 33%. This indicates that the model lacks generalization. For the current project the dataset used was for hospitals. Therefore, all the images were indoors and people were not wearing jackets, gloves and no hands were on pockets, which might be one of the reasons of the shortcoming of the classifier in this image, as the hand outline, for example, could be an essential parameter in the correct classification of the images. Using a more suitable dataset for the context of the projects might significantly improve results. Further work could also be done to prevent the system from having to crop photographs into the size of the images in the training dataset, which heavily distort the test images and could significantly hinder the ability of the model of obtaining correct classifications.

## 6. Discussion and conclusions

This project has investigated the initial steps towards creating a machine learning model to control phase decisions of traffic lights in pedestrian crossings. The proof-of-concept was demonstrated as a potential solution to fully map the state of traffic, and it investigated the functionality of using convolutional neural networks for the classification of pedestrians into age and disability classes for their allocation of a customised minimum safety crossing time. Classification models and subsequent integration with the object detection model yielded positive initial results, which could be further improved with more training.

This project serves as a proof of functionality of the system, which, under more training and with better datasets could be fit for purpose.

### 6.1. *Datasets*

The main shortcoming and learning points of this report lie around the construction of datasets. Many of the issues encountered in this report come as consequences of suboptimal dataset construction. The disability classification network showed much higher testing accuracies than the age classification network partially because its dataset was more balanced, and its images had greater resolutions. However, the disability classification network also showed low accuracy results when tested with the cropped images of pedestrians, perhaps because of a lack of generalizability due to the dataset being small, and because its images portray another context and environment than that of pedestrians.

Limitations such as time and storage constraints restricted the capabilities of this project of constructing optimal datasets. However, this shows the vital importance of training convolutional neural networks with large datasets that are built up of high-resolution images, balanced classes, and diverse images for each class for a good generalizability and high test accuracies. Furthermore, for optimal results, the objective of the dataset should align with the objective of the classifying network. For this, spending extra time and effort to create datasets specific to the project whenever possible, rather than trying to edit pre-made ones created for different contexts, would be beneficial.

### 6.2. *Convolutional Neural Networks*

Neural networks have proven to be incredibly powerful tools for this context due to their ease of implementation, as they are self-improving, and only require a small number of hand-designed parameters. Furthermore, they are fast to train and yield high accuracies. A found shortcoming of convolutional neural networks for classification of images is that it can only process same-size images. This limits its usability, as it requires inputs to be resized into the desired format, which generally involves distorting the image and leads to an incorrect classification of the image. This also limited the accuracy of the integrated model as the size of the cropped images of pedestrians significantly differed from the ones used to train the disability classifier.

### 6.3. *Future Work*

The successful implementation of the LabelImg [16] software to create custom labels for object classification would enable the creation of face detection capabilities for this model, which would allow the integration of the object detection model with the age classification module. This would enable

detected pedestrians to be classified based on age and disability as intended, to provide each one with more suitable minimum waiting times.

The use of this real-time traffic mapping method can then be implemented to train a deep reinforcement learning agent to optimize phase decisions and policies to reduce traffic waiting times.

Data safety should also be considered. Firstly, an end-to-end encryption of the database where images taken by the system are stored should be implemented. This data could serve to locate people, which could lead to safety issues (blackmailing, extortion are possibilities). Furthermore, past images which do not serve to train the system should be instantly deleted once used for that respective phase decision.

## 7. Project Management

A number of things were done to ensure the completion of the project before the deadline. Firstly, a Gantt chart at the start of each term was created, outlining the tasks with their respective deliverables and deadlines. The progress of each activity was tracked with the percentage completion to ensure everything was completed within the deadline.

Figure 24 and show the Gantt timeline for the first and second term respectively.

Furthermore, a risk register was created at the start of the project to identify potential issues that could be encountered throughout the project so that they can be assessed and mitigated. This is shown in table 8 below.

<b>Risk</b>	<b>Possibility</b>	<b>Impact</b>	<b>Mitigation</b>
Datasets harder to find/construct than expected	Fair	Delay in the project	Start research early
Issues with debugging code for CNNs	Fair	Delay in the project	Allocate extra time for coding CNNs
Object Detection models complex to implement	High	-Delay in project -No implementation of fundamental feature	Allocate extra time to research and code models
Errors in code from libraries used	Low	-Delay in practical work due to debugging	-Select reputable libraries -Allocate time to research and test libraries
Sickness	Fair	Delay in the project	Allocate free catch-up time at end of each term

Table 8. Risk assessment of the Project



Task Name	Duration in Weeks	Start Date	End Date	Deliverables	Percentage Completion	Week1	Week2	Week3	Week4	Week5	Week6	Week7	Week8	Week9	Week10	Week11
<b>TERM 1</b>																
Research of Conventional Methods of Traffic Control	1	26/09/2022	01/10/2022	Gained Understanding on Methods, Advantages, Disadvant	100											
Research of I Methods of Traffic Control with AI	2	03/10/2022	16/10/2022	Gained Understanding on Methods, Advantages, Disadvant	100											
Research of Key AI Vocabulary From Literature	1	17/10/2022	22/10/2022	Gained Deeper Understanding of Literature	100											
Research of Neural Network Literature	2	24/10/2022	04/11/2022	Gained Understanding on Optimal CNN Architectures	100											
Risk Assessment and Gantt Chart Creation	1	07/11/2022	12/11/2022	Created Initial Risk Assessment and Gantt Charts	100											
Research for Motivation for the Project	1	14/11/2022	19/11/2022	Found Key Information for Abstract and Introduction	100											
Initial Draft on Introduction and Literature Review	2	28/11/2022	03/12/2022	Created Initial Draft for Introduction and Literature Review	100											
<b>TERM 2</b>																
Research of Practical Methods for Face Detection	1	14/01/2023	18/01/2023	Chose YoloV5 as library for face detection	100											
Search of Dataset for Age Classification	1	15/01/2023	19/01/2023	Selected best dataset out of those found	100											
Adaptation of Dataset for Age Classification to Project	1	21/01/2023	25/01/2023	Removed pictures irrelevant to the task	100											
Creation of CNN to Classify Faces Based on Age	2	28/01/2023	10/02/2023	Completion of optimum network architecture	100											
Search of Dataset for Disability/Injury Classification	1	13/02/2023	17/02/2023	Selected best dataset out of those found	100											
Adaptation of Dataset for Injury Classification to Project	1	20/02/2023	24/02/2023	Removed pictures irrelevant to the task	100											
Creation of CNN to Classify People Based on Mobility	1	27/02/2023	03/03/2023	Completion of optimum network architecture	100											
Research of Real Time Object Classification	1	06/03/2023	10/03/2023	Found method of using OC in video	100											
Application of Real Time Object Classification	2	13/03/2023	24/03/2023	Created function to return arrays of objects detected	100											

Figure 24. Gantt Chart for the First and Second Term Respectively

# Appendix

Appendix available on request from Prof: A. Baldycheva

## References

- [1]] Lomas, N. and Dillet, R. (2020). How four European cities are embracing micromobility to drive out cars. [online] TechCrunch. Available at: [https://techcrunch.com/2020/11/20/how-four-european-cities-are-embracing-micromobility-to-drive-out-cars/?guccounter=1&guce\\_referrer=aHR0cHM6Ly93d3cuZ29vZ2xILmNvbS8&guce\\_referrer\\_sig=AQAAACEcgNuQWwZi1H1WuE36sOI5TmkGrdqAaANnwCnNssYX1u0z4ozRuTeCBPm7424UmlJEUg3i-Hf1mLOS4cR8VlOy8opWUxE98ds1Nf9AKEJYRZ6Jpe8PBa1VCJr56S1re5orvFinSaV-f\\_5Nhf9F\\_o8K7QuIofHsI-fqBTLCHMQ](https://techcrunch.com/2020/11/20/how-four-european-cities-are-embracing-micromobility-to-drive-out-cars/?guccounter=1&guce_referrer=aHR0cHM6Ly93d3cuZ29vZ2xILmNvbS8&guce_referrer_sig=AQAAACEcgNuQWwZi1H1WuE36sOI5TmkGrdqAaANnwCnNssYX1u0z4ozRuTeCBPm7424UmlJEUg3i-Hf1mLOS4cR8VlOy8opWUxE98ds1Nf9AKEJYRZ6Jpe8PBa1VCJr56S1re5orvFinSaV-f_5Nhf9F_o8K7QuIofHsI-fqBTLCHMQ)
- [2] Garg, D., Chli, M. and Vogiatzis, G. (2022). Fully-Autonomous, Vision-based Traffic Signal Control: from Simulation to Reality. [online] publications.aston.ac.uk. Available at: <https://publications.aston.ac.uk/id/eprint/43543/>
- [3] Auto Express (2016, June 2). Traffic jams cost the UK £6.9bn last year. [online] Available at: <https://www.autoexpress.co.uk/car-news/consumer-news/94871/traffic-jams-costs-in-the-uk>
- [4] GovTech. (2023). How much did traffic congestion cost the U.S. last year? [online] Available at: <https://www.govtech.com/question-of-the-day/how-much-did-traffic-congestion-cost-the-u-s-last-year#:~:text=Answer%3A%20More%20than%20%2481%20billion.&text=Traffic%20congestion%20in%20the%20U.S>
- [5] Department for Transport. (2021). Reported Road Casualties Great Britain: Pedestrian factsheet 2021. [online] Available at: [https://www.gov.uk/government/statistics/reported-road-casualties-great-britain-pedestrian-factsheet-2021/reported-road-casualties-great-britain-pedestrian-factsheet-2021#:~:text=In%202021%2C%20361%20pedestrians%20were,11%2C261%20slightly%20injured%20\(adjusted\)](https://www.gov.uk/government/statistics/reported-road-casualties-great-britain-pedestrian-factsheet-2021/reported-road-casualties-great-britain-pedestrian-factsheet-2021#:~:text=In%202021%2C%20361%20pedestrians%20were,11%2C261%20slightly%20injured%20(adjusted))
- [6] PyTorch. (n.d.). [online] Available at: [https://pytorch.org/hub/ultralytics\\_yolov5/](https://pytorch.org/hub/ultralytics_yolov5/)
- [7] Morrison, J. (2008). The Eccentric Engineer. [online] Available at: <https://doi.org/10.1049/etr:20081518>
- [8] FLIR Systems, Inc. (2019, June 19). The Evolution of Intersections: From Inductive Loops to Artificial Intelligence. FLIR Traffic. [online] Available at: <https://www.flir.co.uk/discover/traffic/urban/the-evolution-of-intersections-from-inductive-loops-to-artificial-intelligence/>
- [9] BBC News. (2013, 4 Sep.). Does Pressing the Pedestrian Crossing Button Actually do Anything? [online] Available at: <https://www.bbc.co.uk/news/magazine-23869955>
- [10] Wei, H., Zheng, G., Yao, H. and Li, Z. (2018). IntelliLight: A Reinforcement Learning Approach for Intelligent Traffic Light Control doi: <https://doi.org/10.1145/3219819.3220096>
- [11] Paul, S. and Acharya, S.K. (2020). A Comparative Study on Facial Recognition Algorithms. [online] papers.ssrn.com. Available at: [https://papers.ssrn.com/sol3/papers.cfm?abstract\\_id=3753064](https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3753064)
- [12] Bhatt, D., Patel, C., Talsania, H., Patel, J., Vaghela, R., Pandya, S., Modi, K. and Ghayvat, H. (2021). CNN Variants for Computer Vision: History, Architecture, Application, Challenges and Future Scope. Electronics, 10(20), p.2470. doi: <https://doi.org/10.3390/electronics10202470>
- [13] Nielsen, M.A. (2018). Neural Networks and Deep Learning. [online] Available at: <http://neuralnetworksanddeeplearning.com/chap1.html>

- [14] Dua, M., Shakshi, Singla, R., Raj, S. and Jangra, A. (2020). Deep CNN models-based ensemble approach to driver drowsiness detection. Neural Computing and Applications. doi: <https://doi.org/10.1007/s00521-020-05209-7>
- [15] COCO. (n.d.). - Common Objects in Context. [online] Available at: <https://cocodataset.org/#home>
- [16] GitHub. (2022). LabelImg. [online] Available at: <https://github.com/heartexlabs/labelImg>
- [17] nicknochnack (2021). YOLO Drowsiness Detection Tutorial [online] GitHub. Available at: <https://github.com/nicknochnack/YOLO-Drowsiness-Detection/blob/main/Drowsiness%20Detection%20Tutorial.ipynb>
- [18] Mehrotra, Kishan & Mohan, Chilukuri & Preface, Sanjay. (1997). Elements of Artificial Neural Nets.
- [19] Goodfellow, I., Bengio, Y. and Courville, A. (2016). Deep Learning. [online] Deeplearningbook.org. Available at: <https://www.deeplearningbook.org/>
- [20] Deepchecks. (n.d.). What is Rectified Linear Unit (ReLU). [online] Available at: [https://deepchecks.com/glossary/rectified-linear-unit-relu/#:~:text=ReLU%20formula%20is%20%3A%20f\(x](https://deepchecks.com/glossary/rectified-linear-unit-relu/#:~:text=ReLU%20formula%20is%20%3A%20f(x)
- [21] DeepAI. (2019). Softmax Function. [online] Available at: <https://deepai.org/machine-learning-glossary-and-terms/softmax-layer#:~:text=Softmax%20Formula&text=where%20all%20the%20zi%20values>
- [22] ] MLPs, Industry 4.0 (ENG2006), University of Exeter (2022)
- [23] Computer Vision, Industry 4.0 (ENG2006), University of Exeter (2022)
- [24] Aktaş, Y.Ç. (2022). Object Detection with Convolutional Neural Networks. [online] Medium. Available at: <https://towardsdatascience.com/object-detection-with-convolutional-neural-networks-c9d729eedc18>
- [25] Kumar, S. (2021). 7 Hyperparameter Optimization techniques every Data Scientist should know. [online] Medium. Available at: <https://towardsdatascience.com/7-hyperparameter-optimization-techniques-every-data-scientist-should-know-12cdebe713da>
- [26] www.kaggle.com. (n.d.). facial age. [online] Available at: <https://www.kaggle.com/datasets/frabbisw/facial-age?resource=download>
- [27] mobility-aids.informatik.uni-freiburg.de. (n.d.). MobilityAids - Autonomous Intelligent Systems. [online] Available at: <http://mobility-aids.informatik.uni-freiburg.de/>