

Improving the Safety of Pedestrians at Road Crossings using IoT Sensors

CMT 404 – Computing and IT Management Dissertation



Cardiff University School of Computer Science and Informatics

Author: **Jamie Emery** (1426360)

Supervisor: **Dr Padraig Corcoran**

Moderator: **Professor Omar Rana**

Acknowledgements

I would like to thank my supervisor Dr Padraig Corcoran who's been very supportive during this project. I have very much enjoyed our discussions and I hope to work with him again in the future. I would also like to thank my family; especially my grandmother, Dr Lyn Huckman. She has been a constant source of inspiration and I'm thankful to have been able to follow in her footsteps by completing a masters.

Abstract

Statistically, the largest proportion of pedestrian road accidents and fatalities occur when crossing the road, especially within cities. We propose an edge network model that aims to improve pedestrian safety by demonstrating how IoT camera sensors can utilise object detection software to alert oncoming drivers when pedestrians are crossing the road while the traffic light is green. We'll investigate the performance limitations of the software as well as finding the optimum setup for the IoT device when considering positioning and pedestrian count. We'll also utilise cloud computing services to communicate this data with endpoint users such as the department for transport who could perform further analysis.

Key words: Pedestrian Safety, IoT (Internet of Things), Object Detection, Cloud Computing

Table of Contents

1. Figures	6
2. Aims and Objectives	7
3. Introduction	8
4. Background Material	10
4.1 - Distinguishing and Quantifying Pedestrian Safety	
4.2 - Road Crossing Designs	
4.3 - Understanding Pedestrian Safety Risks	
4.4 - Understanding a complete IoT Network	
4.5 - Improving Safety with IoT	
4.6 - Improving Transport Safety with IoT	
4.7 - Improving Pedestrian Safety with IoT	
4.8 - Detecting Objects	
4.9 - Edge and Cloud Computing	
5. Methodology	14
5.1 - Approach	
5.2 - Traffic Light Model Construction	
5.3 - Setting up the YOLOr	
5.4 - Completing our Edge Network	
5.5 - Obtaining Results	
5.6 - Moving Data from Edge Network to the Cloud	

6. Results and Analysis	20
6.1 - Simulating a range of pedestrian crossing scenarios	
6.1.1 - Low Pedestrian Count	
6.1.2 - Medium Pedestrian Count	
6.1.3 - High Pedestrian Count	
6.1.4 - Changing the Camera Positioning around Z-axis Circumference plane	
6.1.5 - Changing the Camera Positioning along Z-axis	
6.1.6 - Analysis of IoT device positioning results	
6.1.7 - Alerting oncoming traffic while pedestrians cross on green light	
6.1.8 - Brief overall note of results and comparison with real people	
6.1.9 - Transferring data to the cloud	
7. Conclusion	34
8. Future Work	36
8.1 - Pavement Pedestrian Detection	
8.2 - Detecting Anomalies in the Road	
8.3 - Signal Communication with Smart Cars	
8.4 - A Complete Cloud Architecture	
8.5 – Vision of a Smart City	
9. Self-Reflection	39
10. Appendices	41
11. References	48

1. Figures

Figure 1: A flowchart that describes the IoT network process

Figure 2: A UML Sequence model of the IoT network process

Figure 3: A snapshot of the road crossing model for applying computer vision

Figure 4: The command for executing the YOLOr software

Figure 5: Table of certainties for physical pedestrian variables

Figure 6: Model setup of 3 Pedestrians

Figure 7: Table of certainties for 3 pedestrians

Figure 8: Model setup for 4 pedestrians

Figure 9: Table of certainties for 4 pedestrians

Figure 10: Model setup for 4 pedestrians with anomaly

Figure 11: Table of certainties for 4 pedestrians with anomaly

Figure 12: Model setup for 8 pedestrians

Figure 13: Table of certainties for 8 pedestrians

Figure 14: Model setup for 9 pedestrians

Figure 15: Table of certainties for 9 pedestrians

Figure 16: Model setup for 10 pedestrians

Figure 17: Table of certainties for 10 pedestrians

Figure 18: Model setup for 13 pedestrians

Figure 19: Table of certainties for 13 pedestrians

Figure 20: Model setup for 15 pedestrians

Figure 21: Table of certainties for 15 pedestrians

Figure 22: Graph of certainties when varying the angle of IoT device around circumference of z-axis plane

Figure 23: Graph of certainties when varying the distance of IoT device to target pedestrian

Figure 24: Model design that illustrates positioning measurements of the IoT device

Figure 25: Terminal output at instance where no people are detected, and lights are 'GREEN'

Figure 26: Terminal output at instances where people are detected but lights are 'AMBER' and 'RED' respectively. No signal sent

Figure 27: Terminal output at instance where people are detected, and lights are 'GREEN', signal sent

Figure 28: 'pedestrian.txt' output file listing instances of pedestrians detected when lights are 'GREEN'

Figure 29: A crowd of live music goers detected using the YOLOr software

Figure 30: A typical SQL query on our pedestrian table in Amazon Athena

Figure 31: The table generated from the SQL query in figure 30

Figure 32: An image of a road crossing in Cardiff divided into three sections

Figure 33: An example of trained software detecting the elderly lady's fall related accident

2. Aims and Objectives

Before we dive into the project, we should establish some aims and objectives that we want to come away from having completed. Firstly, we want to demonstrate an automated process that will alert oncoming drivers when pedestrians are crossing at a green traffic light. Once the process is launched, it should require no further input from us until we want the demonstration to stop. Secondly, we want to investigate the performance limitations of the computer vision-based object detection software as insight into the ideal design of an optimised real-world setup when using a computer vision IoT device relative to moving pedestrians. We then want to demonstrate how useful information from this model can be extracted and sent to the cloud for analysis by possible endpoint users. Lastly, we want to provide possible solutions to the limitations found in the investigation as well as other future work that could build on this project.

3. Introduction

When we consider pedestrian safety, we're thinking about the condition in which someone travels on foot (or by mobility aid), within the vicinity of a road, and their immediate threat to injury or fatality.

In 2019, pedestrians accounted for a staggering 27% of all road deaths: a 3% increase from the previous year while the rate of fatalities has stayed fairly constant between 2009 – 2019 as illustrated in *appendix A* (Department for Transport, 2019, p. 9-12). Due to higher populations “the majority of pedestrians killed or seriously injured (KSI's) occur on urban roads” where pedestrians and moving vehicles are more likely to come into contact. The department for transport has stated that walking “has a higher risk than driving, probably because of the lack of pedestrian-based technologies” (Department for Transport, 2015, p. 1-3). Although the fatality rate has stayed relatively constant between 2009 and 2019, the rate of those seriously injured has seen a gradual decrease as illustrated in *appendix B*. It's only in recent years that this rate has started to plateau which might suggest that the current measures we have in place for pedestrian-road safety is reaching its minimum KSI rate which cannot be reduced further without the introduction of new processes for improving pedestrian safety. Demonstrating a model that focuses on the safety of pedestrians with respect to oncoming traffic is an important research objective that has the potential to tackle this statistic, acting as a new approach to improve both pedestrian and road safety.

“Unsurprisingly, pedestrians are more likely to be killed or seriously injured while crossing the road” (Department for Transport, 2015, p. 6). Whether it's during peak commute-to-work times, a late Saturday night back from the pub or a busy Sunday of shoppers, pedestrians will inevitably cross the road when traffic lights are green. Pedestrians caught in this scenario are most commonly in accidents due to factors such as “failing to look properly, ... careless reckless or being in a hurry” and, for ages 16-19, being “impaired by alcohol” (Department for Transport, 2015, p. 10). Due to the fact pedestrians that specifically cross the road pose the greatest threats to safety, this model only considers a road crossing. Because ‘the risk for pedestrian KSI casualties is 22 times higher than for car occupants’ (Department for Transport, 2015, p. 3), we want to use the model to help reduce this factor to the point where it's considered safer for pedestrians to travel on foot/by mobility aid than to use vehicles, particularly in urban areas. With the increased efforts to encourage the public to ‘go green’ by walking or using public transport, this a timely project.

An IoT (Internet of Things) device is a piece of hardware that can read and communicate data within a network of other devices, most commonly via the internet. When we talk about an IoT device, we're not referring to standard computing devices such as desktops, laptops, or smartphones, but rather devices that non-traditionally use the internet and form part of a communicative network such as a smart watch or sensor camera. In this project, we will explore, demonstrate, and discuss the use of IoT devices to reduce the threat towards pedestrian safety.

The report is structured as follows. In section 4 we explore some existing work that seeks to improve road safety which this project will follow on from or work in collaboration with. We'll also look at the relevant technologies that could be used to carry out this project. In section 5, we'll justify the tools used to carry out the demonstration while describing the process of achieving a fully functioning edge network that can improve pedestrian safety and how useful

information can be extracted from it and analysed in the cloud. In section 6 we'll discuss the output result with different dependent variables such as pedestrian quantity, road layout and how these results could be used for further analytics by various endpoint users connected to the edge network. Having concluded this work in section 7, we'll then go onto discuss future work in section 8 that builds on this model in a real-world setting as well as additional features that could further improve safety. In section 9 we'll self-reflect on this project and how the MSc course has provided me with the tools and skills to carry this project and prepare for future work.

4. Background Material

4.1 - Distinguishing and Quantifying Pedestrian Safety

Pedestrian safety should be identified as separate to crime safety. We're aiming to prevent casualties and fatalities due to road traffic accidents as opposed to intentional harm caused by criminal activity. An example method of preventing crime safety would be a CCTV camera that aids identifying and potentially preventing a criminal act from taking place. The following existing work focuses on models designed to aid pedestrian safety.

Pedestrian safety can be quantified based on the individual's decision making. "The pedestrian chooses a path that maximises its utility in terms of two criteria: overall length and risk of a road crash incident". When considering a model where traffic lights are green and a pedestrian has decided to cross the road, this decision making has effectively been made and can no longer be considered a variable but a constant (Corcoran, 2018, p. 5).

4.2 - Road Crossing and Pedestrian Model Designs

The following information on road crossings will help us when it comes to designing the road crossing model simulation. The minimum width for a pedestrian road crossing is 2.4 metres, however, if the crossing is considered popular (most urban crossings), the minimum should be 4 metres. A busy crossing is one where the flow of pedestrians is over 600 per hour, in which case a wider crossing should be put in place. For a pelican crossing, "which uses far-side pedestrian signal heads and a flashing amber/flashing green crossing period", it has a maximum width of 10 metres. Beyond this distance, an island must be introduced to split the road into two crossings: one for each traffic direction. When considering visibility, pedestrians must be able to see oncoming traffic with no obstruction from objects such as parked cars or trees. Furthermore, instances such as wheelchair users and children should also be considered when ensuring full visibility for drivers (Department of Transport, 1995, p. 1-3).

When considering the design of our pedestrian models, the average height for a male in the UK is 1.78m and 1.64m for a female (WorldData.info, No date).

4.3 - Understanding Pedestrian Safety Risks

The threat towards pedestrian safety can be further broken down into two categories: fatality risk and accident risk, the former causing death, the latter resulting in injury. With the pedestrians' decision-making factor kept constant (to cross while the traffic light is still green), there are certain variables that can increase the risk of fatality. Darkness of the road can increase this risk especially when there's no street lighting as well as the road being bi-directional as opposed to one-way. The speed limit of the road is also a key contributing factor where studies show that increasing the limit by 10km/h increase the probability of death by 37% (Olszewski, 2015, p. 89).

4.4 - Understanding a Complete IoT Network

There are various technologies involved in IoT. These technologies are placed at different tiers: Sensors, Microcontrollers, and Internet Connectivity/Service platforms (Kumar, 2019). The sensors that communicate with the microcontroller (which runs the specific software) is connected to an IoT platform, "which integrates data from the different devices and applies

analytics to share the most valuable information with applications built to address specific needs” (Clark, 2016). This explains how these 3 components must work together to not only detect in real time for immediate threat to safety but use the relevant information extracted from this action for analysis to future improve safety.

An IoT designed for transportation purposes is built on a 5-layer architecture in which we can apply our three-tier technology as illustrated in *appendix C*. Firstly, a perception layer made up of sensors, with the relevant software installed on the microcontroller, collects information, for example speed, positioning, or quantity of a particular object. The co-ordination layer provides the means of securely communicating this information to the rest of the network. In real-world applications this could be via Bluetooth, Wi-Fi or 4G. The “brain...responsible for storing, processing, and analysing the information” is found at the artificial intelligence layer and most commonly in the form of a cloud computing platform, responsible for big data analysis. The application layer is based on what we assign this network to do. This could be something that improves traffic safety or perhaps analysis to improve traffic. Lastly, the business layer’s “major responsibility...is to foresight strategies for the development of business models based on the application usage data and statistical analysis of the data” (Kaiwartya, 2016, p. 5360-5361). In a transportation sense, this could be, for example, analysing the number of central London traffic users against unpaid congestion charges to investigate what proportion of users forget to pay their congestion charge.

4.5 - Improving Safety with IoT

Human beings are almost exclusively the focus when considering safety within any environment. In manufacturing industries, a vast majority of accident-related injuries and fatalities are due to human error when incorrectly following control procedures, also known as Lockout tagout (LOTO) procedures, when handling electrics and machinery. In this case study, the introduction of IoT based LOTO devices on machinery is aimed to improve a worker’s obedience to procedures by alerting a supervisor or manager if their action is incorrect and hence a threat to their safety. An application of IoT such as this proves the importance of minimal latency when communicating with a supervisor or manager. “The SMS must reach within standard/defined time and actuation must happen within standard/defined time” to minimise the threat to a user’s safety (Kumar, 2019).

4.6 - Improving Transport Safety with IoT

Using IoT devices to improve safety in transportation has the potential to reduce KSI rate. AWS are a leading company in improving the overall performance of transportation networks. One of these performance-aims is to improve safety by “delivering real-time decision support”. It takes a large amount of various data processing to ensure that someone travels as safe as quick and efficient as possible. Another aim looks to absorb this “data from multiple mobile end-points”. This is an idea that using millions of IoT sensors can be made accessible for real-time support. To reduce traffic-related fatalities, AWS suggest capturing not only crash incidents but “high risk behaviours” such as near misses and red-light violations. This captured data is processed through AWS analytic and Machine Learning services to better understand how roads could be improved or what measures, such as road signs, should be in place (AWS, 2020).

4.7 - Improving Pedestrian Safety with IoT

An emerging technology from recent years is the internet of vehicles (IoV's) which is a blanket term that covers various communication between vehicle and other devices including "vehicle-to-vehicle, vehicle-to-roadside, vehicle to infrastructure of cellular networks, vehicle-to-person devices and vehicle-to-sensors" (Kaiwartya, 2016, p. 5359).

Pedestrian IoV detection technologies are surfacing in newer models of vehicles. For Volvo, their cars display a "collision warning... when there is a risk of colliding with a pedestrian, cyclist or vehicle". A drawback to this technology, which poses thought for future work, is that the IoV device requires a full-bodied view of the pedestrian combined with stationary or "normal human pattern of movement". This drawback may result in the inability to detect pedestrians that use mobility aid or bicycles. Another detection drawback is that the pedestrian must be at least 80cm. A result of this drawback could obstruct the detection of small children (Volvo, 2020).

In Melbourne, camera sensors are being introduced to detect cyclists and pedestrians in real time at road crossings. The aim is to adjust the length of crossing times in direct correlation with the number of pedestrians waiting on the pavement to cross. The model focuses on the number of pedestrians as its independent variable. The idea is to benefit drivers and pedestrians: the former to reduce unnecessary delays to drivers when minimal or no pedestrians are crossing and the latter to give a safer amount of time to cross when it's busy (Bicycle Network, 2021).

4.8 - Detecting Objects with Computer Vision in Transportation

Traditionally, object detection software applies its "model to an image at multiple locations and scales. High scoring regions of the image are considered detections" (Redmon, 2016). In transportation, a pipeline methodology is used for describing computer vision installation and application as described in *appendix D*. Firstly, for "Image and Video Capture" we must consider "field of view..., ruggedness and cost". This is followed by "Data Pre-Processing" where the camera is correctly positioned, and its plane of sight covers the entire surface area of the road crossing. This also considers factors such as "brightness...and motion stabilisation". "Feature Extraction" identifies the relevant pixels within the real-time image that's captured by the camera. For this model, we're interested in the pixels that correspond to the pedestrians. Our software must be able to detect heads, arms, and bodies etc. The "inference engine" is the point at which the sensor "takes as input the feature descriptors and emits a hypothesis or decision". This is the detection certainty that the software displays on each bounding box having detected an object. "Data Presentation and Feedback" for our software will be outputting the percentage likelihood that the local feature extracted is a pedestrian (using a minimum certainty to allow a positive ID). This publication goes on to explain the application of computer vision for counting pedestrians. A potential challenge posed by them is the fact that "people tend to walk and wait in platoons" (Loce, 2017, p. 8-10).

A study made in Rwanda used computer vision to track pedestrians crossing foot-trail bridges. In this use case, computer vision is used to observe people while tracking their movements, including the direction of travel based on tracks. When monitoring people in real time it's important to get approval from an ethics or data community to install a camera in public. They used an "open source Darknet implementation of the YOLO (You Only Look Once) object detection deep neural network that is pretrained to, among other things, detect people at frame rate". When comparing the detection performance between image stills and videos, they found

that “the motion-activated video-clip files provided greater support for the computer vision algorithm compared to the stills” (Thomas, 2020).

YOLO, a real-time object detection tool of revolutionary speed and accuracy, “divides the image into regions and predicts bounding boxes and probabilities for each region”. The benefits of using YOLO is that it improves on classifier-based (“predicting the class of given data points”) by making a prediction using “single network evaluation” to apply bounding boxes to the object in real-time thus feeding an output as the software sees it (Redmon, 2016).

A problem that currently stands with deep learning object detection methods are “the effects of small pedestrian ratios and considerable differences in the aspect ratios of input images”. Performance decreases exponentially as the distance between pedestrians and camera increases (Wei-Yen, 2021, p. 934).

4.9 - Edge and Cloud Computing

Previously, when we discussed the 5-layer architecture of a Transportation IoT network, the third-tier technology is a centralised IoT platform in which data generated by each device is sent for analysis. Cloud platforms offer scalable, “high-computational capacity with moderate response time” making it suitable for this network of data to communicate with (Munoz, 2018, p. 1420).

A particular study uses cloud computing services to create a “real-time ECG monitoring system” IoT device. Data that’s read by the ECG is sent and stored in an AWS S3 Bucket for analysis. ‘Message *‘Queuing Telemetry Transport (MQTT)’* protocol and *‘Hyper Text Transfer Protocol (HTTP)’* provide the gateway between device and cloud platform (Manju, 2021).

A fundamental analytics tool within AWS is Amazon Athena: ‘an interactive query service that makes it easy to analyse data in S3 using standard SQL. Athena is serverless so there is no infrastructure to manage, and you pay only for the queries you run’ (AWS, 2021).

The study by Munoz et al. that we mentioned before, challenges IoT devices using the cloud considering an exponential increase of IoT devices is predicted to be deployed in the upcoming years. “One core cloud infrastructure is not a long-term scalable solution”. A more suitable method of distributing the analytics to reduce the workload on the cloud is by sharing some of this analysis on the edge. In transportation, it is “mission critical IoT applications, with very stringent delay requirements, may require performing IoT analytics in the edge in order to perform real-time actions” (Munoz, 2018).

5. Methodology

In this section, we'll describe the method carried out to build, demonstrate and experiment with our pedestrian crossing model. In section 4.1, we outline the design of this model using the 5-layer architecture discussed in section 3. In 4.2 we'll discuss the physical setup of our model, including the design of our pedestrian model. From there, we explain the setup of the object detection software in 4.3. The traffic light simulation component is explained in 4.4 as well as the overall edge network that responds to it. We then explain how we obtain results in 4.5 while moving this important data to the cloud in sub-section 4.6.

5.1 - Approach

In this section, we propose a model that aims to reduce the threat of pedestrian safety by alerting oncoming traffic that pedestrians are crossing the road when traffic lights are green. The flowchart in figure 1 explains the IoT network process that occur dependent on the traffic light colour.

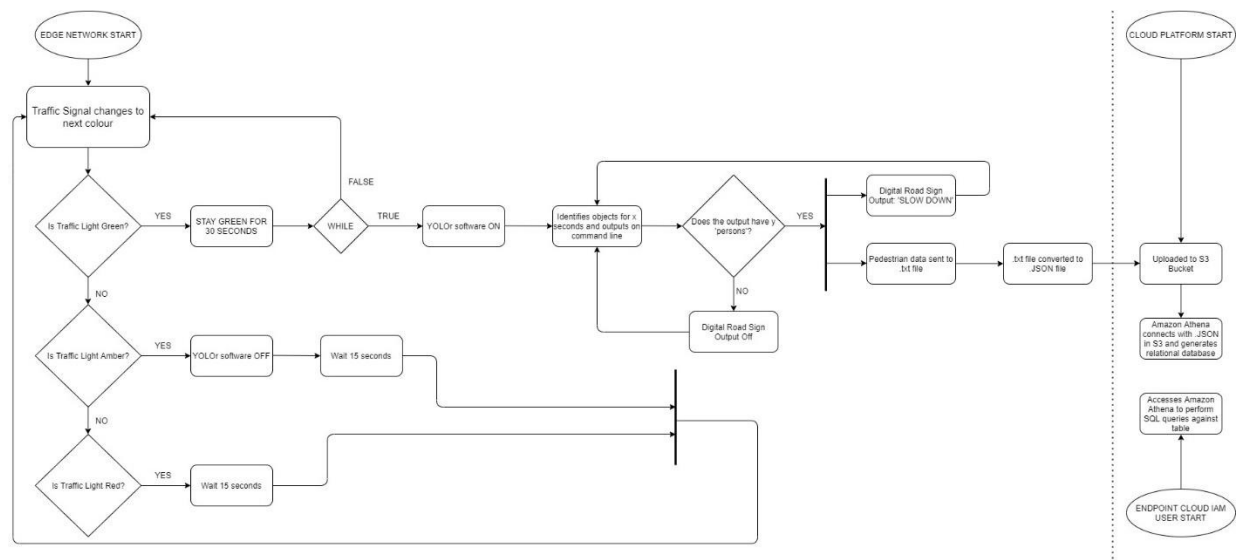


Figure 1: A flowchart that describes the IoT network process

The basis of our design is inspired by the technology components described in [12] as well as the 5-layer architecture explained in [11]. Given that this project is a demonstration, we will use tools suitable for a project of this magnitude with further discussion on how it translates and scales up to make it suitable for real-world application. Our 'perception layer' is an IoT device that uses a laptop webcam and 'YOLOr' software: a vision-based object detection system that uses machine learning to identify certain objects. The appropriateness of using this version of object detection is the fact that it can run in real-time and is extremely fast compared to previous models. It also has the benefit of off-the-shelf use for users not familiar with writing computing vision, machine learning scripts. As this model, mission-critically, must have minimal latency (and considering research made from [15]) this IoT device will communicate with the traffic light and digital road sign within an edge network (part one of the 'artificial intelligence' [11] layer). This edge network makes our 'communication layer' [11] relatively straight forward, but what's more, is that it keeps in theme with [15] by only sending streamlined, relevant data across to our cloud platform: Amazon S3 which is connected with Amazon Athena (part two of

the ‘artificial intelligence’ ^[11] layer) to reduce data traffic and possible delays. Our final layer: the ‘business layer’, will be based on our discussion of potential, analytical use cases for cloud endpoint users such as the department for transport in ways in which they can improve road layout and traffic. We can summarise the sequence like so:

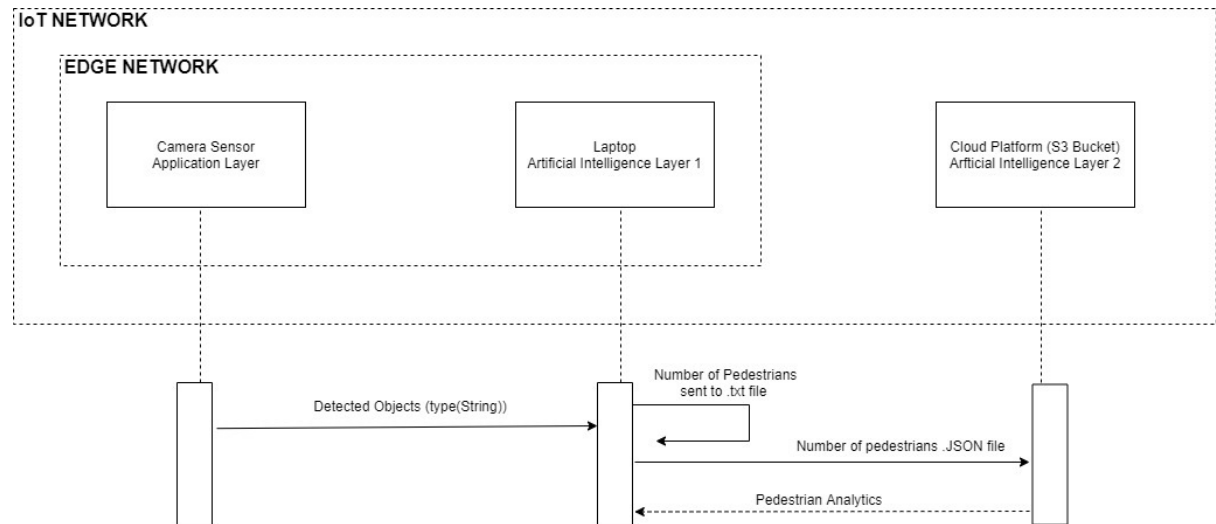


Figure 2: A UML Sequence model of the IoT network process

To test the model’s reliability, we will investigate the effect of variables like pedestrian numbers, physical features of the pedestrians and various positioning of the IoT device along different axes. These processes are described in the following sub sections

5.2 - Traffic Light Crossing Model Construction

The IoT device is a 2015 MacBook Pro laptop that uses a 720p FaceTime HD camera. Using photographed faces attached to card cut-outs (approximately 20 cm in height and scaled to represent the average male height), we’re able to setup stills of pedestrians crossing the road bi-directionally. The direction of travel is illustrated by faces or back of heads. In this model we use a range of crossing lengths that’s scaled, approximately, to the size of our pedestrian cut outs. We’ll then deploy our pedestrians in various numbers while changing the width of roads. These different volumes of pedestrian clusters will allow us to test the performance of the computer vision to see if denser volumes affect its ability to identify every pedestrian. Our number of pedestrians in each scenario will range from a minimum of 1 pedestrian, to represent “quieter” periods, to 15 for “busy” periods. A photograph of this set up is presented in figure 3.



Figure 3: A snapshot of the road crossing model for applying computer vision

5.3 - Setting up the YOLOr

As this model doesn't require modifications to the code to train new objects (humans are already trained), we can install an off-the-shelf version of YOLOr. As we're running the object detection software from the CPU, we need to ensure that PyCharm and Anaconda are installed as prerequisites. We're then able to initiate a conda environment in which the YOLOr software will run as well as pip installing any dependencies that we don't currently have.

To test the object detection is working correctly, we input the following command within our environment to detect objects from a specified image, in this example, a horse:

```
python detect.py --source inference/images/horses.jpg --cfg  
cfg/yolor_p6.cfg --weights yolor_p6.pt --conf 0.25 --img-size 1280 -  
-device cpu --save-txt
```

Figure 4: The command for executing the YOLOr software [1]

Once the software has executed the command, an output image with detection boxes is saved. *appendix E* confirms the object detection is functioning correctly. To run the software on a webcam in real time, we change the value for '--source' to '0'. Now when we run the command a new window will display the output with object detection applied (*appendix F*) and a description of what object (and quantity) has been detected in that frame (*appendix G*).

5.4 - Completing our Edge Network

With YOLOr installed, we use Python to create a complete edge network that allows this software to interact with a traffic light simulation and subsequent digital road sign.

The basis of the traffic light simulator uses real-time to achieve automated signal change which we allocate to variable called 'current_light'. The first 30 seconds of a minute sets this variable to 'GREEN', between 30-45 it's changed to 'AMBER' and the final 15 seconds changed to 'RED'.

We begin by executing the command from figure 4 (source set to '0'). When this runs, a subsequent 'pedestrian.txt' file is opened. A while loop is used to run the object detection software for 12.2 seconds before it's killed. It's found through trial and error that for this laptop's CPU, it takes 12.2 seconds for the process to initiate, detect and record one line of data. If we configured it to allow more than one output per loop, when it comes to extracting this information to the cloud, our system will think more pedestrians are present than what is true.

While the process is running and recording output data on the CLI, it also saves the information to another .txt file under the output directory. It's from this output.txt file that we read from to send the correct signals to the digital road sign. If the line records a 0, which represents a person, we will print 'PERSON DETECTED' and set a 'person_seen' variable to 1. This is mainly for our benefit to ensure the script is running correctly when comparing this output to the road sign. Now, if the 'person_seen' variable is equal to one and the 'current_light' variable is equal to 'GREEN', the digital road sign will emit 'SLOW DOWN, PEDESTRIANS CROSSING'. When this isn't the case, we print 'SIGNAL OFF' to know that the sign isn't displaying anything.

During this event, we have another for loop that reads the output.txt file to count how many pedestrians have been detected in this instance. This information is then written to the pedestrian.txt file format. It's important that the contents of the output.txt file is truncated during each 'GREEN' signal instance.

5.5 - Obtaining Results

When demonstrating the pedestrian detection model, we want to challenge some of the drawbacks found from previous research as well as a series of possible factors that could affect the computer vision software's performance to investigate its limits. We will explain these results for each scenario in a set of sub sections.

We begin by placing the IoT device to one end of a rectangular table (with grey sheeting to match road colour) that will imitate the road layout or plane of view. The background will remain constant. We'll attempt to keep lighting as constant as possible although results may be taken from various points of the day.

As an initial test of YOLOr's performance (before introducing the sub section scenarios), we take careful consideration into the appearance of our pedestrian figures to see whether any factors would reduce detection capability. These factors are (image captured in *appendix H.1-J.3*):

- Varied pedestrian size
- Some pedestrians wearing a hat
- The back of the head (short male hair, long female hair)
- Varied hair lengths

We run the object detection software five times on each physical variable and obtain an average certainty as recorded. We ensure that the distance from the camera is kept constant.

Following this, we arrange a setup with low numbers of pedestrians to represent our first scenario. Three and four sets of pedestrians will be placed within the plane as well as a scenario that looks at the effect of an anomalous object against desired targets. We'll then increase this population count to medium and high numbers of pedestrians: eight, nine and ten for the medium population scenario, thirteen and fifteen for the high population. It's important that we aim to scale each model as accurately as possible. By measuring the length of the pedestrian figures, and setting those lengths to average heights, we can scale out minimum and maximum road crossing distances according to research found in section 3.2. We can also adjust the IoT device's elevation to improve its performance through improved pedestrian certainty.

Our next scenario looks at finding the optimum angle at which our device could be positioned to maximise certainty in real life. We want to explore the maximum limitations so by positioning two pedestrians close together to imitate obstructed views will help find this. Once this angle has been found within a certainty of roughly 5 degrees, we move onto the next scenario that investigates the effect of certainty with distance of the device from the target.

Having explored performance, we let our IoT device run against a group of pedestrians to seek confirmation that:

- 1) The traffic light simulation correctly changes according to real time
- 2) The IoT device and digital road sign correctly communicate with one another in the instance that pedestrians are detected crossing the road when lights are 'GREEN'
- 3) The correct information (the circumstance in point two) is saved to 'pedestrian.txt'
- 4) Our 'JSON_converter' script correctly converts the 'pedestrian.txt' file to JSON format

5.6 - Moving Data from Edge Network to the Cloud

We want to initiate a network connection between our IoT and cloud platform for delivering the pedestrian data for reading online, downloading, or running SQL queries against. Therefore, we need to convert our pedestrian.txt file into JSON format to make it readable for SQL querying. *appendix J* is a snapshot of our script created to convert .txt to JSON. Note this JSON document must be slightly altered to the correct, simplified format that works for Amazon Athena. The script splits each line and assigns each column of values to the fields described which are contained within a set of brace brackets. No comma should be at the end of the curly brackets. We must also ensure that each field value is set as the correct data type.

Now we have the correct format of our data, ready for sending to the cloud. We want to demonstrate a conceptual process in which various endpoint users can access this information. We therefore need three resources from Amazon Web Services (AWS):

- 1) Amazon S3
- 2) Identity Access Management (IAM)
- 3) Amazon Athena

Logging into the AWS console as a root user, we begin by creating an S3 bucket. We configure the unique bucket name: 'pedestrian-crossing-s3' and assign it to the eu-west-2 region as all our endpoint users would be based in the UK. For this scenario, we will disable server-side encryption of objects (our files) for the purpose of this demonstration. With this bucket created, we upload the .txt and .json file to the bucket.

We now want to create a mock endpoint user, in this case, the UK government's department for transport. Under the IAM resource, we first have to setup Multi-factor Authentication (MFA) as the root user to improve security. We then add a new user and configure their access type, in this case, password for accessing the console. A User Group is created which allows us to apply permissions that each user within the group will inherit. In this case, we've permitted our 'Department_for_Transport' user to change their IAM User Password and have Read Only access to Amazon S3 as illustrated in *appendix K*. We also have the option to make the bucket public if, in real-world, it was decided that the data be made to the public.

With S3 and IAM in place, we're ready to point Amazon Athena in the direction of the .JSON file as our data source. We create a table and list the field names to those matching within our 'JSON-converter' script. We also have the option to save query output results to either our 'pedestrian-crossing-s3' bucket or a different bucket. For this demonstration we'll use a separate bucket as we want our endpoint users to only see the queries made by them and not others. Now we can use the Athena dashboard to run SQL queries against our table with results saved to S3. This completes our cloud demonstration.

6. Results and Analysis

When reviewing the aims and objectives in section 2, the sets of results that we set out to find were, firstly, to demonstrate an automated process that will alert oncoming drivers when pedestrians are crossing at a green traffic light. Secondly, to find performance limitations of the computer vision based YOLOr software. Thirdly, demonstrate how useful information within this model can be extracted and sent to the cloud for further analysis. Lastly, to identify any other limitations, for instance in the hardware, to support possible solutions for future work.

We discovered one setback with our IoT device in which the camera lens was not particularly wide angled which, unavoidably, limited our width to which pedestrians could be placed if we wanted to keep the model scaled to size. This limit was found to be 4 pedestrians. We also found that the IoT device should be raised to approximately twice the height of the pedestrians to give the device optimum potential to detect all pedestrians in each scenario (illustrated in *appendix L*)

6.1 - Simulating a range of Pedestrian Crossing Scenarios

Figure 5 displays the results of average certainties when detecting pedestrians with various physical attributes.

Variable Factor	Average Certainty of Person
Varied Pedestrian Size (Larger)	$(0.82+0.83+0.80+0.82+0.79)/5 = 0.81$
Varied Pedestrian Size (Smaller)	$(0.85 + 0.73 + 0.87 + 0.74 + 0.80)/5 = 0.80$
Pedestrian Wearing a Hat	$(0.75 + 0.83 + 0.76 + 0.81 + 0.84)/5 = 0.80$
Back of the Head (Male, Short Hair)	$(0.78+0.82+0.75+0.67+0.84)/5 = 0.77$
Back of the Head (Female, Long Hair)	$(0.71+0.69+0.72+0.62+0.67)/5 = 0.68$

Figure 5: Table of certainties for physical pedestrian variables

All variables provided no change to the YOLOr's ability to detect the individual which provided us with confidence to proceed. However, the average certainty for each variable type did vary. The larger and smaller pedestrian models as well as the pedestrian wearing a hat had the highest value with only 0.01 difference between them. The male back of the head had a slightly lower average of 0.77 however the female back of the head had a considerably lower score of 0.68.

An immediate drawback from our model was the frame rate. The IoT device would typically capture a frame, on average, every 3 seconds. This was due to the performance of the computer vision software when running on our laptop's particular CPU. Ideally, the software would be running on a Graphics Processing Unit (GPU) to increase the frame rate. However, the ability to capture motionless pedestrians with bounding boxes in real-time by a degree of high certainty (< 0.8) remained unaffected which suffices for this demonstration.

6.1.1 - Low Pedestrian Count

Camera position = eye level (9.5 inches)

Figure 6 demonstrates our typical setup with a low number of pedestrians crossing. To ensure the model was correctly scaled in keeping with the distances mentioned in section 4.2, the road width must be greater than 11.8 inches. For this scenario, we didn't want to exceed the length of 19.67 inches (minimum width for a popular crossing). The pedestrians were then scattered evenly within these two constraints. We scanned 3 (two camera facing and one walking) and 4 pedestrians (two camera facing and two walking away) for this model as it resulted in minimal overlap when positioning the camera adjacent to the crossing direction. We also arranged an anomalous object behind three pedestrians.



Figure 6: Model setup of 3 Pedestrians

Reading	Pedestrian Detection Certainty			
	Pedestrian 1	Pedestrian 2	Pedestrian 3	Average
1 st	0.85	0.89	0.60	0.78
2 nd	0.59	0.69	0.70	0.66
3 rd	0.83	0.72	0.59	0.71
4 th	0.75	0.86	0.78	0.80
5 th	0.82	0.84	0.50	0.72
				0.73

Figure 7: Table of certainties for 3 pedestrians

For the results in figure 7, there were occasional dips in certainties throughout the 5 readings for each pedestrian, however, the average certainty remains at a reliably high score and always detects the correct number of pedestrians. Comparing these results to the pedestrians' positioning in figure 6, we noticed that pedestrian 3 has noticeably lower certainties when no obvious reason would explain it doing so.



Figure 8: Model setup for 4 pedestrians

	Pedestrian Detection Certainty				
Reading	Pedestrian 1	Pedestrian 2	Pedestrian 3	Pedestrian 4	Average
1st	0.66	0.63	0.83	0.82	0.74
2nd	0.60	0.62	0.67	0.80	0.67
3rd	0.42	0.47	0.75	0.76	0.60
4th	0.51	0.63	0.73	0.84	0.68
5th	0.41	0.66	0.80	0.89	0.69
					0.68

Figure 9: Table of certainties for 4 pedestrians

In figure 9, results show that the average certainty decreased by approximately 7%. Noticeably this time, the certainties of each pedestrian would increase going from left to right. This could be due to the uneven distribution of indoor artificial light.



Figure 10: Model setup for 3 pedestrians with anomaly

Reading	Pedestrian 1	Pedestrian 2	Pedestrian 3	Average
1st	0.59	0.67	0.51	0.59
2nd	0.59	0.61	0.48	0.56
3rd	0.54	0.66	0.41	0.54
4th	0.81	0.81	0.46	0.69
5th	0.68	0.81	0.50	0.66
				0.61

Figure 11: Table of certainties for 3 pedestrians with anomaly

The detection certainty for the anomalous object remained consistently high (between 0.85 and 0.88) for each reading whereas certainties for each pedestrian had dipped further compared to the results in figure 9 and 10. There was no evidence for a change in contrast when introducing this object indicating that the addition of a different object type could affect the certainties of other objects in the plane but that would require further investigation.

Our IoT output detected the correct number of pedestrians in each scenario. The average certainty for detected pedestrians was 0.73, 0.68 and 0.61 respectively. The graph in *appendix M* illustrates how the average certainty decreases by an average of 0.12 as the number of pedestrians are increased or when introducing an anomalous object.

6.1.2 - Medium Pedestrian Count

Camera Position = 16 inches

In keeping with the model's scalability, the minimum distance from the camera's position was 19.67 inches which classifies this scenario as a popular road crossing. We scanned 8 pedestrians (4 camera facing and 4 walking away), 9 pedestrians (5 camera facing and 4 walking away) and 10 pedestrians (5 camera facing and 4 walking away) in various arrangements as illustrated in figures below. Now that the number of pedestrians has increased, we'd expect to see a greater diversity in certainties given the results. This subsection looks at how large these differences are.



Figure 12: Model setup for 8 pedestrians

	Pedestrian Detection Certainty								
Reading	1	2	3	4	5	6	7	8	Average
1st	0.70	0.35	0.75	0.67	0.67	0.75	0.62	0.86	0.67
2nd	0.70	0.28	0.59	0.63	0.60	0.77	0.56	0.86	0.62
3rd	0.69	0.34	0.70	0.58	0.76	0.76	0.48	0.88	0.65
4th	0.63	0.29	0.75	0.70	0.79	0.81	0.68	0.89	0.69
5th	0.68	0.41	0.34	0.63	0.63	0.75	0.65	0.86	0.62
									0.65

Figure 13: Table of certainties for 8 pedestrians

Pedestrian 2 had the lowest range of certainties for each reading. This could be a combination of this pedestrian being at the lower end of sizes as well as the partial obstruction by pedestrian 1 and 3. Pedestrian 8 had consistently the highest certainty. We would have expected pedestrian 8 to be highest given his position, but the fact that he's walking away could cost the software's ability to detect as high as it could than if he was facing forward as we learnt from our initial 'pedestrian variables' testing.

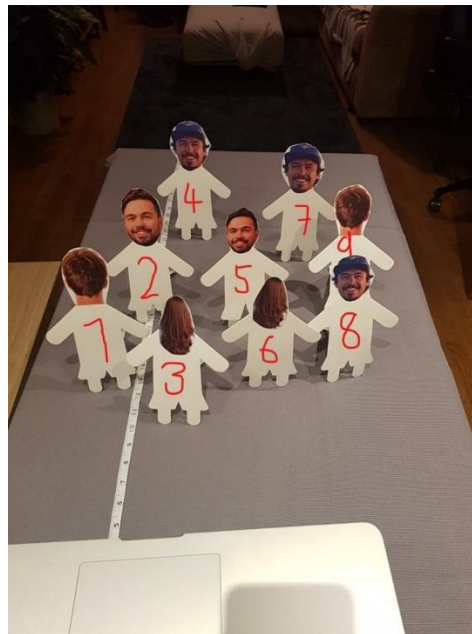


Figure 14: Model setup for 9 pedestrians

	Pedestrian Detection Certainty									
Reading	1	2	3	4	5	6	7	8	9	Average
1st	0.80	0.40	0.78	0.68	0.75	0.89	0.70	0.84	0.39	0.69
2nd	0.81	0.41	0.82	0.68	0.63	0.90	0.66	0.83	0.45	0.69
3rd	0.73	0.40	0.80	0.70	0.61	0.90	0.64	0.83	0.33	0.66
4th	0.77	0.30	0.64	0.66	0.70	0.88	0.67	0.78	0.41	0.65
5th	0.78	0.51	0.61	0.77	0.58	0.84	0.66	0.84	0.39	0.66
										0.67

Figure 15: Table of certainties for 9 pedestrians



Figure 16: Model setup for 10 pedestrians

	Pedestrian Detection Certainty										
Reading	1	2	3	4	5	6	7	8	9	10	Average
1st	0.85	0.84	0.42	0.73	0.46	0.49	0.83	0.90	0.73	0.57	0.68
2nd	0.86	0.85	0.38	0.69	0.47	0.41	0.85	0.91	0.76	0.72	0.69
3rd	0.84	0.84	0.44	0.59	0.48	0.37	0.87	0.87	0.82	0.57	0.67
4th	0.86	0.85	0.43	0.75	0.48	0.33	0.76	0.91	0.74	0.60	0.67
5th	0.84	0.88	0.44	0.74	0.62	0.39	0.83	0.89	0.62	0.66	0.69
											0.68

Figure 17: Table of certainties for 10 pedestrians

For the results in figure 17, pedestrians positioned to the rear of the crossing had the lowest range of certainties. Pedestrian 3 and 6, both in similar positions however one facing, the other walking away, had a very similar range of certainties. This could suggest that beyond a certain distance, the forward/back facing attribute is governed by the distance rather than appearance but would require further investigation.

The correct number of pedestrians were detected in each scenario with occasional drop out within one frame. The average pedestrian detection certainty for a medium pedestrian count remained reliably consistent with a fluctuation of just 0.03

6.1.3 - High Pedestrian Count

Camera level = 20 inches

We discovered that the camera's height was too low for this scenario. The video output, *appendix N*, illustrates how the bounding boxes were too bunched so that we were unable to read the certainties. The CLI output was also unable to detect all 13 or 15 pedestrians at this height due to obstruction, so the camera was raised to 20 inches to resolve this.



Figure 18: Model setup for 13 pedestrians

	Pedestrian Detection Certainty															
Reading	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	Average
1st	0.85	0.64	0.44	0.56	0.46	0.72	0.41	0.56	0.60	0.67	0.76	0.61	0.41	0.52	0.58	0.59
2nd	0.86	0.65	0.41	0.52	0.60	0.80	0.44	0.53	0.62	0.68	0.77	0.55	0.51	0.51	0.52	0.60
3rd	0.86	0.68	0.48	0.58	0.69	0.74	0.42	0.54	0.61	0.68	0.73	0.53	0.38	0.54	0.52	0.60
4th	0.86	0.67	0.51	0.55	0.60	0.73	0.43	0.59	0.68	0.81	0.60	0.37	0.61	0.41	0.54	0.60
5th	0.85	0.67	0.35	0.55	0.68	0.76	0.50	0.54	0.59	0.83	0.62	0.38	0.62	0.76	0.57	0.62
																0.60

Figure 19: Table of certainties for 13 pedestrians



Figure 20: Model setup for 15 pedestrians

Reading	Pedestrian Detection Certainty													Average
	1	2	3	4	5	6	7	8	9	10	11	12	13	
1st	0.86	0.49	0.67	0.51	0.34	0.57	0.73	0.85	0.63	0.60	0.73	0.42	0.61	0.62
2nd	0.88	0.55	0.70	0.59	0.39	0.52	0.74	0.85	0.61	0.62	0.63	0.49	0.45	0.62
3rd	0.85	0.58	0.67	0.45	0.32	0.54	0.72	0.70	0.59	0.71	0.59	0.44	0.49	0.59
4th	0.89	0.44	0.60	0.55	0.28	0.51	0.73	0.85	0.63	0.60	0.73	0.42	0.61	0.60
5th	0.87	0.43	0.75	0.57	0.31	0.52	0.72	0.79	0.64	0.51	0.64	0.48	0.50	0.59
														0.60

Figure 21: Table of certainties for 15 pedestrians

For the high-count scenario, the range of certainties very similar to the medium count scenario, however the fluctuation of average certainties was reduced to 0.

Looking back at the results for each pedestrian count scenario, as the pedestrian count increases, the average detection certainty of pedestrians decreases. The fluctuation also decreases. This evidence shows that although the object detection software becomes less certain on whether the object is in fact a pedestrian, it's compensated by greater reliability that it has given the score it should be. Even though we see a reduced average, we think the IoT device using this software is a reliable method for real world application as it has proved it can detect a wide number of pedestrians consistently with occasional loss in detection on pedestrians that never exceed one frame.

6.1.4 - Changing the Camera Positioning around Z-axis circumference plane

If two pedestrians walk directly in line with one another, we want to find the angle at which the computer vision IoT device can no longer distinguish between them. We'll assume a realistic distance between them (scaled down to 4 inches). We'll also position the camera height at eye level to investigate the true maximum limit of performance in this given scenario. Now we re-position the camera along a 13 inch + laptop length radius (total of 22 inches) to the front pedestrian along the z axis plane.

As illustrated in *appendix O*, when the IoT is positioned perpendicular to the front pedestrian, it is unable to detect the rear pedestrian. Figure 21 displays a graph of pedestrian detection certainties of the rear pedestrian when the IoT device is positioned between various angles.

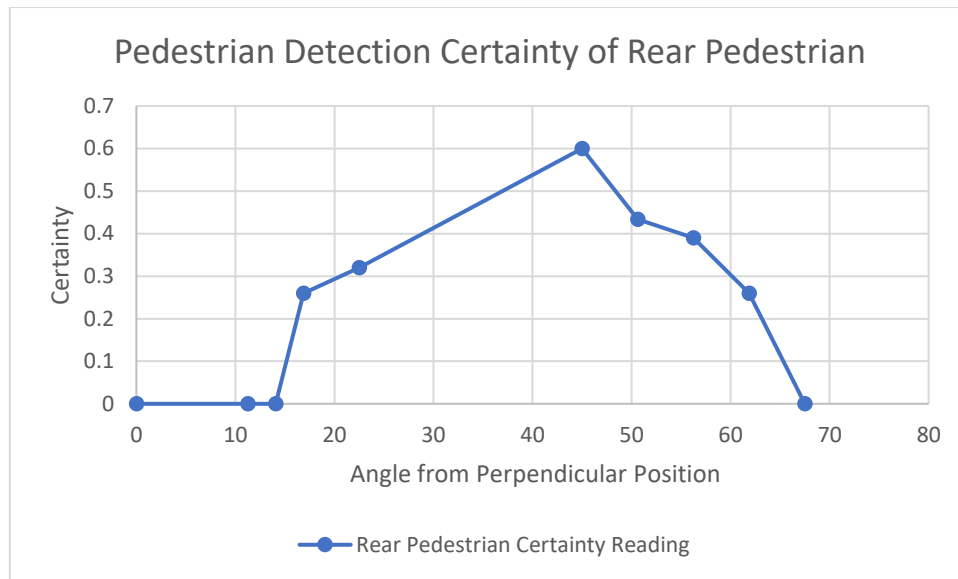


Figure 22: Graph of certainties when varying the angle of IoT device around circumference of Z- axis plane

The optimum angle at which we achieved the optimum certainty was ~ 45-degree angle. When comparing the resulting certainty reading to that of previous scenarios, it was at the lowest range of averages. The minimum angle that the IoT device camera must be positioned to distinguish between both pedestrians is within the range of 14-16 degrees. The maximum angle was found to be ~65 degrees.

6.1.5 - Changing the Camera Positioning along z-axis

With a similar setup to our z-axis plane scenario, we want to investigate the effects of varying IoT device distances from the target for detection. We'll still use our pedestrian model for this setup. We positioned the IoT device at the optimum angle of ~45-50 degrees, which was discovered from the previous scenario. The device was placed at varying distances both smaller and greater than 22 inches.

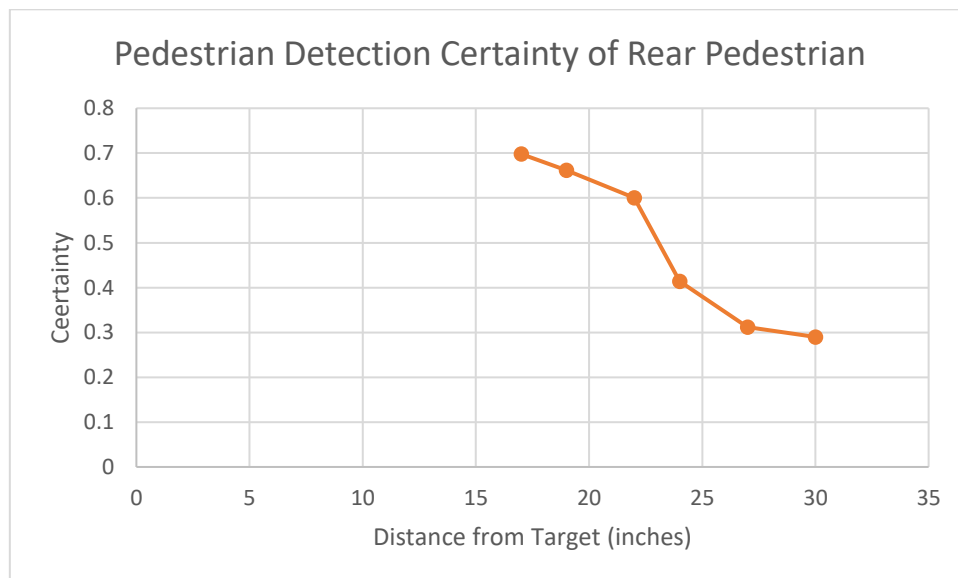


Figure 23: Graph of certainties when varying the distance of IoT device to target pedestrian

The graph shows that the closer the IoT device is brought to the two closely positioned targets, the higher the certainty becomes.

6.1.7 - Analysis of IoT device positioning results

We can produce a scaled design of the optimum real-world setup for this model. If we consider the average number of pedestrians as the independent variable, we can use the results based on IoT device positioning to produce a design.

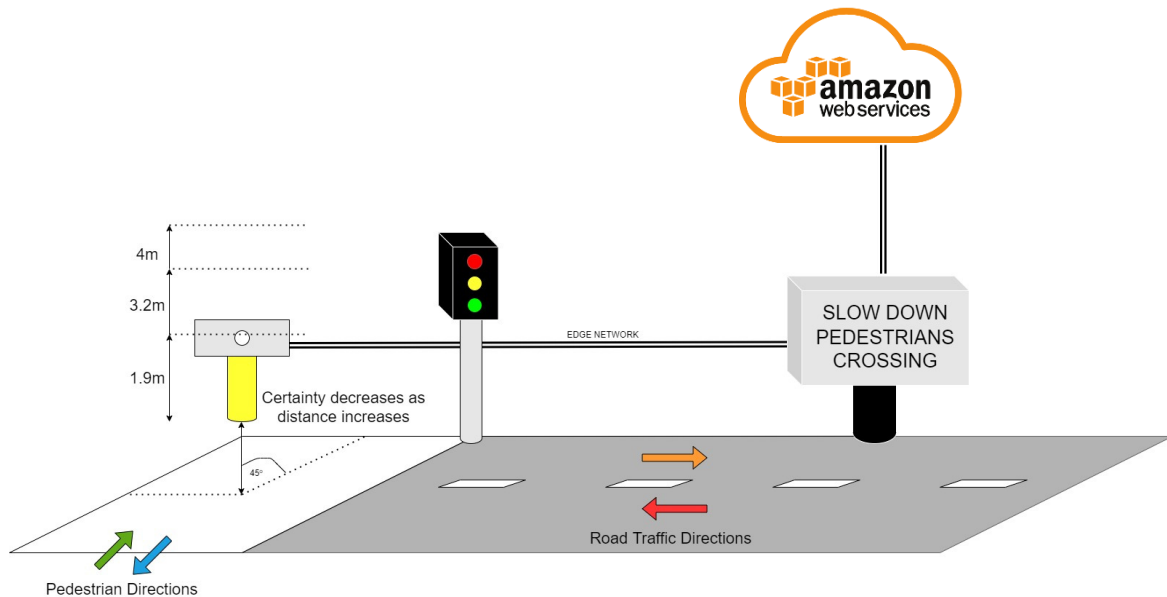


Figure 24: Model design that illustrates positioning measurements of the IoT device

Figure 24 illustrates the possible setup of our IoT device within a real-world scenario. When we up-scale our measurements from inches to real-world scale, we find that the suitable height for an IoT device at a low count pedestrian crossing is 1.9 meters. For a medium and high-count crossing, the suitable height is 3.2 and 4 meters respectively. Further investigation may look at suitable heights for extreme pedestrian counts such as 30 or more. The camera is positioned at a 45° angle from the typical direction of travel. Not all pedestrians will walk directly along this axis, but this angle remains the most appropriate. Distance to the crossing has not been specified due to its dependency on the surface area of the crossing that the IoT device must cover within its plane of sight.

6.1.6 - Alerting oncoming traffic while pedestrians cross on a green light

When executing our 'detect.py' script, the terminal produces a simulation of a traffic light system. The terminal successfully displays 4 outputs for each iterative start-to-stop loop of our detection software, these are:

- 1) The number of persons detected (including any anomalous objects that have been trained for detection)
- 2) Whether a person has or has not been detected
- 3) The traffic light status
- 4) The resulting display from the digital road sign

When placing no pedestrians within the plane of sight, the following output was displayed which confirms that our IoT device can distinguish the difference between people and no people while not falsely alerting oncoming traffic.

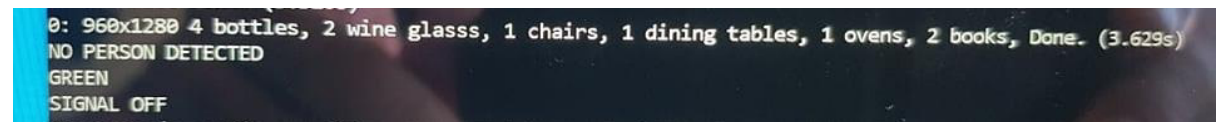


Figure 25: Terminal output at instance where no people are detected, and lights are 'GREEN'

Figure 25 displays an instance where objects other than people were detected. Although these were detected while the traffic light displayed 'GREEN', our IoT device sent no communication to falsely alert oncoming drivers which we can see is evident by the 'SIGNAL OFF' output.

We then executed the run.py script with pedestrians to demonstrate our real-world process. As displayed in figure 26, when our traffic light simulation displayed 'RED' or 'AMBER', pedestrians are detected but no signal is sent to the digital sign.

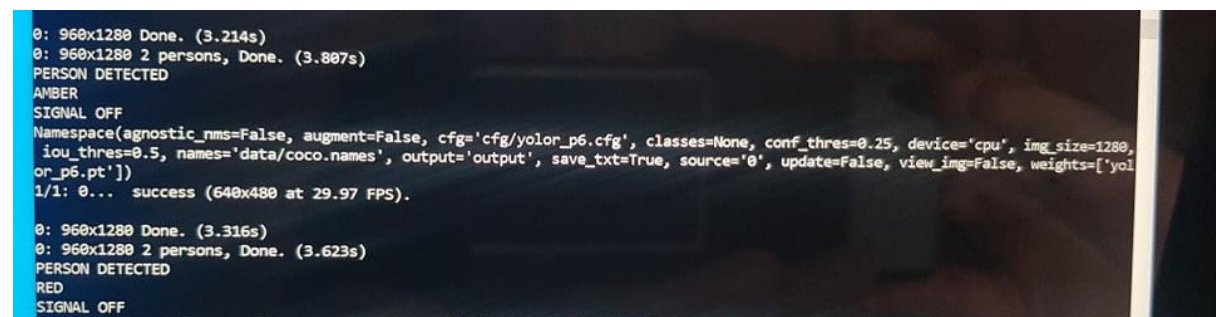


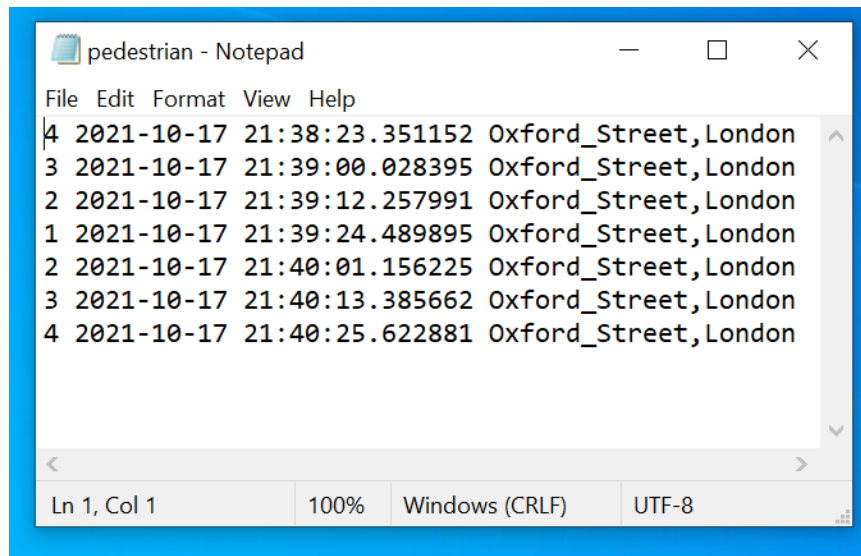
Figure 26: Terminal output at instances where people are detected but lights are 'AMBER' and 'RED' respectively. No signal sent

Continuing with this execution, as the colour display changed to 'GREEN', as well as detecting pedestrians in the plane, a signal is now sent to the digital road sign alerting oncoming drivers to 'SLOW DOWN, PEDESTRIANS CROSSING' as illustrated in figure 27.



Figure 27: Terminal output at instance where people are detected, and lights are 'GREEN', signal sent

We allowed the process to run through from 'GREEN' to 'RED' three times while taking away and adding the number of pedestrians within the plane of sight ensuring the correct number were being detected.



```
File Edit Format View Help
4 2021-10-17 21:38:23.351152 Oxford_Street,London
3 2021-10-17 21:39:00.028395 Oxford_Street,London
2 2021-10-17 21:39:12.257991 Oxford_Street,London
1 2021-10-17 21:39:24.489895 Oxford_Street,London
2 2021-10-17 21:40:01.156225 Oxford_Street,London
3 2021-10-17 21:40:13.385662 Oxford_Street,London
4 2021-10-17 21:40:25.622881 Oxford_Street,London
Ln 1, Col 1 100% Windows (CRLF) UTF-8
```

Figure 28: ‘pedestrian.txt’ output file listing instances of pedestrians detected when lights are ‘GREEN’

Figure 28 displays the contents of the pedestrian.txt file that was created for saving records of pedestrian detection when lights are ‘GREEN’. We confirmed that the script did not save irrelevant data, i.e., pedestrians detected at ‘RED’ or ‘AMBER’ lights as well as 0 pedestrians detected when lights are ‘GREEN’. Our script successfully counts the correct number of objects in each instance and translates this to a readable output. The file also includes information with regards to the time, date, and location of road crossing.

To note, we experienced a minor problem with light contrasting. When the IoT device begins recording at each interval (from the window in the background). Any rush of light obscures the device’s ability to detect all objects at the forefront of its plane of sight. We therefore directed the camera at an angle that’s more towards the road layout to reduce this contrast.

6.1.7 - Brief overall note of results and comparison with real person

A notable set back found when recording the certainty results was the software’s tendency to not detect pedestrians that are positioned within the plane of sight. Various factors that could cause this were:

- 1) The pedestrian designs
- 2) The camera resolution
- 3) Contrast
- 4) Anomalous objects

Although detected with sufficient confidence, some results were unexpectedly lower, again, possibly due to the factors mentioned. Therefore, at this point we felt it was important to test the software against real people. As this investigation was undertaken during the COVID-19 pandemic, we resorted to testing an image still.



Figure 29: A crowd of live music goers detected using the YOLOr software [1]

The figure above is an image of a crowd at a music gig. We applied the software to the image which outputs bounding boxes to each detected person. Members in the foreground typically have higher detection certainties (~ 0.73) to those positioned midway (~ 0.62) and those towards the back (~ 0.44) or considerably obstructed (~ 0.32). The man displayed in the bottom right corner of the image has not been detected indicating that his positioning is too close but not because he is out of focus, as people out of focus by a greater degree are detected. When comparing these certainties to our setup for both various numbers of pedestrians and distance-to-target, it would suggest that our results follow a similar trend, however images of real-life people compared to our model show proportionally higher certainties. The fact that smaller, more out of focus, or more obstructed people were detected in this image suggests that better image resolution improves detection rate.

6.1.8 - Transferring data to the cloud

Appendix P displays the output of our .JSON file when we applied the 'JSON converter' script to our 'pedestrian.txt' file. We had to slightly rearrange the output so that only one row of data was displayed per line. A future study could look at how we can automate this within the script. This simplified expression of our data sits in our S3 bucket. We then use Amazon Athena to direct our 'pedestrian_crossing' database to extract this data to a table titled 'pedestrian' as captured in *appendix Q*. A terminal within the Athena dashboard allows us to perform SQL queries.

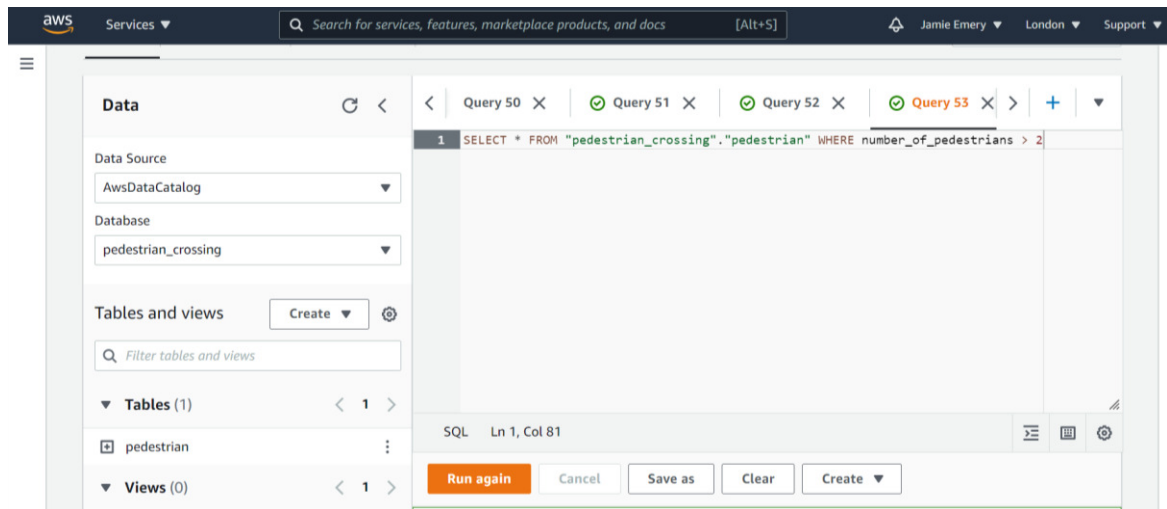


Figure 30: A typical SQL query on our pedestrian table in Amazon Athena

Figure 30 displays a typical query that an endpoint user may perform to find, for instance, the date, time, and locations when the number of pedestrians exceeded 2. The result of this query is illustrated in figure 31.

<div> Completed Time in queue: 0.162 sec Run time: 0.4 sec Data scanned: 1.71 KB </div>			
<div> Results (4) <div>Copy</div> <div>Download results</div> <div>Search rows</div> <div>< 1 ></div> </div>			
number_of_pedestrians	date	time	location
4	2021-10-17	21:38:23.351152	Oxford_Street,London
3	2021-10-17	21:39:00.028395	Oxford_Street,London
3	2021-10-17	21:40:13.385662	Oxford_Street,London
4	2021-10-17	21:40:25.622881	Oxford_Street,London

Figure 31: The table generated from the SQL query in figure 30

As well as improving safety, from a climate perspective, this data could be used to help towns and cities go “greener”. The data could be used to find the regions that have consistently high pedestrian walkers for adding vehicle congestion charges to encourage these drivers to travel by pavement instead of road.

The root user of this AWS account can also keep track of bills incurred from running queries per TB of data as well as costs for storing the objects in S3.

7. Conclusion

This dissertation describes how we were able to achieve the objectives set in section 2. We achieved the goal of demonstrating a means of improving pedestrian safety at road crossings. This project achieves the objectives set out at the beginning of the investigation by providing a comprehensive testing of an emerging technology within a model that can help improve the safety of pedestrians. Using an IoT device and automated processing when executing a python script within an edge network, we demonstrated how oncoming traffic, via digital road signs, can be alerted that pedestrians are crossing the road when traffic lights are green. We were able to demonstrate how this process does not require any human input once the initial command is executed.

This project also seeks out and demonstrates effects and limitations on the device's performance when varying the IoT's position, as well as varying the population and positioning of the targets. The setup worked well when finding an average detection certainty for each pedestrian count. Results confirmed the device's reliability in detecting various numbers due to minor fluctuation in averages. However, limitations begin to show when pedestrians become partially obstructed from one another and certainty decreases. We were able to find the optimum angle and height of the IoT device, relative to pedestrians, as well as investigating the effects on detecting objects when varying the distance from the target. Clear trends were found when altering the IoT's position along these different axes. Results showed that an angle between 45 and 50 degrees provides the best detection performance. Also, the closer the device is to the target, the better the performance, however the minimum distance is relative to the number of pedestrians that must fit the IoT's field of sight. This could be overcome by using a multi camera system that works in parallel so that a real-world setup has the advantage of maximising detection certainty without having to compromise on unnecessarily moving the cameras away from the targets. Additionally, as the pedestrian count increased, our results showed that increasing the height of the IoT device improves performance by reducing the effect of obstruction.

Our next objective was achieved by creating another python script within the edge network, that can extract information about pedestrians that cross the road on a green traffic light. From there we were able to upload the information to AWS on an Amazon S3 bucket as well as demonstrate how endpoint users are granted access using IAM and perform analysis using Amazon Athena.

Our results also showed that, although the performance trends correlate to that of a real-life image of people, the certainties were impacted. A possible solution for this (for investigation in a future project) is to use a camera with better resolution and frame rate as well as arranging the model in a real pedestrian crossing setup. This project was also limited to capturing stills in real-time. Future work might look at the effects on detection certainties when pedestrians are in motion. We also carried out the demonstration indoors. Future work could look at the effect of running the object detection software at various daylight/night-time hours to help further investigate the reliability of the IoT device.

Collectively, these results provide evidence for the key components of a model as well as its details on performance optimisation which has the potential to improve pedestrian safety. As we begin to introduce deadlines on cutting carbon emissions, there's a much higher campaign for travellers to walk on foot/disability aid where suitable making this project a timely solution to help protect the public.

The process described in this project is universally scalable. If introduced for real world application, this setup can be scaled from one or a group of crossings all the way up to a countrywide network of devices, all working within their edge network but sending useful information to one receiving point in the cloud. As demonstrated within AWS, access can be granted to both individual users and/or the public for analysis. This analysis can further improve safety by highlighting road crossings that have high pedestrian numbers crossing at certain times which may trigger the need to introduce more crossings or improve road layouts.

To make this model suitable for real-world application, there needs to be improvement on the IoT device's camera so it can capture video in higher resolution. This will increase its range of distances in which it can detect pedestrians with higher certainty. As this setup would be outdoors, the device must work for all weathers. The process itself must pass regulations with authorities, such as GDPR, to ensure there's no data breach when sending information to the cloud. There must be some degree of awareness made to the public that this system is in place, perhaps, introducing the digital sign type to driving theory tests. The data may require encryption when transferring from the edge network to the cloud to protect the information against unpermitted users. This can be done within AWS when setting up the S3 bucket (receiving point). The data is then automatically de-crypted when accessed by permitted users. Alternatively, to limit this threat to data theft, the data could be simplified by simply sending a 0 or 1 if a person is detected. This method, however, has the disadvantage of not specifying the number of pedestrians for SQL queries in the cloud. Future work could investigate the software's performance limitations when varying the brightness which relates to different times of the day, both day and night. Different weather conditions such as heavy rainfall could also be introduced as a variable to test against the IoT device's ability to detect pedestrians.

8. Future Work

8.1 - Pavement Pedestrian Detection

Whereas this project proposes and demonstrates how pedestrian safety can be improved when travelling across a road crossing, it does not consider the safety of pedestrians at roadside. A possible study could look at how we improve the safety of pedestrians standing at either side of the crossing by modifying the model in this project. The YOLOr software uses machine learning to detect objects. In section 4, we learnt that it divides an image into sections to predict bounding boxes on objects. We could find a method of training the software to distinguish and detect the difference between pedestrians standing roadside to those using the road crossing as illustrated in figure 31.



Figure 32: An image of a road crossing in Cardiff divided into three sections

The image would be divided into three segments for training. For the two segments highlighted in red, we would train our machine learning model to detect pedestrians that are standing still or have minimal movement within a certain degree. The segment in the middle would be trained to detect pedestrians that are on the move and have intention to cross the street.

But how would this improve the safety of pedestrians that aren't in the direct obstruction to oncoming traffic? If this is during peak work commute hours for instance, the number of pedestrians waiting to cross may exceed area allocated for them to do so. This could risk an overflow which would either encourage pedestrians to cross at a green traffic light or be unintentionally pushed into the road. Our edge network system would identify when this scenario occurs and subsequently change the traffic lights to red and/or alert drivers that there's a high number of pedestrians at roadside. The middle segment would act accordingly to the demonstration provided in this project.

8.2 - Detecting Anomalies in the Road

We briefly touched on introducing an anomalous object when finding the effect it has on the IoT's ability to detect pedestrians. A future investigation could look at training the YOLOr software to detect anomalies within the pedestrian object.



Figure 33: An example of trained software detecting the elderly lady's fall related accident

Figure 32 illustrates a scenario where an elderly woman has fallen over. In this project, our model would detect the pedestrian but would not distinguish whether they're moving or not. The model in this future investigation could train the YOLOr software to distinguish between moving and motionless or barely moving pedestrians in the road. The subsequent response could be to alert drivers or change the traffic lights to red. The investigation described in this sub section follows on very well from section 8.1. It goes one step further to detect moving and non-moving pedestrians in all three segments.

This method of training for anomalies also has the potential to improve the safety of oncoming drivers. Any anomalous objects such as car debris, roadkill or objects dropped by pedestrians could put drivers at risk from accidents involving aversion or tyre punctures. This future investigation could train the software to detect these types of objects as well as fallen or motionless pedestrians. The edge network can then alert these drivers to slow down.

8.3 - Signal Communication with Smart Cars

Within our edge network, the IoT device communicates with a digital road sign to alert oncoming traffic that pedestrians are crossing. A future project might focus on finding other communicative methods that could bypass a digital road sign and be sent directly to the vehicle via the user interface screen. Not only could information on object types detected be alerted directly to the driver but other information such as:

- The chances that traffic is congested due to the number of pedestrian commuters using road crossings
- Information on pedestrians late at night that could be under the influence of alcohol crossing roads within their driving route
- "Greener" areas that drivers should avoid driving within as it's highly populated with on-foot travellers

The communication in this scenario isn't restricted to a car and one IoT device within the edge network. An entire network of devices could feed information to the driver providing useful information

8.4 - A Complete Cloud Architecture

In this project, we used Identity Access Management to grant endpoint users permissions to read and analyse data uploaded to Amazon S3 while stored in a database on Amazon Athena. These are just three of the fundamental resources that make cloud computing a beneficial use for this model. Future work made by researchers or industry professionals might look at how we can include cloud architectures to make even better use of the information received from this model. In subsection 8.3 we discussed how this data could be communicated directly to drivers via their in-built user interface screens. We could use AWS to distribute information within a pedestrian's current location region to their smartphone with updates on "pavement-traffic" to help avoid busy crowds of people: a particularly good use during a pandemic.

8.5 - Vision of a Smart City

The work discussed in sub section 8.4 precedes this section well having looked at improving the safety of both pedestrians and drivers. A smart city uses a collection of connected IoT devices to monitor and extract data using a range of different sensors. So how else can we use IoT devices within a smart city to improve the safety and wellbeing of people?

8.5.1 – Construction

Construction is a major part of any city with constant improvements and renovations made to buildings. IoT devices could be introduced to improve the safety of workers by identifying events in real time that could prevent serious injuries or even death. An IoT device that uses machine learning object detection could be trained to identify safety risks and breaches on site. For example, a hazardous tool left on the floor or piece of equipment precariously positioned is detected by the device, subsequently sending an alert to the site manager who can act accordingly to avoid accident, injury, and death.

8.5.2 – Medicine

IoT is already very commonly used in medicine, most prominently for monitoring patients remotely. An example use case is for cardiologists to identify heart arrhythmia. An IoT implant is surgically inserted into the patient and over time it will monitor and send information on heart rhythm to the doctor for analysis that they wouldn't otherwise be able to do in an in-person appointment with the patient. Future work could look further at innovative ways in which IoT devices can be used to remotely monitor a patient with respect to other organs.

9. Self Reflection

Having taken Dr Padraig Corcoran's module: 'Distributed and Cloud Computing' and having a keen interest in cloud computing and edge networks, I wanted to use this opportunity to work with him on a possible follow-on project from a previous publication. This publication looked at 'modelling pedestrian safety with respect to road traffic crashes by estimating the safety of paths' (Corcoran, 2018), making this a suitable extension to his research by looking at pedestrian safety from a different angle by focusing on road crossings. I've thoroughly enjoyed investigating and reporting this project because it finds a solution to help improve of pedestrian safety.

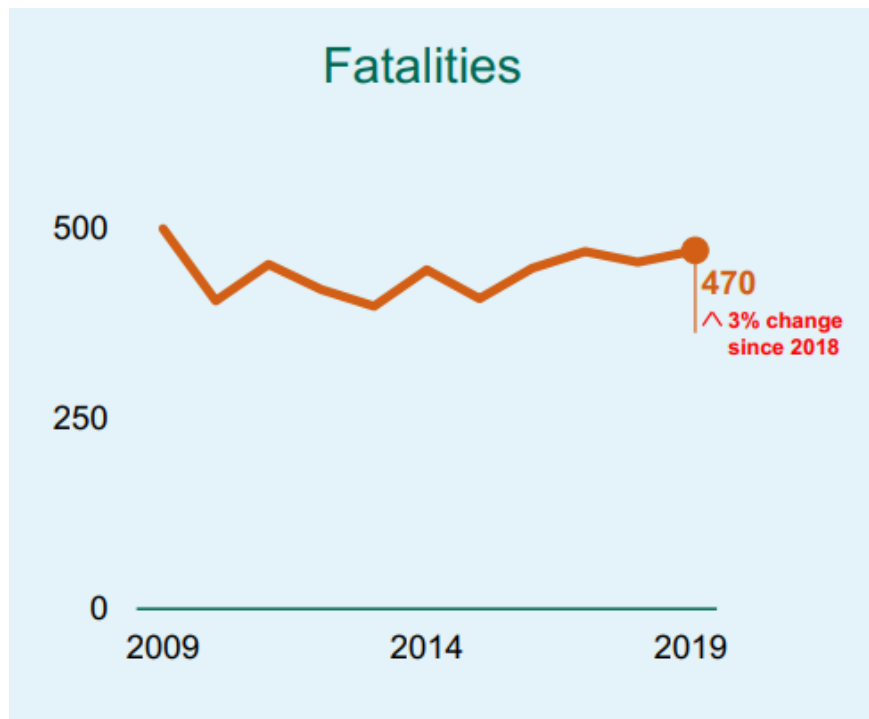
When starting this project, I wanted to ensure that its goals passed each criterion specified in the SMART objectives model. This stands for: Specific, Measurable, Achievable, Realistic and Time frame. We weren't trying to improve pedestrian safety by covering all regions of the road. Instead, we chose to focus on a specific road feature in which pedestrians often put themselves in danger when using incorrectly. Our primary task, to construct a process that would alert oncoming drivers when pedestrians are crossing on a green light, wasn't a quantitate objective. But by investigating the performance limitations of the object detection software, we could set ourselves a measurable objective. To make this project achievable, I wanted to build on the skillset that I had developed through the learning side of this course. For example, using Python to automate the communicational process between IoT device and digital road sign, and computational thinking to abstract irrelevant output information within the .txt file. I think these examples cross over partially with the criteria that the project is realistic. When using the YOLOr software, it was very much off the shelf as the ability to train the software to detect new objects or traits within objects is not currently within my skillset and easily learnt within the time given to complete the project. I also didn't have the resources to train a deep learning model as this would require a graphics processing unit (GPU). A GPU can take days if not weeks to train a deep learning model which would add further issues with time. In terms of the time frame criteria, an important skill I practiced in this project was time management. With the scope of the task, I found it useful to allocate deadline goals and keep to them to the best of my ability. Having started a new job that overlapped with this course, I had to remain, somewhat, strict between time each allocated to the job and the project, but this did prove a difficult task.

Throughout the project, I had weekly meetings with Padraig to discuss progress and planning for upcoming stages. It was useful to have these discussions to brainstorm ideas on what variables and methods sufficiently test the performance of the computer vision-based object detection software as well as the possible future projects that could precede this work, some of which I'd be very keen to be involved with. One method that I used to advance the project was using iterative processing. I first worked towards achieving a minimum viable process that successfully achieves the goals of this objective. I then looked back at how this process could be improved. For example, the output of the 'pedestrian.txt' file. As well as the useful pedestrian information, I had some background terminal output generated by the software that would have no use for uploading to the cloud or to be analysed by end users. With that removed, I now had the number of pedestrians detected at a date and time. I wanted to think about what else also could be useful for analysis or to keep better record of which IoT device within the network this occurred. That's where I decided to also include location as another output. The discussions with Padraig were iterative in the sense that we built on ideas during each meeting, and it encouraged me to think outside the box, especially when writing the future work section. For the future section, I wanted to find topics for research on additional IoT use cases such as improving driver safety.

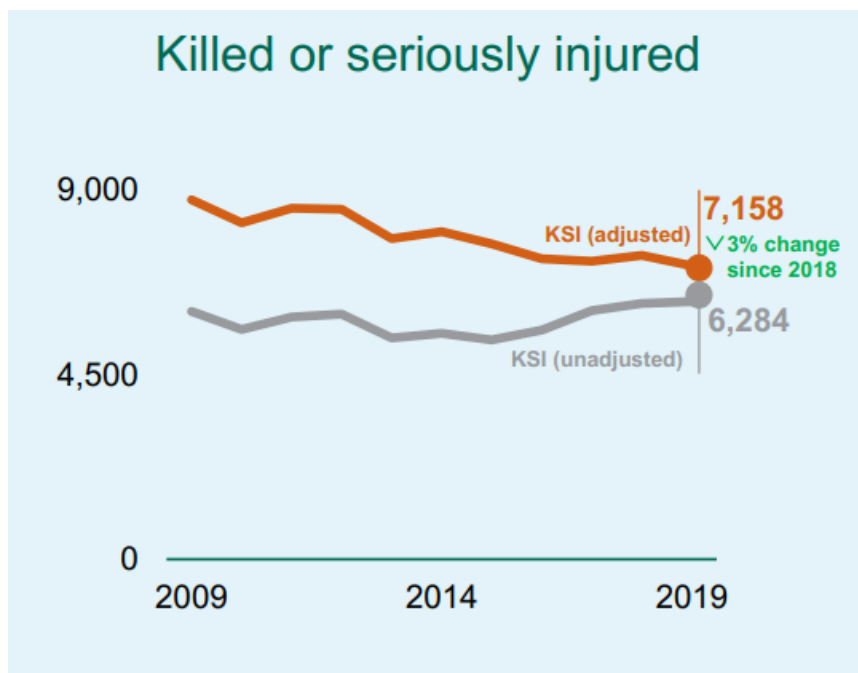
I decided to write this dissertation at the same time as creating and carrying out the model demonstration as each element provided new inspirational ideas for the other component. I would describe the writing and demonstration components as a collaborative process. The table of contents acted as the plan when writing this dissertation which was helpful given the size of the piece of writing. It would allow me to add notes and ideas conjured throughout the project. I also made good use of the comments to remind myself of things that should be added. I wrote the aims and objectives just before I started carrying out the demonstration. It provided me with a clear idea as to what outcomes we should be achieving having completed the practical side.

I believe this project lays good fundamental groundwork for using computer vision-based object detection and machine learning to provide a solution that improves pedestrian safety. I hope it can encourage peers to continue working on this solution for it to be, one day, applied to the real world.

10. Appendices



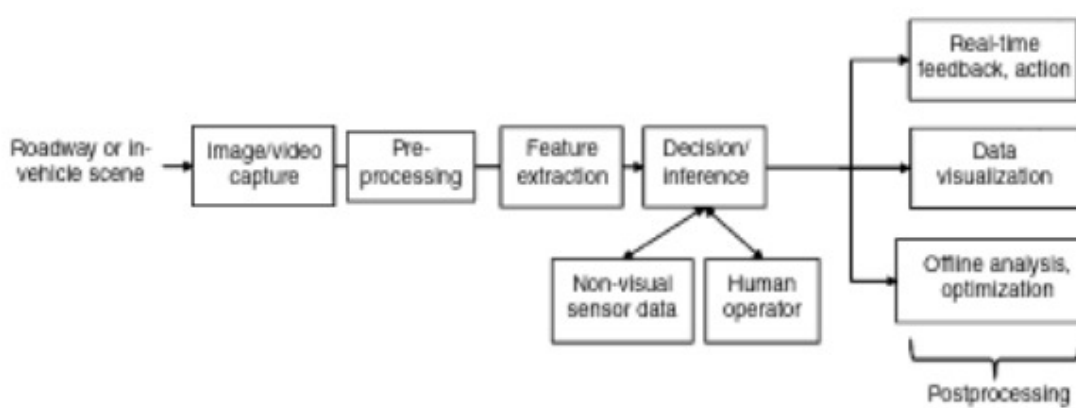
[A]



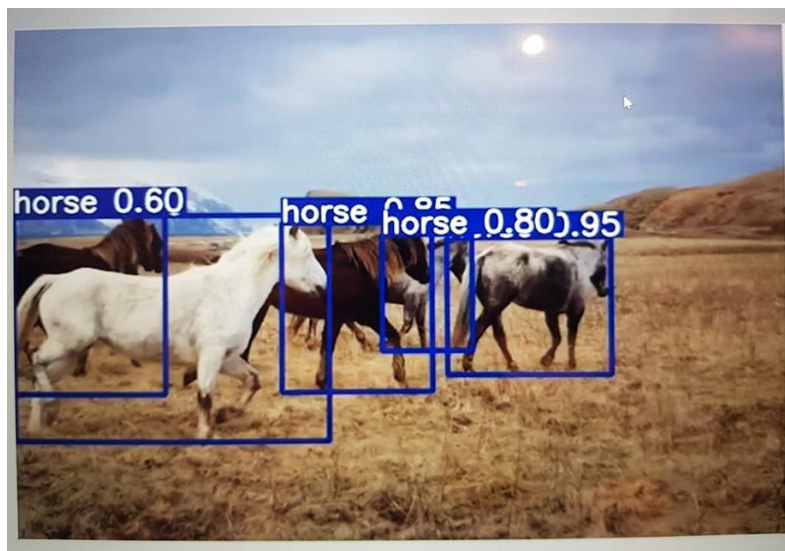
[B]

Layers	Representation	Functionalities
Business	Graphs, Flowchart, Table, Diagram	<ul style="list-style-type: none"> • Business model and investment designs • Resource usage and application pricing • Budget preparation, data aggregation
Application	Smart applications for vehicles and vehicular dynamics	<ul style="list-style-type: none"> • Smart, intelligent services to end users • Service discovery and integration • Application usage data and statistics
Artificial Intelligence	Cloud computing, big data analysis, expert systems	<ul style="list-style-type: none"> • Storing, processing, analysis of data • Analysis based decision making • Service management based on profit
Coordination	Heterogeneous Networks: WAVE, WiFi, LTE	<ul style="list-style-type: none"> • Unified structure transformation • Interoperability provisions • Secure transportation of information
Perception	Sensor and actuator of vehicles, RSU, personal devices	<ul style="list-style-type: none"> • Data gathering: vehicle, traffic, devices • Digitization and transmission • Energy optimization at lower layers

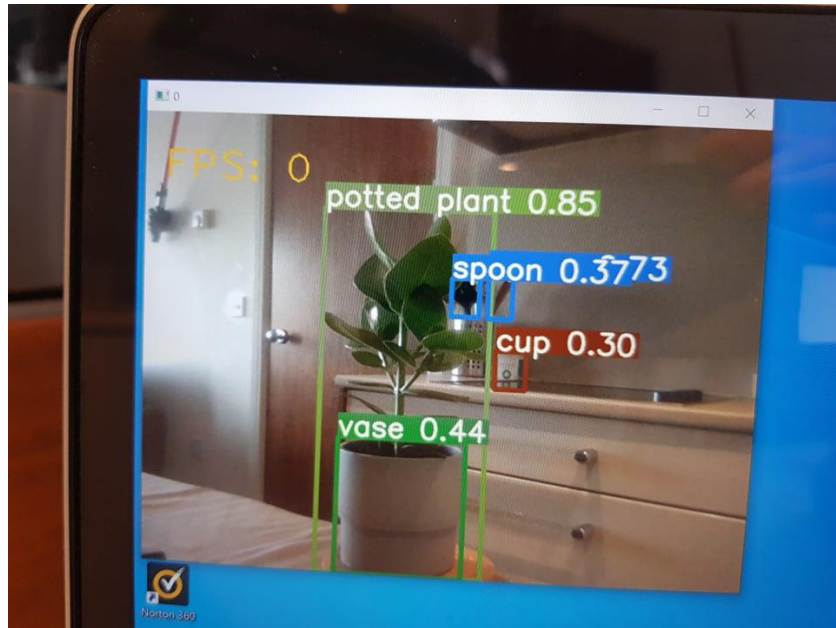
[C]



[D]



[E]



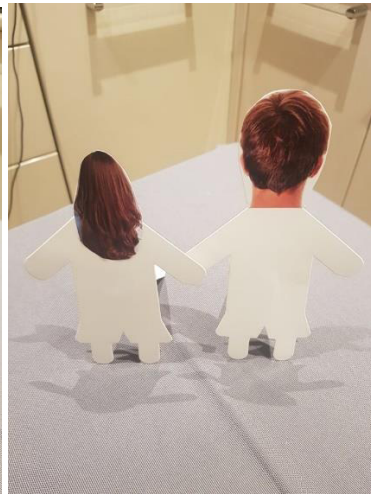
[F]

0: 960x1280 1 cups, 2 spoons, 1 potted plants, 1 vases, Done. (3.210s)

[G]



[H.1]



[H.2]



[H.3]


```

json_converter.py > ...
1  import json
2
3  filename = 'pedestrian.txt'
4  fields = ['number_of_pedestrians', 'date', 'time', 'location']
5
6  with open(filename) as fh:
7
8      out_file = open("test2.json", "w")
9
10     for line in fh:
11         description = list(line.strip().split(None, 3))
12
13         description[0] = int(description[0])
14         description[1] = str(description[1])
15         description[2] = str(description[2])
16         description[3] = str(description[3])
17
18         dict2 = dict(zip(fields, description))
19
20         json.dump(dict2, out_file, indent=4)
21         out_file.write('\n')
22
23     out_file.close()
24

```

[J]

Dashboard
Access management
User groups
Users
Roles
Policies
Identity providers
Account settings
Access reports
Access analyzer

IAM > Users

Users (1) Info
An IAM user is an identity with long-term credentials that is used to interact with AWS in an account.

<input type="checkbox"/>	User name	Groups	Last activity	MFA	Password a...
<input type="checkbox"/>	Department_for_Transport	Endpoint_Users	Never	None	1 minute ago

Permissions policies (2 policies applied)

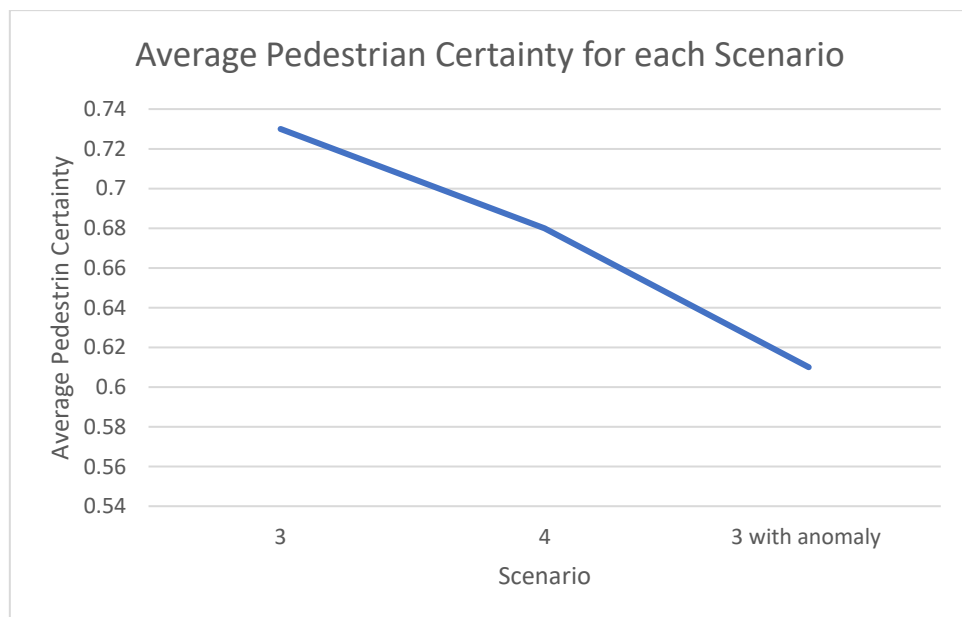
Add permissions
Add inline policy

Policy name	Policy type
Attached directly	
IAMUserChangePassword	AWS managed policy
Attached from group	
AmazonS3ReadOnlyAccess	AWS managed policy from group Endpoint_Users

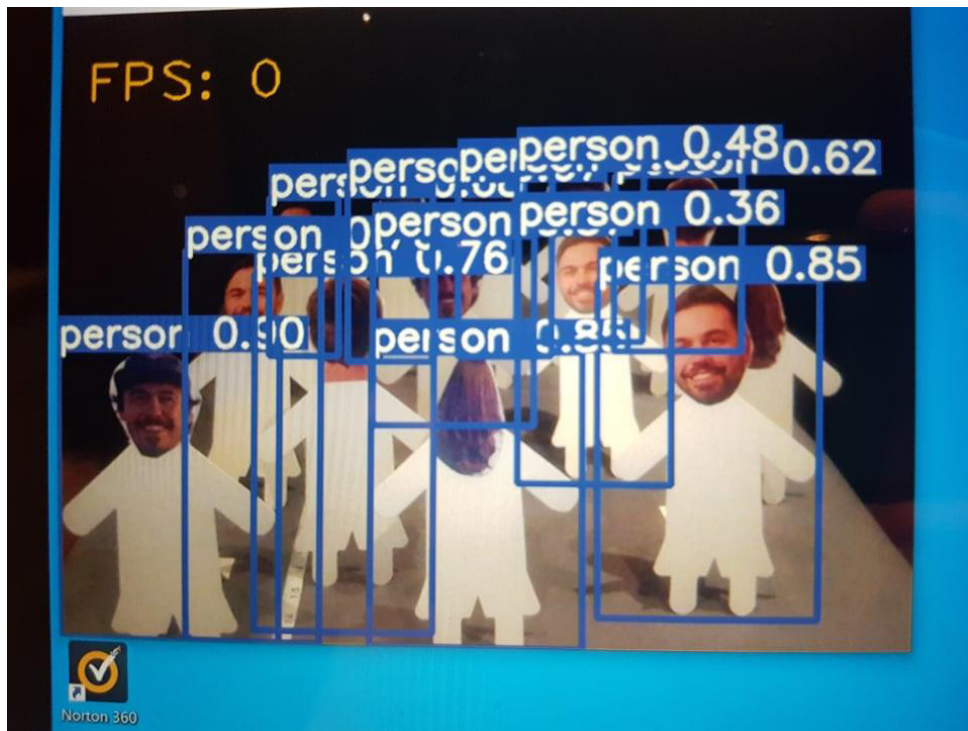
[K]



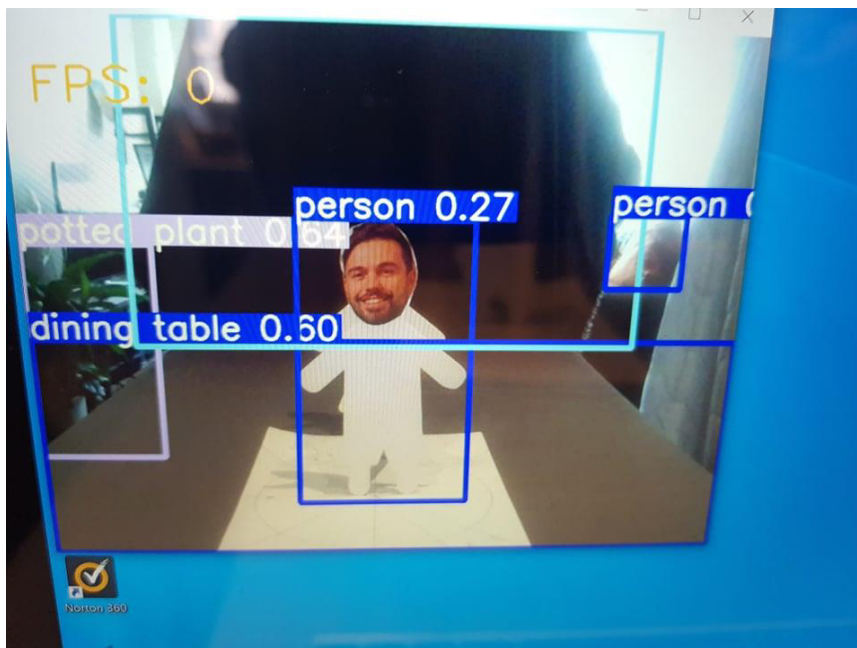
[L]



[M]



[N]



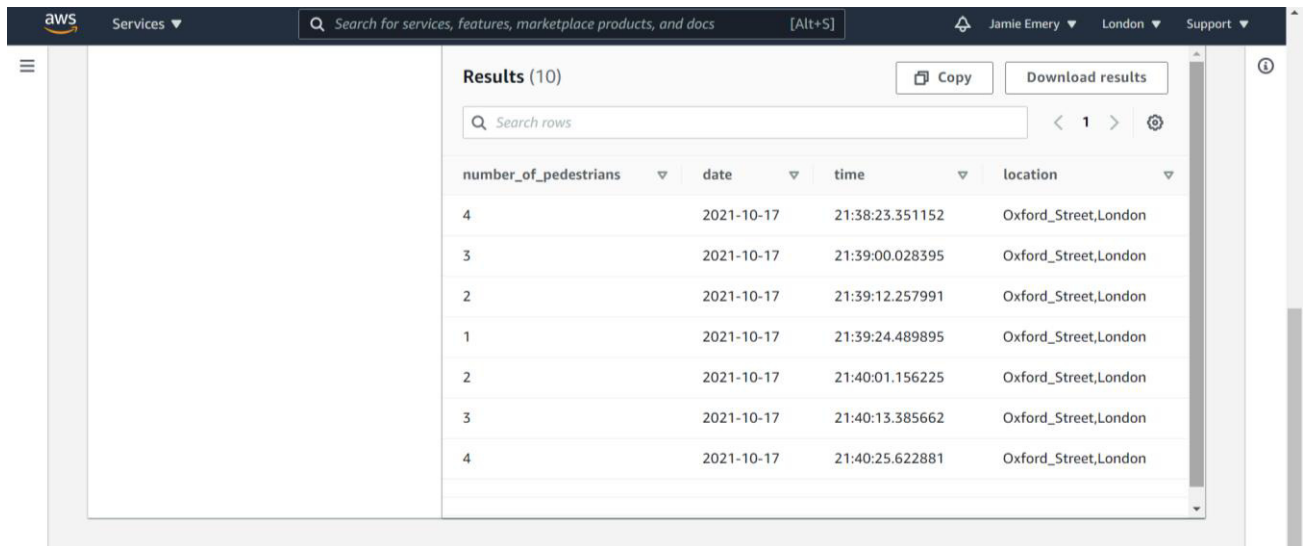
[O]

```

{} pedestrian.json > ...
1  {"number_of_pedestrians": 4,"date": "2021-10-17","time": "21:38:23.351152","location": "Oxford_Street,London"}
2  {"number_of_pedestrians": 3,"date": "2021-10-17","time": "21:39:00.028395","location": "Oxford_Street,London"}
3  {"number_of_pedestrians": 2,"date": "2021-10-17","time": "21:39:12.257991","location": "Oxford_Street,London"}
4  {"number_of_pedestrians": 1,"date": "2021-10-17","time": "21:39:24.489895","location": "Oxford_Street,London"}
5  {"number_of_pedestrians": 2,"date": "2021-10-17","time": "21:40:01.156225","location": "Oxford_Street,London"}
6  {"number_of_pedestrians": 3,"date": "2021-10-17","time": "21:40:13.385662","location": "Oxford_Street,London"}
7  {"number_of_pedestrians": 4,"date": "2021-10-17","time": "21:40:25.622881","location": "Oxford_Street,London"}
8

```

[P]



Results (10)

Search rows

number_of_pedestrians	date	time	location
4	2021-10-17	21:38:23.351152	Oxford_Street,London
3	2021-10-17	21:39:00.028395	Oxford_Street,London
2	2021-10-17	21:39:12.257991	Oxford_Street,London
1	2021-10-17	21:39:24.489895	Oxford_Street,London
2	2021-10-17	21:40:01.156225	Oxford_Street,London
3	2021-10-17	21:40:13.385662	Oxford_Street,London
4	2021-10-17	21:40:25.622881	Oxford_Street,London

[Q]

11. References

- [1] Augmented Start-ups, 2021, 'YOLOr Pro', <https://www.augmentedstartups.com/YOLOR-You-Only-Learn-One-Representation-PyTorch>
- [2] AWS Public Sector, 2020, 'How do transportation customers use AWS for improving safety', <https://www.youtube.com/watch?v=jsiU6za4rsM>
- [3] AWS, 2021, 'Amazon Athena: Overview', <https://aws.amazon.com/athena/?whats-new-cards.sort-by=item.additionalFields.postDateTime&whats-new-cards.sort-order=desc>
- [4] BicycleNetwork.com, July 2021, 'New smarter bike detectors to improve crossings', <https://www.bicyclenetwork.com.au/newsroom/2021/07/29/new-smarter-bike-detectors-to-improve-crossings/>, Bicycle Network
- [5] Clark, J., November 2016, 'What is the Internet of Things (IoT)?', <https://www.ibm.com/blogs/internet-of-things/what-is-the-iot/>, IBM
- [6] Department of Transport, The Welsh Office, The Scottish Office, The Department of the Environment for Northern Ireland, 1995, 'The Design of Pedestrian Crossings', The Stationery Office, Crown Copyright 1995
- [7] - Department for Transport, 2015, 'Facts on Pedestrian Casualties', https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/448036/pedestrian-casualties-2013-data.pdf, p 1-10
- [8] - Department for Transport, 2019, 'Reported Road Casualties in Great Britain: 2019 annual report', https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/922717/reported-road-casualties-annual-report-2019.pdf p 9-12
- [9] Hannah, C., Spasic, I., Corcoran, P., 2018. 'A computational model of pedestrian road safety: the long way round is the safe way home', p. 5, doi: 10.1016/j.aap.2018.06.004
- [10] Hannah, C., Spasic, I., Corcoran, P., 2018, 'Modelling pedestrian safety with respect to road traffic crashes by estimating the safety of paths', Cardiff University, <http://orca.cardiff.ac.uk/id/eprint/110139>
- [11] Kaiwartya, O. et al. 2016, 'Internet of Vehicles: Motivation, Layered Architecture, Network Model, Challenges and Future Aspects', IEEE Volume 4, p. 5359-5361
- [12] Kumar, S., Tauseef, S. M., 2018, 'Development of an Internet of Things (IoT) based Lockout/Tagout (LOTO) device for Accident Prevention in Manufacturing Industries', IOP Conf. Ser.: Mater. Sci.Eng, p. doi: 10.14419/ijet.v7i3.12.19894
- [13] Loce R. et al. 2017, 'Computer Vision and Imaging in Intelligent Transportation Systems', Wiley – IEEE Ser, p. 8-10
- [14] Manju, L. S., Mithilesh, A., Mitul K. A., Afsar, A., May 2021, 'IoT-enabled cloud-based real-time remote ECG monitoring system', Journal of Medical Engineering & Technology, 45:6, <https://doi.org.abc.cardiff.ac.uk/10.1080/03091902.2021.1921870>

- [15] Munoz, R. et al. April 2018, '*Integration of IoT, Transport SDN, and Edge/Cloud Computing for Dynamic Distribution of IoT Analytics and Efficient Use of Network Resources*', IEEE, p. 1420, doi: 10.1109/JLT.2018.2800660
- [16] Olszewski, P., Szagała, M. Wolański, A. Zielińska, 2015, '*Pedestrian fatality risk in accidents at unsignalized zebra crosswalks in Poland*', p. 89, 10.1016/j.aap.2015.08.008
- [17] Redmon, J., 2016, '*Darknet: Open Source Neural Networks in C*', <http://pjreddie.com/darknet/>
- [18] Thomas, E., Gerster, S., Mugabo, L., Jean, H., Oates, T., 2020, '*Computer vision supported pedestrian tracking: a demonstration on trail-bridges in rural Rwanda*', Plos One, Vol 15, <https://doi.org/10.1371/journal.pone.0241379>
- [19] Volvo, 2020, '*Collision Warning System – Detection of Pedestrians*', <https://www.volvocars.com/uk/support/manuals/v40/2018w17/driver-support/collision-warning-system/collision-warning-system---detection-of-pedestrians>, Volvo
- [20] Wei-Yen, H., Wen-Yen, L., 2021, '*Ration-and-scale-aware YOLO for Pedestrian Detection*', IEEE Transactions on Image Processing, p. 934, doi: 10.1109/TIP.2020.3039574
- [21] WorldData.info, 2021, '*Average height and weight by country*', <https://www.worlddata.info/average-bodyheight.php>