University of Western Australia

GENG5512 Thesis

Detecting Phone Focused Pedestrians For Collision Avoidance in Autonomous Busses

David Gregory 21724425

School of Electrical, Electronic and Computer Engineering

Supervised by

Prof. Thomas Bräunl
School of Electrical, Electronic and Computer Engineering

Word Count: 7529

19 December 2020

1 Abstract

Avoiding pedestrians and preventing collisions are fundamental requirements for the widespread adoption of autonomous vehicles. However, avoiding collisions can injury passengers on-board busses, delaying user acceptance and regulatory compliance.

This project inspects the efficacy of using pose estimation to determine if pedestrians are distracted by their mobile phone. Due to the rise in mobile phone use and injuries to pedestrians using mobile phones, these pedestrians pose the greatest threat to cause injuries from executing manoeuvres to avoid collisions. Mobile phone fixated pedestrians were selected to be the most unpredictable pedestrian type and are the focus of this project. The project is implemented on the UWA renewable energy vehicle (REV) autonomous bus (nUWAy) to increase passenger and pedestrian safety.

The method utilises phone detection pipeline consisting of image conditioning, a pose estimation model and a fast trees classifier for pose joint angles to determine if a pedestrian is distracted on their mobile phone. The fast trees classifier implemented by training hand-labelled images collected by the REV team. Limitations from OpenPose such as phantom and combined poses are corrected in post-processing to allow the pose estimation model to run in tandem with other algorithms on an embedded computer. Robot Operating System (ROS) is used as the middleware to communicate to the onboard cameras, display the algorithm on nUWAy's displays and output the presence and locations of pedestrians on their mobile phones.

Acknowledgements

I would like to first and foremost thank my supervisor, Professor Thomas Bräunl, for the opportunity to work on such an exciting and challenging project. Many thanks to Mr. Thomas Drage, Mr. Craige Brogle, and the rest of the REV Autonomous Team for their support and friendship over the project. I am fortunate to have been able to learn so much from this experience.

Contents

List of Figures

List of Tables

2 Introduction

2.1 Background

Autonomous vehicles have the potential to fundamentally change our modern society by alleviating problems associated with congestion and limited parking, providing drivers with more free-time, potentially reduce fatalities by 90% [1]. However, one of the largest barriers for the adoption of autonomous vehicles is the ability to demonstrate that they are safer than human drivers [2] as edge cases not seen before and unexpected behaviour of other non-autonomous road users becomes a larger risk than to human drivers [3].

Part of the University of Western Australia's (UWA) contribution to the future of autonomous vehicles is through the Renewable Energy Vehicle (REV) team's nUWAy bus. The nUWAy bus is a modified EasyMile EZ10 bus with additional computational power from an Nvidia AGX Xavier and other safety features. It aims to be autonomously operated on the UWA campus to transport students across campus.

As there are safety considerations for busses that are not present for cars, it is important to make the bus as safe as possible for passengers and pedestrians alike. One of the largest bus-specific form of injury is non-collision injuries [4].

Non-collision injuries can result from hard accelerating and/or sudden braking. Sudden braking is the highest risk of injury for bus passengers aside from collisions. In a 2019 study, it was found that 62% of falls from driver manoeuvres were from hard braking [4]. With respect to the nUWAy project, hard acceleration is unlikely to be a source of potential non-collision injuries due to a low maximum speed that the bus will operate at. Falls from hard braking are a potential hazard as the bus may require hard braking to avoid crashing. Therefore, reducing hard braking from avoiding pedestrians is an important consideration to protect the passengers on board the bus.

In past two decades, there has been a rise in the number of injuries of pedestrians using mobile phones [5] [6]. University students are particularly at risk to being injured as they are more likely to be distracted by mobile phones leading to a higher rate of collisions with vehicles [7]. This poses a problem to autonomous busses as now on university campuses are people walking around on their mobile devices not actively seeking out the avoid a collision with the bus. Therefore, it is crucial to be able to identify these pedestrians and provide this information to the autonomous systems on the bus.

2.2 Aim

The aim of the nUWAy project provides an avenue to create models to predict pedestrian intentions to determine if they are distracted and to apply them to a dense mixed-use road in order to increase the safety for passengers and pedestrians. Mixed-use roads are roads that are cohabited by pedestrians and vehicles alike. It also allows further contribution to improving the situational awareness for autonomous vehicles in a lower risk environment due to the slower speed of the shuttle bus compared to cars on urban roads. This project provides a novel solution to detecting pedestrians who are distracted by their mobile phone in computationally constrained environment.

3 Literature Review

3.1 Machine Learning

3.1.1 Deep Learning and Neural Networks

Artificial Neural networks (ANNs) try to mimic how biological neurons interact but instead ANNs activate other neurons with some computational logic function [8] [9]. ANNs are the foundation of deep learning. The first layer of an ANN is the input layer, the last layer is the output layer and the n layers in between are hidden layers. Figure 1 shows the basic structure of an ANN.

In a fully connected neural network, all of the neurons are connected to each of the neurons in the layer ahead and behind it. The values of these neurons are some mathematical function of the neurons that connect to. Often each of the connections will have some weighting which is adjusted by the network as it trains. To train the network, the neuron weights are intialised and the first set of input values are sent through the input layer, through the hidden layers and the output (predicted value) is compared to the labels or classes (true value). The difference is back-propagated through the network, adjusting the weights of the neurons such that it minimises the difference between the predicted and true value. This forward and backward propagation cycle is repeated for the whole training time. The resulting structure of the hidden layers is nonsensical to a human but allows the ANN to learn what features of the data correspond to what classes.

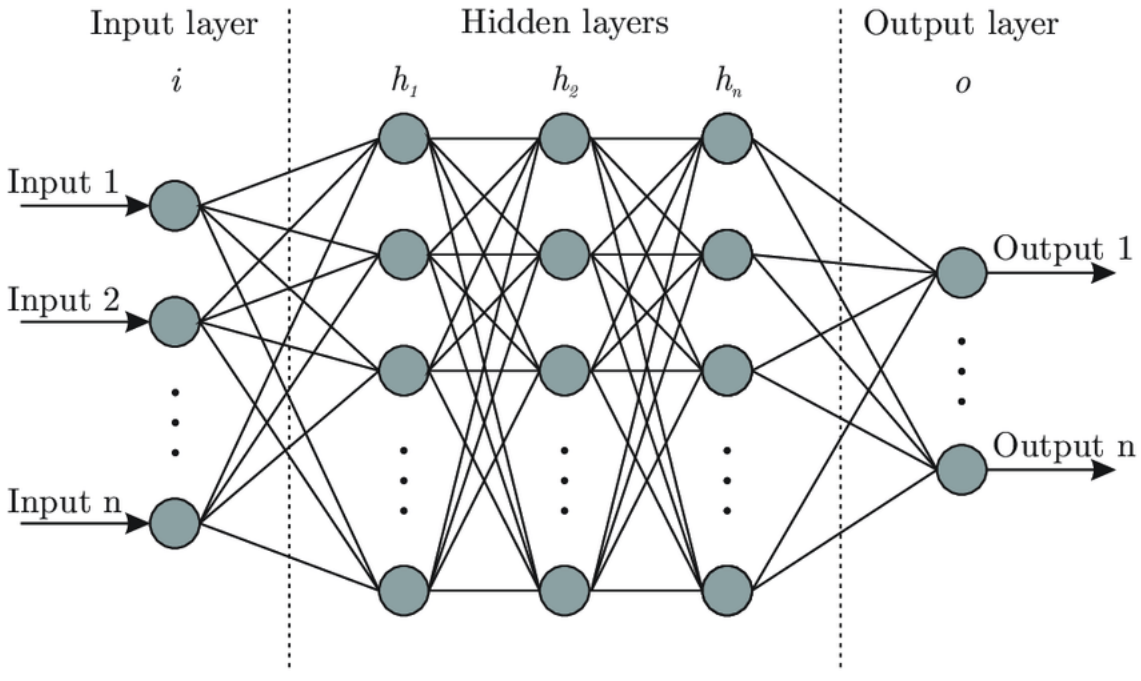

Figure 1: Basic artificial neural network structure. From [10].

3.1.1.1 Convolutional Neural Networks

An especially powerful application of ANNs are Convolutional Neural Networks (CNNs) which specialise in object and image recognition due to their special type of hidden layers. These hidden layers are convolutional layers, pooling layers and fully connected layers. The convolutional

layers scan through the input data creating feature maps or filters that describe the image. One filter may be to describe the horizontal lines and it will quantify this by how "horizontal" the pixels in its receptive field are. This convolutional layer is then pooled to reduce the resolution, and therefore, computational complexity, and it also strips away location dependence of the CNN. That is, the pooling layer allows the CNN to recognise the feature maps no matter where they are on new images. The fully connected layer equips the CNN with the ability to fully connect all the information that forms the image [8].

CNNs have come a long way since their inception and there are now many pre-trained networks trained on many large datasets. These networks include VGG-19 [11], Multiped-500 [12], and Inception-v2 [13] which are trained on datasets such as Common Objects in Context (COCO). The CNN architecture of VGG-19 is seen in Figure 2 where the green layers are the convolutional layers, the blue are the pooling layers, and the purple are the fully connected layers. The receptive field sizes for VGG-19 are 3x3 convolutional filters.

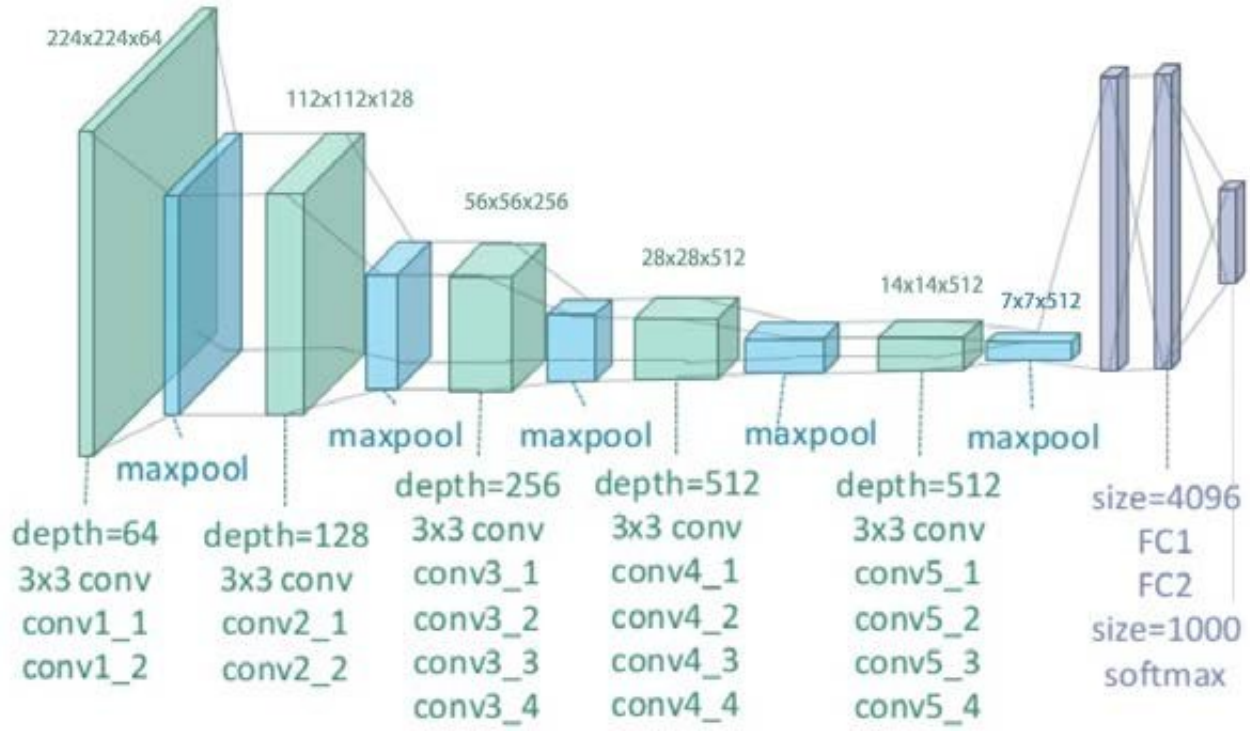

Figure 2: Structure of the VGG-19 CNN. From [14].

3.1.1.2 OpenPose

There are numerous sophisticated pose detection algorithms available currently, each with its own implementation. Pose detection uses CNNs to determine the presence and location of body parts in an image. The actual methodology varies between the pose detection networks. OpenPose is state-of-the-art pose detection algorithm developed by [15]. It uses a top-down approach using part affinity fields (PAFs) which allows the algorithm to track which limbs belong to which people. OpenPose uses the VGG-19 CNN [11] as its convolution process. A massive advantage of OpenPose compared to other pose detection algorithms is its run-time invariance to the number of people in the image.

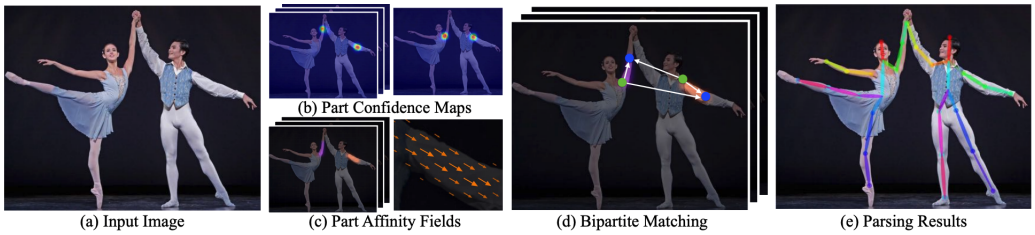

Figure 3: The OpenPose pipeline. (a) Takes an entire image into the CNN (b) generation of confidence maps (c) PAF generation (d) bipartite matching to associate body parts (e) final assembly of the full body onto the image. From [15].

3.1.2 Decision Trees

Decision trees are a type of regression and classification algorithm that segments classes based on the values for different features. This provides it with the ability to predict a class based on a certain set of feature values. Decision trees begin with a root node and make binary decisions from a set of features as to the class of the features presented. The branches are extended till all the decisions are exhausted. An example decision tree with the iris dataset [16] is shown in Figures 4 and 5 where the binary decisions can be seen on each node depending on the feature values.

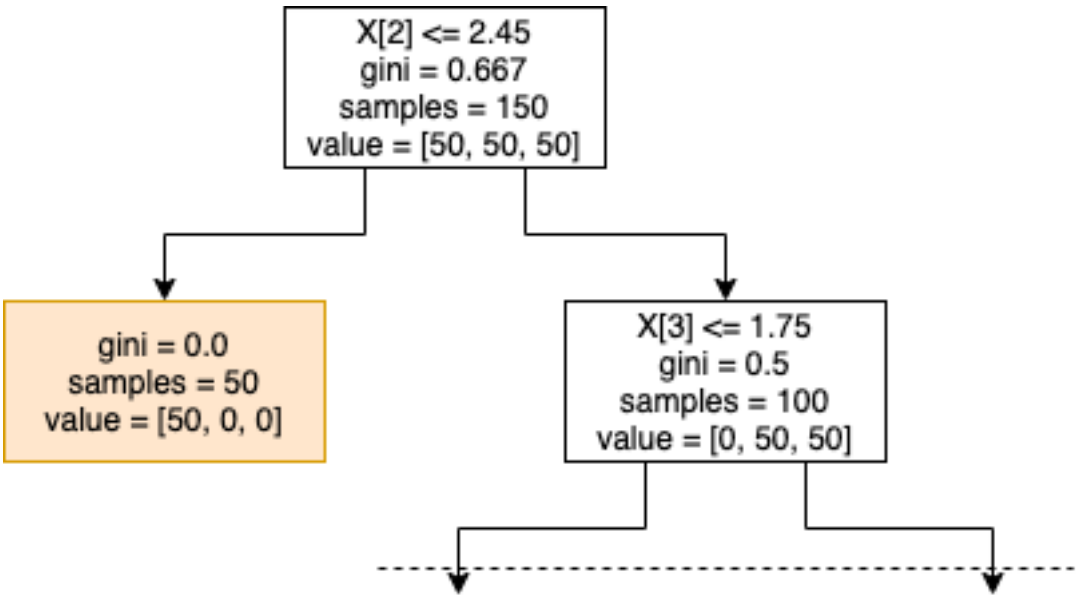

Figure 4: A decision tree on the iris dataset visualised. Note, the rest of the tree has been cropped below the dotted line to make it easier to visualise. From [17].

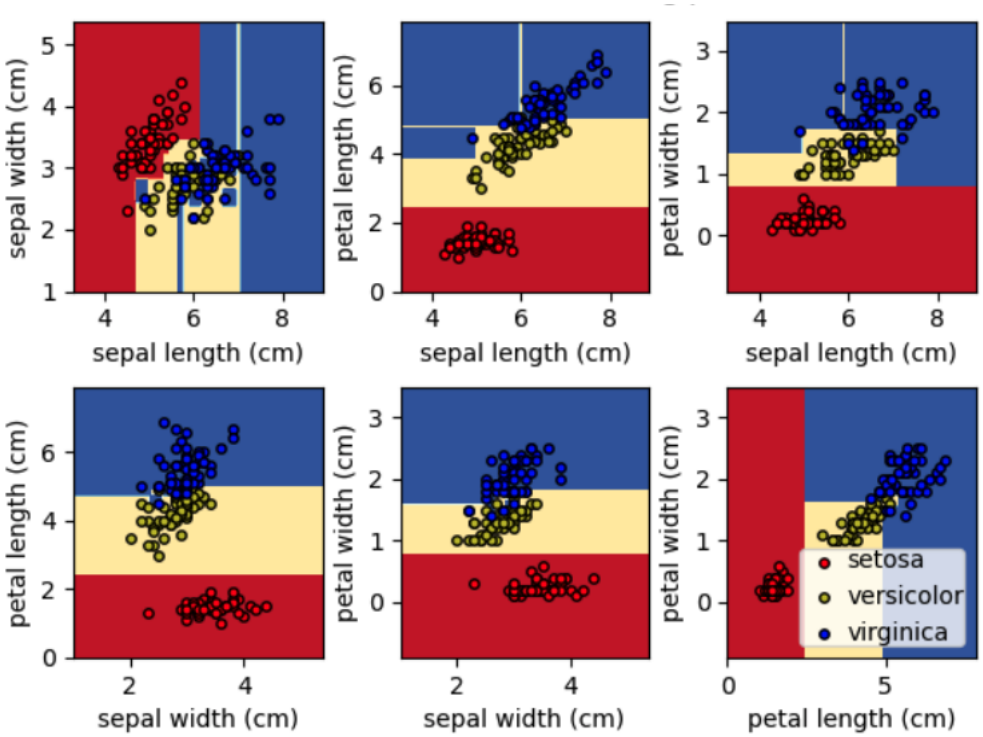

Figure 5: A decision tree on the iris dataset visualised graphically. From [17].

3.1.3 Ensemble Learning

Ensemble learning is combining machine learning models together to produce a better result than a single model can achieve [18]. The boost in performance is from models making different errors to the other models in the ensemble [17]. Each model in the ensemble votes for the class that it predicts and the ensemble learner aggregates all of the votes to produce the overall prediction.

3.1.3.1 Random Forests

Random forests are an ensemble of decision trees which allows them to become better classifiers than just a single decision tree alone [19]. The classifier is powerful as it is simple to train, robust to errors and outliers, and is resilient to overfitting. When an individual decision tree splits its node, the random forest dictates that the best feature amongst a selection of random features is selected which provides a lower variance with the tradeoff for higher bias [8]. This characteristic is responsible for its enhanced performance as an ensemble learner.

3.1.3.2 Extremely Randomised Trees (Extra Trees)

Random forests are extended through the extremely randomized trees (also known as extra trees) classifier which can provide better results than through the traditional random forest classifier [20]. It builds upon the random forest by randomising the threshold value rather than using the best threshold to determine the branch splitting [8]. Randomising the threshold speeds up the training of extra trees compared to random forests as threshold optimisation has a large computation cost to it.

3.2 Object Detection

Object detection is a fundamental component of being able to detect pedestrians in this application. There are currently many state-of-the-art methods of object detection, namely; Histogram of Gradients (HOG) [21] combined with a Support Vector Machine (SVM), You Only Look Once [22] and Single Shot Detector (SSD) [23].

From Zhong-Qiu [24] a list of possible algorithms was obtained and the top ranking for speed and performance (benchmarked by mean average precision (mAP)) and these were tested on the Nvidia AGX Xavier, the local compute unit. These algorithms selected were Multiped-500, Pednet-100 and SSD-Inception-v2.

3.3 Detecting Phone Focused Pedestrians

3.3.1 Hazards of Distracted Mobile Phone Use

Due to the rise in social media and the capabilities of mobile phones, younger people have become so engrossed with their mobile devices that phones are now a source of addiction as shown by Yu-Hsuan et al. [25]. This has led to an increase in the phone focused pedestrian injuries in the last two decades especially among university students [5] [7]. Pedestrian deaths as a result of using mobile phones has increased more than three-fold from less than 1% in 2004 to more than 3.5% in 2010 [5].

3.3.2 Detection of Phone-Based Activities

3.3.2.1 Rangesh et al. 2016

Rangesh et al. first proposed a framework for detecting phone-based activities in 2016 [26]. Their framework is presented in Figure 6 which consisted of obtaining the pose for pedestrians, using Newell, Yang & Dang hourglass method[27], then locating their hands using a distance of their arm measurement and obtaining a frame of the pedestrian's hand. The presence of a mobile phone was determined using an SVM over its HOG features. The pose and presence of a mobile phone was then combined in their belief network to classify if they were engaging in phone based activities.

The images were obtained using four Go-Pro cameras that provided images with a resolution of 2704x1520 pixels. They managed to achieve an accuracy of 0.9120 on a test set of 182 pedestrians using this method. The graphics processing unit (GPU) that they used to run the program was not specified.

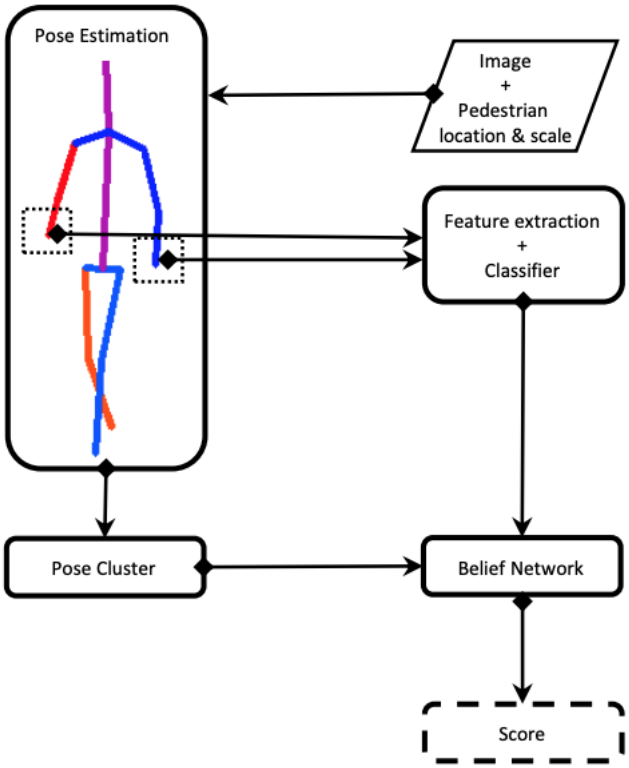

Figure 6: Rangesh et al.'s framework for detecting phone-based activities. From [26].

3.3.2.2 Rangesh & Trivedi 2018

Rangesh and Trivedi [28] extended upon the model presented in Rangesh et al. 2016 [26] by proposing a multi-cue framework to detect phone-based activities of pedestrians using pose, hand, and face estimators. In this method, they use Convolutional Pose Machines proposed by Wei et al. [29] to determine the pose. Only the upper body poses were extracted as these contained the key information for the framework. They also implemented a tracking filter to reduce the computational burden of the pose estimation. Exemplar SVMs are used to train the hand analysis. A benefit of using an ensemble exemplar SVMs is that regular SVMs would overfit the data whereas an Exemplar SVM has a separate classifier for each class.

Similar to the previous study [26], the images were obtained at a resolution of 2704x1520 pixels by four Go-Pro cameras which provided a dataset of 1586 pedestrians. Overall for hand, face, and pose they achieved an accuracy of 0.946, for pose and hand it had an accuracy of 0.916, and for pose alone, an accuracy of 0.858. The SVM was run on a sixth generation Intel i7 central processing unit (CPU) while the pose estimation was performed on a Nvidia Titan X. Despite the powerful GPU, Rangesh et al. still recommends avenues to speed up the pose model such as using different pose estimators such as OpenPose [15].

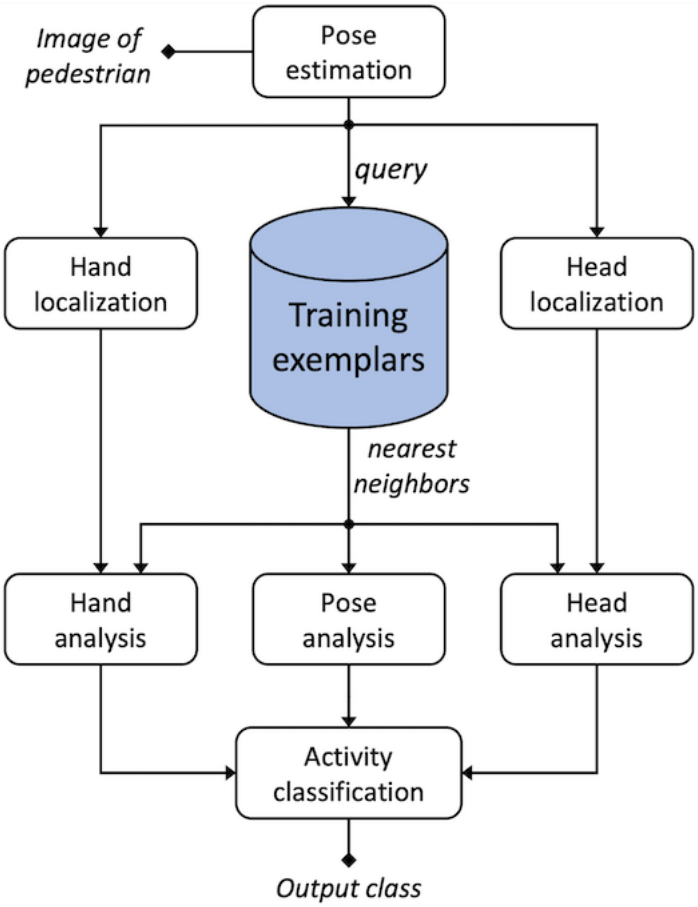

Figure 7: Rangesh & Trivedi's multi-cue framework for detecting phone-based activities. From [28].

4 Design Process

4.1 Design Tools

4.1.1 Hardware

4.1.1.1 nUWAy EZ10 Bus

The nUWAy bus (Figure 8 is based in EasyMile's EZ10 bus [30] and serves to provide power and a chassis for the hardware listed below.

Figure 8: nUWAy on the UWA campus.

4.1.1.2 Cameras

Point Grey Grasshopper 3 (model number GS3-PGE-23S6M-C) [31] (referred to as PG Cameras) monochrome cameras are used as the vision component of the project. They are capable of producing an image size of 1920x1200 pixels but it has been reduced to 960x600 pixels for increased deep learning processing speed and reduced bandwidth so other sensors such as the LiDARS can be queried.

Figure 9: A Point Grey GS3-PGE-23S6M-C camera. From [31]

4.1.1.3 Nvidia Jetson AGX Xavier

An Nvidia Jetson AGX Xavier is used as the compute unit as the project requires a deep learning framework. The AGX Xavier is used to accelerate the deep learning frameworks using Compute Unified Device Architecture (CUDA®) and CUDA Deep Neural Network (CuDNN) which boasts substantially faster processing for deep learning tasks than on a Central Processing Unit (CPU) alone. The unit runs Ubuntu 18.04 which allows for easy development and deployment and connects to the Bus PC via Ethernet. Due to the specialisation of the Jetson devices as embedded machine learning computers, there are optimised implementations of the networks VGG-19, Multiped-500, Pednet-100 and SSD-Inception-v2 [12].

Compared to GPUs present in other frameworks such as the Nvidia Titan X [28] with 3072 CUDA cores while the AGX Xavier only has 512 CUDA cores. To obtain the best performance from the AGX Xavier, the device is set to draw its maximum power of 30W. Additionally, all 8 CPU cores are turned on, and the CPU and GPU cores are set to their maximum clock frequency at 2.27 GHz and 1.38 GHz respectively. However, even with these adjustments, the project is still heavily constrained on computational ability and it is necessary to reduce computational load as much as possible.

Figure 10: An Nvidia Jetson AGX Xavier compute unit. From [32]

4.1.1.4 nUWAy PC

There is a computer on-board the nUWAy bus with a core i7m processor used for all the low level processing such as obtaining all the sensor data. The camera images are obtained by the nUWAy PC.

4.2 Software

This section will explore the algorithms and applications built to support the project. The majority of the designs will use Python given the cross-platform compatibility and ease-of-use which enables rapid development.

4.2.1 OpenPose

OpenPose was used as the pose detection algorithm due to its popularity and multiple implementations across Caffe and PyTorch.

4.2.2 Robot Operating System (ROS)

ROS is an open-source node-based middleware that the project is implemented on which facilitates the high level access to sensor data and information between programs. This allows the transfer of images from the cameras to the AGX Xavier for processing and classification [33]. Although the project can run independent of ROS, it allows significantly easier development of each of the separate programs is linked through ROS so that they can pass on the information from one to another.

ROS will run the nodes that contain the end-to-end final model and publish the data to ROS message streams so that other applications can access the pre and post processed data. Regardless of the nuances of the final model, as each of the compute units are running Ubuntu 18.04, ROS Melodic is the distribution that will be used throughout the project.

4.2.3 Scikit-learn

Scikit-learn (sklearn) is a flexible and simple machine learning library built for Python [17]. As sklearn is built in mind with interoperability with Numpy and Scipy, it allows for fast execution times and time to develop the code. An extra trees, random forest and SVM classifiers for phone pose recognition were built using sklearn due to its ease of use and fast development time.

4.2.4 OpenCV

OpenCV is ubiquitous, open-source library used for computer vision with C++ and Python implementations [34]. The library will be used extensively for the Python based manipulation of images collected by the cameras. The foundation of the labelling program is built upon OpenCV.

4.3 Evaluation Criteria

The key areas that the final design will be benchmarked against to determine which model implementation to use and to evaluate its efficacy.

4.3.1 Model Scoring

Scoring the models is crucial to finalise the framework for the project as it provides the ability to benchmark each of the proposed solutions. The metrics used are precision, recall, F1 score, Matthews Correlation Coefficient (MCC) and speed. Aside from speed, all of the metrics use the following parameters: true positives (TP), false positives (FP), true negatives (TN) and false negatives (FN). There are no true negatives in CNNs due to the nature of the network. It is not logical to compute a true negative on an image which is possesses a near infinite combination of pixels. Thus only true positives, false positives and false negatives can be provided.

As there are two sets of learning models which are classified as the Object Conditioning and OpenPose model, and the phone pose detection extra trees model, both of these models will be evaluated independently of each other. However, the accuracy of the Phone Pose Detection model will depend on the accuracy of the posenet model that feeds into it. The intent to separate the scoring is due to the fact that the models can be optimised independently of each other and retraining of the pre-pose conditioning and OpenPose network.

4.3.1.1 Accuracy

Accuracy is a popular metric in scoring machine learning models as it gives the ratio of correct predictions compared to the total number of predictions [8]. However, accuracy is a poor measure when scoring datasets asymmetric distribution of labels. It was observed during training that there are significantly more negative labels than positive labels for the dataset collected which means that accuracy will not be used to score the models, only to compare the model to other frameworks that use accuracy as a benchmark. Accuracy is given by Equation 1 below.

$$accuracy = \frac{TP + TN}{TP + FP + FN + TN} \tag{1}$$

4.3.1.2 Precision

Precision is the ratio between the number of predicted observations (TP) and the actual number of observations (TP+FP) (Equation 2). This measure shows how well the model correctly predicts a label given that it has predicted it as that label. For example, it shows the ratio of how many people are truly on their phone compared to the number of people the model predicts are on their phone. A precision of above 0.5 results in a model that for a given true label, it can predict it correctly half the time.

$$precision = \frac{TP}{TP + FP} \tag{2}$$

4.3.1.3 Recall

Recall measures the ratio between the number of predicted labels against the total number of labels there actually were (Equation 3). For example, it measures the number of people predicted on their phone against the true number of people on their phone. A recall of above 0.5 results in a model that correctly predicts half of the total number of a certain label.

$$recall = \frac{TP}{TP + FN} \tag{3}$$

4.3.1.4 F1 Score

F1 score is the weighted average of precision and recall [8]. It results in a score between 0 and 1 serves to show the inverse measure of false positives and false negatives. A score closer to 1 means that there are a low number of false positives and false negatives. This is the metric that will be used to evaluate the deep learning models.

$$recall = 2 \times \frac{precision \times recall}{precision + recall} \tag{4}$$

4.3.1.5 Matthews Correlation Coefficient (MCC)

The Matthews Correlation Coefficient, given in Equation 5, provides the overall score of how well the model has performed as it helps compensates for poorly trained models as it takes into account the size of the positive and negative labels [35]. A major advantage provided by using

MCC is that it scores binary classifiers better than other metrics such as F1.

$$\frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \tag{5}$$

MCC scores range between -1 (perfect misclassification) and 1 (perfect classification). Only models with an MCC above 0 will be considered for use in the final design. As the MCC is dependent on if all four parameters (TP, FP, TN, FN) are good, failing to reach 0 means that it has failed the scoring criteria. Given that MCCs use true negatives, this cannot be used to score the CNNs and F1 will be used instead.

4.3.2 Speed

The faster the algorithm runs, the better the method will score. Any speed below 1 frame per second will fail this criteria as that is deemed too slow to provide any value for the project. To measure the speed, the time taken to load the deep learning models into memory will be ignored as this is a once off processing expense. Only the time taken to process the image frames will be measured. The metric for speed will be frames per second (FPS) and a higher FPS is better.

4.4 Requirements

For the project to be successful, there are a number of requirements that it needs to fulfil. These are listed below:

R1. Classify pedestrians that are on their mobile phone.
The crux of the project is to be able to identify pedestrians on their mobile phones so this is the most important requirement for the design to have.

R2. Run in real-time on nUWAy hardware.
It is important that the final design to be able to run on the nUWAy bus's hardware due to limited financial budgets to increase the computational capacity of the bus. Therefore, the project needs to be able to fully run on the nUWAy PC and the AGX Xavier. Real-time is at least at the same FPS as the camera stream of 15 FPS.

R3. Implementation on nUWAy's ROS stack.
As ROS provides the framework for all of the programs on the bus, it is crucial that the final design can transmit and receive ROS messages.

R4. Provide a way to store and label images.
Data is essential to training any sort of machine learning model for the project so a method to store and label these images is required.

R5. Obtain image frames from nUWAy.
The final design should accommodate images streams from the nUWAy PC where the cameras are connected.

4.5 Constraints

Due to the physical limitations of hardware on-board the nUWAy bus such as finite compute resources, there are several performance based constraints on the project.

4.5.1 Hardware

Given the limited budget for the project, the only hardware and cameras that can be used in the final design are what is already on the nUWAy bus itself. For training and testing however, there is no constraint on what hardware can be used.

4.5.1.1 Computational Power

There will be multiple projects and therefore programs running on both the AGX Xavier and the Bus PC so device usage, especially on the GPU, needs to be kept as limited as possible. This may impose limits to what deep learning frameworks and neural network resolutions can be used in the final model.

4.5.1.2 Cameras

The Bus PC which is handling the camera feed also needs to be capable of processing a variety of other sensor information so the camera data needs to be as low resolution as possible. The cameras currently on the nUWAy bus are greyscale which is another constraint on the project.

4.5.2 COVID-19

Due to the global impacts of COVID-19 there are constraints placed in the collection of camera images for the phone focused estimation as the number of students on campus was reduced which in-turn reduced the total number of students in the camera images.

5 Final Design

This section will explore the final software and hardware design for the project. Figure 11 details the software stack that allows the images obtained by the cameras to become poses and subsequently detect if pedestrians are on their phone. The Bus PC will obtain a camera stream from the PG Cameras through the ROS Blackfly Driver. This publishes the images to a ROS topic which is accessed over Ethernet by the AGX Xavier and passed into object conditioning, then pose detection and subsequently either into the labelling program or to the extra trees classifier to be classified as on their phone or not.

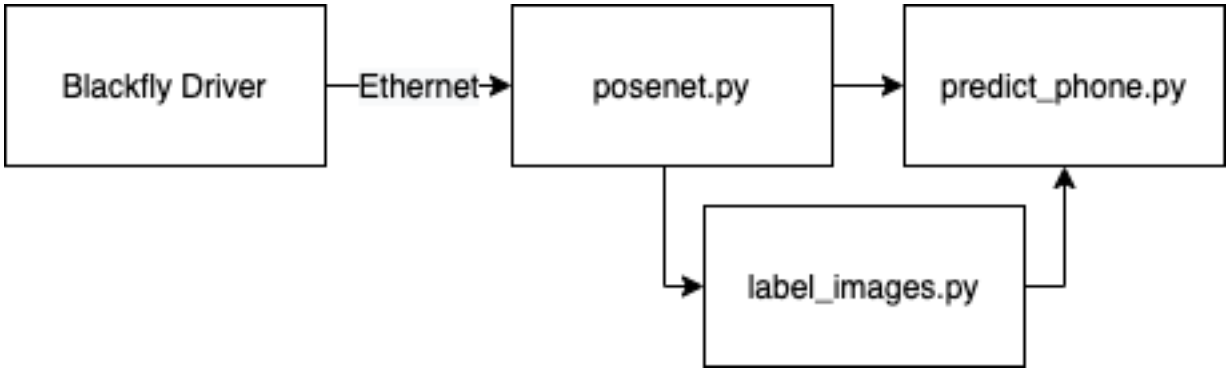

Figure 11: High level overview of how the camera image becomes a label.

5.1 Blackfly Driver

5.1.1 Camera Image and Blackfly Driver

The images were obtained from two cameras that are mounted on the front and the back of the bus. The images are obtained through a C++ ROS camera driver developed by the REV team which publishes them to two topics, one for each of the front (/front_image) and back (/back_image) cameras. To comply with the constraint imposed in Section 4.5.1 the images are of size 960 x 600 pixels in greyscale. The Blackfly Driver corrects for the barrel shape due to the camera's lens.

5.2 posenet.py

This file is the ROS integrated script that does some pre-pose conditioning on the image to get it ready for OpenPose and subsequently pass the output of OpenPose to the labelling or prediction scripts. A flowchart of this program is contained in Figure 12.

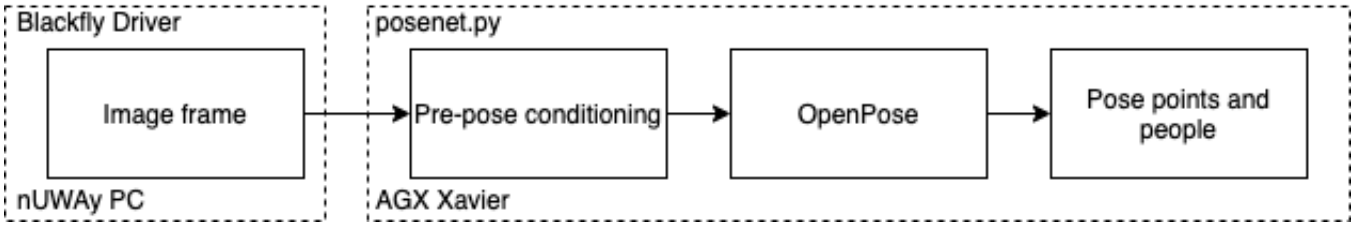

Figure 12: The architecture of posenet.py; the method to turn raw camera frames into the pose points.

5.2.1 ROS Interfacing

5.2.1.1 Input

As the Blackfly Driver publishes the frames of the camera to a ROS message topic, the frames are retrieved from this stream and converted into an OpenCV image matrix.

5.2.1.2 Output

The output are two ROS message streams, /phone_pose_image returns an image stream of

the people on their mobile phones to be displayed on the nUWAy screens, and /phone_focused which returns the position on the image of the distracted pedestrians.

5.2.2 Pre-Pose Conditioning

During testing OpenPose, it was found that there were a high rate of false positives that needed to be reduced. In particular, these false positives came from passing trees or building facades which are shown in Figure 13. To reduce this in the project setup, there were a few different methods that were trialled namely, passing in pixels below the horizon, removing poses that were detected above the horizon of the image, and inserting a pre-trained object detection algorithm before pose detection.

Figure 13: Examples of a false positives (shown in the yellow box).

5.2.2.1 Half Image

By cropping each frame and turning it into a 960 x 300 image impacted the performance of the whole model substantially. The frames processed per second dropped from 3.21 FPS to 1.24 FPS. The advantage of this method is that it completely eliminates the false positives from above the horizon. This method is shown in Figure 14 and it is seen that the false positives from below the horizon are still present. The large impact to speed is eliminated through the pose removal method below.

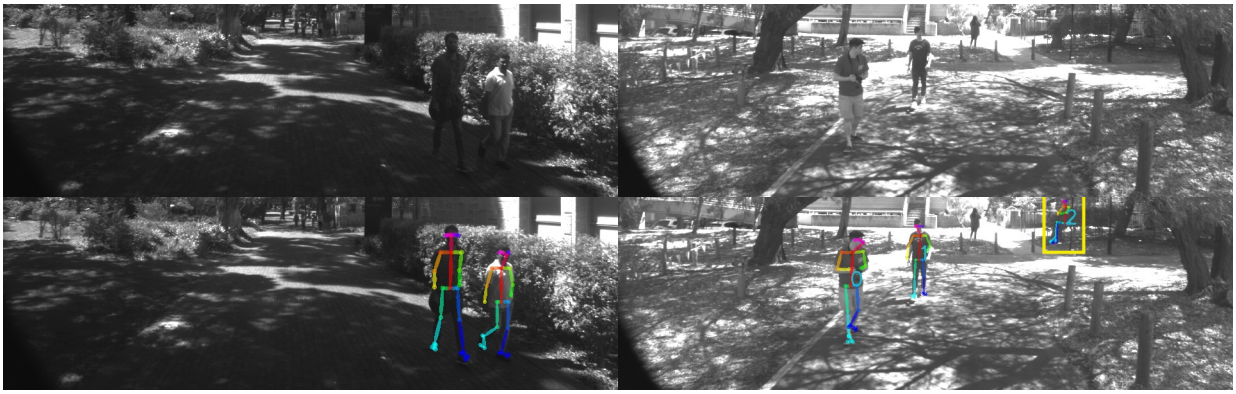

Figure 14: Passing in a half image method. Top row is the pre-processed half image and bottom row is the post-processed half image.

5.2.2.2 Pose Removal

This method passes in the full 960 x 600 image into OpenPose but just does not record the poses that have joints above the horizon. This is a better method than passing in only the half image as it retains the speed of the original network while reducing the false positives above the horizon. However, the problem of false positives below the horizon still remains.

5.2.2.3 Object Detection

Another way to reduce the false positives from OpenPose is to pass the frame through an object detector and only pass the people that the pre-trained object detector detects 15. The hypothesis is that it will behave similar to ensemble learners for other machine learning classification and regression problems as ensemble learners increase the overall performance of the model by leveraging the strengths of each model in the network. Therefore, it is proposed that if the object detection models and OpenPose are trained on different networks then the probability that both networks show the same false negative or false positive is lower than the individual network. That is $A_{FP} \cap B_{FP} < A_{FP}$ where A_{FP} is the false positive rate of OpenPose and B_{FP} is the false positive rate of the object detector.

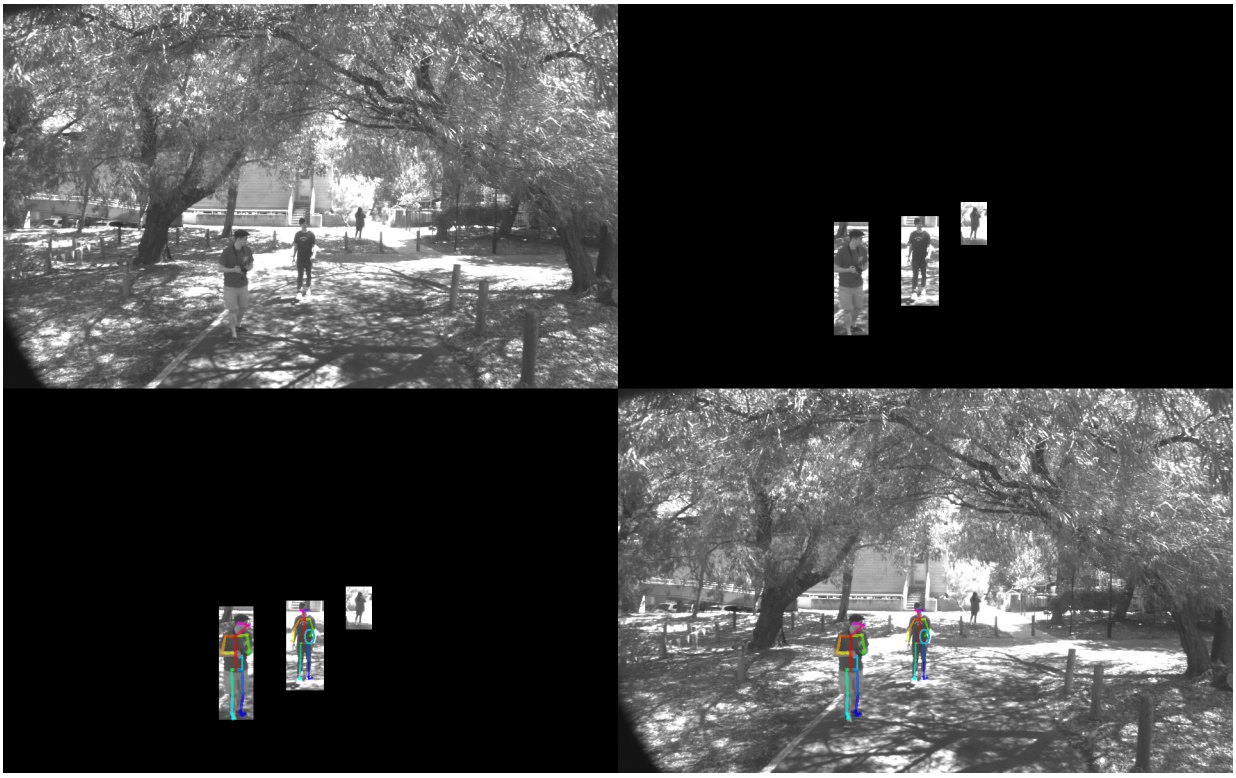

Figure 15: Conditioning through a pre-trained object detector. Clockwise from top left: raw frame from camera, multiped person detected image, pose detection, final output.

There were three CNNs that were tested on the data and benchmarked with speed shown in Figure 1, multiped-500, pednet-100, and SSD-Inception-v2. Multiped-500 was determined to be the classifier to use due to its performance and speed. Overall, the object detection pre-processing managed to reduce the false positive rate but it also reduced the true positive rate as multiped-500 was unable to detect some pedestrians that OpenPose could. This method

can be combined with the removal of poses detected above the horizon to eliminate all false positives in the top half of the image and reduce false positives in the bottom half of the image.

Figure 16 contrasts the different object detectors after being processed by OpenPose. In the multiped-500 image, it can be seen that the pedestrian pose denoted by "5" is not present in the other images despite the networks recognising the pedestrian. The bounding box provided by multiped-500 is a few pixels taller than the bounding boxes from the other two networks which has provided OpenPose with the ability to classify the object as a pose. A false positive detected by multiped-500 that the other two networks did not show can be seen in Figure 16 above pedestrian "2". OpenPose did not detect this as a pose which lends credibility to the ensemble CNN hypothesis.

Table 1: Performance comparison of different object detection models. Tested on 300 images.

Model	**FPS**
SSD-Inception-v2	11.71
Pednet-100	11.40
Multiped-500	11.73

Figure 16: Comparison of pednet-100 (top left), SSD-Inception-V2 (top right), multiped-500 (bottom) on the posenet framework. Note, the images have been cropped so that the details are easier to see.

5.2.3 OpenPose

Various implementations of OpenPose were tested where different deep learning frameworks, pre-trained models and net resolutions were tested and the results are shown in Table 2. Body 25 with -1x368 net resolution had the fastest speed at 3.21 FPS with the tradeoff of a lower F1 score than the Body 135 model. The marginal gain in F1 score from Body 135 is not worth a 48% reduction in speed of the model from 3.21 FPS to 1.68 FPS. Hence, Body 25 will be used as the OpenPose network in the final model.

Table 2: Performance comparison of different methods of OpenPose regarding F1 score and speed. Tested on 300 images. Note, a net resolution parameter of -1 dictates the resolution will depend on the size of the image.

Framework	**Pre-Trained Model**	**Net Resolution**	**F1 Score**	**FPS**
Caffe	Body 25 [15]	-1x368	**0.838**	**3.21**
Caffe	Body 135 [15]	-1x480	**0.861**	**1.68**

Table 3: Performance comparison of different methods to boost the accuracy of the pose detection. Tested on 300 images.

Method	**Precision**	**Recall**	**F1 Score**	**FPS**
Body 25 + Half Image	0.972	0.901	0.935	1.24
Body 25 + Pose Removal	0.859	0.822	0.838	3.21
Body 25 + Multiped	**0.827**	**0.994**	**0.903**	**2.07**

5.3 label_images.py

The labelling program was built using Python and OpenCV to perform a basic labelling function for labelling two sets of data. The program is shown in Figure 17. The graphical user interface provided by OpenCV is shown in Figure 18. Both datasets were from the output of posenet.py where the first set was to provide labels for predict_phone.py and second set was to compare the performance of different models within posenet.py.

5.3.1 Labelling Mobile Phone Distracted Pedestrians

The first set of labelling was to determine which person on the screen was on their mobile phone. The user is prompted to type in the number that corresponds to the figures that are on their mobile devices and press enter to confirm their selection. The selections are stored in a comma-separated values (CSV) file to later be fed into the Phone Pose Detector.

5.3.2 Labelling Performance of posenet.py

The second set involved entering three values, how many people had poses (TP), how many people did not have poses (FN), and how many other non-people objects had poses (FP). There is no TN metric in image and object recognition. From these three measured metrics, the precision, recall and F1 score could be computed so that the models can be benchmarked.

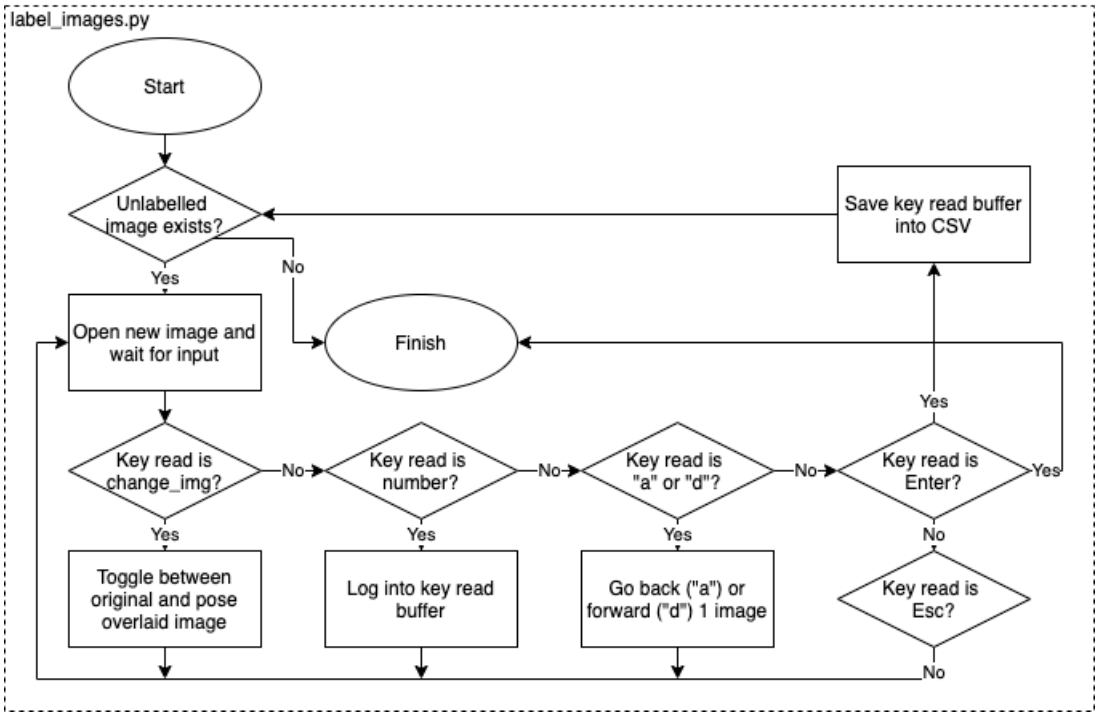

Figure 17: Labelling program description.

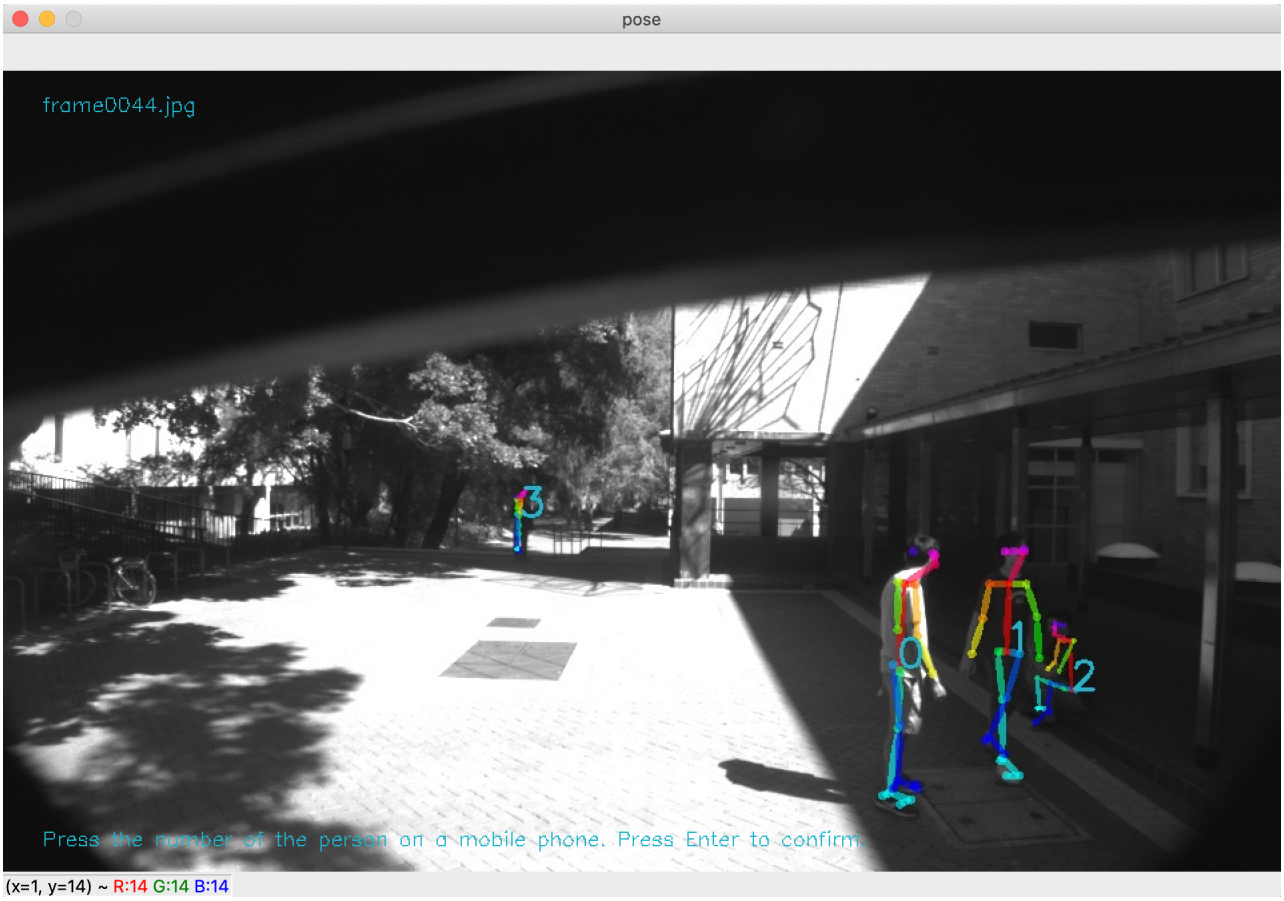

Figure 18: OpenCV based labelling program on Mac OS.

5.4 predict_phone.py

The phone pose prediction was created using the extra trees classifier in sklearn. A random forest classifier and SVM were also used but the extra trees classifier had the best results.

The purpose of the classifier is to determine if a detected person is distracted on their phone or not. The model is demonstrated in Figure 19. The distraction model has two functions, one is to train the extra trees classifier and the other is the evaluate if the input poses are classified as distracted or not. The key features used were the angles created by the pose joints predicted by OpenPose. Positions a, b, c are obtained through OpenPose, lines $\overrightarrow{AB}$, $\overrightarrow{BC}$, $\overrightarrow{AC}$ and the angle of $\sphericalangle ABC$ are calculated by the Phone Pose Detector Python program. The angles are calculated by using the cosine rule (Equation 6) where the joints, lines, and angle is described in Figure 20.

$$\sphericalangle ABC = \frac{b^2 - a^2 - c^2}{2ab} \tag{6}$$

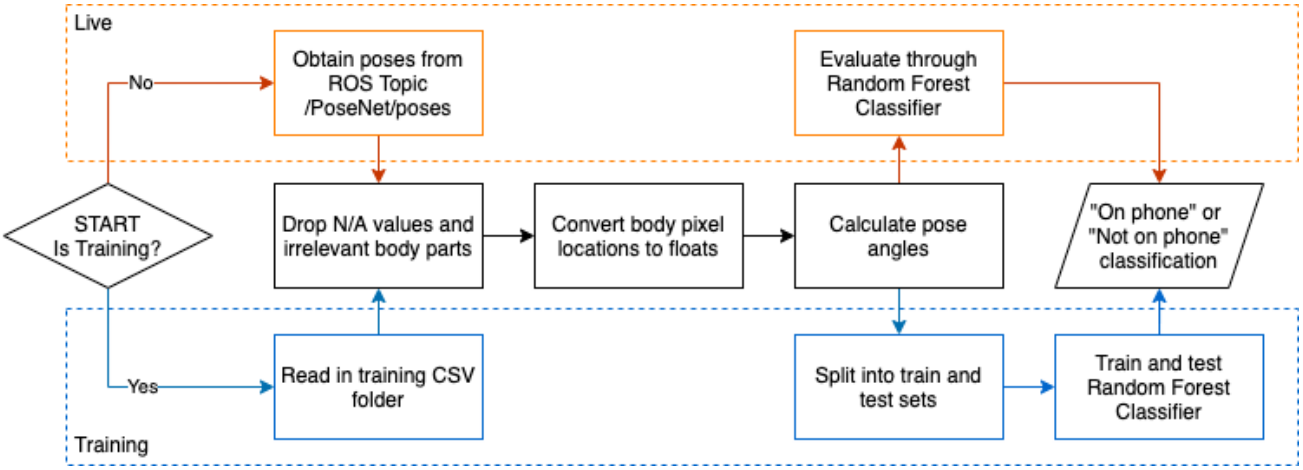

Figure 19: Flow of the extra trees classifier.

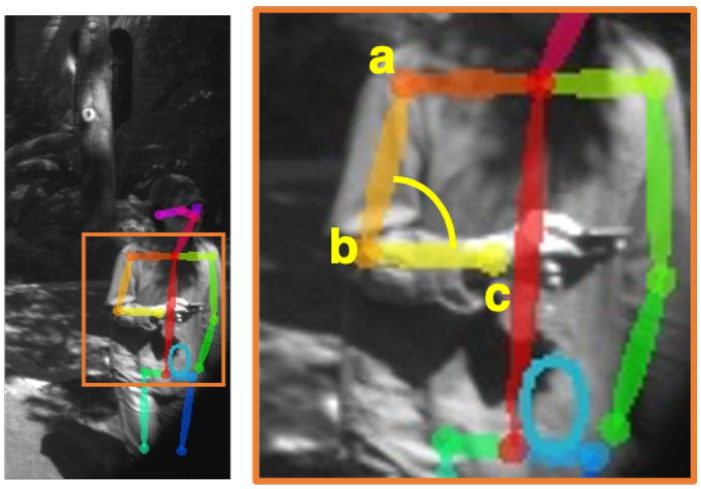

Figure 20: Pose angle features to predict phone focused pedestrians.

6 Results

6.1 Posenet

The final posenet model pipeline was chosen to have the Multiped-500 and pose removal conditioning as outlined in Figure 21. The scores for the model are shown in Table 3 above.

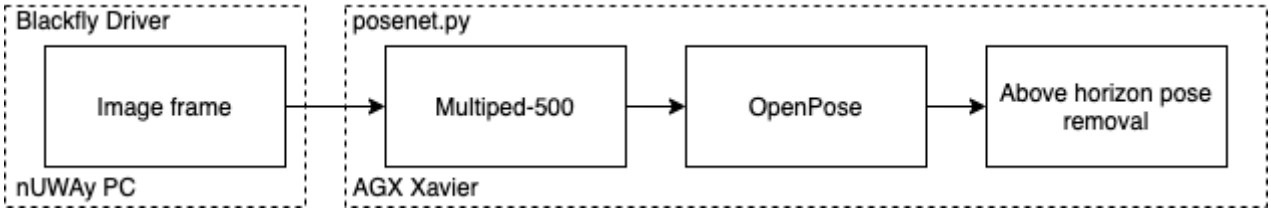

Figure 21: The final model for posenet.py.

6.1.1 Scoring

6.1.1.1 Precision

A precision of 0.827 was one of the lower precision scores among the considered models but overall is high. This represents a relatively high rate of detecting and obtaining the pose of pedestrians. This is above the required threshold of 50% so it meets the minimum criteria for inclusion in the framework.

6.1.1.2 Recall

The final posenet model had a high rate of making predictions evident of the high recall of 0.994 and meets the 0.5 threshold for inclusion.

6.1.1.3 F1 Score

The high F1 score of 0.903 was bolstered by a significantly higher recall than precision score. High recall and low precision scores indicates that the model classifies a lot of pedestrians but the probability that the predicted was made correctly is not as high. However, a precision of 0.827 should not be classified as low. Rather, the model is better at producing pedestrian poses rather than ensuring the predictions are correct. This suggests that there are improvements to be made to increase the precision score.

6.1.1.4 Speed

The requirement outlines in Section 4.4 was that the final model needed to run in real-time. Given that the speed of the overall posenet model was 2.07 FPS, the project does not meet the real-time requirement but it appears sufficient for the current state of the nUWAy project. Given that the bus currently travels at approximately 5km/h, the bus will move approximately 70cm in the time it takes to fully process each frame. Due to the constraints of limited computational power, true real-time performance was a very difficult task when pose classification was involved.

6.2 Phone Pose Predictor

The dataset used for the extra trees classifier had a total of 2800 pose classifications and an 80/20 training/testing split was used obtaining the distributions shown in Figure 22. The hyperparameters for the extra trees classifier were optimised using a grid search method and the best results were produced by using information entropy as the measure of node inequality instead of a Gini coefficient. The summary of predictions for the phone pose detector is given by the confusion matrix in Figure 23 scores for the extra trees classifier are given in Table 4.

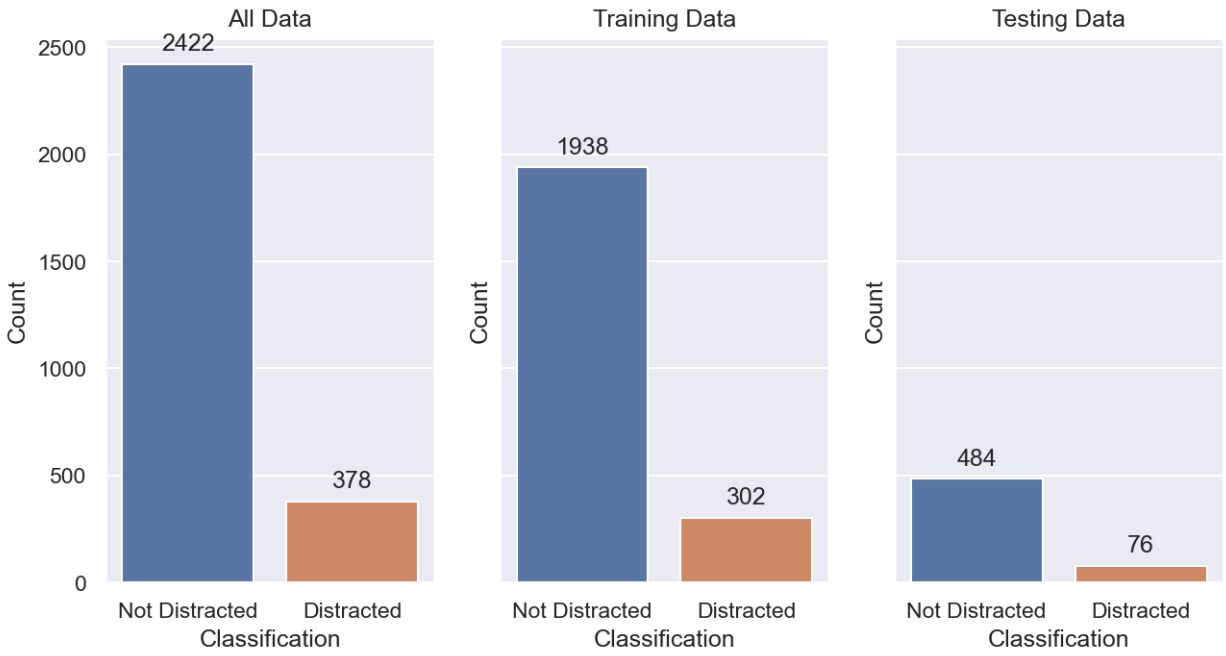

Figure 22: Size of the dataset including the training and testing datasets for the random forest phone detector. The 0 label is *Not Distracted* and 1 label is *Distracted*

6.2.1 Scoring

6.2.1.1 Precision

The extra trees classifier had a high rate of predicting the correct *Not Distracted* or *Distracted* class shown by the high precisions of 0.948 and 0.862 respectively. The higher precision for the *Not Distracted* class is possibly due to the skewed dataset.

6.2.1.2 Recall

The recall is worse for the *Distracted* labels than for the *Not Distracted* labels which means that for all the people in the images that are on their phone, it only predicted 66% of them actually being on their phone or 50 of 76 people. The recall is above the required threshold of 50% which means as this is above 50%, we are still able to use this model. The predicut_phone.py model has a high rate of making predictions for the *Not Distracted* label with a recall of 98%.

6.2.1.3 MCC

A MCC of 0.721 means that the relationship between the extra trees model and correctly

predicting a person's *Distracted* or *Not Distracted* state is strongly correlated. The MCC satisfies the boundary condition from Section 4.3.1.5 where the MCC must be above 0. This means that using this extra trees model is a success for predicting the mobile phone distracted state of poses.

Table 4: Performance of the phone pose detector.

Label	**Precision**	**Recall**	**F1 Score**	**Support**
0	0.948	0.983	0.966	484
1	0.862	0.658	0.746	76
accuracy			0.929	560
macro avg	0.905	0.821	0.856	560
weighted avg	0.937	0.939	0.936	560
SVM MCC				0.608
Random Forest MCC				0.691
Extra Trees MCC				**0.721**

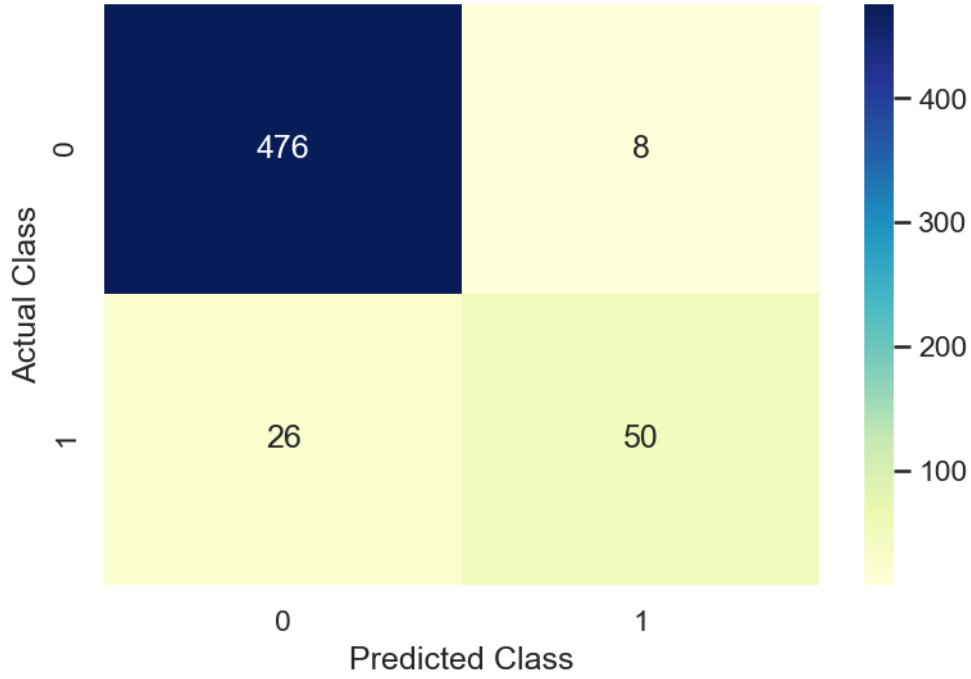

Figure 23: Confusion matrix of the random forest phone detector. Clockwise from top left: true negatives, false negatives, true positives, false positives.

6.2.1.4 Speed

The fast trees classifier performs classification on CPU cores in under 0.1s which gives it a speed of more than 10 FPS. In terms of real-time performance, this would not be sufficient to run in real-time as the cameras stream at 15 FPS. The slow speed could be as a result of using the CPU rather than the GPU to perform the classification task.

7 Discussion

7.1 Satisfaction of Requirements

The final design for the project satisfies most of the requirements from Section 4.4 except for real-time operation. A summary table is outlined in Table 5. Due to high computational requirements for OpenPose with the larger deep learning net resolution, the real-time performance requirement was not able to be achieved. Instead of a minimum of 15 FPS, the final model can only achieve a maximum speed of 1.7 FPS which involves fast trees classification on the posenet poses.

Table 5: Project requirement satisfaction.

Requirement	posenet.py	label_images.py	predict_phone.py
R1. Classify phone focused pedestrians			✓
R2. Real time on nUWAy	✗		✗
R3. ROS implementation	✓		
R4. Image storage and labelling	✓	✓	
R5. Obtain image frames from nUWAy.	✓		

7.2 Scoring Bias

To be able to benchmark and interpret the results, it is important to understand the labelling process for the posenet model as the methodology is different to the phone pose prediction model. The dataset for the phone pose predictor had binary classes, either the pose was *Distracted* or *Not Distracted* however, for the images itself, it is difficult to produce metrics for TP, FP and FN, and not feasible for TP. Only pedestrians who were mostly unobstructed and above 40 pixels were considered. The TP, FP, FN scores for the image in Figure 24 would be 1, 0, 4 respectively.

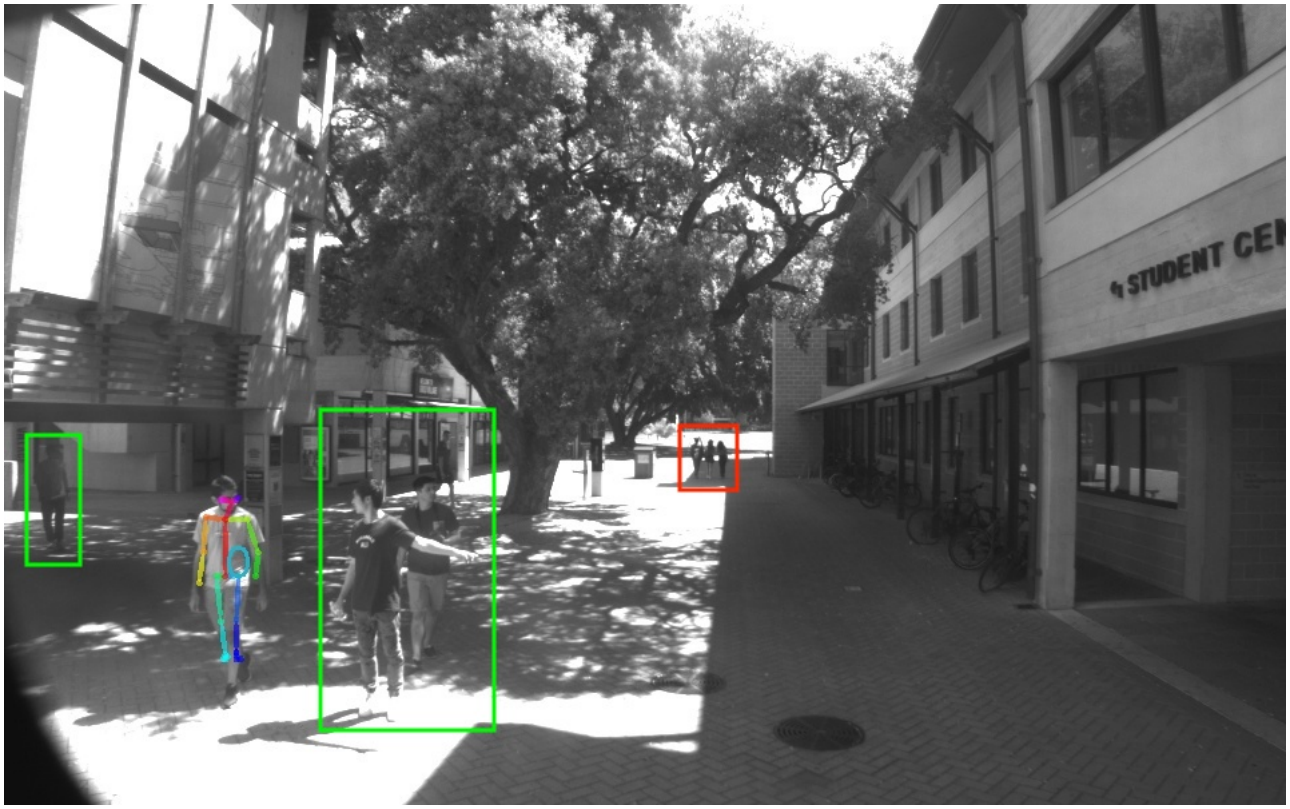

Figure 24: The scoring methodology for posenet. The pedestrians in the green rectangle are eligible for scoring while the pedestrians in the red rectangle are not.

7.3 Features

The predict_phone.py classifier scored well based on the metrics by only processing angles created by upper body joints. However, other information that could potentially improve the performance of the could be gaze or CNNs trained on images of phones as used in Rangesh & Trivedi's [28] work in 2018. However, due to the resolution of the current cameras, and the extra computation time for gaze and hand CNNs, the reduction in speed needs to be compared for potentially only marginal improvements to F1 score.

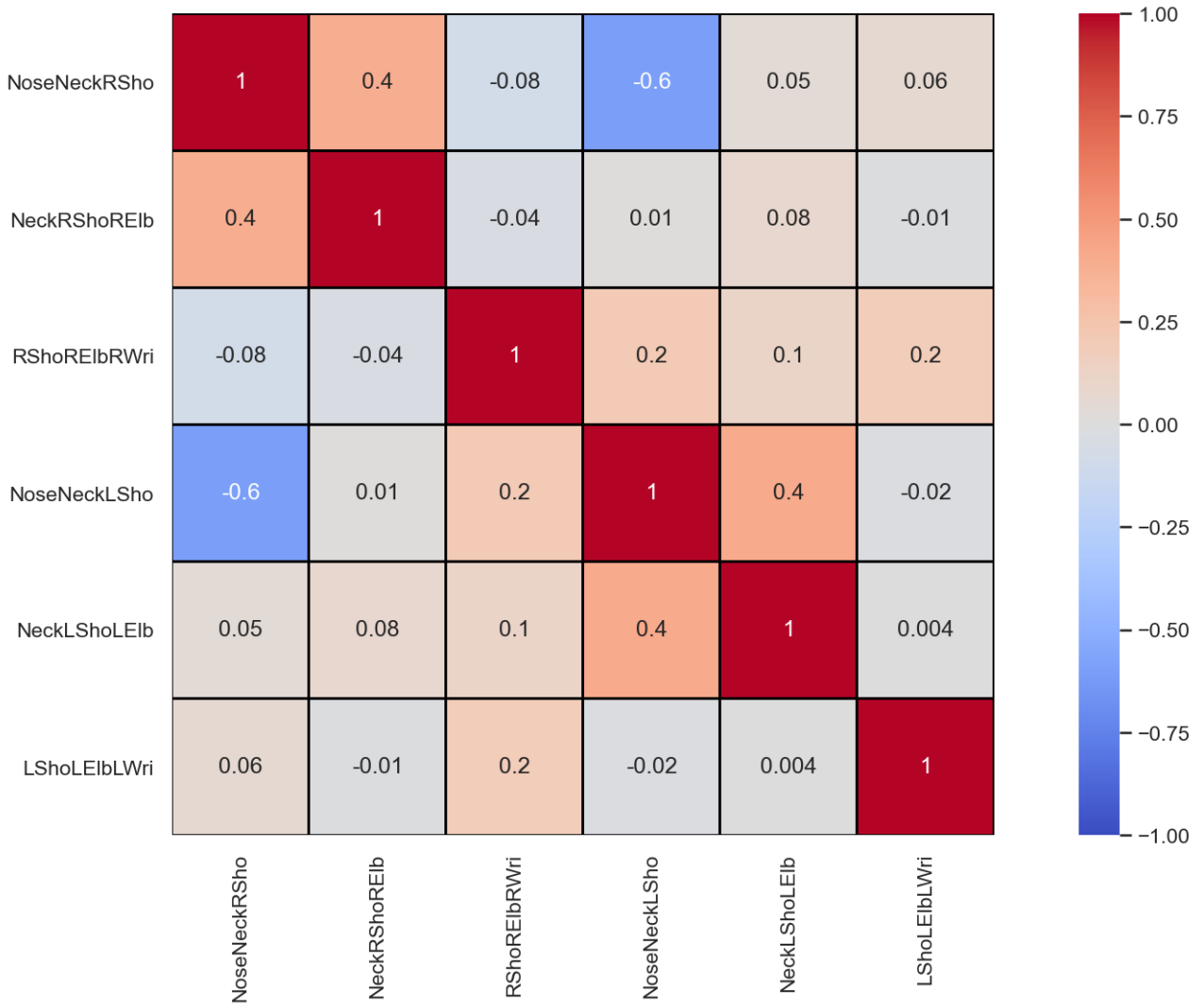

Figure 25: Feature heatmap of the extra trees phone detector.

7.4 Data Quality

Observing the feature importance given in Figure 26, it is seen that there is a heavy bias in the dataset for right handed individuals which is likely due to natural bias in the population for right handed individuals. A lot of the images obtained from the cameras featured members of the REV lab who are right handed which has skewed the feature importance of right handed people. As the number of left handed people in the dataset is low, this retraining of the network to include a higher number of left-handed people is not within scope of this project but it can be extended upon in the future. Although the dataset was comprised of 2800 images, the quality of the dataset suffered due to a high bias due to a high proportion of images consisting of the same pedestrians.

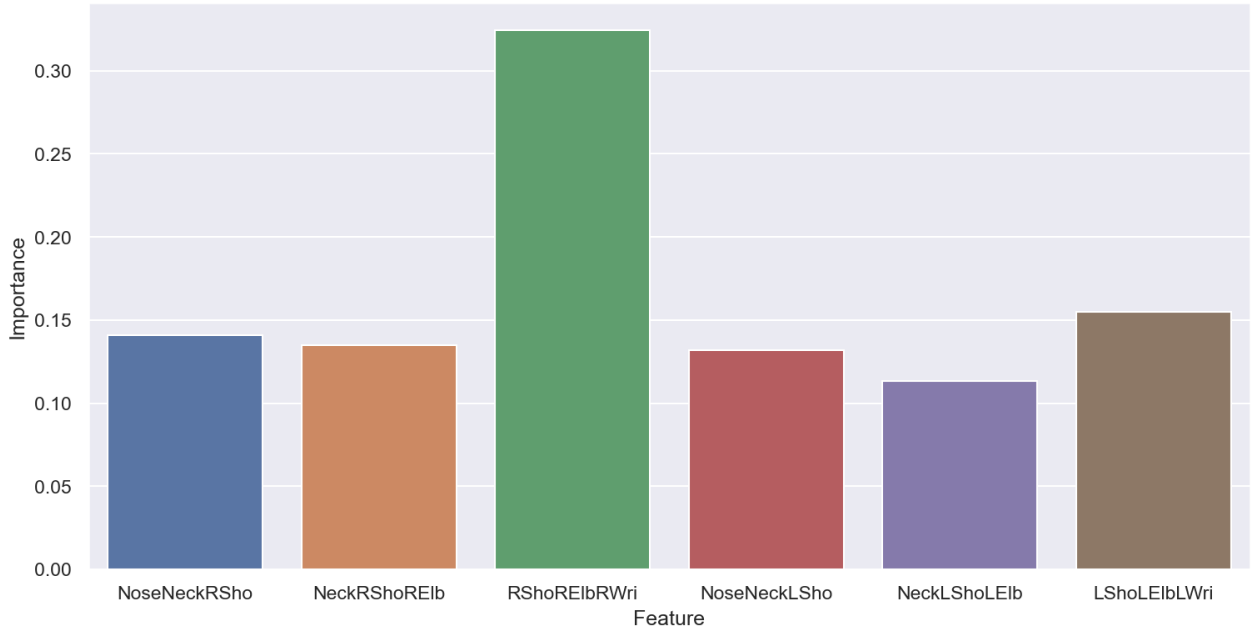

Figure 26: Feature importance of the random forest classifier. Note, the feature names are made up of three four letter works which reference the $\sphericalangle ABC$. For example, NoseNeckRSho is the angle between the nose, neck and right shoulder.

7.5 Comparison to Rangesh & Trivedi's 2018 work

The framework in this project only detects pedestrians who Rangesh & Trivedi would classify as *Texting*. Through pose alone, Rangesh & Trivedi achieved an accuracy of 0.65 for *Texting* and 0.90 for the *None* class compared to an accuracy of the phone pose predictor of 0.94 in this project. Rangesh & Trivedi's framework involves a mutli-class classification whereas this project only involves binary classification therefore, the accuracy for *Distracted* and *Not Distracted* are the same in the phone pose predictor.

The final model in Rangesh and Trivedi's framework involves pose, gaze, and hand classification which resulted in accuracies of 0.88 and 0.97 for *Texting* and *None* respectively and an overall accuracy of 0.946 when including the *Phone Call* class. Their dataset comprised of 1586 pedestrians and due to Go-Pro cameras, pedestrians were as large as 1000 pixels. The dataset used by Rangesh & Trivedi therefore was more robust as it had a higher resolution and likely lower bias due to significantly higher diversity. It is important to also consider the difference in computational ability between the Titan X GPU and the AGX Xavier which provided Rangesh & Trivedi the ability for such a strong framework. As outlined above, the quality of the dataset for this project needs improving which can be achieved by either higher resolution and colour images (reducing speed of the framework), or more diversity in the dataset's pedestrians (improved bias).

7.6 Poor Performance at Long Ranges

The framework suffers from poor performance at detecting and classifying pedestrians at long distances from the nUWAy bus. This is due to the limited pixel information at distances further

than 20m as pedestrians are around 30 and 10 pixels in height and width at 20m respectively. An example of this is provided in Figure 27 at a distance of approximately 30m. This could potentially be mitigated by larger resolution images. Using a 1920x1200 pixel image would double the height and width, providing 300% more pixels to process than in the original image. However, this would also reduce the speed of the framework. Pedestrians 20m away are not as important to classifier as closer pedestrians where processing time is crucial so the poor performance at long ranges is negligible to the overall effectiveness of the framework.

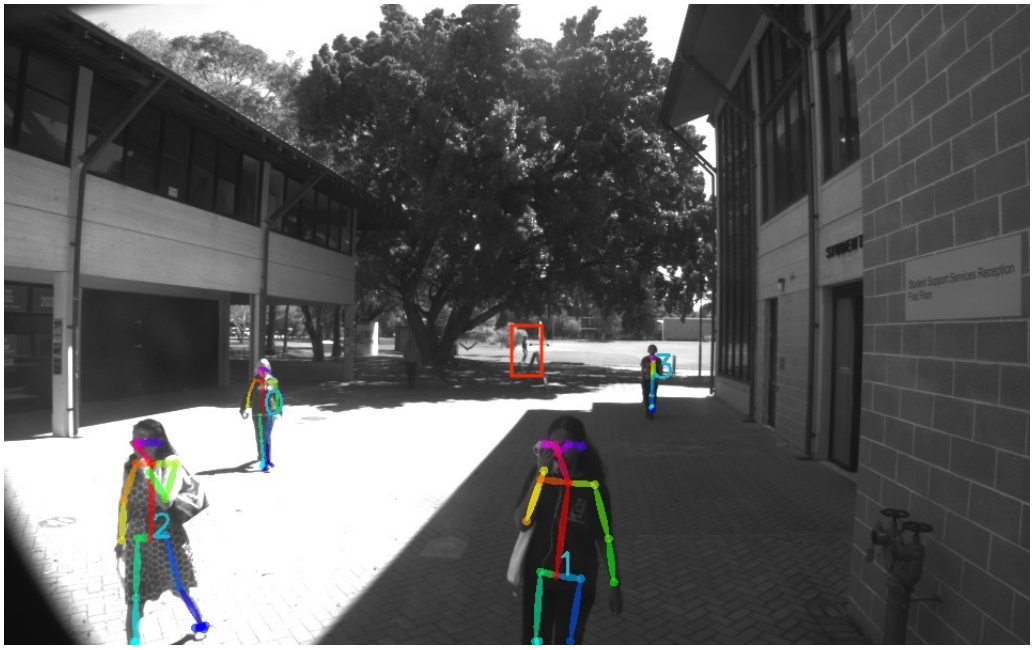

Figure 27: Example of failure to detect pedestrians at long distances. The pedestrian in the red box is 30m away and measures 23 pixels in height and 5 pixels in width.

7.7 Colour Cameras

Colour cameras could be used to further improve the pose detection due to the increased number of features available to the deep learning model. Instead of a greyscale image with only 1 sub-pixel per pixel, it can instead have 3 sub-pixels per pixel (RGB). However, as the size of the images would triple in size, it could potentially slow down the model overall.

7.8 Different Models

OpenPose was chosen as the pose classifier due to the invariance in performance to the number of people in the image. Other pose detection frameworks could be used depending on the number of people detected by the multiped-500 classifier. If the number of people is large, then OpenPose could be used whereas if the number of people detected is small, algorithms such as Convolutional Pose Machines [29] could be used. Additionally, OpenPose with smaller networks on its convolutional stage such as TinyVGG [11] or Mobilenet [36] could be used [37]. The gain in speed from these smaller CNNs could be coupled with a larger image size or colour images to increase the features available to these networks.

7.9 Changes to Physical Environment

Changes to the environment such as time of the year may have an effect on the performance of the posenet classifier. The dataset consisted of images in both overcast and sunny weather but not during rain. In the sunny images, the cameras sometimes had trouble adjusting to the different exposure levels in the shade and in the sun which caused some of the images to be under or over exposed which reduced the capacity of pose detection. Figure 28 provides an example of this. The top frame shows the pedestrian in the red box has been classified but in the following frame, OpenPose is unable to recognise the pedestrian. In this scenario, the camera is in shade while the pedestrian is in the sun which causes the pedestrian to become overexposed and washed out. The pedestrian wearing a black shirt which provides the greatest resistance to overexposure in the sunny area. Despite the dark clothing, OpenPose is no longer able to recognise the pedestrian in the second frame.

To mitigate this issue, changes to the camera settings or image processing would need to be performed. Potential changes to the camera could include reducing the exposure overall to balance the difference in brightness levels. This may underexpose the pedestrians in the shade would reduce the performance of the posenet model.

Figure 28: Effect of partial overexposure due to sunny weather shown in the red boxes. The bottom image is the image directly after the top image.

8 Future Work

8.1 Sophisticated Labelling Techniques

More data for the random forest classifier needs to be collected to be able to further improve the performance metrics of the overall model. As the training set available to the phone focused pedestrian classifier grows, the labelling needs to match in order to increase the accuracy of the model. One method to do this is the let the bus passengers label the images which can also serve as security to prevent bots from accessing the bus's services similar to the methods in

current CAPTCHA verification [38]. Porting the labelling program to a web application and subsequently to mobile devices is crucial for a distributed labelling system.

8.2 Retraining OpenPose

Retraining OpenPose by including images collected by the nUWAy bus could improve its performance metrics. Retraining OpenPose into a more modern deep learning framework such as PyTorch may also result in improved performance metrics, particularly to speed.

8.3 Integration with Path Planning

The scope of this project did not include integration with path planning as at the time of completion, the nUWAy bus did not have a sophisticated path planner. As the UWA campus is relatively flat, this may not be too complex to implement without LiDAR and by using monocular vision. However, edge cases and gradients would need to be accounted for.

8.4 Cloud Computing

As the pose estimation took up the bulk of the computing time, the project could be extended to implement a pipeline that uses 5G or low bandwitdh methods to stream important features of the image to a GPU with greater computational power through the Internet or local network. The time to transfer the information needs to be considered in addition to the improvements in computation time to determine if this would be an effective solution. This method could provide workstation level performance to any embedded computing device in an autonomous vehicle which would massively boost the deep learning capabilities of any autonomous vehicle.

9 Conclusion

The project demonstrated the feasibility and implementation of phone focused pedestrian detection on an embedded system. However, there are still areas for improvement the project failed to meet a requirement of real-time processing. The ensemble posenet method proved to be a slighly better model to use than OpenPose alone with a worthwhile offset in speed with an F1 score of 0.903 and speed of 2.07 FPS. The extra trees classifier on the upper body pose angles demonstrated high performance in multiple metrics with a MCC score of 0.721 and a speed of 10 FPS. Further investigation into model fine-tuning and cloud computing is required for real-time performance of the system while maintaining or improving the performance of the model presented. The key priorities for improvements are accuracy, speed and quality of natively obtained datasets for the model.

10 References

References

[1] M. Bertoncello and D. Wee, *Ten ways autonomous driving could redefine the automotive world*, Jun. 2015. [Online]. Available: https://www.mckinsey.com/industries/automotive-and-assembly/our-insights/ten-ways-autonomous-driving-could-redefine-the-automotive-world#.

[2] N. H. T. S. Administration, *Automated vehicles for safety*. [Online]. Available: https://www.nhtsa.gov/technology-innovation/automated-vehicles#issue-road-self-driving.

[3] A. Reschka, 'Safety concept for autonomous vehicles,' in. May 2016, ISBN: 9783662488454. DOI: 10.1007/978-3-662-48847-8_23.

[4] A. P. Silvano and M. Ohlin, 'Non-collision incidents on buses due to acceleration and braking manoeuvres leading to falling events among standing passengers,' eng, *Journal of Transport & Health*, vol. 14, pp. 100 560–, 2019, ISSN: 2214-1405. [Online]. Available: http://dx.doi.org/10.1016/j.jth.2019.04.006.

[5] J. L. Nasar and D. Troyer, 'Pedestrian injuries due to mobile phone use in public places,' eng, *Accident Analysis and Prevention*, vol. 57, pp. 91–95, 2013, ISSN: 0001-4575. [Online]. Available: http://dx.doi.org/10.1016/j.aap.2013.03.021.

[6] S. M. Simmons, J. K. Caird, A. Ta, F. Sterzer and B. E. Hagel, 'Plight of the distracted pedestrian: A research synthesis and meta-analysis of mobile phone use on crossing behaviour,' *Injury Prevention*, vol. 26, no. 2, pp. 170–176, 2020, ISSN: 1353-8047. DOI: 10.1136/injuryprev-2019-043426. eprint: https://injuryprevention.bmj.com/content/26/2/170.full.pdf. [Online]. Available: https://injuryprevention.bmj.com/content/26/2/170.

[7] D. Stavrinos, K. W. Byington and D. C. Schwebel, 'Distracted walking: Cell phones increase injury risk for college pedestrians,' eng, *Journal of Safety Research*, vol. 42, no. 2, pp. 101–107, 2011, ISSN: 0022-4375. [Online]. Available: http://dx.doi.org/10.1016/j.jsr.2011.01.004.

[8] A. Géron, *Hands-on machine learning with Scikit-Learn and TensorFlow : concepts, tools, and techniques to build intelligent systems*. Sebastopol, CA: O'Reilly Media, 2017, ISBN: 978-1491962299.

[9] J. Schmidhuber, 'Deep learning in neural networks: An overview,' *Neural Networks*, vol. 61, pp. 85–117, 2015, ISSN: 0893-6080. DOI: https://doi.org/10.1016/j.neunet.2014.09.003. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0893608014002135.

[10] F. Bre, J. Gimenez and V. Fachinotti, 'Prediction of wind pressure coefficients on building surfaces using artificial neural networks,' *Energy and Buildings*, vol. 158, Nov. 2017. DOI: 10.1016/j.enbuild.2017.11.045.

[11] K. Simonyan and A. Zisserman, *Very deep convolutional networks for large-scale image recognition*, 2015. arXiv: 1409.1556 [cs.CV].

[12] L. R. Barba Guamán, J. Naranjo and A. Ortiz, 'Deep learning framework for vehicle and pedestrian detection in rural roads on an embedded gpu,' *Electronics*, vol. 9, p. 589, Mar. 2020. DOI: 10.3390/electronics9040589.

[13] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens and Z. Wojna, *Rethinking the inception architecture for computer vision*, 2015. arXiv: 1512.00567 [cs.CV].

[14] Y. Zheng, C. Yang and A. Merkulov, 'Breast cancer screening using convolutional neural network and follow-up digital mammography,' May 2018, p. 4. DOI: 10.1117/12.2304564.

[15] Z. Cao, G. Hidalgo, T. Simon, S.-E. Wei and Y. Sheikh, *Openpose: Realtime multi-person 2d pose estimation using part affinity fields*, 2019. arXiv: 1812.08008 [cs.CV].

[16] D. Dua and C. Graff, *UCI machine learning repository*, 2017. [Online]. Available: http://archive.ics.uci.edu/ml.

[17] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot and E. Duchesnay, 'Scikit-learn: Machine learning in Python,' *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[18] D. Opitz and R. Maclin, 'Popular ensemble methods: An empirical study,' *Journal of Artificial Intelligence Research*, vol. 11, pp. 169–198, Aug. 1999, ISSN: 1076-9757. DOI: 10.1613/jair.614. [Online]. Available: http://dx.doi.org/10.1613/jair.614.

[19] L. Breiman, 'Random forests,' *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, Oct. 2001, ISSN: 0885-6125. DOI: 10.1023/A:1010933404324. [Online]. Available: https://doi.org/10.1023/A:1010933404324.

[20] P. Geurts, D. Ernst and L. Wehenkel, 'Extremely randomized trees,' *Mach. Learn.*, vol. 63, no. 1, pp. 3–42, Apr. 2006, ISSN: 0885-6125. DOI: 10.1007/s10994-006-6226-1. [Online]. Available: https://doi.org/10.1007/s10994-006-6226-1.

[21] N. Dalal and B. Triggs, 'Histograms of oriented gradients for human detection,' in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1, 2005, 886–893 vol. 1. DOI: 10.1109/CVPR.2005.177.

[22] J. Redmon, S. Divvala, R. Girshick and A. Farhadi, 'You only look once: Unified, real-time object detection,' in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 779–788. DOI: 10.1109/CVPR.2016.91.

[23] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu and A. C. Berg, 'Ssd: Single shot multibox detector,' *Lecture Notes in Computer Science*, pp. 21–37, 2016, ISSN: 1611-3349. DOI: 10.1007/978-3-319-46448-0_2. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-46448-0_2.

[24] Z.-Q. Zhao, P. Zheng, S.-T. Xu and X. Wu, 'Object detection with deep learning: A review,' *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 11, pp. 3212–3232, Nov. 2019, ISSN: 2162-2388. DOI: 10.1109/tnnls.2018.2876865. [Online]. Available: http://dx.doi.org/10.1109/TNNLS.2018.2876865.

[25] Y.-H. Lin, Y.-C. Lin, Y.-H. Lee, P.-H. Lin, S.-H. Lin, L.-R. Chang, H.-W. Tseng, L.-Y. Yen, C. C. Yang and T. B. Kuo, 'Time distortion associated with smartphone addiction: Identifying smartphone addiction via a mobile application (app),' eng, *Journal of psychiatric research*, vol. 65, pp. 139–145, 2015, ISSN: 0022-3956.

[26] A. Rangesh, E. Ohn-Bar, K. Yuen and M. M. Trivedi, 'Pedestrians and their phones - detecting phone-based activities of pedestrians for autonomous vehicles,' in *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, 2016, pp. 1882–1887. DOI: 10.1109/ITSC.2016.7795861.

[27] A. Newell, K. Yang and J. Deng, *Stacked hourglass networks for human pose estimation*, 2016. arXiv: 1603.06937 [cs.CV].

[28] A. Rangesh and M. M. Trivedi, *When vehicles see pedestrians with phones:a multi-cue framework for recognizing phone-based activities of pedestrians*, 2018. arXiv: 1801.08234 [cs.CV].

[29] S.-E. Wei, V. Ramakrishna, T. Kanade and Y. Sheikh, *Convolutional pose machines*, 2016. arXiv: 1602.00134 [cs.CV].
[30] Easymile, *Ez10 passenger shuttle*, 2020. [Online]. Available: https://easymile.com/vehicle-solutions/ez10-passenger-shuttle.
[31] F. Systems. [Online]. Available: https://www.flir.com.au/products/grasshopper3-gige/.
[32] *Jetson AGX Xavier Developer Kit*. Nvidia. [Online]. Available: https://developer.nvidia.com/embedded/jetson-agx-xavier-developer-kit.
[33] Stanford Artificial Intelligence Laboratory et al., *Robotic operating system*, version ROS Melodic Morenia, 23 May 2018. [Online]. Available: https://www.ros.org.
[34] G. Bradski, 'The OpenCV Library,' *Dr. Dobb's Journal of Software Tools*, 2000.
[35] D. Chicco and G. Jurman, 'The advantages of the matthews correlation coefficient (mcc) over f1 score and accuracy in binary classification evaluation,' *BMC Genomics*, vol. 21, Jan. 2020. DOI: 10.1186/s12864-019-6413-7.
[36] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto and H. Adam, *Mobilenets: Efficient convolutional neural networks for mobile vision applications*, 2017. arXiv: 1704.04861 [cs.CV].
[37] L. Mai, *Hyperpose*, Nov. 2020. [Online]. Available: https://github.com/tensorlayer/hyperpose.
[38] W. Hasan, 'A survey of current research on captcha,' *International Journal of Computer Science Engineering Survey*, vol. 7, pp. 1–21, Jun. 2016. DOI: 10.5121/ijcses.2016.7301.