University of Southern Queensland

Faculty of Health, Engineering & Sciences

Pre-Emptive Detection of Mobile Phone Use In-Vehicle

A dissertation submitted by

Andrew Hopkins

in fulfilment of the requirements of

ENG4112 Research Project

towards the degree of

Bachelor of Electrical & Electronic Engineering

Submitted: October, 2020

Abstract

Distraction while driving currently makes up 14% of crashes and 10% of fatalities on roads in NSW, Australia on average every year (Transport for NSW 2020*a*). Through the increase in reliance on mobile devices, these numbers are sure to increase. There are currently methods involved in the reduction of this issue, however there is currently nothing in place for detection while driving.

This research works to prove the hypothesis that using a fixed camera installed in a car, and with some image pre-processing and machine learning algorithms applied, a machine can detect the act of reaching for a phone. To prove this hypothesis, the research required a database to be created, and algorithms used in order to determine the accuracy of detection.

With the analysis of previous literature, this research was able to identify some key methods required for classification of pre-emptive mobile phone usage. These techniques were tested against a database created with varying vehicle types and driver behaviour. The results of the trained models were then tested against a previously "unseen" data set, to verify the accuracy of the machine.

The results of this project showed promise, with three different machine learning techniques applied against unseen data. Each reviewed 240 samples of reaching events as well as a varied length of driving videos. The bagged tree method detecting 27.92% with a false positive rate of 3.45 per minute, the cosine KNN method detecting 29.17% with a false positive rate of 4.74 per minute and the cubic SVM method detecting 29.17% with a false positive rate of 6.13 per minute.

This research was limited by hardware capabilities, as well as data limitations. With an increased database across multiple vehicle types and drivers, the results would show more

significance. Further research in this area would also be required to limit the time of which the pre-processing algorithms required, as 1 second of data is still currently requiring 2.8 seconds in processing time.

Suggestions were made that additional research with a more varied database should be conducted to ensure validity across different vehicle types and driving styles. With more significant training conducted, a machine could then be developed in order to make these calculations real time and test them against a driver in a vehicle. This technique would show significant importance in detecting the shortfalls of the machine training and identifying different areas for potential improvement. Additional trials on pre-processing techniques were identified and some ideas around the training to identify more than just three categories to help in determining areas of which data is lacking.

University of Southern Queensland

Faculty of Health, Engineering & Sciences

ENG4111/2 *Research Project*

Limitations of Use

The Council of the University of Southern Queensland, its Faculty of Health, Engineering & Sciences, and the staff of the University of Southern Queensland, do not accept any responsibility for the truth, accuracy or completeness of material contained within or associated with this dissertation.

Persons using all or any part of this material do so at their own risk, and not at the risk of the Council of the University of Southern Queensland, its Faculty of Health, Engineering & Sciences or the staff of the University of Southern Queensland.

This dissertation reports an educational exercise and has no purpose or validity beyond this exercise. The sole purpose of the course pair entitled "Research Project" is to contribute to the overall education within the student's chosen degree program. This document, the associated hardware, software, drawings, and other material set out in the associated appendices should not be used for any other purpose: if they are so used, it is entirely at the risk of the user.

Dean

Faculty of Health, Engineering & Sciences

Certification of Dissertation

I certify that the ideas, designs and experimental work, results, analyses and conclusions set out in this dissertation are entirely my own effort, except where otherwise indicated and acknowledged.
I further certify that the work is original and has not been previously submitted for assessment in any other course or institution, except where specifically stated.

Andrew Hopkins

Acknowledgments

I would like to thank Tobias Low for offering guidance and assistance as my supervisor over the length of this project. Notably for the level of feedback at different stages throughout the works, which led to both my presentation and project success.

I would also like to thank my partner Emma Moore, for the assistance in recording, as well as allowing me to use her vehicle for recording. As well as Emma's sisters Georgia Moore and Laura Moore for allowing me to use their vehicles.

Andrew Hopkins

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Overview

This chapter outlines the background information that the project is built around. This includes the dangers of driving, some current methods of protection and the identified problem with the current scenario. This is followed by the aims and objectives of the project and concluded with some of the expected benefits and outcomes that are to come from the completion of the project. This project aims to determine if the pre-emptive detection of mobile phone usage is feasible using image pre-processing techniques and machine learning algorithms.

1.2 Background

1.2.1 Dangers of Driving

Driving poses risks for many reasons, with the leading causes of road accidents including speeding, driving under the influence of drugs/alcohol and distractions. It has been found that a 1% increase in speed results in a 4% increase in crash fatality risk. In a car-to-car side impact, there is a fatality rate of 85% for cars travelling at 65km/h. Driving under the influence of alcohol varies significantly depending on the user and their reaction to the substance. Results have shown a factor of five times more likely to be involved in a crash when under the influence of amphetamines.

Libya is a great example for the danger of driving, where road rules are often no followed and standards are lower and less strict for what passes as a road worthy vehicle. Infromation gathered by the World Health Organization shows Libya leading the most road traffic deaths in 2015, with approximately 73 deaths per 100,000 population estimated, double that of any other country. These numbers help to show the requirements of enforcement and strict road rules. It should be noted that at the time, the law did not prohibit the use of a mobile phone while driving in Libya. As of 2015, there were 138 countries that had banned the use of hand-held mobile phone use while driving. This has increased to 150 countries as found in the report released in 2018 (World Health Organization 2018).

Distraction while driving is a growing issue, with mobile phone usage a large concern. The World Health Organization have found that using a mobile phone makes a driver four times more likely to be in a crash than drivers not using a mobile phone (World Health Organization 2020). This is due to a slowing down of reaction time and difficulty staying in the correct lane. More concern comes from statistics released in 2018 by U.S. department of Transportation of which a report from the National Highway Traffic Safety Administration shows that 10% of all fatal crashes caused by distraction were from drivers between the age of 15 to 19 years (US Department of Transportation 2018). A value of 10% of the population over a four year range, one of which is under the supervision of a licensed driver, shows overwhelming statistics of the effect of distraction while driving.

1.2.2 Methods of Protection

The below figure shows the importance of road protection. It shows a significant decay in road crash fatalities since the enforcement of seatbelts, random breath testing (RBT) and speed limits. This was also resolved with the introduction of speed cameras in 1991, and a graduated licensing scheme in 2000 (Transport for NSW 2020*b*). Without these methods the fatality rate would have only climbed, especially with the requirement for work self-transport increasing.

Other methods of protection include the overall protective framing of the cars, with new and improved devices such as shatter resistance glass, anti-lock braking systems, stability control and airbags. These features continue to grow year by year, with the prospect of driverless cars seeming more feasible. NSW has recently introduced the use of traffic light cameras that can detect people using their mobile phones.

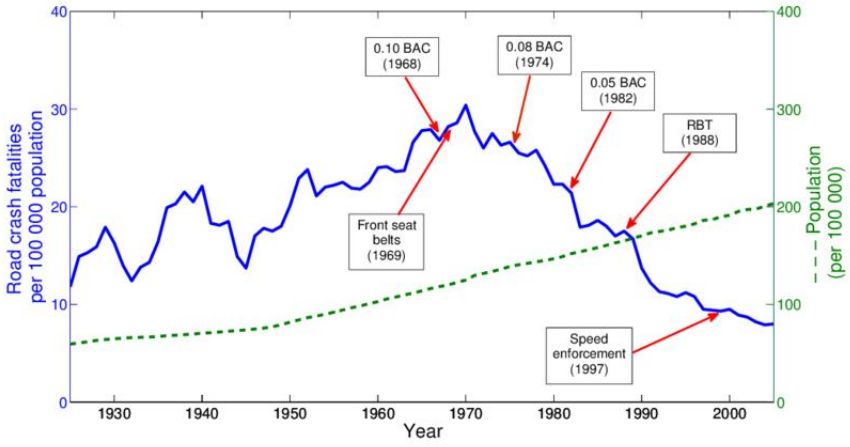

Figure 1.1: Evolution of population and road fatalities in Australia from 1925 to 2005 (Larue & Gregoire 2010)

Many methods aim to protect the people in the vehicle if it were to come into a situation that may result in an accident, while many others protect the people in the vehicle after an accident has occurred. However, the most critical methods of protection seem to be the deterrent ones, such as RBT and speed camera, that reduce the number of accidents on the road, not the severity. For the safety of drivers and pedestrians, it is more important to work on the safety features that prevent an accident from occurring, rather than a soft control such as the ones mentioned above. While commonly related to the work industry, the hierarchy of control is a good tool to show the importance of elimination. It shows the level of importance, where elimination (removing the possibility to use a phone while driving) would be rated at much higher effectiveness than engineering controls (shatter resistance glass and airbags) that are used to minimise the effect after an accident has occurred.

1.3 Problem

While there are methods of deterrent currently in place for speeding and driving under the influence of drugs and alcohol, the detection of mobile usage while driving is rather minimal. Drivers know the difficulty in being caught using a phone in any area other

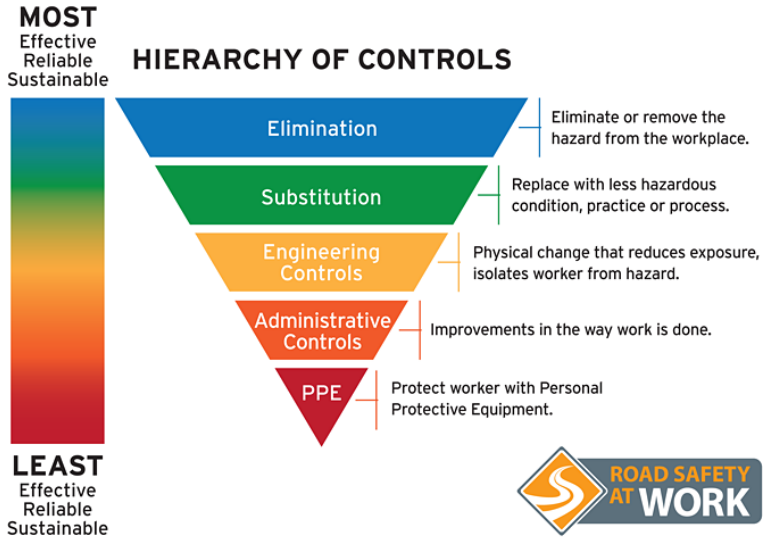

Figure 1.2: Hierarchy of Controls (Road Safety at Work n.d.)

than traffic lights (due to the install of NSW new detection cameras). Due to this, drivers are still comfortable using their phones while driving, putting themselves and the people around them at risk.

Without the development for a method to detect the use of mobile phones at all times while driving, it is unlikely that the situation will get better, especially with a younger generation coming through that have always grown up with mobile phones. The numbers of road crashes, injuries and deaths will be likely to increase per 100,000 population with younger drivers coming onto the roads in the years to come.

With advancing technology and the exposure that the current generation of children get to different devices, it is becoming identified as "acceptable" to use a mobile phone while driving. The risk increases significantly with this generation coming through, with previous data from U.S. Department of Transportation showing that the age group of 15 to 19 years old are accountable for 10% of all road accidents caused by distraction.

1.4 Aim and Objectives

The aim of this project is to determine if machine learning and image processing techniques will allow for the rapid detection of pre-emptive mobile phone usage while driving. That is, the algorithms will be working to detect a driver reaching for their phone, instead of detection after it has already been used, or while in use. The requirement for this to

be high speed is due to it only taking seconds to reach for a phone.

The hypothesis is that the pre-emptive detection of mobile phone usage can be achieved, although it will be rather difficult to get the algorithms and techniques down to an output speed at which the results are calculated before the phone has been picked up. A likely result is that over-detection will occur, and the driver will be "suspected" to be reaching for their phone when they change gears or scratch their leg for example.

It is expected that a rather large database will be required in order to make the detection affective across all different vehicle types and drivers. While this is not achievable, the hypothesis is that the pre-emptive detection can work in a hand full of different vehicles with one driver. Once completing this a larger scale of the project would be required to cover other actions, driving behaviours and vehicle types etc, although this work is out of the scope of this project.

For the system to be feasible, the time taken to process an image for categorization needs to be less than 0.5 second from the event occurring. This is to ensure that enough images can be detected as "reaching for phone" in order to improve the confidence of the output. Without this time period being met, proof of the method and training is possible, but more work would need to be done in the area of optimization to achieve the goal. While pre-processing may take longer than this in the early stages of the project, work will be conducted to reduce this processing time, after all it is more important to prove that it can be done first, before optimizing the system and processes taken to make it possible.

Below are a list of the major objectives of the project:

1. Obtain legal footage of road users using their mobile phones as well as just driving normally.
2. Review and categorize footage.
3. Use MATLAB deep-learning tool to set a benchmark on video with minimal processing.
4. Find shortfalls in deep-learning tool and areas for optimization.
5. Apply multiple levels of processing on images, to compare to the benchmark approach.

6. Use MATLAB deep-learning tool to see improvements.

7. Test finalized methods on previously "unseen" data

1.5 Outcomes and Benefits

This project intent is to help to determine the effectiveness of using machine learning algorithms and image processing techniques to detect the use of mobile phones in-vehicle. It will help highlight that the technology is there, and there could be a method of protection to be introduced into the car industry in the near future. It will show that the detection of mobile phone usage, like speeding and driving under the influence, can have a deterrent of similar strength. If it is successful it could be further developed to stop the use of mobile phones while driving, before the phone is even touched.

The outcomes of the project will be:

1. Increased understanding of the ability to further detect mobile phone usage while driving.

2. Increased understanding of the ability to perform pre-emptive detection with machine learning.

3. Encourage further development and research in the area of in-vehicle mobile phone detection.

If successful, the development of this research could stretch further than just the car industry. With concerned parents potentially given the ability to install a small unit for the detection while their children are driving. This will not only ease the minds of parents but also encourage the upcoming generation to stamp out the bad habit.

Any advancement that this work could make to the implementation of in-vehicle detection of mobile phone usage would have a significant impact on the safety of road users, including drivers, pedestrians and cyclists. Further study in this area past my initial project could result in life saving installations for new or old vehicles.

Chapter 2

Literature Review

2.1 Overview

To ensure the feasibility of this project, a literature review has been conducted. This is to investigate and understand the benefits of a range of different machine learning algorithms, image processing techniques, and some methods to optimize speed and accuracy in both areas.

These research areas are quite broad. Due to the early stages of the project requiring an overview and comparison of all techniques, before narrowing this down to the more suitable ones for this project. These areas have been grouped into their main areas, leaving four categories of different machine learning algorithms.

Similarly, image processing techniques have been reviewed rather broadly, with intent to focus this more into the areas of importance as the project advances. Research into both main areas are rather limited in terms of both mobile detection and pre-emptive detection, showing a real gap in the area. While this is the case, there is still significant research in the areas of machine learning and image processing techniques that will allow this project to remain feasible.

2.2 Machine Learning Algorithms

Machine learning is an application of artificial intelligence (AI), which began development in the 1950's. This was able to grow over time as machines became more powerful and the cost of computing became cheaper. The first found implication of machine learning on road safety came from a paper about determining the road suitability for the transportation of dangerous substances by J.M. Matías and J. Taboada and C. Ordóñez and P.G. Nieto (Matías, Taboada, Ordóñez & Nieto 2007). Machine learning for vehicle safety has only continued to grow, with papers predicting motor vehicle crashes, correlation of driver distraction and intentions, pedestrian collision avoidance and many more. With the need to travel to work and the dangers of driving as traffic and road speeds increase, and the increasing population density, the need for machine learning for road safety is growing.

Below is a broad review of the different categories of supervised machine learning algorithms for classification. This review has included a large range of information to ensure the categories are understood before proceeding with the project, and ruling algorithms out. By correctly understanding all algorithms and the methods to optimize them, the final output of the project is likely to result in higher accuracy rates with faster classification speeds. The main algorithm classes that have been researched are; decision trees, support vector machines, discriminant analysis and nearest neighbour classifiers.

2.2.1 Decision Trees

While reviewing a paper by Fang, Gao and Zhao, the significance of decision trees was examined. The results of this experiment showed an 83.7% accuracy while using a combination of decision trees and other machine learning techniques. The paper also displayed the ability to combine decision trees with other methods when the outlying factors become more significant. This is an area that will be taken into heavy consideration, as the first step of recognition may be much faster with decision trees. This paper on sign language recognition was useful in determining some different methods to detect video-based classifiers. (Fang, Gao & Zhao 2003). Figure 2.1 displays the way in which a decision tree can be broken down into multiple categories, at which point it can be taken over by different machine learning techniques to categorize from that point.

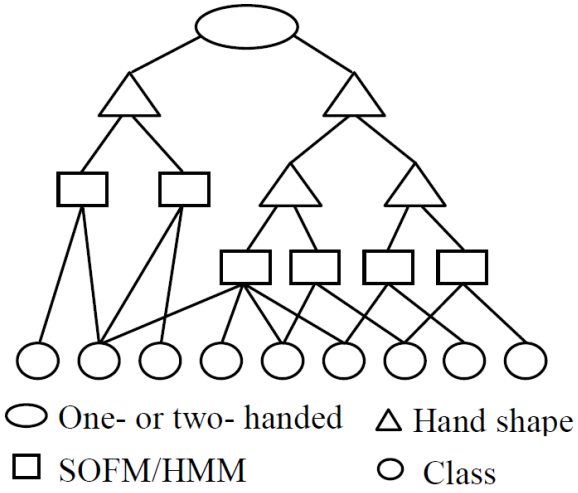

Figure 2.1: The hierarchical decision tree for sign languauge recognition (Fang et al. 2003)

2.2.2 Support Vector Machines

Support vector machines (SVM) can be used for classification by determining a parallel slab on a plane to separate two classes (Xu, Li, Fang & Zhang 2018). Due to this, and the fact that this project separates data into three classes, the SVM algorithm will require a two-step process to work correctly. This information is critical in early phases as it is important to recognise which classifications will be "grouped" at the first stage. Figure 2.2 displays how an SVM detects the optimal plane, by separating the two classifications by with the larges slab, in multiple dimensions.

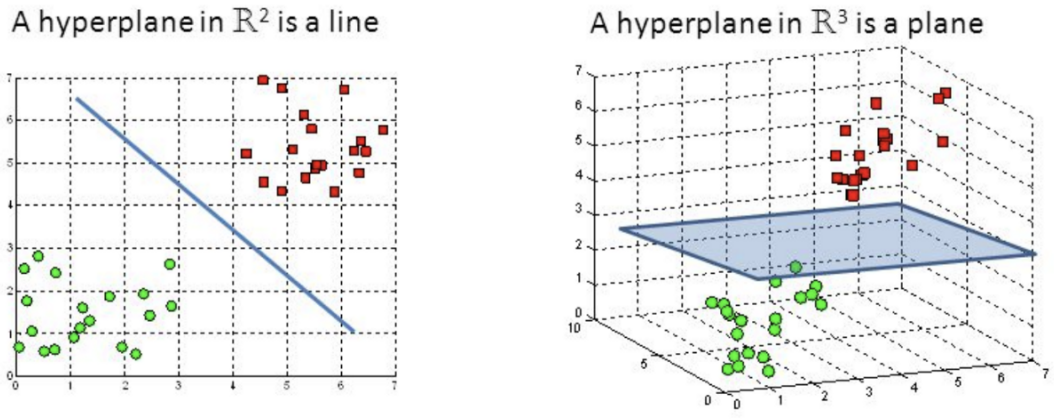

Figure 2.2: SVM displayed in 2-D and 3-D data plot (Xu et al. 2018)

It can be seen in both planes, that the SVM algorithms detects the largest field between the classifiers separated by parallel slabs or lines. As stated earlier, this can only apply to two classifications, so in order to make this algorithm work, it will be important

to determine the largest separator between all three classification, while grouping two together. After separating the most significant classifier, the final two can be again tested to determine the best separator.

It is important to note that while the SVM may require multiple layers in order to be successful, it showed an accuracy of 93.5% on Honghua's work on recognition of human movement. This paper included 9 different classifiers and required significant learning time.

Significance is seen in this approach as for potential classification of a low movement frame may be easily detected by an SVM. It may be found to be a useful classifier for the early stages, as a significant separation may be detected between classifiers like driving with minimal image movement, and a largely moving image flow such as "reaching for phone" or possibly abnormal driving events. These minimal movement frames can potentially be ruled out at a much earlier stage with the help of SVM's, significantly speeding up the speed of classification.

2.2.3 Discriminant Analysis

A paper on the linear discriminant analysis of human movement showed a great insight to the effectiveness of the method. It is a closely related method to regression analysis, as it attempts to create an expression where one variable is a linear combination of others, where in this case the dependant variable is the classifier. That is, it is trying to create a linear relationship between a classifier and the dependant variables input. One paper showed an effective insight into this by gaining an accuracy of 83.47%, while also comparing the outputs of multiple different methods of discrimination. This paper has found extremely useful as it displays some significant methods of not only machine analysis, but also image processing methods and frame scanning techniques. It should be noted that this analysis method and SVM show similar techniques, the main difference is that SVM assumes that all groups are totally separable, where discriminant analysis assumes data is normally distributed (Clemmensen, Hastie, Witten & Ersbøll 2011). Figure 2.3 shows how linear discrimination is used to find the best separation of two classifiers, note the significant differences from SVM. Showing that the second image is the favourable linear relationship as it has significantly less overlap between the two classifiers.

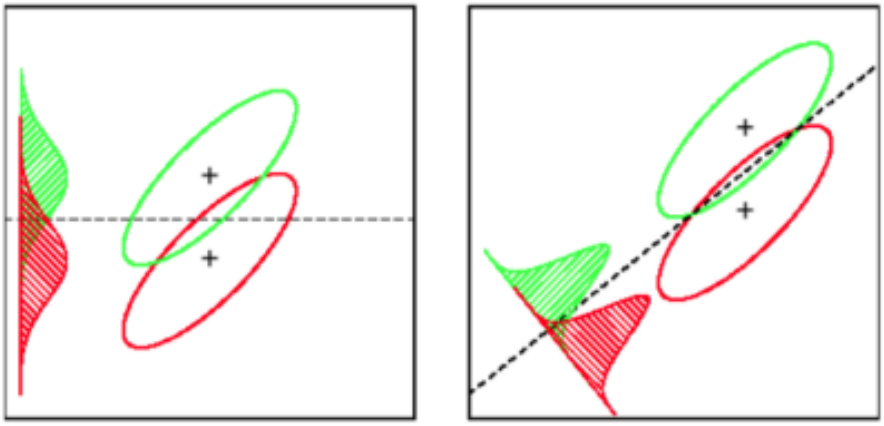

Figure 2.3: Linear Discimination Analysis of Two Classifiers ((https://stats.stackexchange.com/users/177677/renel chesak) 2019)

Discriminant analysis may find useful in similar ways to SVM's. If the case arises that seperation cannot be detected by an SVM, disciminant analysis can be used in its place. It should be noted that if there is an overlap of information, it is known that accuracy will be effected. The aim is to keep the effect to a minimum and ensure a fast algorithms. This information of the operation of a discriminant analysis will be useful in determining what stages it will be effective and when it should be avoided at further stages in the project.

2.2.4 K-Nearest Neighbor Classifiers

In a book by Hastie, Tibshirani and Friedman, they discuss the elements of statistical learning. Insight has been found into the method of nearest neighbour classifications. Due to the nature of these classifications, it has been found that the method, while it has its other benefits, will be rather slow in comparison to others (Hastie, Tibshirani & Friedman 2009). The figure below shows how the technique works as a 2D plot. A line separates the classifications based on k-nearest neighbours (KNN). The issue being that the plotted line cannot be defined, and for every variable, it is required to use all other data points to find the closest value and classify itself. Given the time constraints of this project, for live detection, the output result of this classifier will be too slow. As more work goes into building a database to train, this approach will become slower and slower as a method for classification.

As seen the Figure 2.4 the test value is classified by the 15th nearest neighbours, and

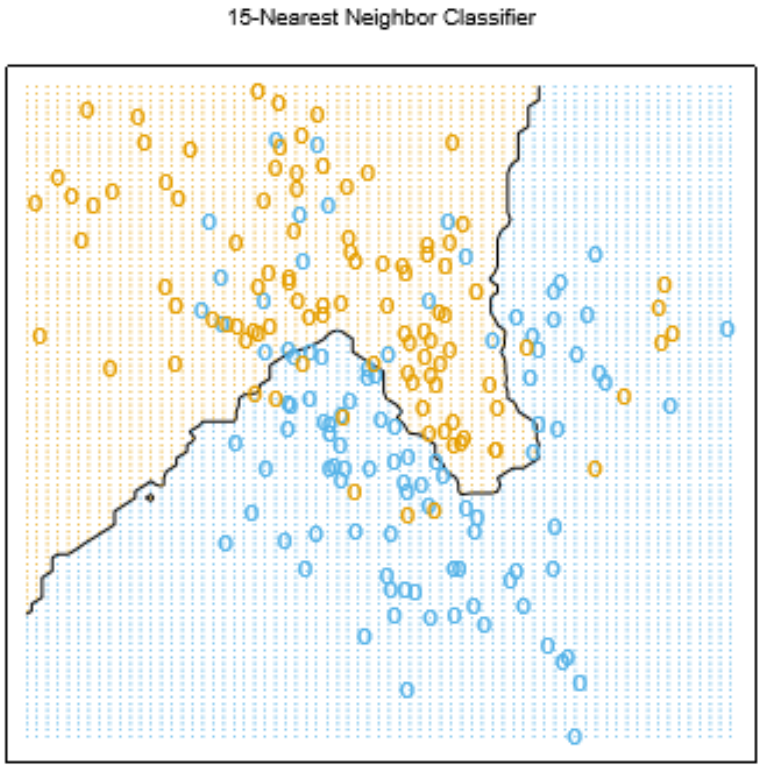

Figure 2.4: Classification by 15-Nearest Neighbors (Hastie et al. 2009)

given the complexity and large dataset, it is impossible to give a value to the line. For each new set of values to be classified, it must compare its distance between all other data values to determine its classification, resulting in a lengthy output time. Due to a large data set to be required for the learning of this complex idea, it is unlikely that the KNN algorithm will be quick enough to output a response, however, without any testing completed, ruling it out too soon would be unnecessary. It may be found a useful tool after using a significant amount of pre-processing. Due to its potential to test all classifiers in one algorithm, it may also be found to be quicker that SVM and decision trees, due to their multi-step process.

2.3 Image Processing

Image processing is an important step in ensuring the data that is received by the machine is simplified and can hold some significance. This enables the machine learning algorithm to work more efficiently. For example, feeding the machine image data, as opposed to flow data would be extremely difficult to comprehend, and likely result in very low accuracies.

Just as you wouldn't feed a machine classifying plant types, flow data of the video as opposed to image data. Image processing takes many steps, as simple as cropping of data which will not be covered, to more in-depth techniques such as motion energy images, motion history images and optical flow energy images.

2.3.1 Motion Energy Images

A motion energy image simplifies an image in a significantly important way for this project. It finds the change from frame to frame, and outputs a binary image (Bobick & Davis 2001). This is very simple output data that becomes rather useful as the aim of the project is to detect the reaching for a phone. Figure 2.5 that explains the use of this technique. This technique may be difficult to implement based on the vehicle movement, any slight movement of the camera would result in all ones in the motion energy image.

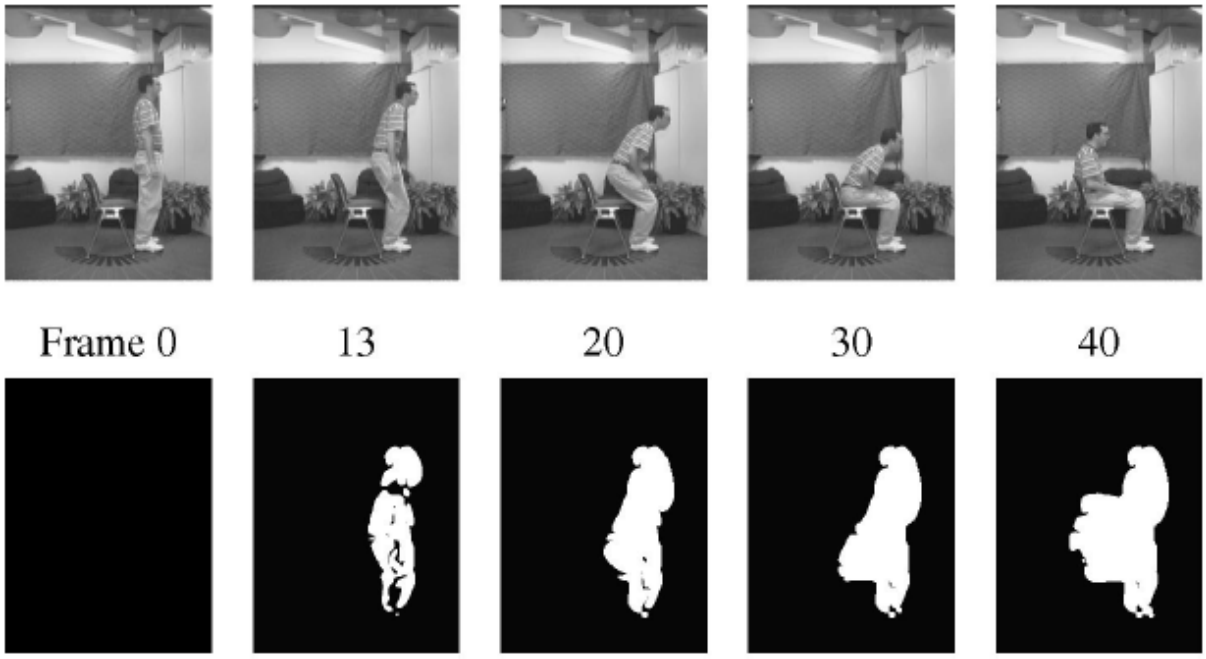

Figure 2.5: Example of a Motion Energy Image (Bobick & Davis 2001)

2.3.2 Motion History Images

Motion History Images hold a similar technique as the energy images. Whereas they output a grayscale image based on movement over multiple frames. These are grayscale as the intensity is an indication of how recently the movement occurred. If the movement occurred in the most recent frame, it would be white. The last frame within the buffer

size would be indicated by a dark grey. This method may also prove useful and is simply an extension of the motion energy image.

This method can have as many or as few previous frames displayed in the history image, which is usually determined by the speed at which movement is generally measured at for the application that it is being used under. For this project, 30 frames would be too large of a history image as this is the time it takes to reac the phone, so potentially somewhere around the 10-15 mark would work well.

2.3.3 Optical Flow Energy Images

Optical flow energy images are a representation of moving framesusing a matrix of vectors. The matrix contains the standard layout of the image, while the vectors inside it determine the direction and magnitude of the movement from the previous frame. This method involves an insolvable equation, and hence creates the need for "restrictions" to be set, allowing the equation to be solved (Bobick & Davis 2001). This has resulted in multiple different solutions or methods, one of which exists in MATLAB as an image processing tool already. Figure 2.6 below shows the tool used in MATLAB, laid back over the top of the last scanned frame.

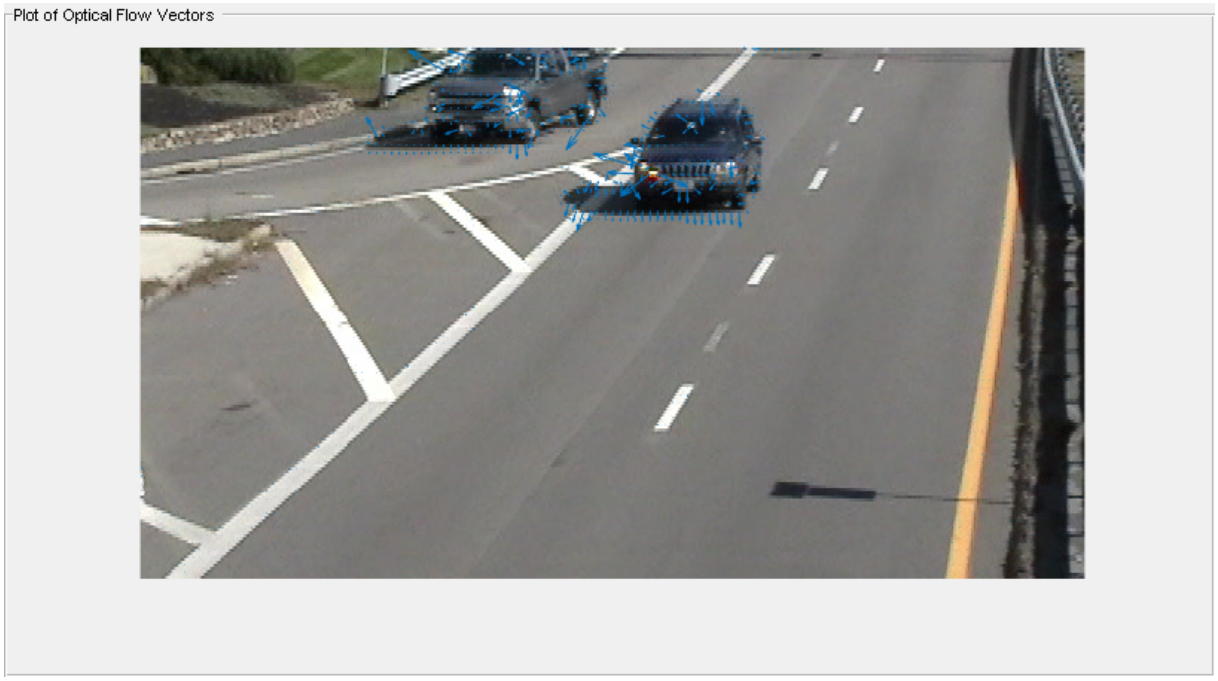

Figure 2.6: Optical Flow Output Layed Over Current Frame (MathWorks 2020)

2.4 Machine Learning Pre-Processing

Data pre-processing is an important step in order to ensure the data passed to the machine learning algorithms has removed any useless data and outputs remaining data in a simple form. This enables the machine to train and identify the classification at a faster rate, and in some cases at the cost of some accuracy. Below are some different methods of pre-processing that may prove useful for the project.

2.4.1 Feature Selection

Feature selection is the process of identifying and removing as much irrelevant and redundant features as possible(Kotsiantis, Kanellopoulos & Pintelas 2006). These features are generally categorized in three ways; Relevant - Significant influence on the output Irrelevant - No influence on the output, data seems to be random for each example Redundant - Feature can take the role of another (doubling up on the same data)

Some filtering can take place with the use of categorization as defined by Blum and Langley(Blum & Langley 1997); Distance - Feature A is preferred over feature B if A creates a greater distance between two classes Information - Feature A is preferred over feature B if the information gain from A is greater (not so useful for this project) Dependence - Uses correlation between a classifier and the feature, higher correlation is preferred. Consistency - Two samples are in conflict if they share values but disagree in the class that they represent (random data)

Chapter 3

Methodology

3.1 Overview

This chapter outlines the aims and objective with more detail, this is an important step in the methodology to ensure the correct methods are followed to achieve the objectives. After the completion of the aims and objectives, it is important to outline the limitations to ensure the project can be adequately complete in the timeframe with minimal interference.

Certain objectives for this project are critical, such as data acquisition. Without this step achieved in a timely manner, it becomes very difficult to set a benchmark and detect areas of which image processing and machine learning algorithms can be adjusted for optimization of the results.

Other areas such as trialling different image processing techniques will be done along the length of the project, and if time permits, more effort will be put into this area when finalizing the project.

This chapter ensures that project is set out with all aspects planned while using different tracking and safety tools, such as a timeline and a risk assessment.

3.2 Outline of Objectives

These were covered in the Introduction, although a more detailed explanation of the objectives are as below:

1. Obtain legal footage of road users using reaching for their mobile phones as well as just driving normally.
    - Data must be evenly distributed, to ensure that over classification is not caused by having a dominant sample.
    - This data will require simulation to ensure no road rules are broken. Data will be collected by myself with the use of a fixed camera to ensure all data is similar.
    - The data will be recorded on a Samsung S8, with a frame rate of 30 fps.
    - More data may be recorded through the length of the project, as data is optimized or results show lacking samples.
    - Data will be varied by the following:
        - Time of day
        - Vehicle used
        - Clothing
        - Position of seat
        - Passenger in vehicle
        - Variety of driving techniques
        - Variety of phone positions
    - Figure 3.1 below is a representation of the cropped image to be used. Important features are as below:
        - Steering wheel
        - Lap area of driver
        - Leaving out window, as this results in false data due to simulation

Figure 3.1: Image outlining camera positioning and resulting image

2. Review and categorize footage

   - Videos will be reviewed frame by frame to detect the moment at which the classification changes.
   - Videos will end after the phone is reached for, as it is overcomplicating the machine to train it to see phones as well.
   - Data is to be categorized into three separate categorizations, these are "Driving", "Reaching" and "Abnormal".
   - Driving includes only normal driving, such as turning corners, three-point turns and indicating.
   - Abnormal is for driving that looks more like reaching for the phone, these include:
     - Reaching for stereo
     - Reaching for gear stick
     - Applying hand/park brake
     - Scratching leg
   - Reaching includes a variety of techniques of reaching for a phone including:
     - Between legs

  - In either pocket
  - Positioned somewhere on dashboard
  - Positioned in or around the glove box
  - Positioned on the passenger's seat

3. Use MATLAB deep-learning tool to set a benchmark on video with minimal processing.

   - There is a possibility that data will require cropping and data averaging or quantization to allow for the first stage of testing.
   - The requirement for this cropping will be determined in the trial phase of machine learning.
   - This benchmark will be achieved using the Horn-Shunck method, which is a solution for the estimation of optical flow between two image frames. This is a build-in function that already exists in MATLAB.
   - Multiple different machine learning techniques will be implemented and carried through the length of the project for consistency.

4. Find shortfalls in deep-learning tool and areas for optimization.

   - As suggested, this will be based around data averaging or quantization and results of the earlier objective.
   - Such areas might detemine that one category did not have enough samples, or one had many more than others.
   - Reviewing data frame by frame might identify areas where further cropping can occur.
   - Further cropping will allow for less averaging or quantization, getting more value out of the left over data. This stage may take multiple steps to identify the significant pixels in each sample.
   - With significant optimization occuring, more data such as a motion history image, or the original image itself may also be included in the training.

5. Apply multiple levels of processing on images, to compare to the benchmark approach.

   - Cropping will occur before the first stage of training, with further potentially applied after initial training is complete, pending feature selectiong success.

- The first stage is through the use of optical flow energy images, other techniques will be applied and compared, such as motion history and motion energy.
- This objective will be applied multiple times, as the results will determine the next step.

6. Use MATLAB deep-learning tool to see improvements.
   - Using more in-depth optimizations of the machine learning techniques.
   - If frame data size can be minimized, more technical options may become available with the memory freeing up.
7. Test finalized methods on previously "unseen" data
   - This data will be recorded in correspondence with the rest, ensuring it is still of the same standard.
   - Data used at this stage will not have been seen by any machine learning through the entire project.
   - This method will eliminate any machine or human bias of data categorization.

3.3 Limitations

A significant limitation in this process is the retrieval of a database. While there are databases in existence for the detection of fatigue, there is a requirement for the images used in this project to look at hand movement more than facial expression. Due to this there is a requirement to develop a database. Growing onto this limitation is the development of COVID-19, and the impact it is having on the ability to get data of driving while following the order, as well as the access to other vehicle types. Data collection will be time consuming, as will the categorization stage. For this reason, it is important to ensure that only necessary data is recorded and categorized, by collecting it at an "as required" basis.

The second most significant limitation is hardware resources. Limitation to the hardware are based around the current computer in possession, due to there being no financial support, and the task being conducted externally to the university. These limitations in hardware will simply slow down the operating time of the processing of data and machine learning, and potentially result in some machine learning techniques not to be able to

handle the large data size. It may result in earlier stages of training requiring cropping and/or quantization, which will be determined through some trial training of built in databases.

3.4 Data Collection

Data collection is a significant area in this project, as there is a requirement for successful machine learning to learn from multiple differemt driving techniques, in different vehicles, with other variables changing, such as clothing and time of day. This data collection will likely occur over the length of the project as the more varied the data that is collected the more effective and realistic the results of the final machine will be.

It is important to ensure that all data collected is similar enough for a simple layer of image processing to eliminate unwanted data. This will require the use of a mounting devide to ensure that all images captured display similar frame positioning. This data will then be reviewed, and any video that does not meet the requirements will need to be removed from the database. Data collection will be done as widespread as achievable, across the length of the project to ensure a good variety of information is received.

While collecting this data, it is also important to note that for legal reasons, the data will not be "real" data of people driving and reaching for their phones. This data will have to be staged, in which real driving will occur, with a passenger in the vehicle starting and stopping the recording, and reaching events will be simulated while the vehicle is stationary.

It is also a significant requirement to ensure that there is an even amount of data samples across the three categories. The abnormal samples are important as they include similar features to that of reaching for a phone. It is important to ensure that not every situation that a hand is moving away from the steering wheel is categorized as "Reaching".

As mentioned in the objectives, the same camera will be used throughout the length of the project, and positioning in the vehicles will be similar to ensure that the project can be further expanded to different varieties of vehicles in the future. The frame rate is 30fps, and important features of data include the lap area of the driver, the steering wheel and the absence of the window.

3.5 Project Legalities

The current legal issues that this project will run into are highly affected by these two areas; road rules and the current COVID-19 pandemic. The information below outlines how these areas are being addressed to ensure that no rules are broken through the collection of data.

In terms of data collection around the COVID-19 pandemic and its current changes to the laws. These new laws have required residence of NSW to stay at home unless going to; work, school or an educational institution, shopping for food or essentials, getting medical supplies/care or exercising (NSW Government 2020). Local law enforcement has been contacted regarding the matter to ensure the correct steps were taken, and laws were correctly understood. Under these new rules, and due to the classification of this project as mandatory study, I have been granted to drive around for the collection of this data.

Similarly, there are many factors of data collection that become an issue when considering the road rules. It is important to ensure that no recording device is used in vehicle without a passenger, who is in control of starting and stopping the device. These information regarding the use of mobile phones while driving was provided by transport NSW, all recording required is to only be conducted in NSW to ensure compliance (Transport for NSW 2020*c*).

While it is not currently a legal requirement in NSW, a mask will be worn when conducting recording in other people's vehicles. The steering wheel will also be sterilized before and after recording take place. This is to ensure that the most care is taken to prevent the spread of COVID-19, if it were to pick up again in the region. The vehicles used will be family members that we stay in close contact with as another level of preventative with spreading of COVID-19.

3.6 Project Planning

Project planning is an essential step in ensuring that key objectives of the project are kept on track. This section includes resources required for the completion of the project, a project schedule and a risk assessment.

3.6.1 Resources

A list of resources is required to ensure the objectives of this project are successful and feasible. Many of the required resources are already in possession, while some, such as MATLAB extensions, which through the USQ licensing, I have access for. Another significant resource for this project is time, in which a minimum of 500 hours is expected to meet the requirements. This will be evenly spread between 10-15 hours per week where possible, with the intent to slowly retrieve and add data to the system and importantly complete further review into research areas of significance as the project advances. Time will continually be used to develop the dissertation, keep in contact with Tobias Low regarding progress, developing processing techniques to enhance the classification time of the machine learning techniques and analysing these different machine learning techniques to determine the most successful for this application.

A significant resource is the data, before starting, it is difficult to know how much data will be required to see good results. In terms of data for machine learning, the more that can be resourced, the more success the machine will have in optimizing its algorithms and gaining a high percent success rate.

A list of resources are detailed in Table 3.1

Table 3.1: Requirements for project

Item	**Quantity**	**Source**	**Cost**	**Reason**
Laptop	1	Student	In possession	Word processing and running MATLAB
Recording Device	1	Student	In possession	For recording data/ videos
MikTex	1	Student	In possession	Used to produce document for submission
Microsoft Powerpoint	1	Student	In possession	Used to create presentation
Microsoft Excel	1	Student	In possession	Potentially required for data analysis and storage
MATLAB	1	Student	In possesson	Main programming tool for project
MATLAB Machine Learning/Image Processing Add-ons	8	Student	In possession	Required for machine learning applications and image processing algorithms
Video of driving	As many as possible	Student	Nil	Required for machine learning from multiple different sources

From preliminary testing, the speed of my performance of the current laptop have been powerful enough to perform the required algorithms such as image processing and machine learning. If it is found at a later date that this is not keeping up, there may be requirements to purchase or borrow a more powerful device to ensure the completion of this project. There is currently no total cost for this proect and if there were later requirements for a new laptop, this will be used outside of the project and still determined as a zero cost to the project.

3.6.2 Project Schedule

Completion of this project will occur in the academic year of 2020. This will include the completion of ENG4111(Research Project Part 1), ENG4112(Research Project Part 2) and ENG4903(Professional Practice 2). To be successful in completion of this project, it is important to break the project down into milestones. Monitoring progress is critical to ensure the project is kept on track for completion. For this reason, the research has been broken down into the following steps:

- Phase 1 - Preparation
- Phase 2 - Data Collection
- Phase 3 - Initial Testing
- Phase 4 - Image Processing Optimization
- Phase 5 - Machine learning Optimization
- Phase 6 - Analyse Performance
- Phase 7 - Documentation and Presentation

Table 3.2: Project Phases 1 - 2

Phase 1 - Preparation	
1A	Project commencement approval - Obtaining official approval from USQ to commence project.
1B	Resource acquisition - Obtain and/or purchase all software and hardware required to complete the project. Ensure licensing is available or purchased for the use of MATLAB applications.
1C	Literature - Perform literature review. Notably identifying pre-processing algorithms and machine learning techniques of benefit.
Phase 2 - Data Collection	
2A	Data Requirement - Gather information regarding legalities of road rules, and currently COVID-19. Determine what features are important and how these will be captured in the most affective way.
2B	Database - Collect variety of different information from received data. This will be determined and re-visited over length of project. Requirements may change as optimization occurs. Importance in variety of vehicle and seat position needs to be taken into consideration.
2C	Categorize data - All data will be reviewed and categorize to allow for supervised machine learning and verification steps. To minimize algorithm complexity, the categories will be; driving, reaching for phone and abnormal driving events(this includes gear change, touching face, scratching leg etc.).

Table 3.3: Project Phases 3 - 4

Phase 3 - Initial Testing	
3A	MATLAB Overview - Test on small database. This step will help to determine how many data inputs the machine learning algorithms can handle, and will give a benchmark for improvement
3B	Documentation -These results set the benchmark for improvement and optimization. It is important to record and refer to these after improvement/changes are made to the image processing and machine learning algorithms.
3C	Experimentation - Adjusting some values in the MATLAB tool to investigate just how they operate. Different layers may help to determine the state of some images quickly. It is important to correctly understand how all the algorithms work.
3D	Review - Initial testing sets the benchmark. It allows determination of what will and will not work. It can help determine how many inputs are allowed, the speed of the process, and which algorithms return the highest accuracy in the shortest time.
Phase 4 - Image Processing Optimization	
4A	Image Cropping - Due to the data being varied by vehicle and driver, images will need to be cropped. Cropping will occur around the steering wheel and lap area as the face and window will represent "false" data as the driver will be stationary.
4B	Data Matrices - Images will be converted to greyscale data values to be input into the database. Due to these being videos, an algorithm for comparing images will also be used to output a magnitude and angle of movement matrix.
4C	Trial Algorithms - While working through different techniques, new data will be added and unnecessary data removed. This will be ongoing to determine what works best for the machine.
4D	Finalize - After reviewing, the optimal image processing techniques will be picked. This will be based around a speed to accuracy ratio. This phase goes hand-in-hand with phase 5 and phase 6.

Table 3.4: Project Phases 5 - 7

Phase 5 - Machine Learning Optimization	
5A	Trial Algorithms - While working through different techniques, layer and algorithms. This will help to determine what aspects of the machine data are "useless" and what data can be taken advantage of. An image that doesn't move significantly from frame to frame likely does not need to go past the first layer/branch of the process.
5B	Finalize - After reviewing, the optimal machine learning algorithm will be picked. This will be based around a speed to accuracy ratio. This phase goes hand-in-hand with phase 4 and phase 6.
Phase 6 - Analyse Performance	
6A	Analyse - Each step and change in processing and machine learning will be closely reviewed. This is in a goal to minimize the required data and time to process and categorize the images
6B	Record - An in-depth record will be taken of every change. This will include what did and did not work. What the results and speeds were for each different algorithm, and exactly what data was passed to the machine after processing. This will help in combining working features and removing unwanted data.
6C	Finalize - After determining the final processing and machine learning techniques, the final "unseen" data set will be put through the finalized machine to determine the accuracy and processing speed. Results will be recorded for documentation and presentation. This phase goes hand-in-hand with phase 4 and phase 5.
Phase 7 - Documentation and Presentation	
7A	Progress Report - Preparation, discussion with supervisor and submission of progress report.
7B	Draft Dissertation - Prepare draft dissertation for submission to supervisor.
7C	Final Documentation - Submit final dissertation after receiving feedback from supervisor and making necessary adjustments.

Many of these phases operate over the length of the project and overlap with other phases at times. It was broken down in this way to ensure it is trackable and easy to follow. To ensure this remains manageable, a schedule was created and can be seen in Appendix B.

Figure B.1 of Appendix B displays what phases require completion before others can commence. It also displays the importance for phase 4, 5 and 6 to be completed in correspondence with each other. This schedule will be referred to regularly through the project to ensure things are kept on track.

3.6.3 Risk Assessment

A risk assessment was adopted using the risk matrix found in research paper; Good risk assessment practice in hospitals (Kaya 2018). Figure 3.2 displays how risk levels are categorized based on the consequences and likelihood of the risk ocurring.

		Consequence				
		Negligible 1	Minor 2	Moderate 3	Major 4	Catastrophic 5
Likelihood	5 Almost certain	Moderate 5	High 10	Extreme 15	Extreme 20	Extreme 25
	4 Likely	Moderate 4	High 8	High 12	Extreme 16	Extreme 20
	3 Possible	Low 3	Moderate 6	High 9	High 12	Extreme 15
	2 Unlikely	Low 2	Moderate 4	Moderate 6	High 8	High 10
	1 Rare	Low 1	Low 2	Low 3	Moderate 4	Moderate 5

Figure 3.2: Risk matrix used to calculate risk levels (Kaya 2018)

The risk assessment is located at Table C.1 - C.7 of Appendix C. This risk assessment was used to identify all potential risks involved with this project. Every step of every phase was risk assessed, and these assessments will be reviewed upon commencement of

these tasks. Due to the outbreak of COVID-19, the risk assessment in certain areas of this project may need to be reviewed to ensure it is up to date with the current legislations.

Due to the potential hazards involved with this project, the risk assessment has displayed a rather large number of high-risk tasks. Although due to the controls put in place, many of these issues have been categorized down to a low potential risk.

Some risks were established late into Semester 2, as two of the Residential school subjects I was enrolled in were converted into online subject, due to COVID-19, requiring time balancing adjustments. As this risk was added at a later date and addressed as it was made aware, it has not been added to the initial project risk assessment. These subjects to data have neem successfully attended and completed as required for the completion of the degree.

Chapter 4

Stage 1 - Initial Design

Initial testing involved the data collection, a minimal amount of image processing, and machine learning, in which all machine learning methods available were attempted. Results were collected and observations made on the success of techniques. These observations were used to determine the changes that would be implemented for stage 2 of testing.

4.1 Data Collection

Initial data collection was aimed around achieving some results for Phase 3 - Initial Testing. For this reason, data collection at this early stage involved only one vehicle, on one day, resulting in approximately a 15-minute video. This video was then broken down into multiple shorter videos. It is important in the data collection stage to ensure that a video ends with reaching, otherwise there is more training involved that is not essential for the outcome of this project.

Data collection for phase 3 of this project included only 3 minutes of usable footage, which results in 5030 samples, due to a recording rate of 30 fps. Once the data was collected, each frame is watched step by step to find the frame in which the driver begins to reach for the phone. See Appendix F for the MATLAB code that was used to determine the correct details for each sample. This code was used after reviewing the data in full speed to jump to and pinpoint the correct frame for categorization. Results after categorizing each value are as follow:

- 672 abnormal samples
- 2087 driving samples
- 2541 reaching samples

The data above can be seen to have been skewed significantly away from the abnormal samples. While this is not ideal, it was helpful in indicating that a better method is required for ensuring a more even spread of data. Initial data collection was used mostly for the initial training, although clips were simple to reuse at a later point as data was still valid. Using the information found above some important notes were taken to ensure that data collection after this point was spread out in a more even manner, and that video samples taken in different vehicles had some method of consistency.

The collection stage involved reviewing of the data as mentioned above, which was then stored into a classification array. This was done with manual entry with a small level of calculation based around the video trimming, a small sample of the code for this application is shown in Appendix G. This code displays how the values were entered manually as well as how they were compiled for testing. This method ensured accuracy as every array had to be the correct length in order to align with the predictor array. This function also did a quick search of different values to see how many samples of each type are stored. As specified the arrays will not line up with each other as a significant amount of data was removed to allow for an understanding of how it worked without reproducing all the used code.

4.2 Data Pre-processing

It is important in early stages of testing to keep the pre-processing to a minimum, although there were restrictions in hardware capabilities that required some level of pre-processing to occur. The initial training stage was more of a verification of approach, and this enabled some data pre-processing techniques to be tested and validated. Initial pre-processing techniques were found to be adequate and allow for rather accurate results in the first stage of testing. A significant amount of work was done in the first stage to make the data gathering and pre-processing techniques easier, however some methods were left at a manual input state to allow ease of change at later stages and make it easier

to verify/debug in the case that the processed data did not line up correctly with the categorized arrays.

After retrieving data and attempting to run it through a machine for learning, it was determined that too much information was being processed. It is also important to ensure that the movement is being detected, for these reasons the preprocessing techniques that were used in early stage training were cropping, optical flow algorithms (using the Horn-Schunk method) followed by data averaging. Video trimming was also involved to ensure that only the data that was valid was fed into the machine. These methods have been detailed below.

4.2.1 Cropping

Cropping in the initial stage involved a 1080x1081 box, due to the method of which this was conducted it was not identified until later that the dimensions were out by one pixel. While it makes for 1080 more points, for the purpose of initial training, the extra row of pixels did not cause any issues. This cropping was performed due to RAM limitations on the computer used to run MATLAB. Cropping was performed on videos to ensure that the lap area of the driver and the steering wheel could be seen and that the window area was removed, for reasons as specified previously. 1080 pixels were simply used as this kept the full width of the image, and only height adjustments were required to be made between different vehicle types. The cropping technique can be detected in Appendix H and was used in conjunction with the trimming of video to avoid double handling of MATLAB function running.

4.2.2 Horn-Schunk Algorithm for Optical Flow Outputs

A method of optical flow detection was required in order to detect the reaching act. This eliminates issues involved in training around car interior colour, clothes worn by driver and other issues that come with reading just the still image. The Horn-Schunk method was used simply due to its accessibility through MATLAB. This is a built-in function in MATLAB's image processing add-on. These methods are as explained in "Determining Optical Flow" by Berthold K.P. Horn and Brian G. Schunck (Horn & Schunck 1981), in which the derivation of the method, and some applications are outlined.

This method was found to take approximately 0.09 seconds per frame using the tic-toc recording method in MATLAB. That is 2.7 seconds of calculation for every second of data.

4.2.3 Data Averaging

As mentioned above, Motion Energy Outputs result in a significant amount of data due to the floating-point value for both the x and y direction of movement. For this reason, a significant amount of averaging was required. As stated previosuly, data was collected 1080x1081 and after some trial and error, it was found that averaging this into blocks of 15x15 pixels was a good solution. This level of averaging allowed for the machine to be fed approximately 15 minutes of video for training, with the other current methods of pre-processing. This averaging of data simply involved averaging the results from the Motion Energy Output in the x direction and in the y direction. This made the current data 225 times smaller than original, which allowed for a significant increase in the samples able to be fed into the machine. Initial testing showed good results which validated the method.

The MATLAB code for both the averaging of data and the Horn-Schunk method are both included in the same file. This code can be seen in Appendix I

4.2.4 Video Trimming

As specified previously, this was conducted at the same time as cropping. A lot of manual work went into this process, this included watching the recording which was one large file and breaking it down into multiple short videos for ease of trimming. The next step was to watch each video frame by frame and note the frames at it which the video needed to be trimmed to. This was also conducted alongside categorization of the videos, similarly, to avoid double handling of data review. Every video was watched, the starting frame recorded, any frame where a change of categorization was detected was recorded as well as the final frame, to ensure array sizes added up. This ensured that all videos were of the correct length and categorization, which was verified during the training as the file sizes were required to be of the same length in order to compare variables to the classification determined.

4.3 Machine Learning Algorithms

MATLAB's classification learning app was used with default settings in the first stage to keep all algorithms consistent with each other. There was an attempt to try all algorithms, some of which failed due to memory issues. With a majority completing successfully it was determined that the data size was still reasonable and that more cropping of data at this stage would only result in worse outcomes. The setting used for these algorithms, that are consistent across all machine learning applications are listed below.

4.3.1 Data Validation

5-Fold Cross validation was used at this stage, this was due to a smaller dataset with no other data to compare results to and test outputs. This method separates the data into 5 different sections, it then looks at 4 of the 5 folds and trains a model based on them, after which it validates these results with the remaining section. That is 5 machines are trained and validated against a small section of the data. This is seen as a good technique as it gives a good estimate of the accuracy. This is the recommended technique through MATLAB with a small dataset.

Later stages of testing may still use some form of k-fold validation, where k is an input variable. Larger values of k result in slower training times, although the results prove more accurate as the number of folds increase. Hold out validation may also be tested at a later stage, though this may depend on the data set size, as hold out validation is only recommended for large data sets. This method will be described in more depth if used.

4.3.2 Misclassification Cost

Misclassification cost is a disciplinary result of incorrectly classifying a value. The misclassification cost was left as seen in the figure below and can be described with the following explanation. Diagonals must be left as 0 as seen in the figure, this is due to the result being correct, in Figure 4.1 below any incorrect classification will result in 1 point. The goal of the machine learning is to minimize the points while still attempting to keep the accuracy high. If the top right hand was a 5 instead of a 1, the misclassification of reaching when the true value was abnormal would result in 5 points. Due to 5 points

being significant in comparison to 1, the machine learning algorithm would attempt to minimize the amount of reaching outputs, when similar to abnormal events.

		Predicted Class		
		Abnormal	Driving	Reaching
True Class	Abnormal	0	1	1
	Driving	1	0	1
	Reaching	1	1	0

Figure 4.1: Misclassification cost use in Stage 1

After completing the initial training, some adjustments were made in the misclassification cost to attempt to minimize the amount of times the machine predicted reaching when the true classification was abnormal. Due to the data set used this resulted in one of two outcomes:

- A low miscalculation value made almost no change, as there was only a small number of misclassification
- A large misclassification value resulted in very inaccurate results, as it skewed all reaching classifications more towards abnormal classifications

While this proved unsuccessful for the first stage of training, with a more even spread of data in the second stage it is likely to have a better result. This method may be used again in the second stage to see if it has more beneficial outcomes.

4.3.3 Advanced Settings

Advanced settings are dependent on the type of training, for example, maximum number of splits for a tree or number of neighbours for the KNN method. These settings were not adjusted as this first stage was simply used to get an idea of the effective methods with minimal adjustments made. These settings may be adjusted at a later stage in the machine learning optimization phase of the project.

4.4 Results

See Tables D.1 and D.2 in Appendix D displaying the output accuracy percentages from the training. Time to train and approximate predictions per second were also recorded. This data was recorded for each training method that was successful, an an example of how this was output by the machine learning MATLAB tool can be seen below in Figures 4.2 - 4.4. This example used the linear discriminant results that were measured in this stage of training. These values were then converted into an table to allow for the graphs as seen in this section.

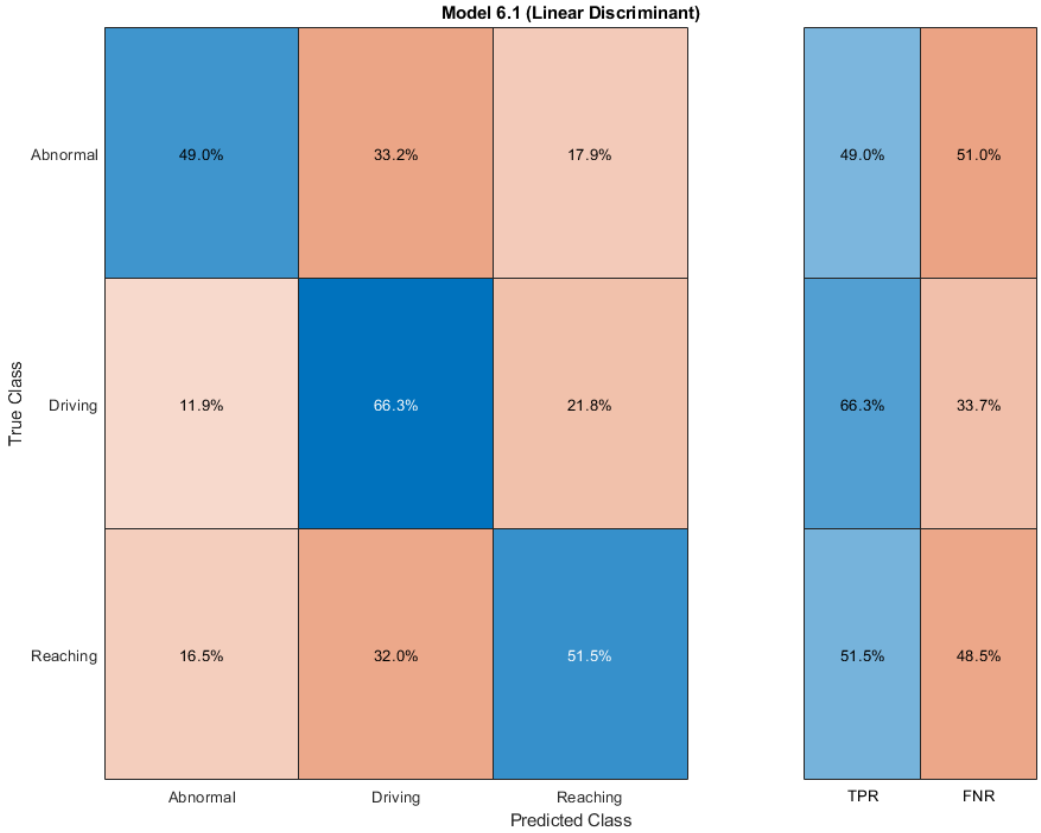

Figure 4.2: MATLAB Machine Learning Output Matrix 1

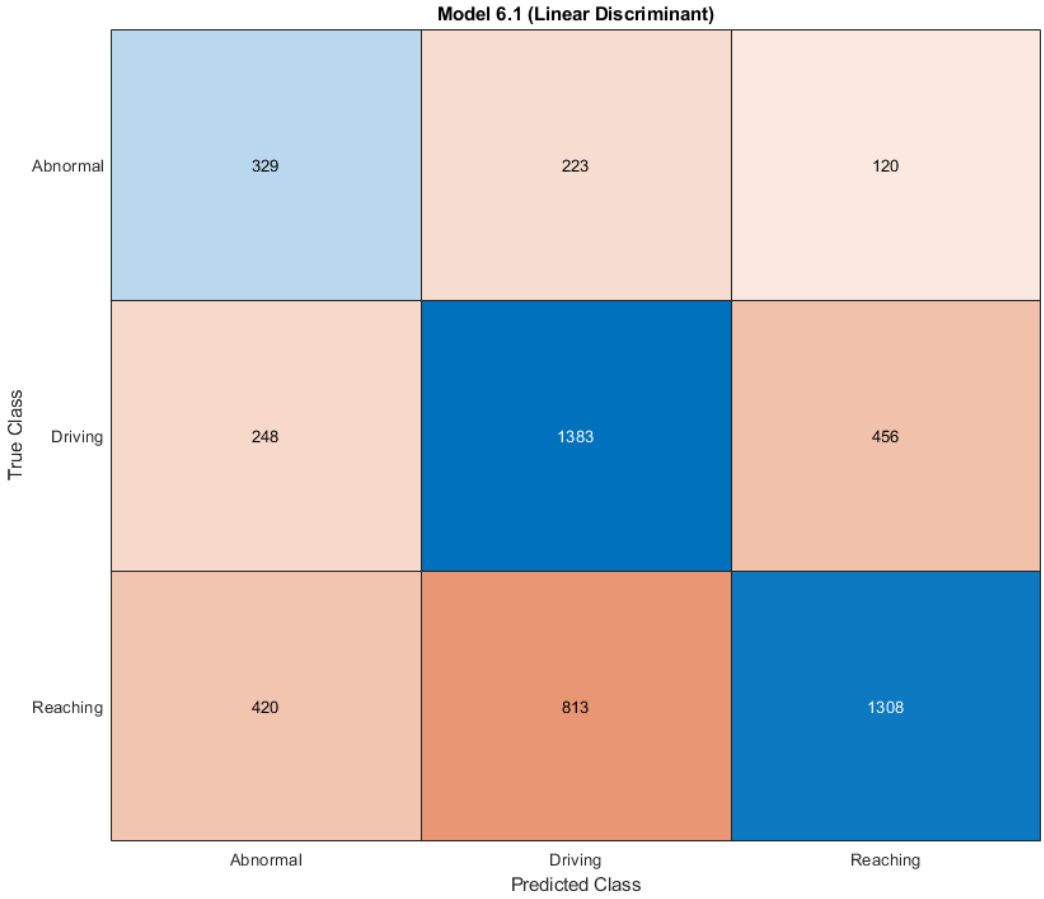

Figure 4.3: MATLAB Machine Learning Output Matrix 2

Model 6.1: Trained

Results

Accuracy	57.0%
Total misclassification cost	2280
Prediction speed	~170 obs/sec
Training time	735.43 sec

Model Type
Preset: Linear Discriminant
Covariance structure: Full

Optimizer Options
Hyperparameter options disabled

Feature Selection
All features used in the model, before PCA

PCA
PCA disabled

Misclassification Costs
Cost matrix: default

Figure 4.4: MATLAB Machine Learning Output Data 3

It is important to note that time to train is only relevant for project time constraints, a large time to train will not have any negative affect on the ability of the algorithm to operate. These larger training methods were often left to run overnight, and the recording at this stage was to allow for the same process to be followed for later stages, knowing which methods could be done with ease and which ones to leave for training at night. Prediction speed is of high importance, as a prediction speed of 30 obs/sec leaves no room for processing time.

A handful of algorithms failed due to memory related issues, and these will not be followed through to the later stages of training. See Figures 4.5 and 4.6 below displaying some of the key finding of the first stage of testing. These display both the overall accuracy and the machine learning prediction speed.

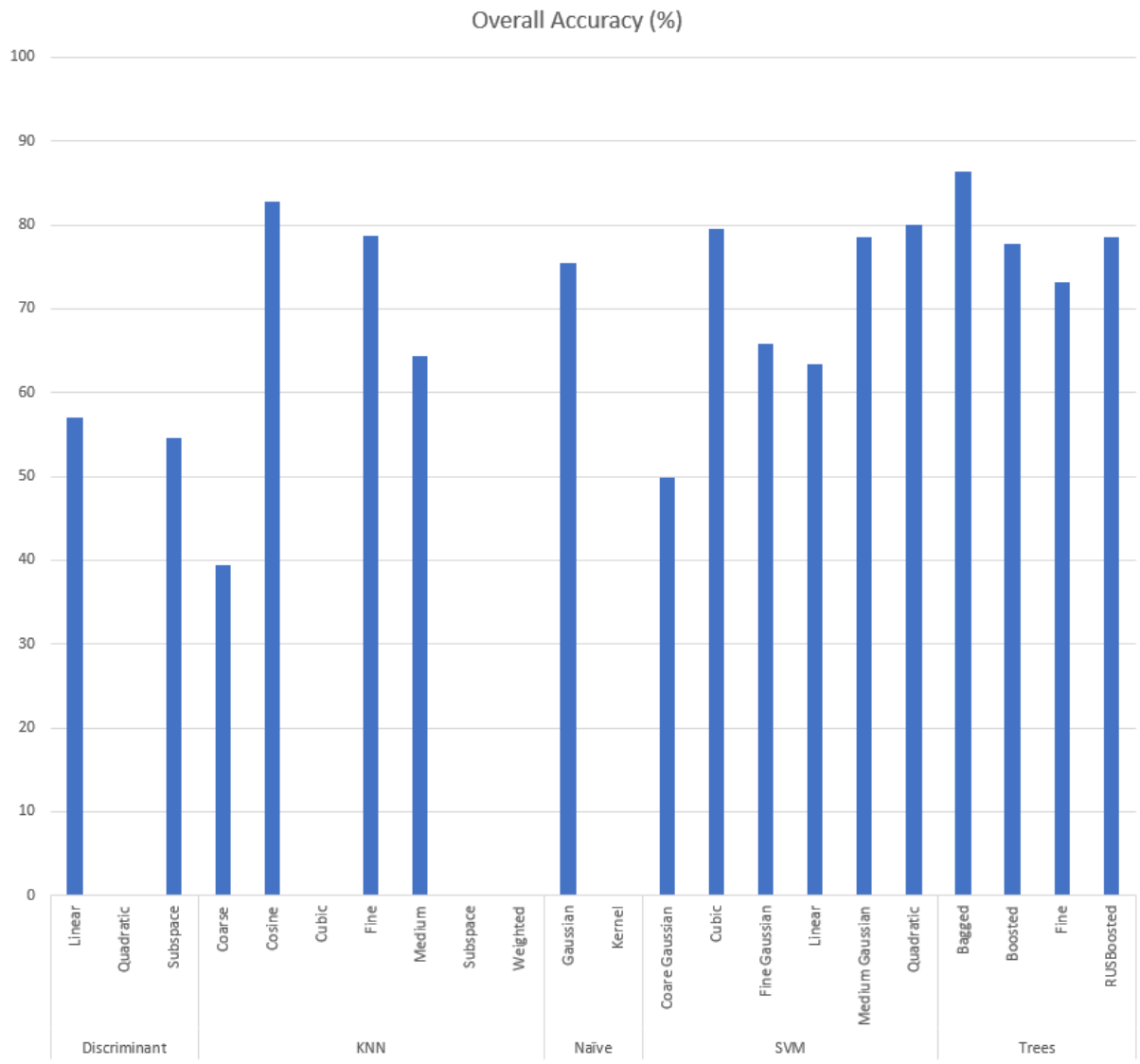

Figure 4.5: Overall Accuracy of Stage 1 Machine Learning Techniques

This first stage of testing was to validate that there was a relationship that could be identified between the reaching movements and a reaching event, that was separable from other methods with machine learning. Although these results are not showing a clear indication, the high rate of success shows some promise that there will be a method to detect the reaching events, potentially after the implementation of some more pre-processing techniques. Any method above showing a zero-result failed on memory constraints.

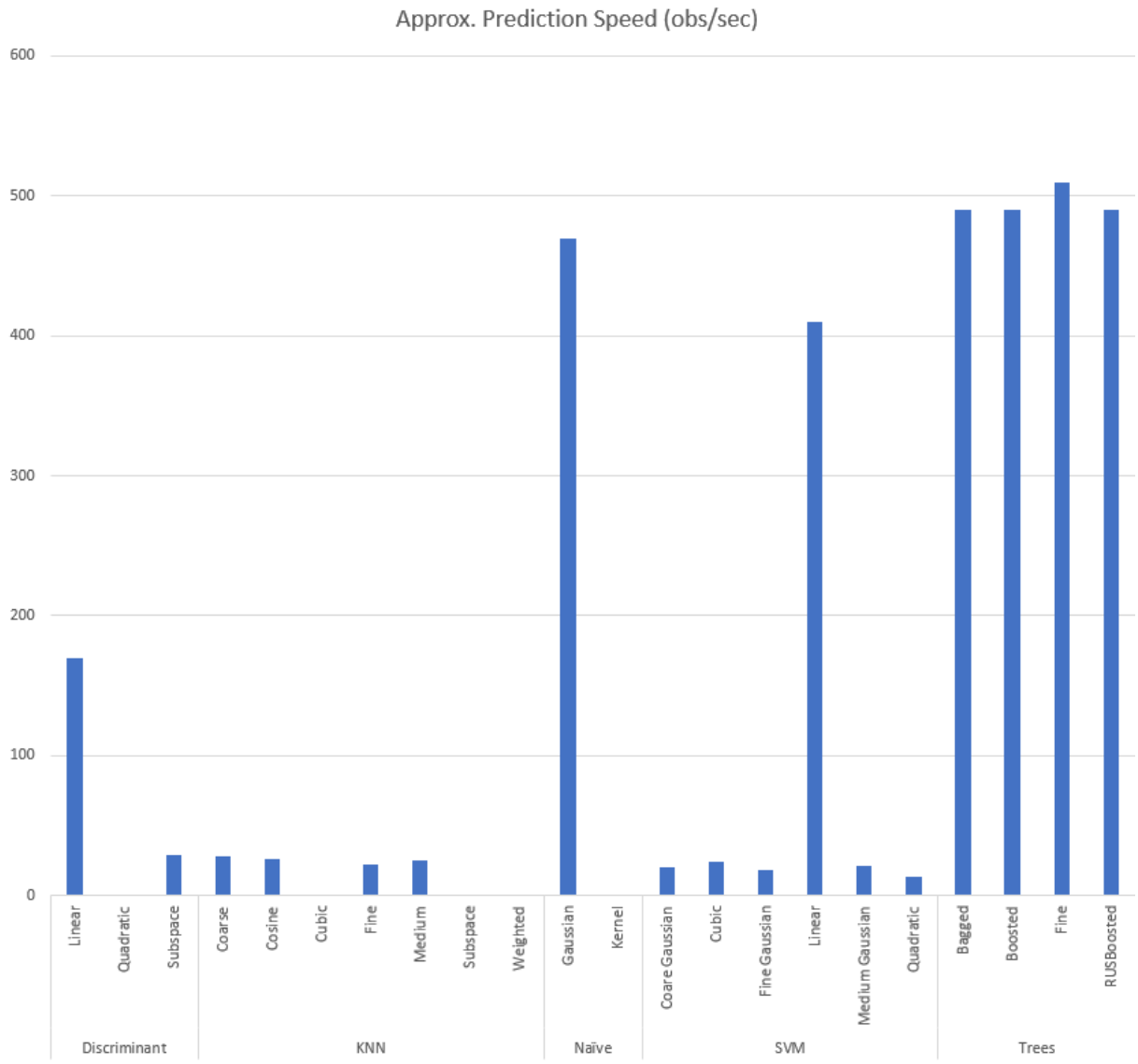

Figure 4.6: Approximate Prediction Speed of Stage 1 Machine Learning Techniques

With a frame rate of 30 frames per second, anything floating around the 30-50 obs/sec is rather slow. The higher this number, the more time that can be applied to the pre-processing techniques. There are a handful of techniques here that show high prediction speeds, and some that sit around the mark of 30-50 obs/sec. This data can be seen in Tables D.1 and D.2 in Appendix D.

With the speed becoming more relevant at later stages in the project, there may be a requirement to view the relationship to speed around the lower end of the scale, however there is currently no need to graph a view that shows more closely the values around that 30-50 obs/sec region. Currently the Tree methods look the best for high prediction speed and the KNN method, as expected due to its approach of comparing to all other values fed into the machine, is rather slow, and would be expected to get worse with an increasing data-set. Similarly to above, any methods showing a zero-result failed on memory constraints.

All methods that successfully trained will still be followed through with the next stage of training. With optimization on these training methods to be reviewed once more relevant results are achieved. RUSBoosted tree was determined as the most successful result, not for its overall accuracy, but for the even spread of accuracy across all three categories. A high accuracy can be achieved by miss-categorization of Abnormal results such as the SVM Medium Gaussian approach has displayed. This is due to a smaller sample size of Abnormal results as compared to other. This sample size issue also results in a significant affect to that of the KNN type training methods. Many results are expected to look more like the RUSBoosted tree method after the second stage of data retrieval.

Ideally keeping the time to train minimal helps in making small adjustments to the pre-processing techniques, as well as an approximate prediction rate greater than 30 obs/s. This is due to the frame rate, a much large number similar to that of the tree methods is desirable.

4.5 Observations

This section outlines the observations made about the above techniques used in the first stage. It also outlines what methods will be taken moving forward into stage 2 of testing. These observations involved training the same data against what was determined as the most successful result (RUSBoosted), and comparing what the predicted class was against the true class.

4.5.1 Data Collection

After initial testing was complete the data collection extended to four different vehicles, to get a better understanding of the affect that different vehicles had on the machine learning, and the ability to successfully determine the actions with a wider variety of data. The four different vehicles are listed below:

- 2015 Honda HRV (Wagon)
- 2011 Mitsubitshi Triton (Utility)
- 2019 Suzuki Swift (Hatchback)
- 2014 Subaru WRX RS-40 (Sedan)

In each different vehicle it is important that positioning of the steering wheel and seat are adjusted, between different recordings. This will be made possible with the use of the following steps; it is important that these steps are ran through multiple times to ensure a valid recording is made for each type. It was also noticed through the first data collection analysis that some videos were rejected as they did not include enough time before the reaching began to allow for a motion history algorithm to be applied. For each vehicle the data below is to be collected:

- Repositioning visor x 5 (Can be done while driving)
- Touching face with left hand x 5 (Can be done while driving)
- Touching face with right hand x 5 (Can be done while driving)
- Touching leg with left hand x 5 (Can be done while driving)
- Touching leg with right hand x 5 (Can be done while driving)
- Pointing or waving x 10 (Can be done while driving)
- Some hand gestures while driving x 5 (Can be done while driving)
- Window up/down x 2 (Can be done while driving)
- Applying brake x 10 (To be done while stationary)
- Changing Gears (To be done while driving in Triton and WRX)

- Changing Radio Channel x 5 (Can be done while driving but will be done while stationary as still seen as distracting)
- Reaching for phone left pocket x 10 (To be done while stationary)
- Reaching for phone right pocket x 10 (To be done while stationary)
- Reaching for phone dashboard x 10 (To be done while stationary)
- Reaching for phone between legs x 10 (To be done while stationary)
- Reaching for phone from passenger x 10 (To be done while stationary)
- Reaching for phone centre console x 10 (To be done while stationary)

It should be noted that there is a large amount of data being recorded above for each vehicle although it will not all be used. All data will be analysed and only relevant data is to be used, while it is important to also keep a good variety. Some data will be simple removed or added to ensure that there is an even spread of different types, to ensure that the training does not focus too heavily on certain classifications. These samples will not be taken one after another to ensure that they are different. Ideally they will be taken on different days, and in each case, some adjustments will be made to the steering wheel positioning and seat positioning to include some variety.

Camera positioning for all recording was placed to ensure that the lap area of the driver could be seen, as well as the hands positioned on the steering wheel. It was also important to ensure that the window could not be seen in the recordings, as mentioned previously. This is a measure to prevent the machine learning algorithms to detect movement outside the window as driving, as any use of phone will be done while stationary. At this stage it has been tested on the favourable method of RUSBoosted tree that 10 minutes of data, at the current size can be used.

The first sample size only included 1.4 minutes of data, meaning the accuracy and validity of results will vary significantly. It is likely that with a large variety in the data-set and with no changes made to the pre-processing techniques, that the machine learning algorithms will output a lower accuracy.

4.5.2 Data Preprocessing

Following into stage 2 of training the data pre-processing techniques will remain unchanged. While it is still important to get the time requirements of pre-processing down, it is more significant to review all the current methods of preprocessing and machine learning under a more value and evenly spread data-set. Time requirements for these areas require a more valid dataset to determine areas where more pixels can be minimized and less quantization is required to occur. There is also the issue based around hardware capabilities that may be slowing down these results. Without first gathering more accurate results, removing more data from these machine learning algorithms is likely to only have a negative impact on the outcomes of the testing.

4.5.3 Machine Learning Algorithms

Similarly to the pre-processing observations, machine learning algorithms will remain unchanged, this is due to the positive results achieved from the first stage of training. It is likely that some methods will fail the training with the increased size of data, and the methods that previously failed will be removed in stage 2. After the completion of this stage some other methods may be incorporated such as misclassification cost, but this will be determined after the completion and observing of stage 2 results.

4.5.4 Post Training Filtering

Some interesting data was found by comparing the trained data back to the real results. It was determined that in a majority of videos, if the last 10 samples were predicted as reaching, or the last 14/15, then the real result was reaching. This technique is planned to be used further in stage 2, as there are cases where reaching is predicted when abnormal is true, and the post filtering stage will rule this out. It is just as important to ensure that the end result does not incorrectly determine reaching, as it becomes just as invalid as if it were not detecting it at all.

Chapter 5

Stage 2 - Database Optimization

The only target area of Stage 2 was data optimization, for this purpose all other techniques were continued as applied in Stage 1, with a focus on more distribution and a variety of vehicle type. The data for this stage returned a slight decrease in validation accuracy, due to the increase in the variety of data, while no changes were made to any other methods.

5.1 Data Collection

The focus of the second stage of testing was around database optimization. This involved the collection of a larger database in a variety of vehicles, to extend the study. It was important at this stage of the testing to ensure that the data collected and categorized has a better distribution than that of Stage 1. Data collection that was used for Stage 2 included approximately 20 minutes of footage that was broken down and categorized. Only 4 minutes and 40 seconds of footage was used as some a lot of data is lost to hands returning to the steering wheel after reaching.

As specified in Section 4.5.1 this data collection involved the use of 4 different vehicle types, being a wagon, sedan, ute and hatchback. However, it was decided before implementing the testing that the Sedan results would be best to be held out and used at a later stage of validation, to be used as a previously unseen vehicle. This will ensure that the methods used are applicable to other types of vehicles with slightly different body shapes.

Results after categorizing each value are as follow:

- 2279 abnormal samples
- 3561 driving samples
- 2541 reaching samples

This data shows a much better distribution of samples. Driving has quite a few more samples, which is inherent due to the actions before every reaching and every abnormal sample including driving. It is important to ensure that a good variety of driving is involved as this includes a lot of samples that may otherwise end up categorized as reaching.

5.2 Results

Stage 2 results were found to be slightly less accurate across the board. More notably when reviewing the validation results, it was noticed that many reaching samples were being incorrectly detected as abnormal samples. It was clear after this stage of training that there is a need for more pre-processing to allow for the machine to correctly categorize its data. Tables D.3 and D.4 in Appendix D shows the outcomes of the machine learning in Stage 2. See Figures 5.1 and 5.2 below displaying some of the key finding of the second stage of testing.

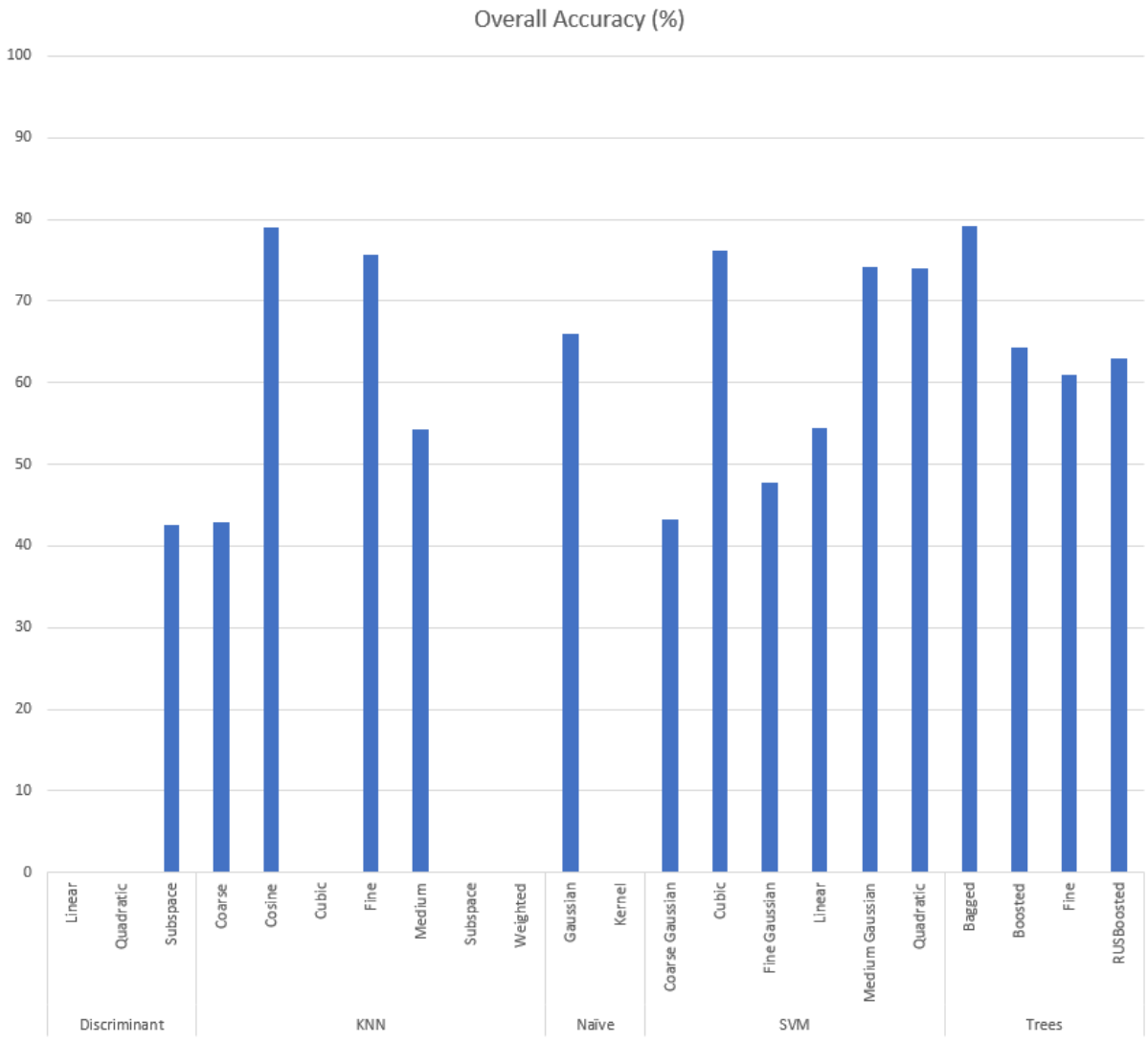

Figure 5.1: Overall Accuracy of Stage 2 Machine Learning Techniques

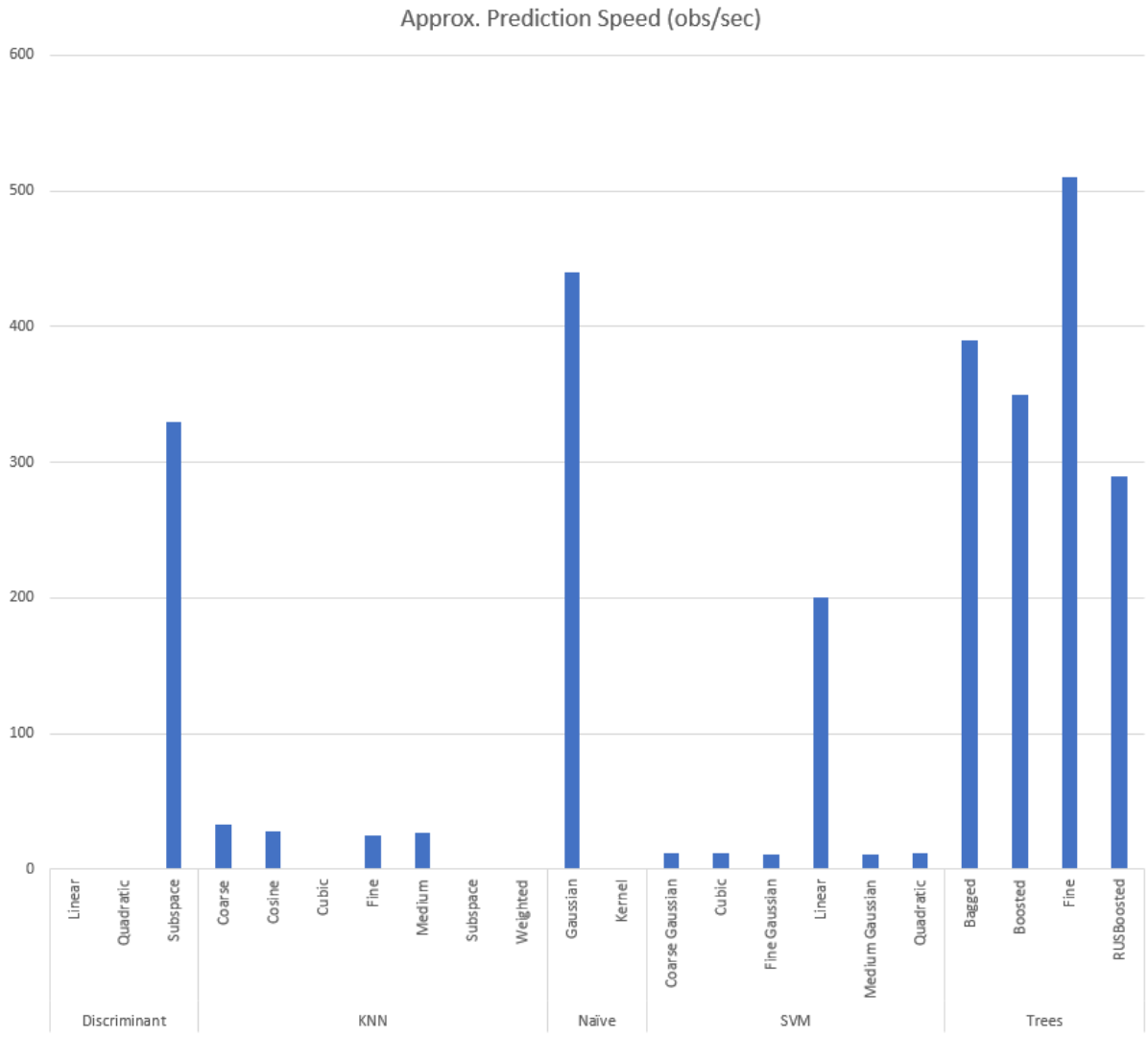

Figure 5.2: Approximate Prediction Speed of Stage 2 Machine Learning Techniques

There is no significance on the most effective method at this stage, as there will be significant changes made to the data that is fed into the machines for further stage training. Although across both stage 1 and 2, it has been noted that the bagged tree method is coming across as a beneficial method due to both its rapid training speed and prediction speed. In both cases this method has resulted in one of the highest overall accuracies.

5.3 Observations

This section outlines the observations made on the techniques that were used in the seconds stage. It also outlines what methods will be taken moving forward into stage 3 of testing. These observations involved training the data and testing its results against some previously unseen data to determine where some possible shortfalls are.

5.3.1 Data Collection

Data collection was the focus for stage 2 and for that reason there were not too many observations made in this area. It was determined that the data collection was significant enough in variety to allow for more pre-processing techniques to be trialled. Data collection may be reviewed again at later stages if it is seen as the area that requires the most improvement in order to develop this project into the correct direction.

5.3.2 Data Pre-processing

Data pre-processing was observed to be one of the downfalls of this stage of testing. While the results are still significantly promising, when these machines are tested against unseen data, it results in the reaching samples getting detected as a variety of different categories. Without some consistency in the outputs, these results are useless in the detection stage. It was determined that this is due to the current data only being a snip of what had changed from the current frame to the previous frame, with no history as to what had happened before that.

To overcome this, an algorithm like that of what would be used for Motion History Images will be implemented in Stage 3. That is, the most recent result as well as the previous

9 result, are combined, to detail a more "historical" view of what the movements were. This algorithm was simply applied to previous data already collected and stored in a new array. This results in slightly less data due to the first 9 samples of every video needing to be removed, as it could not have the full history applied and would therefore result in invalid data for the machine.

5.3.3 Machine Learning Algorithms

As previously mentioned, it was difficult to comment on the success of the machine learning algorithms based on the requirement for some more pre-processing techniques. However, it has been noticed that the success of the bagged tree method, in both training speed and prediction speed, along as its high accuracy. If this continues into further stages of testing, it may be the most successful algorithm for this study.

5.3.4 Post Training Filtering

A method of post training filtering was applied at this stage and did return results which indicate a potential need for it. This included a true result being 10 out of the last 10 samples to have resulted as reaching or 14 out of the last 15. With a better pre-processing implementation this is expected to return some useful results. This method will be implemented and explained in more depth when the results are validated.

Chapter 6

Stage 3 - Pre-processing Optimization

The focus for Stage 3 of the project was pre-processing optimization. It was determined in Stage 2 that the data being delivered to the machine learning algorithms was not sufficient enough in allowing for classification, and when one these trained systems were implemented on unseen data, the results were poor. By optimizing the data going into the machines it is expected to have a higher validation percentage and therefore a better outcome.

6.1 Data Pre-processing

As specified in Section 5.3.2, a method for improving the data going into the machines was required. Due to the storage of previous data, it was ideal to reuse what already existed and simply apply another layer of processing to that data. This was achieved by accessing the information stored from Stage 2 and manipulating it as required. The MATLAB code that was used for this stage can be seen in Appendix J. It is the application of the following equation 6.1 on the averaged data that was stored from Stage 2 of testing, a very similar technique to the Motion History Image application that was researched in Section 2.3.2, however the magnitude is not a fixed value of 1, it is the magnitude based around the speed of movement of the previous frame.

$$R = n + 0.9(n-1) + 0.8(n-2) + 0.7(n-3) + 0.6(n-4) + 0.5(n-5) \\ +0.4(n-6) + 0.3(n-7) + 0.2(n-8) + 0.1(n-9) \quad (6.1)$$

Where R is the result and n is the current frame.

6.2 Database Adjustments

Due to the pre-processing method that was implemented, some consideration had to be taken for the currently categorized data. These arrays needed adjustment, in which the first 9 frames of every videos array had to be removed, as this information lacked the historical requirements that all other data had, making it invalid in this application. Due to the manual method that was used to store this original data, each array could be accessed individually, making it very simple to removing the first 9 elements.

6.3 Results

Stage 3 results improved significantly, with validation tests showing methods as high as 98%. Notably the bagged tree method stood out above all others. This method achieved the second highest overall accuracy of 96.2%, while maintaining the fastest training time of 244 seconds. It also has one of the highest prediction speeds of 560 observations per second. While the results of the fine KNN method showed a higher percentage, it has an approximate prediction speed of 30 obs/sec. Due to the framerate being the same as this, it leaves no room for pre-processing or post filtering of the data. All the time is consumed in the prediction. With a prediction speed of 560 obs/sec it only consumes approximately 5.4% of the time allowable for processing and prediction of the data in order to achieve 30 obs/sec from processing right through to filtering.Tables D.5 and D.6 in Appendix D shows the outcomes of the machine learning in Stage 3. See Figures 6.1 and 6.2 below displaying some of the key finding of the second stage of testing. Figure 6.3 is also displayed, in which it is also the approximation speed, but caps the y-axis at 60, to determine if any under this range are also suitable.

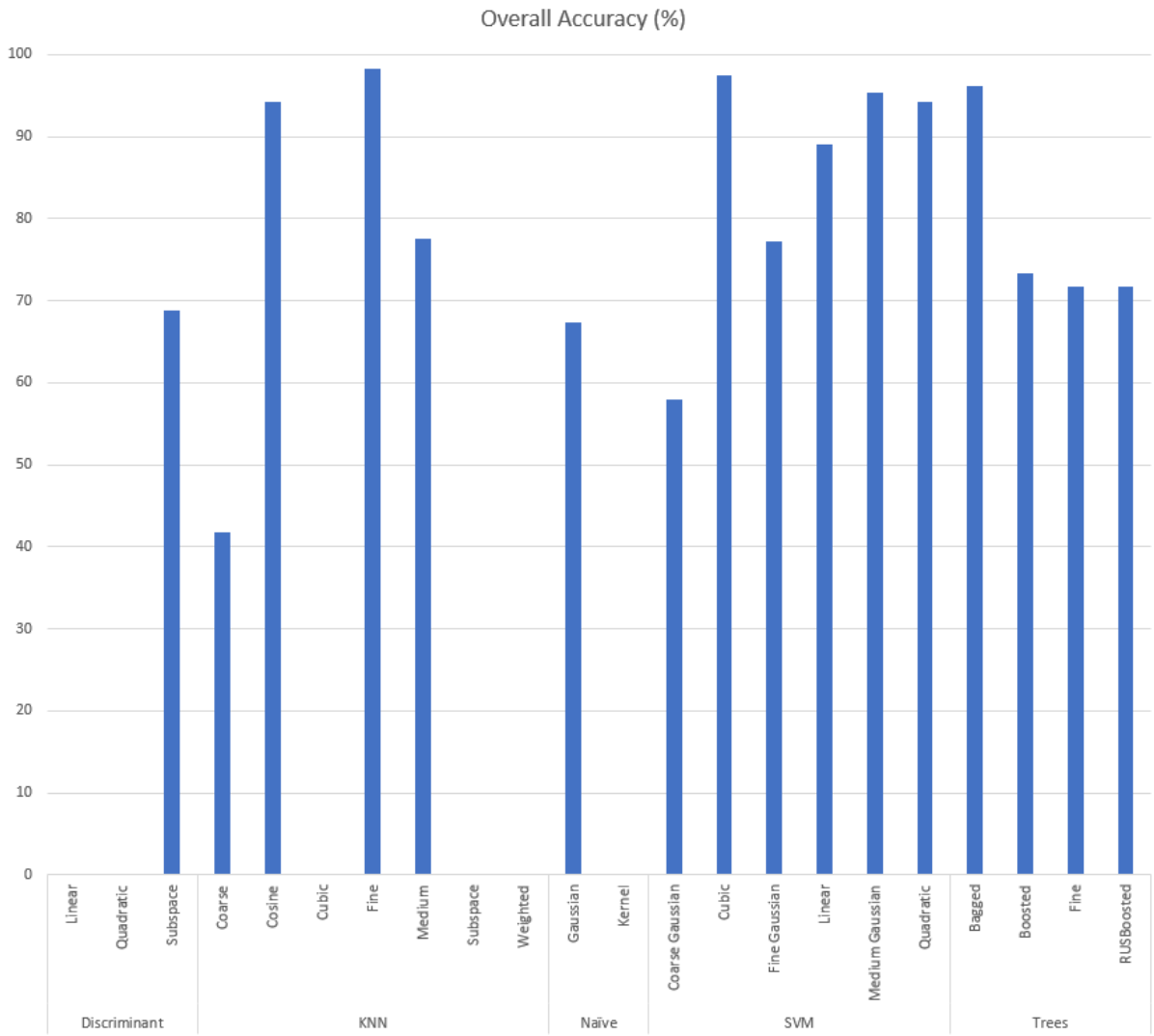

Figure 6.1: Overall Accuracy of Stage 3 Machine Learning Techniques

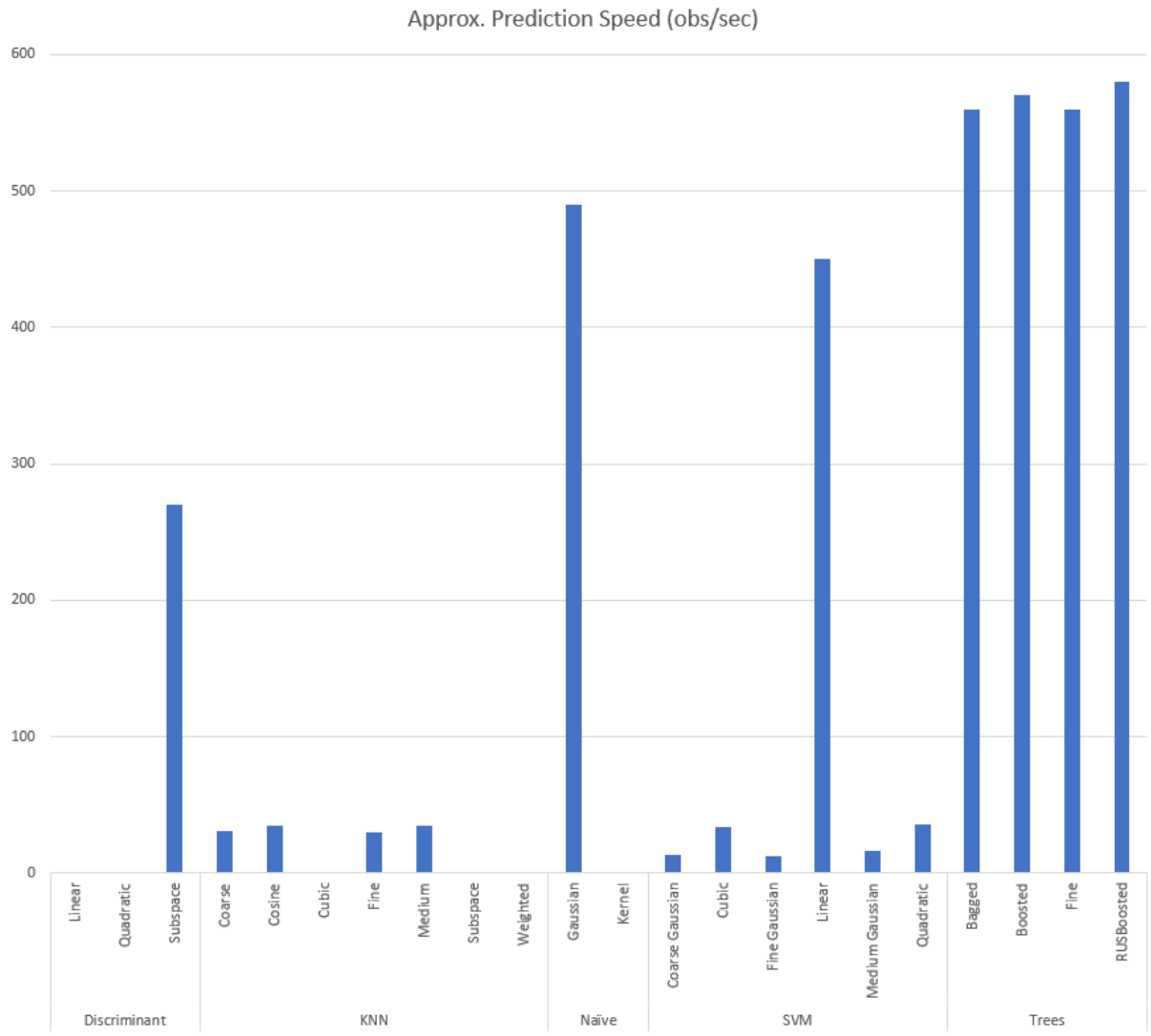

Figure 6.2: Approximate Prediction Speed of Stage 3 Machine Learning Techniques

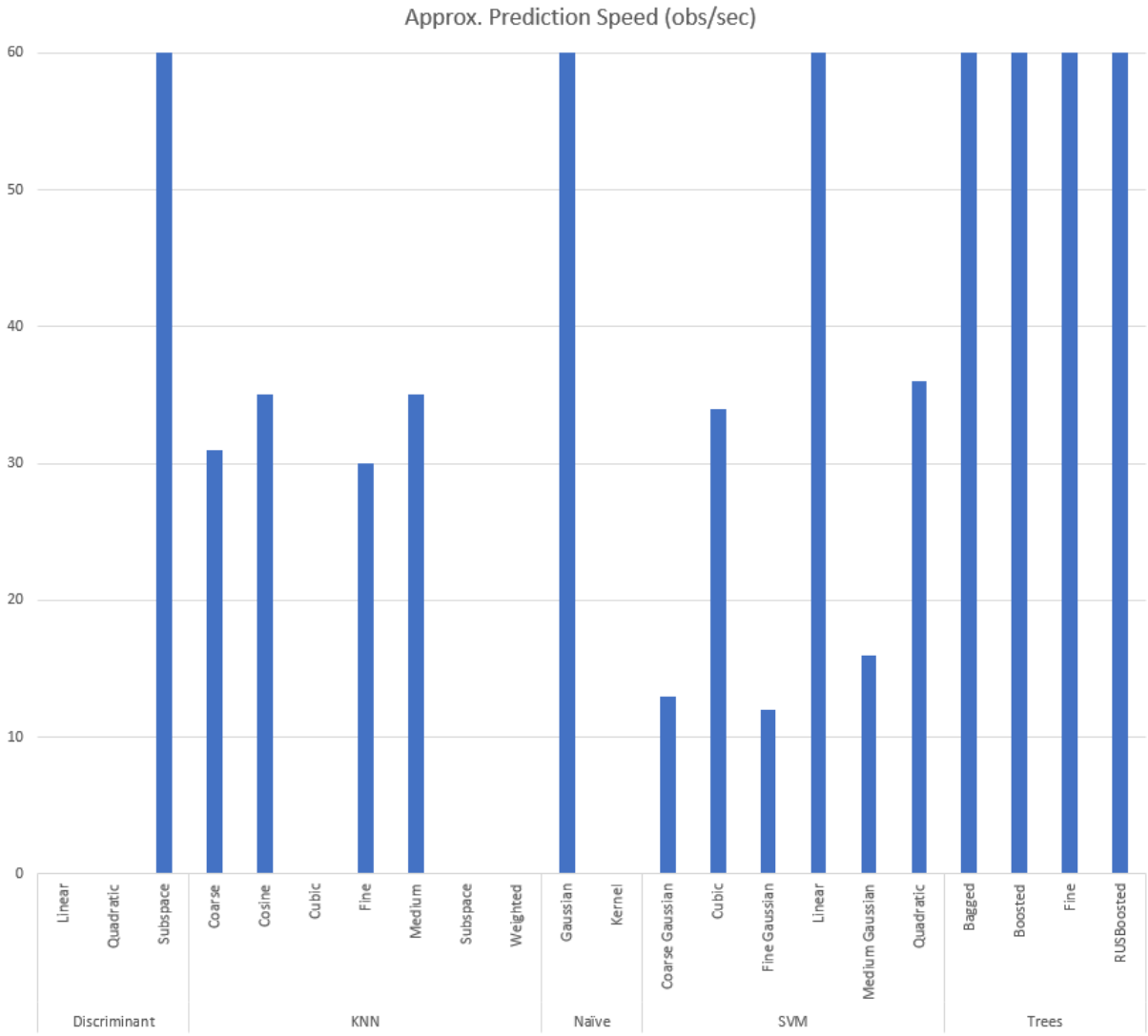

Figure 6.3: Clipped Approximate Prediction Speed of Stage 3 Machine Learning Techniques

Values are capped to 60 as this plot was only used as a tool to compare the values that were on the less acceptable end of the speed scale. Cosine and Medium KNN as well as Cubic and Quadratic SVM can be seen to have speeds that are slightly higher than 30 obs/sec, making them acceptable with the requirement of much quicker pre-processing speeds.

6.4 Observations

This sections outlines the observations made on the techniques that were used in the third stage. It also outlines what methods will be taken moving forward into stage 4 of testing. These observations involved training the data and testing its results against some

previously unseen data to determine where some possible shortfalls are.

6.4.1 Data Pre-processing

The focus of this stage was data pre-processing, and the results proved very effective. The pre-processing applied increased the accuracy by a significant amount. The areas that showed the incorrect results in this application seem to be based around the starting of movement events. Due to minimal movement in the first few frames of a reaching or abnormal event, these were often still just classified as driving.

These observations are not of concern as they have simply applied a 10-frame maximum lag on the output result. Given the high accuracy and success of these outputs, a small lag at the beginning of the movement will be overcome by the accuracy and less requirement for post train filtering.

6.4.2 Machine Learning Algorithms

Similar to Section 5.3.3 the bagged tree method was seen as a well-rounded solution to apply to a stage of validation. Due to its rapid training speed and high prediction rate, it meets the criteria for the training method required. It has also returned the second highest accuracy of 96.2%. Two other methods have also been selected, the Cosine KNN and Cubic SVM. The MATLAB code developed by the classification learner for these three techniques can be seen in Appendix K, L and M. Stage 4 of testing, due to the success of the machine learning and pre-processing techniques, will be a validation stage. There is minimal room for improvement with the current information that can be found without testing this algorithm on a previously unseen source.

Prior to reviewing the results against unseen data, the bagged tree method is preferred due to its combination of high accuracy and high speed. Both the KNN and SVM methods were chosen as compared to others in their category, they had high accuracy and a prediction speed slightly above 30 obs/sec.

While these results have high accuracy, it is still a small data set, and minimal changes in camera positioning and movements may result in inaccuracy. It is important to verify these techniques to detect where the shortfalls are and determine areas for improvement.

Chapter 7

Stage 4 - Verification on Unseen Data

This stage uses previously unseen data ran through what was determined as the three most successful machine learning algorithms (based on prediction speed and accuracy) and analysed the results after applying the post-processing algorithm that was determined in Section 4.5.4. This data has been broken down into many different aspects including percentage by vehicle, reaching type and machine learning algorithm, as well as false detection causes.

7.1 Results

Analysis took place across the 4 different vehicle types, each of which included 10 samples of the following reaching types, as well as a variety of abnormal and regular driving activity.

- Reaching between legs
- Reaching right pocket
- Reaching left pocket
- Reaching to Passenger
- Reaching to Console

- Reaching to Dash

As mentioned in 6.4, the three techniques listed below were applied in the validation trial. These methods have been named based on their category for the purpose of these sections;

- Cosine KNN (KNN)
- Cubic SVM (SVM)
- Bagged Tree (Tree)

All methods used for pre-processing were taken from Stage 3, with the addition of a post-processing filter applied to only indicate a reaching event if 10/10 of the last samples were reaching or 14/15 of the last samples were reaching.

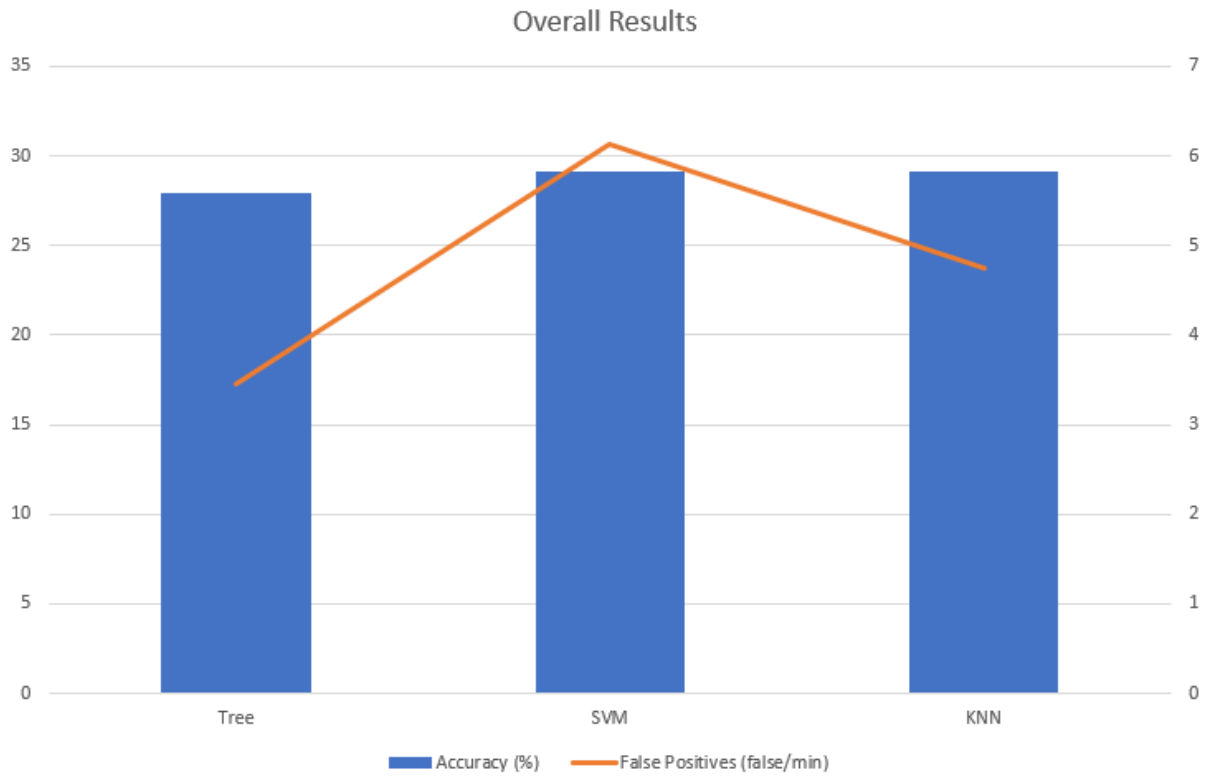

Figure 7.1: Overall Results of Stage 4 Validation Tests

Figure 7.1 displays the overall results of all testing in the four different vehicle types. These results show an accuracy of 27.92% for the Tree method and 29.17% for both the SVM and KNN methods. These results also indicate the error coming with the different techniques, with Tree having 3.45 false positive per minute, SVM having 6.13 false positives per minute and KNN having 4.74 false positives per minute.

These results are very promising, it is clearly showing the potential to detect reaching events while driving, however a significant amount of work may still be required to get these values to an acceptable minimum. Further results are broken down and displayed in 7.2 as they are showing information at lower levels that may be observed to have had a significant impact on the outcomes.

Results tables of all data collected for the validation stage can be seen in Appendix E.

7.2 Observations

7.2.1 Breakdown By Vehicle

Figure 7.2 display the accuracy percentage broken down by vehicle. It also displays the rate of false positives/minute of data.

There are two significant observations that came as a result of this breakdown;

- Wagon had a significantly higher percentage than other vehicles on average across all three methods of detection.
- Sedan results do not look "out of place" even though no training was done with this vehicles data.
- False positive rate very low by comparison in Hatchback.

The wagons results were much higher than other vehicles, with still a relatively low false detection rate. The false positive rate is similar for the ute and sedan for the Tree and SVM method, however it did have a significantly higher KNN false positive result. Evidence by breakdown shows that the likely issue in this case was the mounting method of the phone. As the wagon was the original test vehicle, recordings were conducted by wedging the phone into the window frame, resulting in a very fixed phone. All other vehicles had the phone positioned in a mounting device, which, through review and categorization of false positives, accounted for movement of the camera.

It is likely that this small movement in the camera was enough to shift it out of the range that original data was captured, making it "unrecognisable" to the machine. It

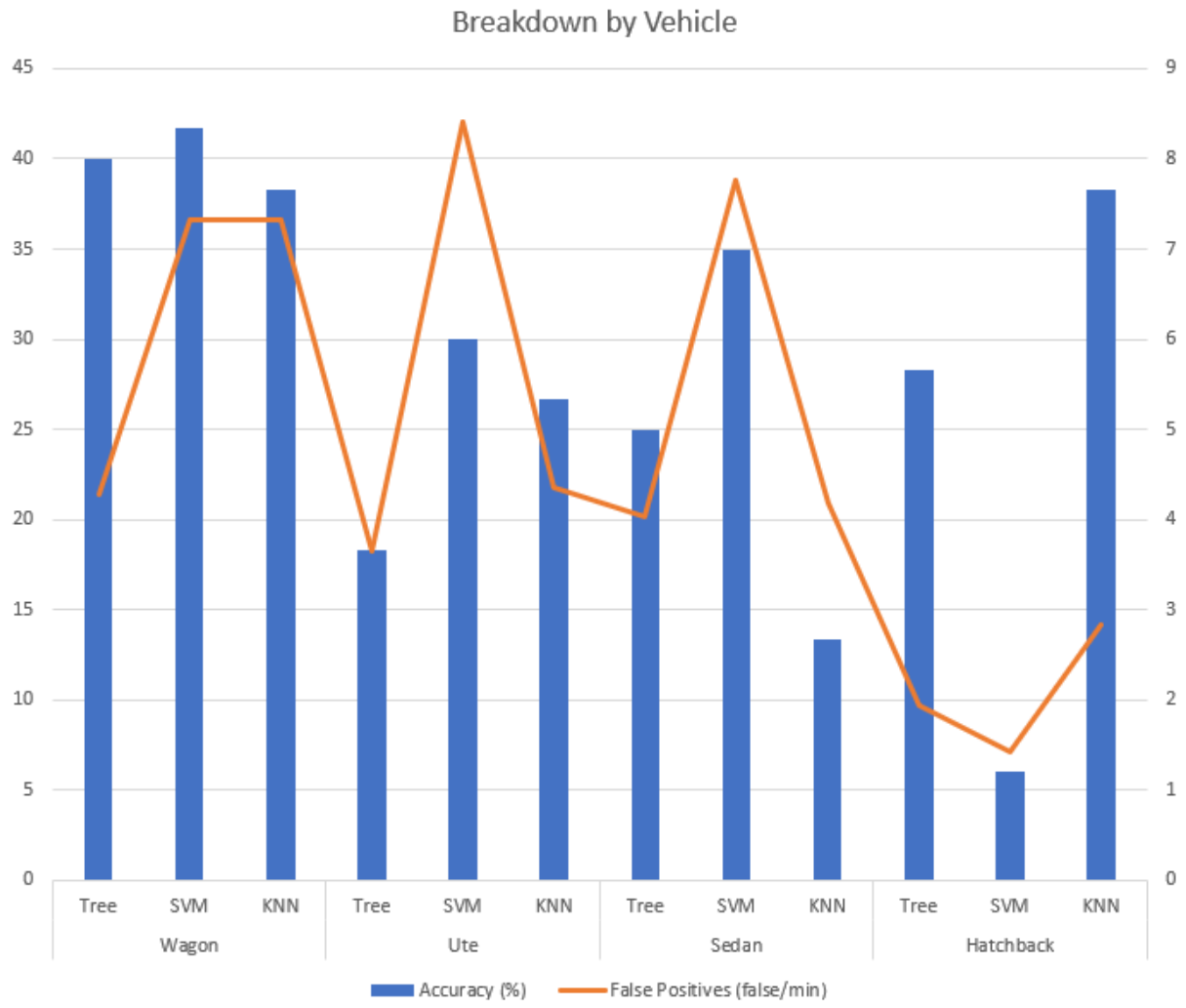

Figure 7.2: Results Broken Down by Vehicle

also potentially causes the original data that was used to train the machine to be slightly inconsistent for the different vehicle types, resulting in a slightly less accurate training as compared to that of the wagon. Overall, it can be determined from this observation that a more secure mounting method is required, as well as a verification test to ensure that all data reviewed is to the same condition (positioning of the camera).

As stated, the Sedan's results looked similar and, in some cases, better than other vehicle types. This shows that there is enough of a relationship between some vehicle types, shapes and designs to allow for the recording in fewer vehicles. This is an important finding as it means a machine can be trained across multiple vehicles, like this project has approached the issue, and potentially apply to hundreds of different models with close to the same features. The method can be reapplied across other vehicle types with a larger data set and likely return good accuracy against several different vehicle shapes.

The false positive rate in the hatchback was significantly lower to those of the other vehicles, however the detection rate was still very similar besides the SVM results. These results may be slightly inaccurate based on availability and access to the vehicle. Based on the results, this vehicle had 2000 frames less than the next closest vehicle meaning by average there was less normal and abnormal driving events recorded.

This issue can be detected across all methods, where the Sedan had the highest false positive detection rate on average along with the largest data set. This observation is significant in the way it shows a need for a larger data set for validation, especially for ordinary and abnormal driving events, and a need for more even distribution of testing. These results did help to show that an increase in driving events compared to reaching shows a noticeable increase in error rate. This clearly outlines a need to implement a larger data set, with more variety in driving behaviour, in order to bring down the error rate under normal situations.

7.2.2 Breakdown By Reaching Type

Figure 7.3 display the accuracy percentage broken down by reaching type. This is still broken down by machine learning technique used.

Observations that were found by breakdown of reaching type determined a significantly higher result in reaching right pocket as a comparison to other types. This observation shows again, a significant need for a larger data set, in which seat positioning and driver styles etc are adjusted. This result is higher since the right pocket is relatively fixed to the same location, whereas there are multiple locations a phone can sit on the dash, in the console and on the passenger seat. This indicates that multiple different angles and speeds will be detected for other reaching events as a comparison to the right pocket events and shows a need for larger datasets to cover a more varied driving behaviour.

This doesn't seem to be indicative in the left pocket scenario, which under the logic applied above should show similar results. Analysis has pointed this to the direction of the gearstick issue. The detection of left pocket is often masked due to its similarity of using the gearstick. More data is required to allow for a higher separation between the use of a gearstick and reaching for the left pocket. By increasing the confidence of the machine based on a larger data set, there could be some reduction in the application of

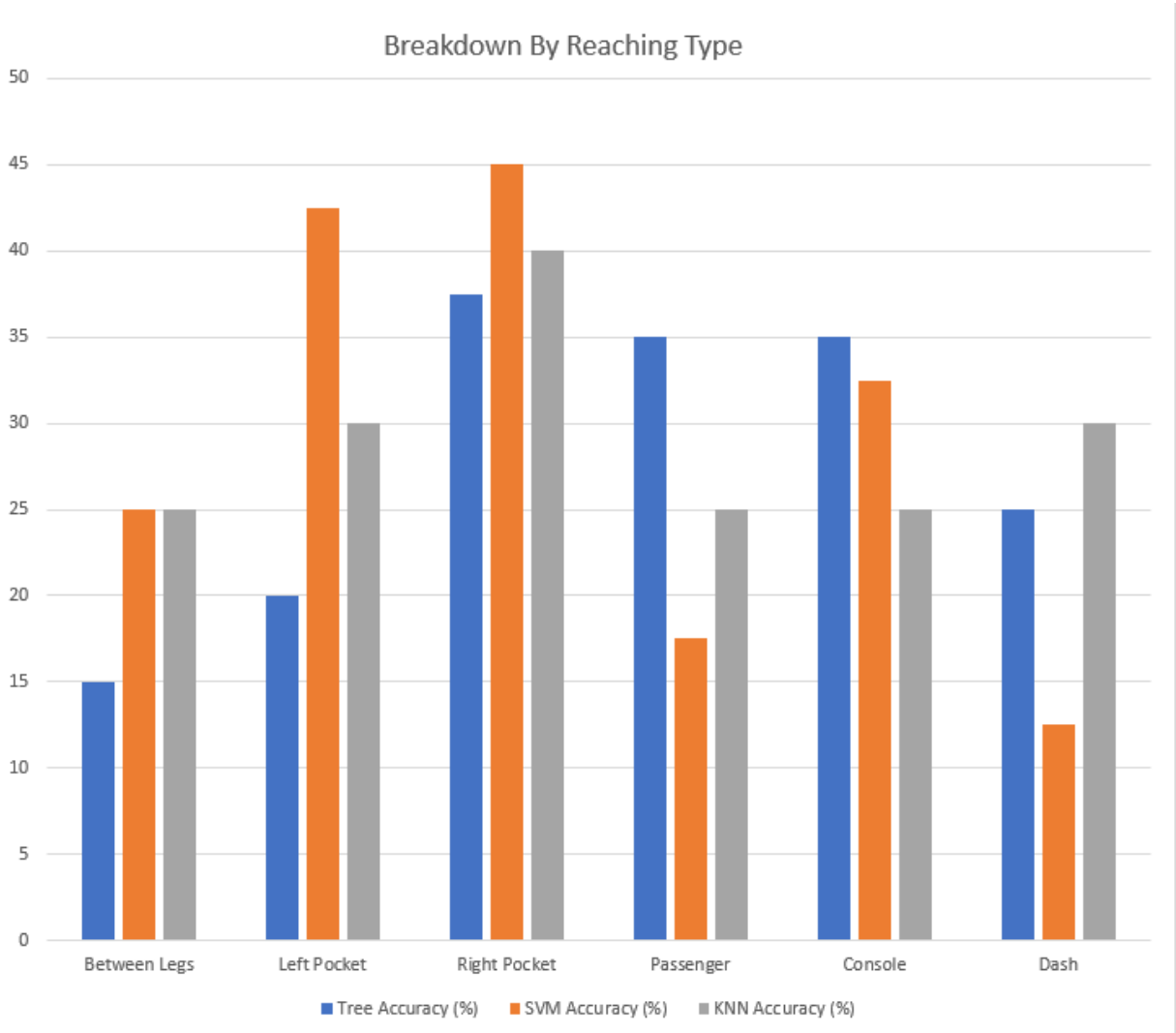

Figure 7.3: Results Broken Down by Reaching Type

post-processing, allowing in reaching events to require less than 10 in a row or 14/15 to be detected. Of course if this is done too soon it will simply result in an increase of accuracy and false positives at the same time.

7.2.3 Breakdown of False Positive Detections

Figure 7.4 display a breakdown of the false positive detections by individual count.

The above breakdown has some interesting outcomes, as it shows the larger issues causing the false positive detection. Interestingly is the hand returning to steering wheel after reaching, which is likely high because it was never used as training data. This graph helps to show areas that do require improvement, such as adjustments to radio/air conditioner,

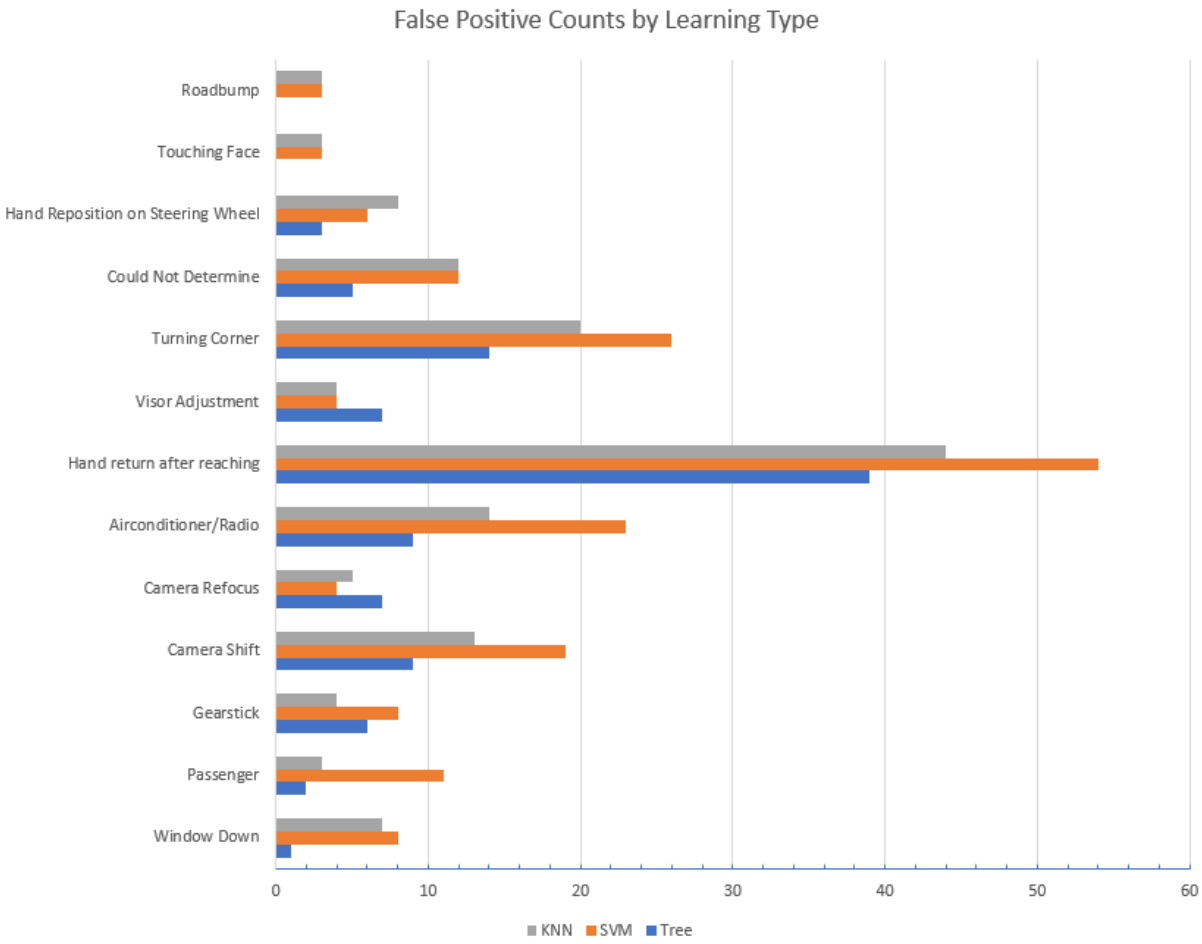

Figure 7.4: Results Broken Down by False Positive Detection

turning corners and camera shifting. Most importantly, it highlights a potential need for further breakdown in the categorization, which could potentially provide more meaningful data in the areas of few training samples. It also, similar to how it has in this instance, can return a percentage of accuracy for specific events, helping to understand exactly what needs more work, or what has hit an acceptable accuracy.

7.2.4 Data Collection

Observations of these results are clear in showing that a much larger, more varied data set is required in order to reach acceptable accuracies. These results are just too inaccurate and while mounting methods also need adjusting, the leading cause was limited data. This can be seen by the number of false positives broken into categories such as turning corner, using gearstick and adjusting air conditioner.

7.2.5 Data Pre-processing

I believe the data pre-processing techniques are acceptable, as they did result in a high accuracy when going through the machine learning algorithms. If anything were to change, there is the possibility of feeding in the true image data as well, although this significantly impacts the amount of footage the algorithms can handle. While this process was affective, there are some other methods that may be worth exploring. Simply increasing the averaging function block size allows for more data into the algorithm while also allowing for some variance in seat and steering wheel positioning. It is difficult to know without trying what affect this will have.

7.2.6 Machine Learning Algorithms

The three machine learning algorithms were all affective and showed similar results when tested. The preferred method is Bagged Tree, based on its higher prediction speed. This method did show a slightly lower accuracy than the other two but also displayed a significantly lower error rate. Interestingly Figure 7.3 shows the SVM method to be much more affective in the detection of reaching the left pocket. I see no reason to rule out methods at this point based on these results, as other methods may have better outcomes under certain circumstances it would be unfortunate to rule them out now and miss those observations.

7.2.7 Post Training Filtering

Post training filtering was an essential requirement at this early stage of design, as without it there would have been thousands of false positives, as well as a much higher accuracy rate. The idea of this filtering was to increase the confidence of the reaching events which it did have success in doing. Ideally as this area of research develops there would be potentially to decrease or remove the need for post training filtering however there may still require a 5 frame filtering as it is very difficult to determine where a hand is going in those early movements off the steering wheel.

Chapter 8

Conclusion and Further Work

This research project had a hypothesis that stated, "through the use of a fixed camera installed in a car and with some image pre-processing and machine learning algorithms applied, a machine can detect the act of reaching for a phone.", which has been validated with the results showing approximately a 29% accuracy in detection. While these numbers are low, they do show a significant potential for improvement, and using observations, areas of further work have been identified in order to increase this percentage. Based on this accuracy and identification of areas of issue, it can be stated that a fixed camera in a car, with image pre-processing and machine learning algorithms, can detect the act of reaching for a phone.

This research stepped through the different stages of training, identifying what worked well and what required more attention. It has highlighted areas of fault that became obvious after the final stage of validation and effectively compared multiple different methods of machine learning algorithms. These areas included the need for a securely fixed mounting device, a validation method for image positioning, a larger and more diverse data set and some other areas for pre-processing application in order to lower the required detection time.

With further research in this area, in vehicle detection of pre-emptive mobile phone usage will become feasible. This detection could help to eliminate the use of mobile phones on the road, significantly decreasing the risk of injury or death to those driving with the phone or drivers/pedestrians in their immediate area. By implementing a device that allows for this detection, similarly to a no seatbelt buzzer, the deterrence of a noise is

often enough to stop people from doing the wrong thing.

8.1 Project Achievements

The achievements of this project will be reflected by comparing results and outcomes to the major objectives outlined in Section 3.2. These objectives have been listed and discussed below.

(i) Obtain legal footage of road users using their mobile phones as well as just driving normally.

Chapter 4 displays the database collection for initial testing, in which there were some requirements around consistency of data gathering and image positioning. In the first instance of collection, as a stage just to verify that the methods were feasible, the data collection was valid. This data was all collected legally, with me as the only driver, and no phone use while driving. This objective was successfully met, allowing for the review and categorization, as well as benchmark setting for the project.

(ii) Review and categorize footage.

The method of review and categorization was outlines in Section 4.1 in which the number of samples were identified. While this number was low and skewed it did allow for initial testing. The review and categorize phase was followed through the length of the project with success, this can be seen through the results at every stage of testing, in which the machine outputs the accuracy based on the review values compared to the calculated values. This phase was extremely time consuming although necessary to ensure the accuracy of results. This objective was met with every frame used for training reviewed and categorized as either reaching, driving or an abnormal event.

(iii) Use MATLAB deep-learning tool to set a benchmark on video with minimal processing.

This area was the focus of Chapter 4, in which data was collected, categorized and put into a machine for training with as minimal processing as possible. This stage required more processing then initially intended due to memory related issues. Pre-processing at this stage required trimming and an optical flow algorithm, both of which were always a requirement to make the data valid and usable. However, it did

also require cropping and data averaging to make the size of each sample smaller in order to feed an appropriate amount of data into the machines for training. While it did use more processing then initially intended, it did meet the requirements of the objective as this was the minimal amount of processing required to set a benchmark (could not be trained without this processing).

(iv) Find shortfalls in deep-learning tool and areas for optimization.

The observations of Stage 1, 2 and 3 were the locations in which optimization techniques were identified after finding shortfalls in the deep-learning tool. This objective included the identification of uneven spread of data across the different categories, the requirement for a Motion History Image approach in data, issues in the mounting and refocussing of the camera. While these optimizations were not all applied due to time constraints, this objective did successfully identify multiple areas of issue that were able to improve the accuracy of the machine learning algorithms, and have identified some areas for future works if someone were to continue this research path.

(v) Apply multiple levels of processing on images, to compare to the benchmark approach.

This objective included the implementation of a motion history image, as well as some post processing requirements to validate the data. The motion history image approach improved the accuracy approximately 20% from the previous stage. There were initial plans to trial more pre-processing techniques however with the high success of this one and the 98% accuracy it seemed more important to apply the validation stage. This was also capped due to time constraints, although the pre-processing techniques applied of trimming, cropping, optical flow, data averaging and motion history image were successful in improving the results when compared to the benchmark approach.

(vi) Use MATLAB deep-learning tool to see improvements.

This stage was conducted at the completion of Stage 2 and Stage 3 in which the data was compared to the benchmark approach. Stage 2 results showed a decrease in accuracy, which was expected due to the increasing data-set size and spread of data. Stage 3 showed a significant increase in accuracy, showing the effect that both database optimization and motion history image applications had on the accuracy of the machines output. This objective was effectively used to see improvements and identify other possible areas for improvement. IT was also successful in identifying

approaches that were not as successful or methods that were no longer followed through the length of the project, such as different training methods and the use of feature selection.

(vii) Test finalized methods on previously "unseen" data.

This was captured in Stage 4 of the project, in which the 3 most effective machine learning algorithms were trained and used to categorize previously unseen data. This included the use of a vehicle which had not been seen before. The results of this data were approximately 29%. While a higher percentage would have been ideal, the results showed that the application can successfully identify reaching events, and with some more optimization, it is expected that these accuracy percentages can become much greater. This objective was also successful in removing human bias as it was fed all data and review in a two-stage process to ensure no manipulation occurred. This objective was successfully met and showed the percent of accuracy of three of the more successful methods.

8.2 Further Work

This section gives recommendations for further work that can be applied in order to continue this research. It has been broken down into categories and detailed information has been given in each.

8.2.1 Data Acquisition

Data acquisition has been identified as an area needing attention. Below is a list of recommendations to ensure that the further development of this research gets off to the right start from the data acquisition stage of the project.

- A method of correctly mounting and ensuring all images have the same features in the same location every time it is recorded.
- Using up the full memory when training on the more successful machine learning methods.
- A more structured method based around vehicle type, such as separating hatchback

and ute into different machines, or researching a specific type alone.

- Removal of refocussing on the camera/device used for recording

These were the issues identified in the data acquisition phase of the project. Fixing up some of these areas will result in a higher accuracy from the machine, as well as allowing for more data to be fed into certain machine learning algorithms. These areas will make the later stages of training much easier to identify issues in recording, as every image will be correctly lined up to the trained data. I believe this will also remove the issue with false positives that could not be determined, as well as significantly reducing the false positive detection all together, based around camera refocussing and camera movement.

8.2.2 Data Categorization

Categorizing data into more sub sections may be beneficial in indicating exactly which areas are causing false positive. This approach will require higher values of data for those events but allows for both false positive detection and false negative detection to have more value. By only using the category, abnormal event, manual viewing needs to occur multiple times to find the source of a false negative. If training and there is a clear indication of events for example, using the gearstick, that result in false positives, it will be much easier to identify a lack of information in that area that needs addressing. Some categories that would have value in breaking down from the observations of Stage 4 are as follows.

- Reaching between legs
- Reaching right pocket
- Reaching left pocket
- Reaching to passenger
- Reaching to dash
- Reaching to console
- Road bumps
- Touching face

- Hand repositioning on wheel
- Turning corners (possibly left and right)
- Visor adjustment
- Hand returning after reaching
- Adjusting air conditioner/radio
- Gearstick movements
- Passenger movement
- Winding window down

More categories will likely increase the complexity of the machine learning algorithms and potentially increase the time to detect, although it will have much larger impact on the observations made after training, it is something that can always be refined down at the latest stage before implementation anyway, but it will help substantially in identifying further shortfalls in the detection methods and some areas for improvement or increased data acquisition.

8.2.3 Data Averaging

Some methods to improve or trial different amounts of averaging could be attempted in order to do two things. It will increase the amount of data that can be fed into the machine learning algorithms if data is increase and allows for different movements in similar locations by widening the pixel size. This method may have an adverse effect in which the average results in a value too near 0 that it becomes useless, but it will need to be trialled to see.

Another potential method is as seen below if Figures 8.1 and 8.2. In which a method of averaging down the columns and across the rows is applied. This method with no further averaging as seen in Figure 8.2 would result in a per sample reduction size from 10,268 down to 4320, and even further with an average across two rows and two columns to 1080. While it may not give an exact location of movement, the values can be crossed in many applications to determine approximate areas of movement. This averaging method would significantly increase the amount of data that can be fed into the machine, meaning

the small loss in data may be outweighed by the large increase in information received. Some values have been randomly placed to show the effect in output in both a single row/column average compared to a double row/column average.

	0	0.03	0.1	0.1	0.1	0.04	0.3	0.3	0.3	0.01
0.03	0	0.3	0	0	0	0	0	0	0	0
0.01	0	0	0	0	0	0	0	0	0	0.1
0.04	0	0	0	0	0.2	0.2	0	0	0	0
0.06	0	0	0	0	0.4	0.2	0	0	0	0
0.04	0	0	0	0	0.4	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0.3	0	0	0	0	0	0	1	1	1	0
0.5	0	0	1	1	0	0	1	1	1	0
0.3	0	0	0	0	0	0	1	1	1	0
0	0	0	0	0	0	0	0	0	0	0

Figure 8.1: Different Averaging Approach Using Single Row/Column

	0.015		0.1		0.07		0.3		0.155	
0.02	0	0.3	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0.1
0.05	0	0	0	0	0.2	0.2	0	0	0	0
	0	0	0	0	0.4	0.2	0	0	0	0
0.02	0	0	0	0	0.4	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0
0.4	0	0	0	0	0	0	1	1	1	0
	0	0	1	1	0	0	1	1	1	0
0.15	0	0	0	0	0	0	1	1	1	0
	0	0	0	0	0	0	0	0	0	0

Figure 8.2: Different Averaging Approach Using Double Row/Column

8.2.4 Data pre-processing Optimization

While the method of applying an optical flow estimation and motion history image like algorithm showed high percentage of returns, the time in which was required for processing was still far too large to be acceptable. This resulted in approximately 3 seconds required for every 1 second of data being processed. Some trialling of other methods, or use of Horne-Schunk parameter adjustment, it is important to try get this time down as much

as possible. Other factors that would impact this are the processing speeds of the device being used, so it is important to minimize this as much as possible to make it a feasible solution for an in vehicle install.

8.2.5 MATLAB Out-Of-Memory

This really only applies to the training of the machine but can simply be enhanced with hardware upgrades. The RAM is the leading impact on how much data can be fed into a machine based on memory implications, however there could be some other changes made around rounding of floats to minimize the data of each sample as well. This memory issue can be linked to the data averaging and many other methods can be applied, some of which will result in the loss of valued data. Some investigation could be conducted into best methods to get the most out of the data for lowest cost, and a simple increase in RAM size is always an option that comes with a once of monetary cost as well.

References

Blum, A. L. & Langley, P. (1997), 'Selection of relevant features and examples in machine learning', *Artificial Intelligence* **97**(1), 245 – 271. Relevance.
URL: *http://www.sciencedirect.com/science/article/pii/S0004370297000635*

Bobick, A. F. & Davis, J. W. (2001), 'The recognition of human movement using temporal templates', *IEEE Transactions on Pattern Analysis and Machine Intelligence* **23**(3), 257–267.

Clemmensen, L., Hastie, T., Witten, D. & Ersbøll, B. (2011), 'Sparse discriminant analysis', *Technometrics* **53**(4), 406–413.

Fang, G., Gao, W. & Zhao, D. (2003), 'Large vocabulary sign language recognition based on hierarchical decision trees', p. 125–131.
URL: *https://doi.org/10.1145/958432.958458*

Hastie, T., Tibshirani, R. & Friedman, J. (2009), *The elements of statistical learning: data mining, inference, and prediction*, Springer Science & Business Media.

Horn, B. K. & Schunck, B. G. (1981), Determining optical flow, *in* 'Techniques and Applications of Image Understanding', Vol. 281, International Society for Optics and Photonics, pp. 319–331.

(https://stats.stackexchange.com/users/177677/renel chesak), R. C. (2019), 'What is the difference between svm and lda?', Cross Validated. URL:https://stats.stackexchange.com/q/432038 (version: 2019-10-21).
URL: *https://stats.stackexchange.com/q/432038*

Iosifidis, A., Tefas, A., Nikolaidis, N. & Pitas, I. (2012), 'Multi-view human movement recognition based on fuzzy distances and linear discriminant analysis', *Computer Vision and Image Understanding* **116**(3), 347 – 360. Special issue on Semantic

Understanding of Human Behaviors in Image Sequences.
URL: *http://www.sciencedirect.com/science/article/pii/S1077314211002074*

Kaya, G. (2018), Good risk assessment practice in hospitals, PhD thesis.

Kotsiantis, S., Kanellopoulos, D. & Pintelas, P. (2006), 'Data preprocessing for supervised leaning', *International Journal of Computer Science* **1**(2), 111–117.

Larue & Gregoire (2010), Predicting Effects of Monotony on Driver's Vigilance, PhD thesis.

Matías, J., Taboada, J., Ordóñez, C. & Nieto, P. (2007), 'Machine learning techniques applied to the determination of road suitability for the transportation of dangerous substances', *Journal of Hazardous Materials* **147**(1), 60 – 66.
URL: *http://www.sciencedirect.com/science/article/pii/S0304389406015196*

MathWorks (2020), 'estimateflow'.
URL: *https://au.mathworks.com/help/vision/ref/opticalflowhs.estimateflow.html*

Road Safety at Work (n.d.), 'Controlling Hazards', `https://roadsafetyatwork.ca/tool-kits/controlling-hazards-and-minimizing-risks/`.

Transport for NSW (2020*a*), 'Driving Distractions and Crash Risk', `https://www.rms.nsw.gov.au/roads/safety-rules/safe-driving/driving-distractions.htmll`.

Transport for NSW (2020*b*), 'Fatality Trends', `https://roadsafety.transport.nsw.gov.au/statistics/fatalitytrends.html`.

Transport for NSW (2020*c*), 'Know The Rules', `https://roadsafety.transport.nsw.gov.au/stayingsafe/mobilephones/know-the-rules.html`.

US Department of Transportation (2018), 'Teen Distracted Driver Data', `https://crashstats.nhtsa.dot.gov/Api/Public/ViewPublication/812504l`.

World Health Organization (2018), 'Global Status Report on Road Safety 2018', `https://www.who.int/violence_injury_prevention/road_safety_status/2018/en`.

World Health Organization (2020), 'Road Traffic Injuries', `https://www.who.int/news-room/fact-sheets/detail/road-traffic-injuries`.

Xu, H., Li, L., Fang, M. & Zhang, F. (2018), 'Movement human actions recognition based on machine learning', *International Journal of Online Engineering (iJOE)* **14**, 193.

Appendix A

Project Specification

This appendix displays the project specification that was developed in April of 2020.

For: Andrew Hopkins

Title: Pre-emptive detection of mobile phone usage in-vehicle

Major: Electrical and Electronic Engineering

Supervisor: Dr. Tobias Low

Enrolment:

ENG4111 – EXT S1, 2020

ENG4112 – EXT S2, 2020

Project Aim:

To investigate the use of machine learning to pre-emptively detect the usage of mobile phones while driving.

Programme: Version 1, 08 April 2020

1. Research the background information relating to the use of machine learning on gestures and human movements. Different techniques for learning will be determined and the most appropriate applied.

2. Assemble a camera recording arrangement in vehicle to ensure all angles of the lap area of a driver can be detected.

3. Record a large dataset. This will require real drive time, as well as periods of phone usage. Methods may involve driving on private property etc to abide by road rules.

4. Reviewing of data. Supervised learning will require the data to be screened and indicators set at the different times leading up to and during phone usage.
5. Training the machine to detect the phone usage.
6. Testing the machines potential on a data set that has not gone through the machine yet.
7. Recording and analysing the results of the machines detection.

If time and resources permit:

9. Detection of areas of weakness in the system, and potential opportunities for improvement.

Appendix B

Project Timeline

This appendix displays the project timeline that was outlined in Section 3.6.2.

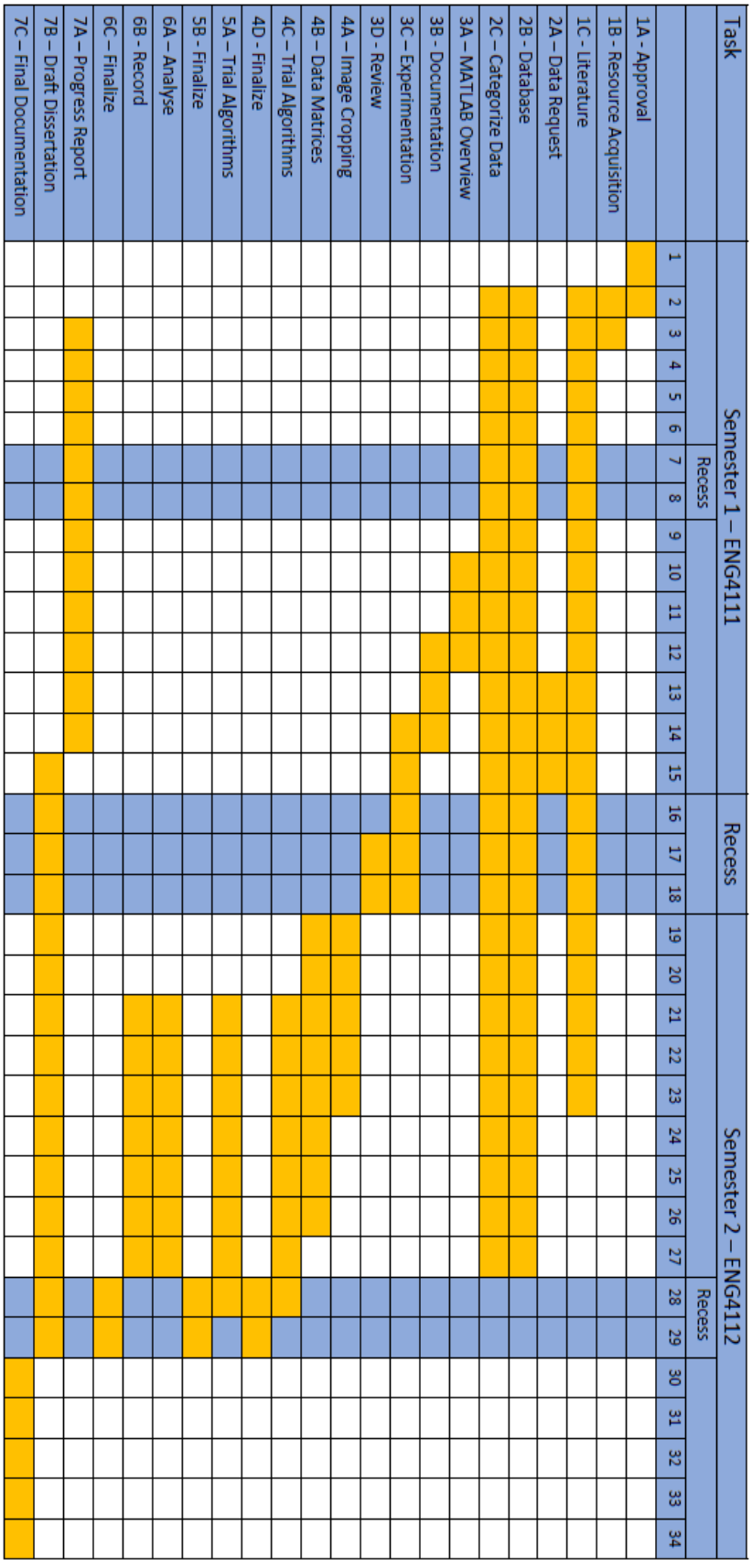

Figure B.1: Project Timeline

Appendix C

Risk Assessment

This appendix displays the risk assessment conducted before beginning work. Section 3.6.3 discussed the procedures of conducting the risk assessment.

Table C.1: Phase 1 - Risk Assessment

Phase Number	**Risk Outline**	**Risk Rating**	**Risk Controls**	**Reviewed Risk Rating**
1A	Approval not received from USQ to begin project.	Extreme 15	Early discussions with supervisor to generate a feasible project. Ensure approvals are obtained before beginning any works.	Moderate 5
1B	Resources not available for commencement.	High 9	Determine camera requirements, programs needed and hardware necessary to complete the project. Obtain required equipment as soon as project is approved.	Low 3

Table C.2: Phase 2 - Risk Assessment

Phase Number	**Risk Outline**	**Risk Rating**	**Risk Controls**	**Reviewed Risk Rating**
2A	No features can be captured effectively with one camera	High 12	Trial multiple areas, consider what features are significant to ensure these are captured. Stick to a plan once determining a position and ensure it is followed correctly	Low 2
2B	Loss of database, and any progress made with refinement/optimization.	High 10	Ensure all data is backed up and stored on two seperate devices. Original copies of all videos will be stored, as well as all data that has been manipulated for all different testing procedures.	Moderate 5
2C	Incorrect/Unstandardized way of categorizing data	High 12	Parameters will need to be created before categorization can commence to ensure all is done in the same way. In time persists, all videos will be reviewed twice. To keep this as simple as possible, videos will be watched and categorized as received to keep project flowing.	Low 3

Table C.3: Phase 3 - Risk Assessment

Phase Number	**Risk Outline**	**Risk Rating**	**Risk Controls**	**Reviewed Risk Rating**
3A	MATLAB machine training takes longer than expected.	Moderate 6	Practice training on some known MATLAB databases with their example programs. This will give a good understanding of hardware capabilities. If system is not handling, discuss early with supervisor if not appropriate, and determine other methods to make the project feasible.	Moderate 4
3B	Loss of documentation	High 10	Similarly to above, it is important to back up the documentation on a spare device, to ensure data cannot be lost.	Moderate 5
3D	Initial testing not thorough enough/needs revisiting.	High 9	Ensure literary review of machine learning is correctly complete before commencing past this initial testing. This will ensure that the benchmark is correctly set and that all algorithms are understood and arranged correctly for initial values.	Low 3

Table C.4: Phase 4 - Risk Assessment

Phase Number	**Risk Outline**	**Risk Rating**	**Risk Controls**	**Reviewed Risk Rating**
4A	Image cropping results in the loss of significant data	Extreme 16	Standard template must be used for image cropping. Due to the use of data from multiple sources, this may need to be done case by case. Discussion with supervisor to ensure method usage is appropriate.	Moderate 6
4B	Large data storage may be time consuming and hard to process	Moderate 6	Complete as the data is received to ensure no back log of work. Ensure stored data in the first instance is simply the original file and its finalized forms of processing techniques ie. greyscale.	Low 1
4C	Unable to find any useful algorithms	High 8	Continue literary review. There are a significant amount of research papers in the area of machine learning of video image. Discuss alternatives with supervisor	Moderate 4
4D	Finalizing done too early, compromising results	High 8	Ensure all potential methods have been reviewed within the time constraints of completing the project. If time becomes an issue, all data should be collected and tested by this point. So it will be a matter of just selecting the most effective method and continuing at this point.	Low 2

Table C.5: Phase 5 - Project risk assessment

Phase Number	**Risk Outline**	**Risk Rating**	**Risk Controls**	**Reviewed Risk Rating**
5A	No algorithms found to work successfully from the beginning.	High 8	Resume literary review to determine other methods for machine learning of this context. Discuss alternative options with supervisor	Moderate 4
5B	Finalizing done too early, compromising results.	High 8	Ensure all potential methods have been reviewed within the time constraints of completing the project. If time ba- comes an issue, all data should be collected and tested by this point. So it will be a matter of just selecting the most affective method and continuing at this point.	Low 2

Table C.6: Phase 6 - Project risk assessment

Phase Number	**Risk Outline**	**Risk Rating**	**Risk Controls**	**Reviewed Risk Rating**
6A & B	Lack of detail in analysis resulting in loss of information.	High 12	Standard method for storing this information will be taken for each step. Significant data such as training time, processing time on test images and accuracy will be tested for every different method applied and stored in the same fashion for each.	Low 2
6C	Final test has extremely poor result.	High 12	By this point, all methods and approaches have been taken to ensure everything was done correctly. Ensure that the final test data was manipulated to the same standard as all other test data. Ensure data set is large enough throughout the entire project. Discuss with supervisor and move on. Time will be limited at this point to be caught up on results.	Moderate 4

Table C.7: Phase 7 - Project risk assessment

Phase Number	**Risk Outline**	**Risk Rating**	**Risk Controls**	**Reviewed Risk Rating**
7A	Progress report not complete on time	High 12	Start report as early as possible and make it the focus of the project until complete. This is the baseline of the project and needs to be treated with high priority	Low 1
7B	Not enough time given for supervisor to provide feedback on draft.	High 9	Dissertation should be completed alongside the project weekly. This will ensure that a significant amount of work is not required in this area alone at the end. Discuss with supervisor the possibility of submitting updates of documentation at more frequent intervals for review of changes.	Low 1
7A	Unable to attend Professional Practice 2 on campus course	High 10	Apply for leave as soon as the dates are known.	Low 1

Appendix D

Stage 1, 2 and 3 Result Tables

This appendix displays the results tables that were developed in Stages 1, 2 and 3 of development.

Table D.1: Stage 1 Machine Learning Results 1 of 2

				Accuracy (%)			
	Classifier Type	Time to Train(s)	Approx. Prediction Rate(obs/s)	Reaching	Driving	Abnormal	Total
Discriminant	Linear	735	170	51.5	66.3	49	57
	Quadratic	Failed	-	-	-	-	-
	Subspace	8087	29	64.4	47.3	40.2	54.6
K-Nearest Neighbour	Coarse	970	28	0.1	100	0	39.4
	Cosine	924	26	76	95.8	68.8	82.8
	Cubic	Failed	-	-	-	-	-
	Fine	1040	22	74.7	89.1	61.6	78.7
	Medium	898	25	19.2	74.9	67.5	64.3
	Subspace	Failed	-	-	-	-	-
	Weighted	Failed	-	-	-	-	-
Naive	Gaussian	221	470	67.9	95.8	60.9	75.7
	Kernel	Failed	-	-	-	-	-

Table D.2: Stage 1 Machine Learning Results 2 of 2

				Accuracy (%)			
	Classifier Type	Time to Train(s)	Approx. Prediction Rate(obs/s)	Reaching	Driving	Abnormal	Total
Support Vector Machines	Coarse Gaussian	2278	20	99.8	4.7	0.3	49.8
	Cubic	1752	24	74.5	96	46.9	79.5
	Fine Gaussian	2317	18	98.2	41.8	18	65.8
	Linear	888	410	65.6	74.8	19.2	63.4
	Medium Gaussian	2135	21	85.8	86.2	28.7	78.6
	Quadratic	2622	13	76.5	96.5	42.1	80
Trees	Bagged	255	490	91.3	88.9	59.2	86.3
	Boosted	5748	490	90.2	77.5	31.8	77.8
	Fine	1427	510	77.8	75.2	48.8	73.1
	RUSBoosted	1139	490	75.2	83.1	76.8	78.5

Table D.3: Stage 2 Machine Learning Results 1 of 2

				Accuracy (%)			
	Classifier Type	Time to Train(s)	Approx. Prediction Rate(obs/s)	Reaching	Driving	Abnormal	Total
Discriminant	Linear	Failed	-	-	-	-	-
	Quadratic	Failed	-	-	-	-	-
	Subspace	2558	330	0	100	0	42.5
K-Nearest Neighbour	Coarse	1130	33	1.9	99.5	0	42.9
	Cosine	1319	28	80.7	91	60.8	79
	Cubic	Failed	-	-	-	-	-
	Fine	1430	25	68.5	86.3	66.8	75.6
	Medium	1438	27	54.4	68.4	32.2	54.3
	Subspace	Failed	-	-	-	-	-
	Weighted	Failed	-	-	-	-	-
Naive	Gaussian	363	440	52.1	68.3	78	66
	Kernel	Failed	-	-	-	-	-

Table D.4: Stage 2 Machine Learning Results 2 of 2

				Accuracy (%)			
	Classifier Type	Time to Train(s)	Approx. Prediction Rate(obs/s)	Reaching	Driving	Abnormal	Total
Support Vector Machines	Coarse Gaussian	5550	12	2.6	99.7	0.3	43.2
	Cubic	5527	12	66.7	92.4	61.2	76.1
	Fine Gaussian	6043	11	13.3	100	4.6	47.8
	Linear	3042	200	31.5	91.1	22.6	54.4
	Medium Gaussian	5502	11	63	92.4	57.7	74.1
	Quadratic	5544	12	63.7	92	57.3	74
Trees	Bagged	371	390	67.1	90	75.9	79.2
	Boosted	7224	350	51.5	83.5	48.3	64.3
	Fine	416	510	53.8	75.2	46.7	61
	RUSBoosted	5558	290	65.7	64.8	56.7	62.9

Table D.5: Stage 3 Machine Learning Results 1 of 2

				Accuracy (%)			
	Classifier Type	Time to Train(s)	Approx. Prediction Rate(obs/s)	Reaching	Driving	Abnormal	Total
Discriminant	Linear	Failed	-	-	-	-	-
	Quadratic	Failed	-	-	-	-	-
	Subspace	5284	270	49.8	94.3	53.4	68.8
K-Nearest Neighbour	Coarse	1015	31	1.6	99.8	2.9	41.7
	Cosine	1012	35	90.1	96.2	96.1	94.2
	Cubic	Failed	-	-	-	-	-
	Fine	1142	30	97.8	98.7	98.1	98.3
	Medium	1030	35	59.8	99.1	66.6	77.5
	Subspace	Failed	-	-	-	-	-
	Weighted	Failed	-	-	-	-	-
Naive	Gaussian	313	490	50.2	71.2	81.9	67.4
	Kernel	Failed	-	-	-	-	-

Table D.6: Stage 3 Machine Learning Results 2 of 2

				Accuracy (%)			
	Classifier Type	Time to Train(s)	Approx. Prediction Rate(obs/s)	Reaching	Driving	Abnormal	Total
Support Vector Machines	Coarse Gaussian	4533	13	37.9	99.5	20.2	57.9
	Cubic	1943	34	97.2	99	95.5	97.5
	Fine Gaussian	5033	12	67.1	99.6	55.9	77.2
	Linear	1566	450	82.8	97.5	83.4	89
	Medium Gaussian	3588	16	94.9	98.6	91.4	95.4
	Quadratic	1913	36	89.7	98.7	93.2	94.3
Trees	Bagged	244	560	94.8	97.6	95.9	96.2
	Boosted	4500	570	63.7	92.7	56.2	73.4
	Fine	301	560	65.7	87.6	55.6	71.8
	RUSBoosted	3818	580	76.9	77.5	57.2	71.7

Appendix E

Validation Result Tables

This appendix displays the results tables that were created in the validation stage of development. The ute had approximately 7 minutes of recording, the Sedan approximately 6.5 minutes, the wagon approximately 8 minutes and the hatchback approximately 5.5 minutes. Each vehicle reviewed 10 samples of each reaching event, and some additional regular driving.

Table E.1: Ute True Positive Counts

	Tree	SVM	KNN
Between Legs	0	3	2
Left Pocket	1	2	4
Right Pocket	2	6	4
Passenger	3	2	2
Console	4	5	2
Dashboard	1	0	2
Accuracy(%)	18.33	30	26.67

Table E.2: Sedan True Positive Counts

	Tree	SVM	KNN
Between Legs	1	2	0
Left Pocket	4	6	3
Right Pocket	3	5	2
Passenger	2	1	0
Console	2	4	2
Dashboard	3	3	1
Accuracy(%)	25	35	13.33

Table E.3: Wagon True Positive Counts

	Tree	SVM	KNN
Between Legs	5	4	5
Left Pocket	2	9	3
Right Pocket	6	6	4
Passenger	2	2	2
Console	5	2	4
Dashboard	4	2	5
Accuracy(%)	40	41.67	38.33

Table E.4: Hatchback True Positive Counts

	Tree	SVM	KNN
Between Legs	0	1	3
Left Pocket	1	0	2
Right Pocket	4	1	6
Passenger	7	2	6
Console	3	2	2
Dashboard	2	0	4
Accuracy(%)	28.33	10	38.33

Table E.5: All Vehicle True Positive Counts

	Tree	SVM	KNN
Between Legs	6	10	10
Left Pocket	8	17	12
Right Pocket	15	18	16
Passenger	14	7	10
Console	14	13	10
Dashboard	10	5	12
Accuracy(%)	28.33	10	38.33

Table E.6: Ute False Positive Counts

	Tree	SVM	KNN
Roadbump	0	0	0
Touching Face	0	0	1
Hand Reposition on Steering Wheel	0	1	1
Could not Determine	2	9	4
Turning Corner	4	2	1
Visor Adjustment	0	2	0
Hand Returning After Reaching	8	20	14
Airconditioner/Radio	1	1	0
Camera Refocus	2	2	1
Camera Shift	6	6	1
Gearstick	2	10	2
Passenger in Frame	1	7	6
Winding Window Down	0	0	0
Errors per minute	3.65	8.41	4.35

Table E.7: Sedan False Positive Counts

	Tree	SVM	KNN
Roadbump	0	0	0
Touching Face	0	0	0
Hand Reposition on Steering Wheel	0	1	1
Could not Determine	5	3	3
Turning Corner	6	7	4
Visor Adjustment	1	0	2
Hand Returning After Reaching	4	21	7
Airconditioner/Radio	1	2	0
Camera Refocus	5	2	3
Camera Shift	4	12	4
Gearstick	0	2	3
Passenger in Frame	0	0	0
Winding Window Down	0	0	0
Errors per minute	4.03	7.76	4.19

Table E.8: Wagon False Positive Counts

	Tree	SVM	KNN
Roadbump	0	3	3
Touching Face	0	3	3
Hand Reposition on Steering Wheel	2	5	6
Could not Determine	0	8	7
Turning Corner	3	9	10
Visor Adjustment	1	2	1
Hand Returning After Reaching	28	26	22
Airconditioner/Radio	0	0	0
Camera Refocus	0	0	0
Camera Shift	1	2	6
Gearstick	0	0	0
Passenger in Frame	0	1	1
Winding Window Down	0	1	1
Errors per minute	4.27	7.32	7.32

Table E.9: Hatchback False Positive Counts

	Tree	SVM	KNN
Roadbump	0	0	0
Touching Face	0	0	0
Hand Reposition on Steering Wheel	1	0	0
Could not Determine	0	0	1
Turning Corner	3	1	2
Visor Adjustment	1	0	0
Hand Returning After Reaching	7	5	15
Airconditioner/Radio	0	1	0
Camera Refocus	1	1	2
Camera Shift	2	3	2
Gearstick	0	0	0
Passenger in Frame	0	0	0
Winding Window Down	0	0	0
Errors per minute	2.76	2.02	4.04

Table E.10: All Vehicle False Positive Counts

	Tree	SVM	KNN
Roadbump	0	3	3
Touching Face	0	3	3
Hand Reposition on Steering Wheel	3	6	8
Could not Determine	5	12	12
Turning Corner	14	26	20
Visor Adjustment	7	4	4
Hand Returning After Reaching	39	54	44
Airconditioner/Radio	9	23	14
Camera Refocus	7	4	5
Camera Shift	9	19	13
Gearstick	6	8	4
Passenger in Frame	2	11	3
Winding Window Down	1	8	7
Errors per minute	3.74	6.65	5.14

Appendix F

Frame Calculating Function

This function was used to determine at which frame movements began, to allow for the detection of first frame of classiication change. It simply outputs the image of that frame, and a frame number so that both can be seen and manually entered into classiciation array. This is referenced in Section 4.1.

Listing F.1: Frame Calculating Function.

```
% Used to determine exact frame where categories change
% This is done after watching full speed and determining approx.
    ↪ time
% Enter video name and start time in lines 7 & 8
% Press enter to move to next frame, Command window outputs
    ↪ current frame
% number

%clc;
%clear;
start = 30;             %always use an integer(seconds into
    ↪ recording)
vidReader = VideoReader('Sedan_2.mp4','CurrentTime',start);
frame = start*30;

% Components of the optical flow calculator .m file
h = figure;
movegui(h);
hViewPanel = uipanel(h,'Position',[0 0 1 1],'Title','Plot of
    ↪ Optical Flow Vectors');
hPlot = axes(hViewPanel);

while hasFrame(vidReader)
    frameRGB = readFrame(vidReader);
    imshow(frameRGB)
    hold on
    frame = frame +1
    input('')
end
```

Appendix G

Classification Array Function

This appendix shows the manual entry method for classification arrays to ensure accurate results. This is referenced in Section 4.1.

Listing G.1: Classification Array Function.

```
%Array creator, take down numbers before trimming/cropping
% Stored at top
% eg. Video6_1 was trimmed between frame 20 to 101
% Up to frame 63 was driving and reaching aften until end

% 1 - 29 Predictors stored in MATLAB Data OP6HS_1

%Video 6_1
% Trimmed to 20 - 101
array6_1(1:43) = "Driving";
array6_1(44:82) = "Reaching";

%Video 6_2
% notused

%Video 6_3
 % Trimmed to 1 - 108
 array6_3(1:14) = "Driving";
 array6_3(15:53) = "Abnormal";
 array6_3(54:108) = "Reaching";

%Video 6_4
% Trimmed to 23 - 99
array6_4(1:17) = "Driving";
array6_4(18:77) = "Reaching";

%Video 6_5
% Trimmed to 135 - 190
array6_5(1:56) = "Reaching";
```

```
W1Console3 = array6_5(10:56);

%Video 6_6
% Trimmed to 1 - 172
array6_6(1:40) = "Driving";
array6_6(41:110) = "Abnormal";
array6_6(111:172) = "Reaching";

%Video 6_7
% Trimmed to 1 - 90
array6_7(1:38) = "Driving";
array6_7(39:90) = "Reaching";

%Video 6_8
% Trimmed to 1 - 66
 array6_8(1:66) = "Reaching";

%Video 6_82
% Trimmed to 1 - 90
array6_82(1:59) = "Abnormal";
array6_82(60:90) = "Reaching";

%Video 6_83
% Trimmed to 1 - 85
array6_83(1:44) = "Abnormal";
array6_83(45:85) = "Reaching";

%Video 6_84
% Trimmed to 1 - 120
array6_84(1:79) = "Abnormal";
array6_84(80:120) = "Reaching";

arrayClassifier = [array6_1 array6_3 array6_4 array6_5 array6_6
   ↪ array6_7 array6_8 array6_9 array6_10 array6_11 array6_12
   ↪ array6_13 array6_14 array6_15 array6_16 array6_17
   ↪ array6_18 array6_19 array6_20 array6_21 array6_22
   ↪ array6_23 array6_24 array6_25 array6_26 array6_27
   ↪ array6_28 array6_29];
arrayClassifier = arrayClassifier';

ab1 = sum(arrayClassifier(:) == "Abnormal");
dr1 = sum(arrayClassifier(:) == "Driving");
re1 = sum(arrayClassifier(:) == "Reaching");
```

Appendix H

Footage Cropper/Trimming Function

This appendix displays the MATLAB code that was used for both cropping and trimming of video. Parts of this code were sourced from the Mathworks open source forum, as specified in the top line of commenting. This is referenced in Section 4.2.1.

Listing H.1: Cropping Function.

```
%https://au.mathworks.com/matlabcentral/fileexchange/63421-
    ↪ cropping-video
%clear all
%clc

start = 1;
stop = 2 %10228;

vid1=VideoReader('Ute_1.mp4');
n=vid1.NumberOfFrames;
writerObj1 = VideoWriter('Ute_1.avi');
open(writerObj1);
i=start;

while i <= stop
  im=read(vid1,i);
  imc=imcrop(im,[0 700 1080 1080]);% x1 y1    x2 y2, 700 crops
     ↪ out any window, 600 includes top of steering wheel.
 % img=rgb2gray(im);
 % [a,b]=size(img);
  %imc=imresize(imc,[a,b]);
   writeVideo(writerObj1,imc);
   %  subplot(1,3,1)
   %  imshow(im)
   %  subplot(1,3,2)
   %  imshow(imc)
     i=i+1;
end
["start" "stop" "count"]
[start stop writerObj1.FrameCount]

close(writerObj1)
```

Appendix I

Optical Flow and Averaging Function

This appendix displays the code used for both the optical flow calculation and the data aeraging method. This is referenced in Section 4.2.3. The optical flow component of this program was sourced from Mathworks, as referenced at the top of the commenting. Some adjustments were made and additional commenting added, some sections were commented out as they were used for debugging and calculating timing of pre-processing techniques.

Listing I.1: Optical Flow and Averaging Function.

```
%https://au.mathworks.com/help/vision/ref/opticalflowhs.html
% Calculates optical flow and performs the quantization
% Each scan of data is then spread accross one row in the array
% Video name to be inserted in line 7 and array name in second
    ↪ last line

%clc;
vidReader = VideoReader('Ute_1.avi','CurrentTime',0);
opticFlow = opticalFlowHS('MaxIteration',10)
frame = 0;
i=1;
C = [];
D1 = zeros(vidReader.NumFrames,10368);  % Preallocated for speed
    ↪  related issues

%DISPLAY-------------------
%  h = figure;
%  movegui(h);
%  hViewPanel = uipanel(h,'Position',[0 0 1 1],'Title','Plot of
    ↪ Optical Flow Vectors');
%  hPlot = axes(hViewPanel);
```

```
%-------------------------------


while hasFrame(vidReader)

    frameRGB = readFrame(vidReader);
    frameGray = rgb2gray(frameRGB);
    frame = frame + 1;

    if(frame == 1)
      % tic;                                          % Used to
         ↪ detect time for flow calculation
       frame = 0;
       flow = estimateFlow(opticFlow,frameGray);
       flowrx = flow.Vx;                              % Flow in x
       flowry = flow.Vy;                              % Flow in y
       t=1;s=1;                                       % Array
          ↪ reset
      % y = toc                                       % Used to
         ↪ detect time for flow calculation

       %DISPLAY-----------------------------------
%          figure(2);
%        imshow(frameRGB);
%          %hold on
%          %plot(flow,'DecimationFactor',[5 5],'ScaleFactor
   ↪ ',60,'Parent',hPlot);
%          figure(1);
%          plot(flowrx,flowry);
%          xlim([0 1]);
%          ylim([0 1]);

        %-------------------------
       % tic;                                          % Used to
          ↪ detect time for quantization
        %Rolution adjustment, currently set to 15 pixels
       while(t<=72)
            while(s<=72)
                flowrxR(t,s) = mean(mean(flowrx(15*t-14:15*t,15*s
                   ↪ -14:15*s)));
                flowryR(t,s) = mean(mean(flowry(15*t-14:15*t,15*s
                   ↪ -14:15*s)));
                s=s+1;
            end
            t = t+1;
            s = 1;
       end
       %z = toc                                       % Used to
          ↪ detect time for quantization
       A2 = reshape(flowrxR',1,[]);                  % Lays data
          ↪ onto one array row
```

```
        B2 = reshape(flowryR',1,[]);                    % Lays data
            ↪ onto one array row
        C = [C A2 B2];                                  % Lays data
            ↪ onto one array row
        D1(i,:) = C;                                    % This step
            ↪ cumulatively gets worse.
        C = [];                                         % Reset C to
            ↪  zero to ensure no excess carried over
        i=i+1
    end

    %DISPLAY-------------------
%       figure(3);
%           plot(flowrxR,flowryR);
%           xlim([0 1]);
%           ylim([0 1]);
%           pause(10^-3)
         %-------------------------------

end



Ute_1 = D1;
beep;                                                   % output
    ↪ noise when complete
```

Appendix J

Optical Flow History Function

This appendix shows the optical flow history created for this project with MATLAB. It uses a very similar approach to that of a Motion History Image, of which the method and calculation can be seen in Section 6.1.

Listing J.1: Optical Flow History Function.

```
% Function used to add Motion History to the image flow data
% Simply input the vector that is the output of Horne-Schunk
    ↪ Optical Flow
% and return will equal the MHI

input = Wagon_2;                                    % Input vector
    ↪ here
MHI = zeros(size(input,1)-10,size(input,2));        % Preallocate
    ↪ for speed reasons

for i=10:size(input,1)
    for j=1:size(input,2)
        MHI(i-9,j) = input(i,j) + 0.9*input(i-1,j) + 0.8*input(i
            ↪ -2,j) + 0.7*input(i-3,j) + 0.6*input(i-4,j) + 0.5*
            ↪ input(i-5,j) + 0.4*input(i-6,j) + 0.3*input(i-7,j)
            ↪  + 0.2*input(i-8,j) + 0.1*input(i-9,j);
    end
end

HMIOutput = MHI;
```

Appendix K

Bagged Tree Machine Learning Function

This appendix shows the bagged tree machine learning function that was output from the classification tool for this project with MATLAB. This is referenced in Section 6.4.2.

Listing K.1: Bagged Tree Machine Learning Function.

```
function [trainedClassifier, validationAccuracy] =
    ↪ trainClassifier(trainingData, responseData)
% [trainedClassifier, validationAccuracy] = trainClassifier(
    ↪ trainingData,
% responseData)
% Returns a trained classifier and its accuracy. This code
    ↪ recreates the
% classification model trained in Classification Learner app.
    ↪ Use the
% generated code to automate training the same model with new
    ↪ data, or to
% learn how to programmatically train models.
%
%  Input:
%      trainingData: A matrix with the same number of columns
    ↪ and data type
%        as the matrix imported into the app.
%
%      responseData: A vector with the same data type as the
    ↪ vector
%        imported into the app. The length of responseData and
    ↪ the number of
%        rows of trainingData must be equal.
%
%  Output:
```

```
%       trainedClassifier: A struct containing the trained
    ↪ classifier. The
%         struct contains various fields with information about
    ↪ the trained
%         classifier.
%
%       trainedClassifier.predictFcn: A function to make
    ↪ predictions on new
%         data.
%
%       validationAccuracy: A double containing the accuracy in
    ↪ percent. In
%         the app, the History list displays this overall accuracy
    ↪  score for
%         each model.
%
% Use the code to train the model with new data. To retrain your
% classifier, call the function from the command line with your
    ↪ original
% data or new data as the input arguments trainingData and
    ↪ responseData.
%
% For example, to retrain a classifier trained with the original
    ↪  data set T
% and response Y, enter:
%   [trainedClassifier, validationAccuracy] = trainClassifier(T,
    ↪  Y)
%
% To make predictions with the returned 'trainedClassifier' on
    ↪ new data T2,
% use
%   yfit = trainedClassifier.predictFcn(T2)
%
% T2 must be a matrix containing only the predictor columns used
    ↪  for
% training. For details, enter:
%   trainedClassifier.HowToPredict

% Auto-generated by MATLAB on 31-Aug-2020 19:42:48


% Extract predictors and response
% This code processes the data into the right shape for training
    ↪  the
% model.
% Convert input to table
inputTable = array2table(trainingData, 'VariableNames', {'
    ↪ column_1', 'column_2', 'column_10367', 'column_10368'});

predictorNames = {'column_1', 'column_2', 'column_10367', '
    ↪ column_10368'};
predictors = inputTable(:, predictorNames);
```

```
response = responseData(:);
isCategoricalPredictor = [false, false, false, false];

% Train a classifier
% This code specifies all the classifier options and trains the
    ↪ classifier.
template = templateTree(...
    'MaxNumSplits', 7668);
classificationEnsemble = fitcensemble(...
    predictors, ...
    response, ...
    'Method', 'Bag', ...
    'NumLearningCycles', 30, ...
    'Learners', template, ...
    'ClassNames', {'Abnormal'; 'Driving'; 'Reaching'});

% Create the result struct with predict function
predictorExtractionFcn = @(x) array2table(x, 'VariableNames',
    ↪ predictorNames);
ensemblePredictFcn = @(x) predict(classificationEnsemble, x);
trainedClassifier.predictFcn = @(x) ensemblePredictFcn(
    ↪ predictorExtractionFcn(x));

% Add additional fields to the result struct
trainedClassifier.ClassificationEnsemble =
    ↪ classificationEnsemble;
trainedClassifier.About = 'This␣struct␣is␣a␣trained␣model␣
    ↪ exported␣from␣Classification␣Learner␣R2020a.';
trainedClassifier.HowToPredict = sprintf('To␣make␣predictions␣on
    ↪ ␣a␣new␣predictor␣column␣matrix,␣X,␣use:␣\n␣␣yfit␣=␣c.
    ↪ predictFcn(X)␣\nreplacing␣''c''␣with␣the␣name␣of␣the␣
    ↪ variable␣that␣is␣this␣struct,␣e.g.␣''trainedModel''.␣\n␣\
    ↪ nX␣must␣contain␣exactly␣10368␣columns␣because␣this␣model␣
    ↪ was␣trained␣using␣10368␣predictors.␣\nX␣must␣contain␣only␣
    ↪ predictor␣columns␣in␣exactly␣the␣same␣order␣and␣format␣as␣
    ↪ your␣training␣\ndata.␣Do␣not␣include␣the␣response␣column␣
    ↪ or␣any␣columns␣you␣did␣not␣import␣into␣the␣app.␣\n␣\nFor␣
    ↪ more␣information,␣see␣<a␣href="matlab:helpview(fullfile(
    ↪ docroot,␣''stats'',␣''stats.map''),␣''
    ↪ appclassification_exportmodeltoworkspace'')">How␣to␣
    ↪ predict␣using␣an␣exported␣model</a>.');

% Extract predictors and response
% This code processes the data into the right shape for training
    ↪ the
% model.
% Convert input to table
inputTable = array2table(trainingData, 'VariableNames', {'
    ↪ column_1', 'column_2', 'column_10367', 'column_10368'});

predictorNames = {'column_1', 'column_2','column_10367', '
    ↪ column_10368'};
```

```
predictors = inputTable(:, predictorNames);
response = responseData(:);
isCategoricalPredictor = [false, false, false, false];

% Perform cross-validation
partitionedModel = crossval(trainedClassifier.
    ↪ ClassificationEnsemble, 'KFold', 5);

% Compute validation predictions
[validationPredictions, validationScores] = kfoldPredict(
    ↪ partitionedModel);

% Compute validation accuracy
validationAccuracy = 1 - kfoldLoss(partitionedModel, 'LossFun',
    ↪ 'ClassifError');
```

Appendix L

Cosine KNN Machine Learning Function

This appendix shows the Cosine KNN machine learning function that was output from the classification tool for this project with MATLAB. This is referenced in Section 6.4.2.

Listing L.1: Bagged Tree Machine Learning Function.

```
function [trainedClassifier, validationAccuracy] =
    ↪ trainClassifier(trainingData, responseData)
% [trainedClassifier, validationAccuracy] = trainClassifier(
    ↪ trainingData,
% responseData)
% Returns a trained classifier and its accuracy. This code
    ↪ recreates the
% classification model trained in Classification Learner app.
    ↪ Use the
% generated code to automate training the same model with new
    ↪ data, or to
% learn how to programmatically train models.
%
%  Input:
%      trainingData: A matrix with the same number of columns
    ↪ and data type
%        as the matrix imported into the app.
%
%      responseData: A vector with the same data type as the
    ↪ vector
%        imported into the app. The length of responseData and
    ↪ the number of
%        rows of trainingData must be equal.
%
%  Output:
```

```
%       trainedClassifier: A struct containing the trained
    ↪ classifier. The
%         struct contains various fields with information about
    ↪ the trained
%         classifier.
%
%       trainedClassifier.predictFcn: A function to make
    ↪ predictions on new
%         data.
%
%       validationAccuracy: A double containing the accuracy in
    ↪ percent. In
%         the app, the History list displays this overall accuracy
    ↪  score for
%         each model.
%
% Use the code to train the model with new data. To retrain your
% classifier, call the function from the command line with your
    ↪ original
% data or new data as the input arguments trainingData and
    ↪ responseData.
%
% For example, to retrain a classifier trained with the original
    ↪  data set T
% and response Y, enter:
%   [trainedClassifier, validationAccuracy] = trainClassifier(T,
    ↪  Y)
%
% To make predictions with the returned 'trainedClassifier' on
    ↪ new data T2,
% use
%   yfit = trainedClassifier.predictFcn(T2)
%
% T2 must be a matrix containing only the predictor columns used
    ↪  for
% training. For details, enter:
%   trainedClassifier.HowToPredict

% Auto-generated by MATLAB on 08-Sep-2020 15:23:26


% Extract predictors and response
% This code processes the data into the right shape for training
    ↪  the
% model.
% Convert input to table
inputTable = array2table(trainingData, 'VariableNames', {'
    ↪ column_1', 'column_2', 'column_10367', 'column_10368'});

predictorNames = {'column_1', 'column_2', 'column_10367', '
    ↪ column_10368'};
predictors = inputTable(:, predictorNames);
```

```
response = responseData(:);
isCategoricalPredictor = [false, false, false, false];

% Train a classifier
% This code specifies all the classifier options and trains the
    ↪ classifier.
classificationKNN = fitcknn(...
    predictors, ...
    response, ...
    'Distance', 'Cosine', ...
    'Exponent', [], ...
    'NumNeighbors', 10, ...
    'DistanceWeight', 'Equal', ...
    'Standardize', true, ...
    'ClassNames', {'Abnormal'; 'Driving'; 'Reaching'});

% Create the result struct with predict function
predictorExtractionFcn = @(x) array2table(x, 'VariableNames',
    ↪ predictorNames);
knnPredictFcn = @(x) predict(classificationKNN, x);
trainedClassifier.predictFcn = @(x) knnPredictFcn(
    ↪ predictorExtractionFcn(x));

% Add additional fields to the result struct
trainedClassifier.ClassificationKNN = classificationKNN;
trainedClassifier.About = 'This␣struct␣is␣a␣trained␣model␣
    ↪ exported␣from␣Classification␣Learner␣R2020a.';
trainedClassifier.HowToPredict = sprintf('To␣make␣predictions␣on
    ↪ ␣a␣new␣predictor␣column␣matrix,␣X,␣use:␣\n␣␣yfit␣=␣c.
    ↪ predictFcn(X)␣\nreplacing␣''c''␣with␣the␣name␣of␣the␣
    ↪ variable␣that␣is␣this␣struct,␣e.g.␣''trainedModel''.␣\n␣\
    ↪ nX␣must␣contain␣exactly␣10368␣columns␣because␣this␣model␣
    ↪ was␣trained␣using␣10368␣predictors.␣\nX␣must␣contain␣only␣
    ↪ predictor␣columns␣in␣exactly␣the␣same␣order␣and␣format␣as␣
    ↪ your␣training␣\ndata.␣Do␣not␣include␣the␣response␣column␣
    ↪ or␣any␣columns␣you␣did␣not␣import␣into␣the␣app.␣\n␣\nFor␣
    ↪ more␣information,␣see␣<a␣href="matlab:helpview(fullfile(
    ↪ docroot,␣''stats'',␣''stats.map''),␣''
    ↪ appclassification_exportmodeltoworkspace'')">How␣to␣
    ↪ predict␣using␣an␣exported␣model</a>.');

% Extract predictors and response
% This code processes the data into the right shape for training
    ↪  the
% model.
% Convert input to table
inputTable = array2table(trainingData, 'VariableNames', {'
    ↪ column_1', 'column_2', 'column_10367', 'column_10368'});

predictorNames = {'column_1', 'column_2', 'column_10367', '
    ↪ column_10368'};
predictors = inputTable(:, predictorNames);
```

```
response = responseData(:);
isCategoricalPredictor = [false, false, false, false];

% Perform cross-validation
partitionedModel = crossval(trainedClassifier.ClassificationKNN,
    ↪ 'KFold', 5);

% Compute validation predictions
[validationPredictions, validationScores] = kfoldPredict(
    ↪ partitionedModel);

% Compute validation accuracy
validationAccuracy = 1 - kfoldLoss(partitionedModel, 'LossFun',
    ↪ 'ClassifError');
```

Appendix M

Cubic SVM Machine Learning Function

This appendix shows the Cubic SVM machine learning function that was output from the classification tool for this project with MATLAB. This is referenced in Section 6.4.2.

Listing M.1: Bagged Tree Machine Learning Function.

```
function [trainedClassifier, validationAccuracy] =
    ↪ trainClassifier(trainingData, responseData)
% [trainedClassifier, validationAccuracy] = trainClassifier(
    ↪ trainingData,
% responseData)
% Returns a trained classifier and its accuracy. This code
    ↪ recreates the
% classification model trained in Classification Learner app.
    ↪ Use the
% generated code to automate training the same model with new
    ↪ data, or to
% learn how to programmatically train models.
%
%  Input:
%      trainingData: A matrix with the same number of columns
    ↪ and data type
%        as the matrix imported into the app.
%
%      responseData: A vector with the same data type as the
    ↪ vector
%        imported into the app. The length of responseData and
    ↪ the number of
%        rows of trainingData must be equal.
%
%  Output:
```

```
%       trainedClassifier: A struct containing the trained
    ↪ classifier. The
%         struct contains various fields with information about
    ↪ the trained
%         classifier.
%
%       trainedClassifier.predictFcn: A function to make
    ↪ predictions on new
%         data.
%
%       validationAccuracy: A double containing the accuracy in
    ↪ percent. In
%         the app, the History list displays this overall accuracy
    ↪  score for
%         each model.
%
% Use the code to train the model with new data. To retrain your
% classifier, call the function from the command line with your
    ↪ original
% data or new data as the input arguments trainingData and
    ↪ responseData.
%
% For example, to retrain a classifier trained with the original
    ↪  data set T
% and response Y, enter:
%   [trainedClassifier, validationAccuracy] = trainClassifier(T,
    ↪  Y)
%
% To make predictions with the returned 'trainedClassifier' on
    ↪ new data T2,
% use
%   yfit = trainedClassifier.predictFcn(T2)
%
% T2 must be a matrix containing only the predictor columns used
    ↪  for
% training. For details, enter:
%   trainedClassifier.HowToPredict

% Auto-generated by MATLAB on 08-Sep-2020 15:24:13


% Extract predictors and response
% This code processes the data into the right shape for training
    ↪  the
% model.
% Convert input to table
inputTable = array2table(trainingData, 'VariableNames', {'
    ↪ column_1', 'column_2', 'column_10367', 'column_10368'});

predictorNames = {'column_1', 'column_2', 'column_10367', '
    ↪ column_10368'};
predictors = inputTable(:, predictorNames);
```

```
response = responseData(:);
isCategoricalPredictor = [false, false, false, false];

% Train a classifier
% This code specifies all the classifier options and trains the
    ↪ classifier.
template = templateSVM(...
    'KernelFunction', 'polynomial', ...
    'PolynomialOrder', 3, ...
    'KernelScale', 'auto', ...
    'BoxConstraint', 1, ...
    'Standardize', true);
classificationSVM = fitcecoc(...
    predictors, ...
    response, ...
    'Learners', template, ...
    'Coding', 'onevsone', ...
    'ClassNames', {'Abnormal'; 'Driving'; 'Reaching'});

% Create the result struct with predict function
predictorExtractionFcn = @(x) array2table(x, 'VariableNames',
    ↪ predictorNames);
svmPredictFcn = @(x) predict(classificationSVM, x);
trainedClassifier.predictFcn = @(x) svmPredictFcn(
    ↪ predictorExtractionFcn(x));

% Add additional fields to the result struct
trainedClassifier.ClassificationSVM = classificationSVM;
trainedClassifier.About = 'This struct is a trained model
    ↪ exported from Classification Learner R2020a.';
trainedClassifier.HowToPredict = sprintf('To make predictions on
    ↪ a new predictor column matrix, X, use: \n  yfit = c.
    ↪ predictFcn(X) \nreplacing ''c'' with the name of the
    ↪ variable that is this struct, e.g. ''trainedModel''. \n\
    ↪ nX must contain exactly 10368 columns because this model
    ↪ was trained using 10368 predictors. \nX must contain only
    ↪ predictor columns in exactly the same order and format as
    ↪ your training \ndata. Do not include the response column
    ↪ or any columns you did not import into the app. \n\nFor
    ↪ more information, see <a href="matlab:helpview(fullfile(
    ↪ docroot, ''stats'', ''stats.map''), ''
    ↪ appclassification_exportmodeltoworkspace'')">How to
    ↪ predict using an exported model</a>.');

% Extract predictors and response
% This code processes the data into the right shape for training
    ↪ the
% model.
% Convert input to table
inputTable = array2table(trainingData, 'VariableNames', {'
    ↪ column_1', 'column_2', 'column_10367', 'column_10368'});
```

```
predictorNames = {'column_1', 'column_2', 'column_10367', '
    ↪ column_10368'};
predictors = inputTable(:, predictorNames);
response = responseData(:);
isCategoricalPredictor = [false, false, false, false];

% Perform cross-validation
partitionedModel = crossval(trainedClassifier.ClassificationSVM,
    ↪  'KFold', 5);

% Compute validation predictions
[validationPredictions, validationScores] = kfoldPredict(
    ↪ partitionedModel);

% Compute validation accuracy
validationAccuracy = 1 - kfoldLoss(partitionedModel, 'LossFun',
    ↪ 'ClassifError');
```