Smartphone Road Surface Monitoring System

Anthony Scerri

Supervisor: Professor Matthew Montebello

Co-supervisor: Dr Dylan Seychell

September, 2023

Submitted in partial fulfilment of the requirements for the degree of M.Sc. in AI.

L-Università ta' Malta
Faculty of Information & Communication Technology

Acknowledgements

Thank you to my supervisors, Professor Matthew Montebello and Dr Dylan Seychell, for providing guidance and feedback throughout this project. Thanks also to my wife Elizabeth, for putting up with me being sat in the study for hours on end.

Abstract

In view of the EU's long-term vision of zero fatalities in road transport by 2050, transport authorities require accurate, up-to-date, and easy-to-use technologies to help them address road anomalies.

Smartphone devices offer a compelling solution because of their availability and various inbuilt sensors. With this in mind, this study shows how smartphone devices can help identify road anomalies. In our study, we used the smartphone's inertial measurement unit (IMU) and camera sensors to track road anomalies. This study uses two identical smartphone devices to capture the data simultaneously. The first hosts the IMU sensing application, whilst the second hosts the object detection component. The IMU road anomaly detection is based on a supervised machine learning model. Meanwhile, the vision solution employs a lightweight object detection architecture to detect anomalies in real-time. In addition, the GPS data provides the backbone to synchronise both applications and also pinpoint the anomaly locations. Finally, we use a simple data fusion technique to merge the IMU and vision results into one reporting system. This simple yet powerful reporting solution allows operators to assess the accuracy of both systems and also provides a richer reporting data set. Our study finds that a dual system yields more conclusive results than conventional mono-sensing systems, as the system provides a straightforward method for verifying both the IMU and vision predictions.

Contents

List of Figures

List of Tables

List of Abbreviations

NN	Neural Network	18
DNN	Deep Neural Network	v
SVM	Support Vector Machine	v
IMU	Inertial measurement unit	iv
GPS	Global Position System	5
CNN	Convolution Neural Network	6
DL	Deep Learning	10
IRI	International Roughness Index	12
GNSS	Global Navigation Satellite System	15
FTT	Fast Fourier Transformation	17
RF	Random Forest	v
GB	Gradient Boosting	18
DT	Decision Trees	18
SSD	Single Shot Detectors	19
LR	Logistic Regression	v
LRM	Linear Regression Model	23
LDA	Linear Discriminant Analysis	23
DT	Decision Trees	18
KNN	K-Nearest Neighbors	v
NB	Naive Bayes	v
ReLU	Rectified Linear Unit Activation	75
YOLO	You Only Look Once	v
SFS	Sequential Forward Selection	
PCA	Principal component analysis	64

Chapter 1

Introduction

Transportation is one of the main pillars of the EU economy as it contributes to economic growth between the EU's states. Notably, the road network is still the primary means of mobility within the states. Governments have invested heavily in road networks for efficient and safe travel. However, road accidents have claimed the lives of over 25,000 people, with a further 135,000 people left seriously injured in 2018 in the EU[1]. As a result, member states must make serious inroads in the next decade to reach the EU's long-term strategic goal of 'Vision Zero'[2].

With volatile weather and heavy usage, road surfaces develop weak spots leading to asphalt cracks, depressions and potholes. Degradation of the road surface can affect the safety and roadworthiness condition of vehicles. In particular, potholes, cracks, and asphalt deformation can influence the safety of motorcycles, according to Bella et al. (2012).

Pothole repair and road maintenance are also costly for road agencies to repair. A study conducted by RAC[3] in the UK shows that one-third of drivers experience pothole damage to their vehicles. For instance, the average repair bill for pothole damage is £142. This is costing the UK's authorities £6 million annually in compensation. In addition, a study published by Local Authority and Road Maintenance UK[4], reveals that the UK government spent £118 million in 2015 to fix urgent potholes.

In another RAC[5] study, one-fifth of motorists claim to have sustained damage from

[1]European Commission (4 April 2019), Publication of preliminary road safety statistics 2018: https://ec.europa.eu/commission/presscorner/detail/en/IP_19_1951 (Accessed 02 September 2021)

[2]REPORT on the EU Road Safety Policy Framework 2021-2030 – Recommendations on next steps towards 'Vision Zero' (22 June 2021), https://www.europarl.europa.eu/doceo/document/A-9-2021-0211_EN.html (Accessed 02 September 2021)

[3]Pothole damage - https://www.rac.co.uk/drive/news/motoring-news/one-third-of-uk-drivers-suffer-pothole-damage-to-their-cars/ (Accessed 05 September 2021)

[4]ALARM - https://www.asphaltuk.org/wp-content/uploads/ALARM_survey_2016.pdf (Accessed 02 September 2021)

[5]Speed bump study - https://www.rac.co.uk/drive/news/motoring-news/drivers-report-speed-bump-

speed bumps. There were over 29,000 speed bumps in the UK and, according to the study, driving over speed bumps without slowing down causes underside and suspension damage. The average repair cost for speed bump damage is £141 per claim.

Overall, road maintenance is a reactive process, which causes budget shortfalls for the authorities. Road agencies primarily rely on road safety audits and onsite inspections to assess the road's condition. These are labour-intensive processes, as they depend on road safety inspectors to report road damage. In contrast, some road agencies proposed different solutions to tackle road anomalies. These include online portals to encourage commuters to report road defects. Dedicated personnel then investigate these images and update the records accordingly. However, the effectiveness of these systems depends on the participation of road users, which might not be active in rural and arterial road networks.

Naturally, road networks will keep expanding, and the number of road users will keep increasing. As a result, it will put more pressure on the authorities and urban planners to deliver safe transportation. These adversities are paving the way for more autonomous reporting systems. The EU directive 2010/40/EU[6] introduces a framework for the deployment of Intelligent Transport Systems (ITS) on road transport. The directive laid the groundwork for artificial intelligence to be included in the transportation framework.

ITS is an area of study that aims to promote innovative services in the transport sector. Its primary aim is to provide commuters with better access to information to make the journey safer, quicker, and smarter. The ITS initiative covers the entire transportation system. However, directive 2010/40/EU puts ITS at the core of its road transport vision. Coupled with information and communication systems, ITS can improve traffic management, mobility, and road safety.

Predominantly, smartphones are one technology segment that is thriving under this ITS initiative. Smartphones have become ubiquitous in modern society. In addition, they can collect high-quality data, such as imagery, vibration, and GPS data. Engelbrecht et al. (2015) conducted a survey to analyse the usage of smartphones in ITS. The study evaluated existing smartphone-based solutions to monitor drivers' behaviours, road conditions, and vehicle telematics. The survey demonstrates that smartphone-based ITS solutions can achieve respectable results. However, there are still several challenges facing the future progress of such systems. One of the key issues is wide-scale deployments, as these systems require a large user base and offer few incentives. Therefore, the adoption of smartphone crowd sensing will be slow.

damage/ (Accessed 10 September 2021)

[6]https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32010L0040&from=EN (accessed 01 February 2022)

Other studies, however, have shown that smartphones can inspect and identify road anomalies with high precision. Mednis et al. (2011) presented a novel threshold-based method for detecting potholes by smartphone built-in three axes accelerometer sensor readings.

Tai et al. (2010) and Mohamed et al. (2015) extended the threshold-based solutions by engaging machine learning solutions in their studies. Moreover, Silva et al. (2017) added additional smartphone sensors to their research to improve their accuracy.

In contrast, one of the major challenges of using a single technique for detecting road defects is that road anomalies have different structural characteristics. Therefore, it is difficult to capture and categorise the anomalies based on one probing method. Vehicle speed, pothole avoidance, and the vehicle's suspension systems can affect the detection process. As a result, stand-alone techniques can return non-precise classifications because of their inability to evaluate other types of ancillary sensors. Hence, this research will incorporate two machine learning techniques: vibration-based models, capitalising on inbuilt mobile accelerometers and gyroscopes sensors, and vision-based models using smartphone cameras. The primary goal is to fuse the vibration and vision data sets to enrich and augment the reporting dashboard. This involves the challenging task of collecting both the visual and vibration readings synchronously. The two data sets can work in sequence by providing validation for each other and providing a firm foundation for drawing a conclusion.

We are now witnessing a shift in the trajectory of smartphones in the ITS field, thanks to the vision machine learning solutions for smartphone devices. Innovative, vision machine learning models can now run directly on a smartphone device, smoothing the path for more powerful machine learning techniques for detecting road anomalies.

Over the last decade, computer vision has emerged as a critical technology for a wide range of applications, including road anomaly detection. Dib et al. (2020) presented a thorough review of different computer vision pothole detection techniques. Their primary aim was to assess the strengths and limitations of the techniques under different environmental conditions. Likewise, Chitale et al. (2020) confirmed that pothole detection using vision deep learning is also possible. However, one of the crucial difficulties of vision techniques is the lack of effective training data sets to train the models. Specifically, supervised machine learning techniques require large data sets of annotated data. The major drawback of annotated data is that this process is often biased, meaning that the classes can be over or under-represented.

1.1 Motivation

The primary motivation of this research is to gather an in-depth understanding of machine-learning techniques for automating road anomaly detection. We investigated various social real-world problems related to road anomalies in the early stages of the study. Road safety is a major headache for Maltese authorities, drivers, and insurance companies alike. For example, there are no available road anomaly data sets to help the authorities prioritise road maintenance. Likewise, local insurance companies cannot distinguish between claims resulting from negligent driving and those resulting from road anomalies. As a result, the lack of detailed data sets endangers the safety of both motorists and commuters.

The goal of this study is to create a simple but effective platform for detecting road anomalies. Similarly, by comparing historical data sets, we can evaluate road conditions chronologically. Ultimately, road authorities can plan maintenance ahead of time. Our initial investigation focused on both traditional and modern monitoring techniques, such as industrial-level sensors and smartphone sensing. It became clear that road monitoring is a fascinating research topic, and researchers have been working on pothole detection techniques for the past two decades. As we will see in this study, smartphones and machine learning techniques can have a significant impact on road monitoring techniques.

1.1.1 Research Questions

This dissertation aims to address the following research question:

Can an efficient and cost-effective smartphone-based system be developed to detect road anomalies, particularly potholes, by integrating inexpensive hardware, IMU, imagery, and GPS data, while ensuring accurate and meaningful anomaly detection?

1.2 Aims and Objectives

The primary objective of this study is to combine different concurrent machine-learning techniques to increase the accuracy of the monitoring operation. We strongly believe that a single approach or algorithm will inevitably have limitations. In this study, we explore how the two methods can handle different aspects of anomaly detection. Our aim is to demonstrate how both systems can complement each other, each highlighting different, but important, aspects of the detection process.

One can always adjust the algorithm's hyperparameters to increase accuracy, however, the literature review shows that every monitoring technique has its limitations. Furthermore, relying on hyperparameter tuning of a single method to increase the accuracy may lead to over-fitting of particular scenarios.

Comparatively, vision systems have demonstrated promising results in identifying road abnormalities and can identify anomalies across a broader road surface than the IMU processes. However, the camera's position, lighting conditions, shadows, and training data can severely impact the accuracy.

As a result, selecting the best hyperparameters and tuning for a single pothole detection method is beyond this research. Our primary goal, on the other hand, is to investigate how two distinct approaches can coexist and work in tandem to improve accuracy. Therefore, our aim is to investigate modern methods to monitor road defects using a combination of smartphones and machine learning techniques. Similarly, we will also examine how we can fuse multiple data sources to validate the classification process.

To achieve the above aims, the following objectives for the proposal are being put forward:

1. The creation of a mobile application to capture road vibration using the accelerometer, gyroscope, and Global Position System (GPS) sensors, including annotation, identification, and classification of road surfacing defects.

2. The creation of a mobile application to capture live images and the classification and recording of different road anomalies, including the GPS position.

3. The comparison of the above two techniques to improve the accuracy of the techniques and the augmentation of the data sets to create a cross-model unified reporting classification data set.

1.2.1 Research Scope

We limit this study to the development of vibration, angular velocity, and vision sensing to address road anomaly issues. To collect the data, we use smartphone devices. This study will collect four types of data: vibration, rotation, GPS, and imagery. We divided the imagery process into two steps. First, we feed the first data stream to a CNN model running on a standard Android smartphone device. Second, we keep the image stream on the smartphone device for the fusion verification and future work. Importantly, we are anonymising the image content using a separate process to conceal faces and vehicle registration numbers.

This study is entirely based on commercially available smartphone devices. With this in mind, we conducted extensive research on the hardware of the smartphone. In addition, we investigated the software interfaces available for accessing the smartphone's sensors. It is important to note that access to the sensors is only available through the device's operating system. As a result, it is important to note that this study is based on pre-processed data.

1.2.1.1 Research Contribution

This research has shown that we can fuse embedded smartphone sensors and imagery techniques to enhance the detection and classification of road anomalies. We delineate the contribution made by this study below:

1. The development of machine learning models to detect road anomalies using IMU sensor data
2. The development of an efficient, smartphone-enabled, vision CNN to detect and classify road anomalies
3. The development of a fusion reporting system combining the accelerometer and gyroscope analysis with the vision-based object detection system
4. The release of the IMU and vision code, including the collected data, as open source software

1.3 Proposed Solution

This study investigates how inexpensive smartphones can detect road hazards. We define road anomalies in this dissertation as road surface depressions and speed control systems, such as potholes and speed bumps. This research proposes a simple and low-cost architecture for identifying and tracking anomalies. In essence, we achieve our goal by fusing together vibration, rotational, imagery processing, and GPS data. We build our solution from the ground up using two native mobile apps. The first application records the vibrations and rotational data of the vehicle. We then transfer this data to a server for processing and analysis. Meanwhile, the second mobile application uses the smartphone's hardware to detect and classify road anomalies using a Convolution Neural Network (CNN) model. Finally, we send the classification and GPS location to the server to be fused with the vibration data. Our proposed solution provides a constantly updated model with up-to-date data feeds to assist road authorities in monitoring the road network.

1.3.1 Research Approach

Figure 1.1 outlines the overall method of the research. The initial research maps out the machine learning methods to detect road anomalies, followed by the methods we will deploy. This involves extensive research and literature reviews on vibration sensors and vision models.

The vibration data analysis part follows this. Strong emphasis is being made on the accelerometers and gyroscope sensors. As a result, we will explore the smartphone development toolkit to develop the applications. Later, we will also investigate the vision models to enhance the road anomaly detection process. For this purpose, we will compare several vision models.

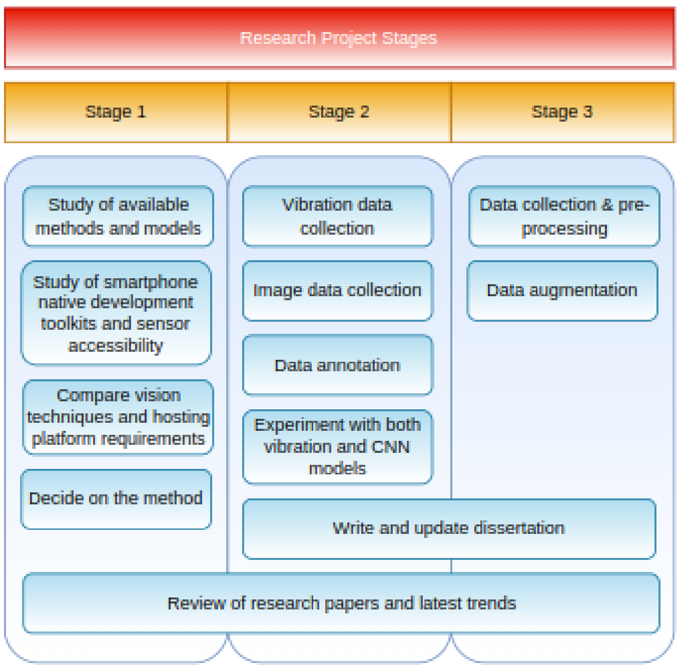

Research Project Stages
Stage 1
Stage 2
Stage 3
Study of available methods and models
Study of smartphone native development toolkits and sensor accessibility
Compare vision techniques and hosting platform requirements
Decide on the method
Vibration data collection
Image data collection
Data annotation
Experiment with both vibration and CNN models
Data collection & pre-processing
Data augmentation
Write and update dissertation
Review of research papers and latest trends

Figure 1.1: Schematic diagram of the research stages

In the second stage of the process, we will collect data from the embedded smartphone sensors using the custom-built mobile application. The subsequent phase includes the analysis of the different data processing techniques. This consists of the familiarisation with time series data sets, data cleansing, re-sampling and feature engineering methods. One of the biggest challenges during the second stage of the research will be the annotation of the vibration data. Consequently, we will use a dual smartphone setup to achieve this task. The first device will capture the IMU data and the second device will annotate the anomalies in real-time. Furthermore, we will carry out several experiments to find the optimal sampling rate based on the smartphone's capabilities.

Similarly, work on the vision part includes a considerable number of tasks. For instance, we conduct thorough research to identify a lightweight CNN model to be hosted on the smartphone device. Also, we develop a prototype to evaluate the model's capabilities and hardware performance.

In the third and final stage of the study, we develop the fusion model based on the data feeds from the vibration machine learning techniques, the visual model's classification, and geo-location data.

1.4 Document Structure

We organised the rest of the dissertation as follows.

In chapter 2, we present the background and literature review on the state-of-the-art techniques for detecting road anomalies. In the first section, we inspect smartphone devices and review how these were employed in previous research to detect road anomalies. We also discuss the main smartphone sensors required to measure road anomalies. We then move to the non-machine learning road anomaly detection techniques. Even though these technologies are outdated, the data-gathering techniques are still relevant in today's models and form the foundation of today's IMU machine-learning techniques. Next, we cover the IMU and vision machine learning techniques and finally, we cover the state-of-the-art road-anomaly detection models.

In chapter 3, we discuss the materials and methods used for our research. We describe all the steps and procedures we carry out to achieve the research objectives, including the design and data analysis. First, we give a general overview of the proposed solution and deliverables. We then delve into the three main components consisting of the IMU, vision and back-end modules. Next, we introduce the synchronisation techniques used in our experiments, including the GPS data and filtering techniques, to improve the geolocation readings. The capturing, processing, and data annotation processes to prepare the data sets for the machine learning process follow this. Afterwards, we explain how we address the vision components' shortfalls. Finally, we conclude by presenting the fusion solution to reinforce both models and improve the results.

Chapter 4 covers the test results. First, we start with a summary of the IMU data set and the machine learning models used in our testing to detect road anomalies. We then move to the feature extraction methods used to build our models and the high-pass filters utilised to remove unwanted vehicle vibrations. Next, we cover the machine learning tests and results obtained, including the classification outcome. The vision experiments follow this and we conclude this chapter by reviewing the data fusion results.

Chapter 5 discusses the evaluation process and suggests how designing machine learning models with evaluation in mind can lead to successful and optimised solutions. Evaluation is a highly structured process and relates to machine learning techniques.

Chapter 6 presents conclusions and future work. This chapter discusses the main contribution of the dissertation and its limitations.

Chapter 2

Background & Literature Overview

2.1 Introduction

The unparalleled success of machine learning, Deep Learning (DL), computer vision and big data analytics in several demanding application domains, such as medicine and fault detection, inspires this research. This dissertation focuses on road anomaly detection and reviews the relevant state-of-the-art approaches and techniques. This chapter investigates how we can combine various machine-learning approaches to develop a hybrid machine-learning approach to overcome the weaknesses of individual methods. Furthermore, we also review the literature related to machine learning approaches in relation to IMU sensory and imagery data.

Machine learning is a subset of artificial intelligence and has become a key technique for solving complex data analysis problems. This technique has become an effective data analytical tool since we can now create complex models and algorithms to analyse huge amounts of data. Detection and forecasting are based on prior learning and tendencies. Today, we find machine learning technologies in the products we use in our daily life, such as natural language processing, image recognition, security, medical diagnosis, and many more.

We divide machine learning into three categories: clustering, classification, and regression. We normally associate clustering with unsupervised learning, where test data sets only have inputs and no corresponding labels. As a result, we do not provide labelled data to the model during training in unsupervised learning. The model attempts to discover the hidden pattern by analysing the statistical properties of the data set and categorising it.

Classification and regression work with training data sets that have corresponding labels. Therefore, when an input is given, the machine can correctly predict the corresponding label. Consequently, in supervised machine learning, we provide labelled data set. To put it differently, we included the desired results in the learning process. As a result, supervised learning is more efficient and accurate. A high-level overview of machine learning methods is shown in Figure 2.1

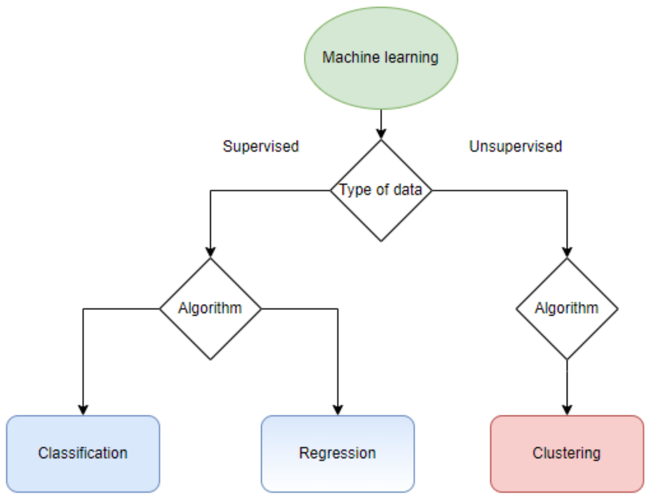


Figure 2.1: Machine learning methods

Depending on the type of data, we use different approaches to model our solutions. Machine learning identifies anomalies or outliers in a data set by locating unusual data points that differ significantly from the overall trends in the data set. In this dissertation, we detect anomalies through the use of supervised IMU and vision methods. However, clustering is also used to help us visualise the collected data.

We organised the rest of the chapter as follows. In section 2.2, we give a brief description of anomaly detection in relation to road surfaces. Section 2.3 discusses smartphone sensing in ITS, including smartphone-based solutions used to detect road anomalies. In section 2.4, we inspect various machine learning solution. Here, we discuss the IMU and vision-based machine learning algorithms to detect road anomalies. In this section, we also discuss the IMU and vision methods, including the classification process.

2.2 Road Anomalies

Anomaly detection is a pervasive challenge encountered across diverse domains, including manufacturing, engineering, transportation, and healthcare. This technique plays a crucial role in identifying rare events or irregular patterns by analyzing data points that deviate significantly from the norm.

The process of anomaly detection primarily revolves around:

- Observing Data Points: Anomaly detection involves carefully observing and gathering data points from the target domain. These data points typically represent the regular behavior or expected patterns within the data set.
- Establishing Baseline: By analyzing the observed data, a baseline or model of normal behavior is created. This baseline serves as a reference point for distinguishing normal from anomalous instances.
- Detecting Deviations: The collected data is then compared against the established baseline. Any data point that deviates significantly from the norm is flagged as a potential anomaly.

Anomaly detection techniques utilize various approaches, including statistical methods, machine learning algorithms, and pattern recognition, to efficiently identify and flag outliers or anomalies.

By harnessing the power of anomaly detection, industries gain the ability to proactively identify irregularities, defects, or potential failures. This empowers them to take timely corrective actions and elevate overall operational efficiency, safety, and quality within their respective domains. Moreover, our road anomaly detection approach is built upon a comprehensive EU[1] study, which meticulously categorizes three distinct types of road anomalies:

- Ruts - linear depressions created by a permanent deformation of the layers of bitumen caused by the load from a vehicle's wheels
- Potholes - depressions brought about by the removal of the superficial layer of tarmac caused by traffic
- Depressions - permanent deformations in the road surface due to ground subsidence

This anomaly classification is based on the International Roughness Index (IRI) which is used widely in civil engineering to classify each group of anomalies. Similarly, vertical

[1]EU Road surfaces
https://www.europarl.europa.eu/RegData/etudes/STUD/2014/529059/IPOL_STU(2014)529059_EN.pdf (accessed 01 October 2021)

traffic calming devices, such as speed bumps and other traffic calming devices, are being classified as road anomalies in our work.

2.3 Smartphone sensing

Advances in computer chip design and manufacturing techniques are constantly improving the performance and energy efficiency of these mobile sensing devices. As a result, smartphone manufacturers are constantly increasing the number and accuracy of onboard sensors. Therefore, the potential of using smartphone devices to build innovative applications in the areas of environmental monitoring, healthcare and ITS is increasing, as suggested by Abualsaud et al. (2019). According to the authors, the accuracy of smartphone IMU sensing components provides an efficient system for crowd sensing applications.

Smartphone-based monitoring applications are therefore of particular interest in ITS studies. One major advantage of using smartphone devices as opposed to using factory-fitted vehicle sensors is that these sensors are easier and cheaper to deploy. Another major advantage is the ease of interfacing. Unlike onboard patented technologies, onboard smartphone sensors are simpler to access and require no proprietary software protocols.

2.3.1 Smartphone sensing accuracy

However, smartphone sensing has its own set of challenges:

- Sensor quality and sampling rates – Built-in sensors are typically less accurate and reliable than industry-rated sensors. The two most marketed features of smartphones are processing power and camera quality. Because of the competitive market, manufacturers use less accurate components as opposed to industry-grade components for accelerometers and gyroscopes sensors to save on cost.

- User interaction and positioning - Users interact in a variety of ways with their smartphones. Given this, the device's orientation and movement have a significant impact on the sensing capabilities and data-capturing accuracy.

- Sensor noise - Smartphone sensors are extremely susceptible to sensor noise. Signal interference, hardware design, and atmospheric conditions affect GPS geolocation accuracy. Furthermore, the user's interaction with the device influences the sensor's reading. In a movement-sensing environment, the accelerometer sensors capture the vibration caused by the user's interaction with the smartphone.

- Battery drain - Heavy CPU usage and high-frequency sensor data collection can cause rapid battery discharge.

Moreover, according to Stisen et al. (2015), the accuracy of smartphone-embedded sensors varies between brands and models. Their research used various smartphone models to compare sensor readings in a human activity recognition system. The study shows that there are data disparities when using different smartphone models, which can impair sensor-based data capture activities.

In contrast, while embedded smartphone sensors are not as accurate as industrial-grade monitoring sensors, research has shown that they can still produce adequate readings for scientific tasks.

2.3.2 Accelerometer Sensor

An accelerometer is a device for measuring and analysing linear and angular acceleration. Accelerometers measure acceleration force in Gs and acceleration in multiple directions. Moreover, accelerometers typically measure acceleration in three directions: X, Y, and Z. When the accelerometer device is stable, a reading of $9.81\,\mathrm{m/s^2}$ is obtained on the axis pointing to gravity.

2.3.3 Gyroscope Sensor

Gyroscope sensors are electronic devices that measure an object's orientation and angular velocity. They can detect the tilt and lateral orientation of the device. We measure angular velocity in degrees per second, which is based on changes in an object's rotational angle per unit of time. The primary distinction between a gyroscope and an accelerometer is that gyroscopes measure angular velocity (rad/sec), whereas accelerometers measure specific gravity force.

Gyroscopes have three angular axes based on the following direction:

- Yaw - Rotation around the vertical axis
- Pitch - Rotation around the side-to-side axis
- Roll - Rotation around the front-to-back axis

In our setup, the Z (roll) gyroscope detects vehicle acceleration and deceleration. The X (pitch) gyroscope detects the vehicle's left and right movement, and the Y (yaw) gyroscope detects the vehicle's up and down movement. Figure 2.2 shows the smartphone's accelerometer and gyroscope sensors and device orientation.

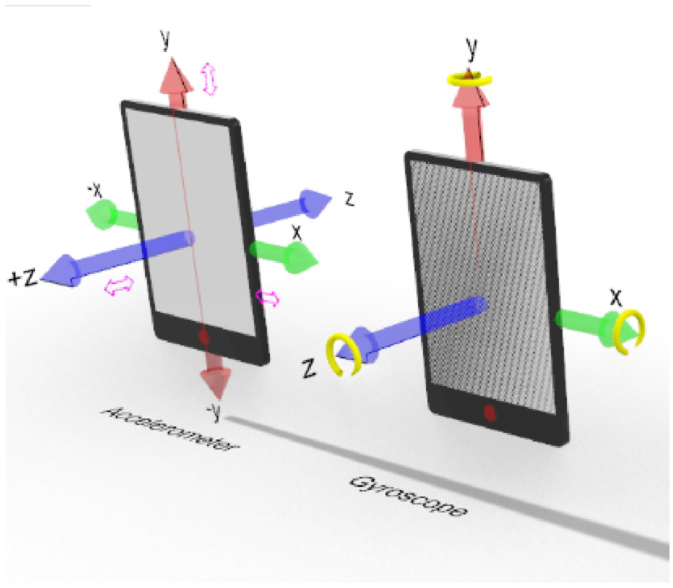


Figure 2.2: Smartphone Accelerometer and Gyroscope Sensors for Precise Orientation

2.3.4 GPS accuracy

Merry and Bettinger (2019) conducted a study on the horizontal position accuracy of smartphone devices in urban environments. The research compared an iPhone 6 to commercially available mapping devices. According to the results, the average position error of an iPhone 6 is between 7 and 13 metres depending on the environment, which is consistent with the general accuracy of recreational GPS receivers.

GPS positioning is determined by listening to radio signals transmitted by GPS satellite systems. Because traditional smartphones use a single frequency Global Navigation Satellite System (GNSS), smartphones receive only a single radio wave frequency from the satellites. The authors of Angrisano and Gaglione (2022) demonstrate how gross errors in measurements, usually related to multi-path and non-line-of-sight phenomena, have a strong influence on GNSS performance in urban locations. As a result, a single frequency is vulnerable to multi-path errors because the signal can bounce off other objects, such as high-rise structures, and generate echo signals. This GPS multi-path distortion can cause GPS reading inaccuracies of up to 5 metres. Even though these errors have no effect on the machine learning models' accuracy, it is important to note that they influence the road anomaly's geolocation data.

2.3.5 Inertial measurement unit (IMU) threshold based solutions

To detect potholes, expansion joints, railroad crossings, and joints, the first generation of IMU ITS solutions relied on simple threshold-based algorithms. Eriksson et al. (2008) used an on-board computer to record vehicle vibration using the X and Z axes and GPS positioning. The authors installed the accelerometers inside the vehicle's cabin and used seven vehicles to cover over 2,400 kilometres of road. A high-pass filter was used to filter door slams, curb ramps turning and breaking. Furthermore, they filtered the data with a Z-peak algorithm to detect road anomalies. Moreover, they developed a XZ-ratio algorithm to identify various road anomalies. When the authors hit a pothole, the Z-peak acceleration produces a noticeable peak on the X-axis. In another observation, the authors discovered a relationship between the vehicle's speed and the accelerometer Z ratio, allowing them to detect small potholes while driving at high speeds. The system correctly identified 90% of the road anomalies that needed to be repaired.

Mohan et al. (2008) were the first to use smartphones in road anomaly detection to identify potholes and bumps. They used the Z accelerometer axis thresholds in their research. Meanwhile, they ran the tests in a strict power management mode to reduce the smartphone's battery usage. Unfortunately, the authors did not explain the classification process for potholes and bumps, instead they focused their paper on the anomalies detection process.

A significant contribution by Mednis et al. (2011) demonstrates how threshold-based algorithms can detect various pothole sizes, gaps, and drains. They compared three different algorithms based on the vertical accelerometer axis in their work: Z-THRESH, Z-DIFF, and STDEV (Z). Their main contribution was the creation of a new algorithm called G-ZERO, which included all three accelerometer axes. They carried out the experiments using a portable computer and a three-axis analogue accelerometer. The sampling rate, which was set to 100 Hz, is also critical. For large potholes, the G-ZERO algorithm can detect road anomalies with 100% accuracy. However, the study concluded that the algorithms missed 7% of the pothole clusters.

A study conducted by Fazeen et al. (2012), emphasised the significance of smartphone placement and orientation when used to detect road anomalies. The experiment uses the accelerometer's X and Z axes with a sampling rate of 25 Hz. The X and Y axes, according to the literature, can measure steering, acceleration, and braking. However, the authors did not provide any additional technical information.

We can find another significant contribution in the work presented by Orhan and Eren (2013). By combining accelerometer analysis with video streaming, the paper presented a unique multi-modal analysis solution. Rather than analysing the entire video recording for anomalies, the accelerometer readings pinpointed the critical recording timelines that are likely to contain a road hazard. Notably, the experiment produced excellent results, with a recall value of 0.82 and a precision value of 0.93.

Furthermore, Douangphachanh and Oneyama (2014) proposed a four-smartphone solution, with two smartphones attached to the vehicle's dashboard, a third device in the driver's shirt pocket, and a fourth device in the car's console. They sampled the three accelerometer axes at a rate of 100 Hz. The authors extracted data from the GPS signal to determine the vehicle's speed. To avoid GPS inaccuracies, they sampled the road condition in 100-meter increments. The authors removed unrelated low-frequency signals, such as breaking and turning, using a high-pass filter. They mostly carried the experiment out in the frequency domain. A summary of the threshold-based characteristics is presented in Table 2.1

Table 2.1: Summary of Previous Research on Threshold-Based Accelerometer Road Anomaly Detection

Paper	**Sensors**	**Axes**	**Freq (Hz)**	**Features**
Eriksson et al. (2008)	Accelerometer	X and Z	380	Accelerometer
Mohan et al, (2008)	Accelerometer	Z	310	Accelerometer
Mednis et al. (2011)	Accelerometer	X, Y and Z	26, 52, 74, 98	Accelerometer
Fazeen et al. (2012)	Accelerometer	X and Z	25	Accelerometer
Orhan & Eren (2013)	Accelerometer	Z	40	Accelerometer
Douangphachanh & Oneyama (2014)	Accelerometer	X, Y and Z	100	Accelerometer

2.3.6 IMU Machine Learning solutions

Perttunen et al. (2011) suggested an Support Vector Machine (SVM) model for detecting potholes and bumps. Instead of using the time domain, the authors experimented with features based on Fast Fourier Transformation (FTT). Bumps and potholes, according to the authors, produce lower frequency components than other road surface anomalies.

The work conducted by Wu et al. (2020) demonstrates that smartphones can monitor road surface conditions without the use of specialised equipment or external hardware setups. Features, data processing, data acquisition, and classification were the main components of the research. Their system used Logistic Regression (LR), SVM, and Random Forest (RF) for classification. They set the sampling rate to 50 Hz. However, because the sampling frequency was inconsistent, they re-sampled the sensor's data using data interpolation. Equally important, they maintained a constant speed of around 30 km/h. Moreover, they implemented a two Hz cut-off Butterworth high-pass filter to remove noise. With the highest F1-score, random forest had the best pothole-detection score. SVM had the highest classification precision for potholes, but its recall rate was the lowest.

As shown in Basavaraju et al. (2020), smartphone devices can predict road conditions using accelerometer and gyroscope sensors. In their experiments, the authors used three different techniques: SVM, Decision Trees (DT), and Neural Network (NN). The authors settled on a sampling rate of 100 Hz. They classified the data into three categories: smooth surface, potholes, and deep traverse cracks. The results show that using smartphones and machine learning techniques to monitor road networks is a viable and cost-effective solution.

Silva et al. (2017) conducted an interesting study on the identification of sewer holes, long and short bumps, and other anomaly categories. The data set included the accelerometer, GPS, and timestamp records. From an algorithm point of view, the authors used NN, tree-based models, RF, Gradient Boosting (GB), DT, and SVM, among other machine learning algorithms. It is also worth noting that the authors were aware of the problems associated with over-fitting. As a result, they used the WEKA tool to outline and reduce the number of features from 39 to 10. Despite this reduction, they obtained an accuracy score of 0.88. Similarly, Fox et al. (2015) used an SVM model and a list of 120 candidate features. The feature characteristics were based on observing the accelerometer signal data properties, such as vertical and lateral accelerations, as well as their ratios and products with vehicle velocity.

Likewise, Brisimi et al. (2016) proposed a system based on smartphones and machine learning algorithms to classify road bumps. With an accuracy of 88%, the system classifies speed bumps and identifies those that require immediate attention. Similarly, Bhoraskar et al. (2012) proposed a machine learning application for detecting traffic conditions and road bumps. The solution uses SVM and K-means clustering algorithms. According to their findings, there is a strong correlation between vehicle make, mobile device, and road condition. As a result, the study concludes that fixed threshold systems are less versatile than machine learning solutions.

Carlos et al. (2018), on the other hand, assessed a set of 30 virtual roads generated by an application called Pothole Lab. The resulting data set had several road flaws, such as potholes and bumps. To identify the anomalies, they then used an SVM classifier model based on various accelerometer-based engineered features. The sensitivity ranged between 0.575 and 1 for the 30 experiments. Furthermore, Cabral et al. (2018) used high-dimensional features and innovative machine learning techniques in a similar study to improve the system's robustness and make it compatible with different vehicle models and smartphone devices. We presented a summary of the machine learning characteristics in Table 2.2

Table 2.2: Summary of Previous Research on Machine Learning-Based Models Road Anomaly Detection

Paper	Sensors	Axes	Freq. Hz	Features	Algorithm
Perttunen et al. (2011)	Accelerometer	X, Y, Z	38	95 - Time and freq domains	SVM
Wu et al. (2020)	Accelerometer	X, Y, Z	50	144 - Time, freq and wavelet domains	LR, SVM and RF
Bhoraskar et al. (2012)	Accelerometer	X, Y, Z	50	Mean, stdev	SVM
Tecimer et al. (2015)	Accelerometer Gyroscope Magnetometer	X, Y, Z	50	Freq domain	SVM, KNN, NB, LMT, MLP
Brisimi et al. (2016)	Accelerometer	X, Y, Z	50	Stats, Freq domain	SVM, RF, LR, Adaboost
Bhoraskar et al. (2012)	Accelerometer	X, Y, Z	50	Time domain	SVM, K-means
Silva et al. (2017)	Accelerometer	X, Y, Z	50	Stats, integral	SVM, RF, GB, NN, DT
Fox et al. (2016)	Accelerometer	X, Y, Z	100	120 - Stats	SVM
Gonzalez et al. (2017)	Accelerometer	Z	50	Bag of words	SVM, ANN, KNN, RF, DT, KR
Carlos et al. (2018)	Accelerometer	Z	50	Mean, stdev, range, threshold	SVM, Ensemble
Cabral et al. (2018)	Accelerometer Gyroscope	X, Y, Z	100	Mean, stdev, range, threshold	SVM, HMM, ResNet, KNN

2.3.7 Vision

Rao et al. (2020) and Baek and Chung (2020) used a video camera to capture the road surface condition. An onboard computer routine inspects the captured images. Consequently, when it detects road surface anomalies, the system sends a notification. These solutions, however, require the use of a dedicated video camera mounted on vehicles.

Maeda et al. (2018) conducted a comprehensive study on road damage detection and classification, comparing two Single Shot Detectors (SSD) using Inception V2 and MobileNet. Inception SSD proved to be twice as slow as MobileNet. The recall patterns were similar, and both scored the same precision results.

Silvister et al. (2019) proposed an enthralling dual mechanism pothole detection system in which the authors combined an SSD and a sensory Neural Network model to improve detection. To train the SSD model, the authors created a custom database of pothole images. The reported accuracy of the Deep Neural Network (DNN) is 96.7%, while an SSD model on the same data set produced an accuracy of 92.9%. In contrast, they reported a major problem with sensory data annotation due to time synchronisation issues.

Tithi et al. (2021). carried out a similar study. In their experiment, they used an SSD TensorFlow Lite model to detect speed bumps and potholes. They collected 2500 sample images of potholes and speed bumps in their experiment. The model detected objects up to 50 feet away with an accuracy varying between 67% and 92%.

In a recent study, Darapaneni et al. (2021) tested the YOLO family (v3,v4 and v5), SSD and Unet models against a Kaggle data set repository containing 666 annotated images. The YOLO family performed well, unlike the Unet and SSD models. The authors recommended improving the SSD algorithm and adding image pre-processing techniques for future work. Camilleri and Gatt (2020) conducted a similar study. The authors compared YOLOv3, YOLOv3 tiny, YOLOv3 SPP, MobileNet V2 and Lite MobileNet V2. The reported figures suggest the YOLO performed better results.

2.3.8 Evaluation benchmark

In Table 2.3 and Table 2.4, we highlight the top-performing state-of-the-art models based on the above IMU and vision research. We will review these standards in the section titled section 5.2 where we compare our results with these benchmarks.

It is worth noting that Brisimi et al. (2016) and Silva et al. (2017), which included speed bumps, are not included in Table 2.3. Brisimi et al. (2016) reported the results in an Area Under the Curve (AUC) metric, with RF receiving a score of 0.697. Meanwhile, Silva et al. (2017)'s result metrics included a score for general overview and percentage of correct classification. To keep the metrics consistent, we excluded these results from the classification table. However, we will include Silva et al. (2017)'s work in our final evaluation to compare our speed bump models.

Table 2.3: Assessment Metrics Employed in Our Research: Evaluation Benchmarks for Performance Analysis - IMU

Source	Algorithm	Acc	Crack		Pothole / Depression		Speed bump		Smooth	
			Precision	Recall	Precision	Recall	Precision	Recall	Precision	Recall
Basavaraju et al. (2020)	SVM	0.886	0.403	0.438	0.722	0.678			0.944	0.947
	Decision tree	0.883	0.435	0.412	0.666	0.671			0.950	0.947
	NN	0.921	0.559	0.611	0.769	0.781			0.969	0.963
Wu et al. (2020)	SVM	0.948			0.908	0.642			0.952	0.992
	LR	0.952			0.851	0.734			0.965	0.984
	RF	0.957			0.885	0.750			0.965	0.988
Silvister et al. (2019)	DNN	0.967			n/a	n/a			n/a	n/a
	SVM	0.929			n/a	n/a			n/a	n/a

Table 2.4: Assessment Metrics Employed in Our Research: Evaluation Benchmarks for Performance Analysis - Vision

Source	Algorithm	Prediction accuracy		Pothole / Depression		Speed bump	
		Pothole	Speed bump	mAP	Recall	mAP	Recall
Camilleri and Gatt (2020)	Yolov3 SPP			0.688			
Tithi et al. (2021)	SSDMobileNetV2	0.670 - 0.920	0.790 - 0.930				
Rani et al. (2020)	SSDMobileNetV2			0.600		0.700	

2.4 Machine learning techniques

2.4.1 IMU Machine learning techniques

Classification is a machine learning technique that categorises data into distinct classes. The goal of a data classification process is to identify and correctly classify new data.

The following is a list of terminologies used in machine learning classification:

- Classifier: An algorithm used to map the input data and classify it
- Classification model: The evaluation technique used to predict the category of new data using the results of a training routine.
- Feature: It is an individual observation property of a phenomenon being investigated
- Binary Classification: In this classification, the outcome has two potential values: 0 or 1
- Multi-class classification: It can classify data into more than two classes. To clarify, this type of classification can only assign one type of class per output.
- Multi-label classification: This is similar to Multi-class classification. However, multiple labels can be assigned to one observation.

2.4.2 Overview of Bias and Variance

Supervised machine learning algorithms construct their models using the training data provided during the model's construction cycle. The primary goal is to best estimate the mapping function (f) from the output label (Y) given an input variable (X).

We can break the prediction error down into two parts:

- Bias error
- Variance error

2.4.2.1 Bias Error

Bias represents the model's simplifying processes used to simplify the target function learning mode.

Linear algorithms have a high bias, which makes them easy to learn and understand but limits their flexibility. As a result, they have poor predictive performance on complicated structures that do not meet the algorithm's bias's simplified assumptions.

- Low Bias: Indicates that the target function has fewer assumptions. Mainly K-Nearest Neighbors (KNN), SVM, and DT
- High Bias: Indicates more assumptions of the target function: Mainly LR, Linear Regression Model (LRM) and Linear Discriminant Analysis (LDA)

2.4.2.2 Variance Error

Variance is the amount by which our estimate of f(X) would change if we estimated it using a different training data set. For instance, the target function estimate should not change much if we use different training data sets. This means that the algorithm has strong underlying logic for identifying the relationship between the input and output variables.

- Low Variance: Shows lower changes to the target function's estimate when we change the training data. (LR, LRM and LDA).
- High Variance: Shows higher changes to the target function's estimate when we change the training data. (DT, KNN and SVM).

2.4.3 Bias-Variance Trade-Off

Ultimately, the main goal of any supervised machine learning algorithm is to create a low bias and low variance function that has good prediction performance.

The rule of thumb is that:

- Linear machine learning functions normally have a low variance but a high bias.
- Nonlinear machine learning functions normally have a high variance but a low bias.

The most difficult aspect of parameterised machine learning algorithms is determining the optimal parameter setting to balance bias and variance. The following two algorithm examples indicate how we can adjust the bias-variance trade-off:

- KNN algorithm (low bias and high variance). We can improve the trade-off by changing the value of k, which increases the number of neighbours that changes the prediction and increases the bias of the model.

- SVM algorithm (low bias and high variance). We can tweak the trade-off by increasing the C parameter that controls the number of infringements of the margin, which increases the bias but decreases the variance.

2.5 Machine learning algorithms

The main idea behind machine learning is to use statistical models to allow computers to learn to use generic programming code. Based on the above research, we will now discuss the machine learning algorithms used in our study.

2.5.1 K-means

K-means is a popular machine learning algorithm for categorising unlabelled data into distinct clusters. The function only needs one hyperparameter: the K value. The number of clusters required is determined by this parameter. A centroid also identifies each cluster. The concept behind the K-means algorithm is to minimise the sum of distances between points and their respective cluster centroid. Iteratively, the K-means process adds a new point to the cluster whose centre is closest to it.

We used clustering as an exploratory technique in our experiments to analyse the hidden structure of our data. This technique allows us to better decompose the data set into different subsets, each of which represents a group of similar points with the same characteristics.

2.5.2 Logistic Regression (LR)

LR is a popular type of statistical analysis used in predictive analytics and machine learning modelling. LR can handle both binary and multinomial regression and is used to estimate probabilities using a logistic regression equation to understand the relationship between a known variable and one or more independent variables. A binary regression predicts a two-state outcome, whereas a multinominal regression can handle multiple options.

Advantages: LR is a popular classification algorithm that was created specifically for classification. It's useful for understanding how multiple independent variables affect a single outcome variable.

Disadvantages: The assumption of linearity between the dependent and independent variables is the major limitation of LR.

2.5.3 Support Vector Machine (SVM)

SVM is a supervised learning technique used for both classification and regression problems. SVM represents training data as points in space separated into different classification by a clear gap between the classes. The theory behind the SVM is to have the gap as wide as possible. We then plot test data into the same space and predict the category according to which side it falls.

Advantages: SVM is memory efficient because it uses a subset of the training data. Therefore, it is effective in high-dimensional spaces.

Disadvantages: Since the algorithm does not provide probability estimates, a cross-validation data process is required.

2.5.4 K-Nearest Neighbors (KNN)

The KNN is a simple data classification algorithm that classified a new data point depending on the neighbouring classification groups. The closest it is to a classification group, the more likely it forms part of that group.

KNN forms part of the “lazy learning” group of algorithms. To put it more simply, they do not rely on models built on training data prior to classifying the data points. Thus, they apply the classification model during execution. The classification of new data points is based on a simple majority vote of the nearest neighbours. For every classification cycle, the process has to search for its nearest neighbours and check the closest class group. Therefore, this algorithm is resource intensive. Furthermore, the KNN algorithm only requires one parameter, the K number. This represents the number of closest neighbours that the algorithm has to check to classify the new data point.

Advantages: The KNN algorithm is simple to implement. It can handle noisy training data, and works effectively with large training data.

Disadvantages: Finding the optimal K value can be tricky, and the computation cost is high since it needs to check the distance of the point to all the training data set.

2.5.5 Random Forest (RF)

RF (Breiman, 2001) is a powerful, yet relatively simple, supervised machine learning technique. It allows fast and automatic identification of relevant information from extensive data sets. The biggest strength of this algorithm is that it relies on the collection of various predictions (trees) rather than trusting a single one.

In classification, each tree casts a vote for the final prediction.

Advantages: The algorithm relies on the collection of various predictions (trees) rather than trusting a single one.

Disadvantages: Random forests can be computationally expensive and require more resources than individual decision trees, especially when dealing with a large number of trees and complex data sets. Training and predicting with a large number of trees can take considerable time and memory.

2.5.6 Naive Bayes (NB) classifier

Bayesian networks are probabilistic graphical models for representing undefined information. These graphs are a collection of interdependent nodes connected via a network of directed connections. Each node represents an attribute of interest in a problem domain, such as vibration levels in a road monitoring application, to measure the likelihood estimation of road anomalies. The node's edges comprise the conditional probability of the matching ransom value. The simplest Bayesian network classifier is called Naïve Bayes, which is based on the Bayes' theorem.

Naive Bayesian network relies on known data to work out the dependencies between the attributes and the class labels and, by using this information, they calculate the outcome probabilities of future events. This classifier uses the Bayes' theorem as shown in (2.1):

$$[P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{2.1}$$

Where

- A and B are events
- P(X) is the probability that event X occurs.
- P(X|Y) is the conditional probability that event X occurs if event Y is true

Advantages: This algorithm does not require large training data sets to establish the required parameters. NB classifiers are extremely fast when compared to other complex classification methods.

Disadvantages: NB is considered a poor estimator, so the probability outputs should be interpreted with caution. Therefore, any probability outputs from the model should be treated with care.

2.5.7 Deep Neural Network (DNN)

Research on road anomalies is largely based on statistical machine learning techniques. These methods require manual interventions to construct distinct features. However, with the emergence of big data, scientists are looking beyond the traditional feature engineering techniques for machine learning.

With techniques such as DL technologies, manual feature extraction is no longer required. DL works directly with raw data. We can find various studies concentrating on DL methods for road anomaly detection.

As shown in Basavaraju et al. (2020), the authors used DL to identify road anomalies. Similarly, Xu et al. (2019) used a DL to detect complex human activity recognition.

Deep learning models are based on artificial neural networks (ANN). Neural networks contain three key components: the input layer, the hidden layer, and the output layer, as shown in Figure 2.3. The first layer provides a channel for our raw sensor readings. The middle layer follows this. Each layer has various nodes connected to the neighbouring layer. This is where the model implements its logic. For more complex problems, we can increase this section to multiple hidden layers. In the final section, we find the output layer. The number of nodes varies depending on the number of classes the model is identifying. In a binary classification mode, the output layer will contain two classes representing a 1 and a 0.

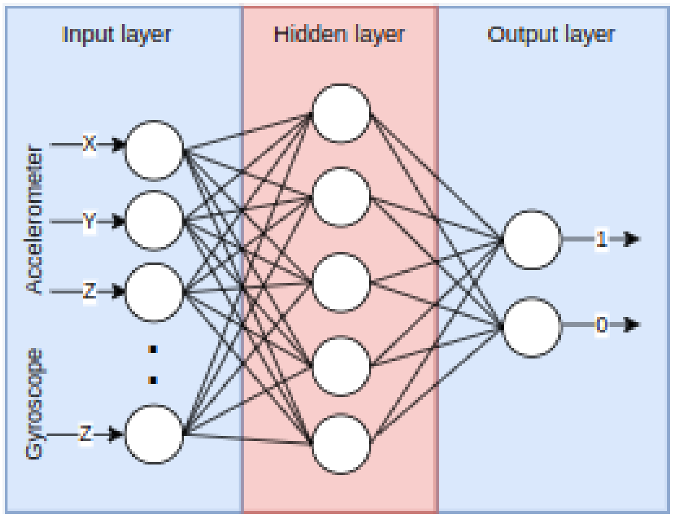


Figure 2.3: Illustration of a Basic Neural Network Architecture

2.5.8 Vision Machine learning techniques

2.5.9 The You Only Look Once (YOLO) family

Redmon et al. (2016), introduced a new concept for detecting objects in images called YOLO. Prior to their work, convolutional neural networks such as Region-Convolutional Network (Girshick et al., 2014), were less efficient. They used to detect objects by first producing bounding boxes around objects. Next, they applied post-processing techniques to remove duplicate detections and improve the bounding boxes positioning. These multi-stage techniques of object detection were inefficient and difficult to optimise.

YOLO introduced new concepts in computer vision that allowed the extraction of features and their predictions to be computed in one process. The main idea behind YOLOv1, the first release of YOLO, was to apply a seven by seven grid cell to the image. The process then checks in which cell the centre of the object is located and assigns the object to that cell. From that point onwards, the other cells will disregard that object. The authors called this architecture Darknet. YOLOv2 (Redmon and Farhadi, 2017) followed in December 2017. It added a Batch Normalisation, which was introduced by Ioffe and Szegedy (2015). The goal was to normalise the features to allow faster and more reliable training of the neural network. YOLOv2 also increased the image resolution from 224x224 to 448x448. Moreover, it introduced the concept of anchor boxes. Anchor boxes replaced the grid cells in YOLOv1.

2.5.9.1 YOLOv3

Redmon and Farhadi (2018) released the third YOLO version in 2018, with several improvements on the previous two releases. Small objects were difficult to detect in the previous versions because of the downsampling of the input image before forwarding to the deep layers. The 2018 release of YOLO addressed this issue by combining YOLOv2's Darknet architecture and the Residual network (ResNet) which was proposed by He et al. (2015).

ResNet introduced the concept of skip connectors to help the activators to access deeper layers of the network with minimal detail loss, as shown in Figure 2.4

The network has a bottleneck structure with a one by one, followed by a three by three convolution layer.

The authors also increased the Darknet architecture to 53 convolutions, as shown in Figure 2.5. For the object detection process, it stacked 53 additional layers onto the Darknet architecture, giving the algorithm 106 fully convolutional layers.

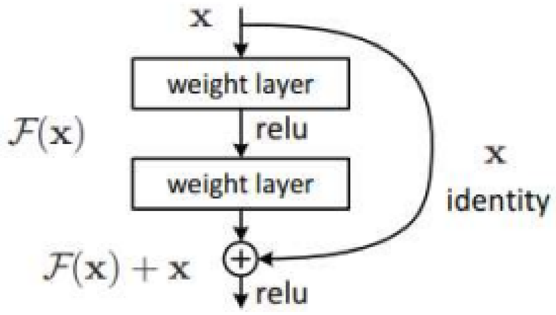
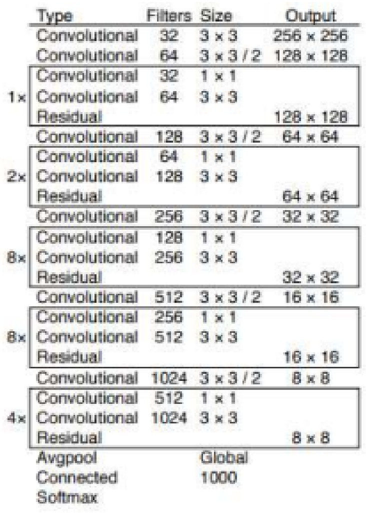

x
weight layer
$\mathcal{F}(\mathbf{x})$
relu
weight layer
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

Figure 2.4: Reset Skip connection - reproduced from He et al. (2015).



	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

Figure 2.5: Darknet-53 - reproduced from Redmon and Farhadi (2018)

2.5.9.2 Multi-scale training

YOLOv3 can detect objects at three different scales, compared to its previous versions. The authors achieved this by using the last three residual blocks by downsampling the image by 32, 16 and 8 at layers 82, 94 and 106, respectively. The author used an input resolution of 416 x 416, therefore, a downsampling stride of 32 will produce a feature map of 13 x 13 Similarly, strides of 16 and 8 produce feature maps of 26 x 26 and 52 x 52. YOLOv3 could detect smaller objects that the previous releases could not. As the map is more detailed, it can now detect fine objects.

In February 2020, Joseph Redmon announces on social media that he was withdrawing from computer vision. Following his departure, the future of YOLO became unclear. Some argue that YOLOv3 is the last YOLO member and subsequent enhancements should not use the YOLO name.

2.5.9.3 YOLOv4

On April 2020, Bochkovskiy, Wang and Liao released YOLOv4. In their YOLOv4 paper, the authors gave a detailed snapshot of the current trends in computer vision. Fig. 2.6 lists all the major components outlined in the paper.

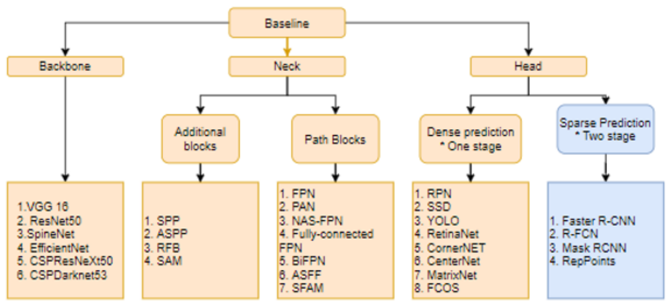


Figure 2.6: Object Detection - Major components

YOLOv4 Architecture:

a) Backbone

Following several experiments and observations, the authors discovered the CSP-Darknet53 produced better results than the most advanced convolutional networks at the time of writing. The two other contenders were CSPResNext53 and EfficientNetB3.

It is important to note that both the CSPResNext50 and CSPDarknet53 are derivatives of the DenseNet Huang et al. (2017) architecture. DenseNet consists of dense block stages and transition layers. The CSP architecture uses the same concepts of the DenseNet, but it separates the input into two portions instead of using a full-size input feature map. CSP preserves the features through propagation and reduces the number of network parameters. YOLOv4 eventually replaced the YOLOv3 Darknet-53 residual blocks with dense blocks.

b)Neck

The primary role of the neck section is to act as a collection point for feature maps from different sections of the backbone. The YOLOv4 head comprises the SPP block and the PANet feature aggregation section.

He et al. (2015) released SPP in 2015. The primary role of the SPP is to solve the problem of handling images of various dimensions. Most CNN models work on pre-defined dimensions, so they introduced SPP to generate a fixed-size output irrespective of the input dimension. SPP also includes a function used to extract important features by pooling multi-scale versions directly from the provided image data. Figure 2.7 shows the same image being copied three times using different max pooling with a kernel of different sizes.

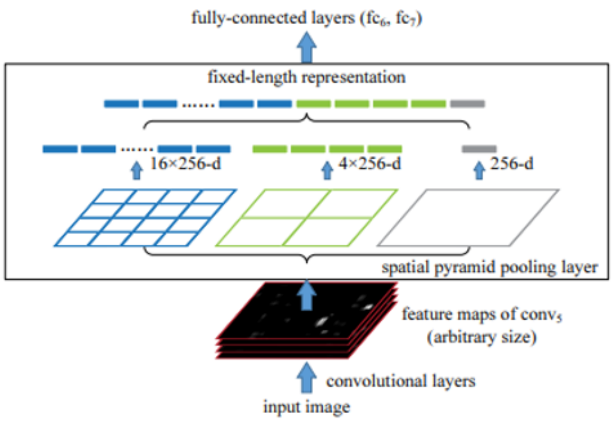

fully-connected layers (fc$_6$, fc$_7$)
fixed-length representation
16×256-d
4×256-d
256-d
spatial pyramid pooling layer
feature maps of conv$_5$ (arbitrary size)
convolutional layers
input image

Figure 2.7: Classical SSP block - reproduced from He et al. (2015)

In YOLOv4, the original SPP block was fine-tuned by concatenating the feature maps with the size of size(fmap) x size(fmap) x 512.

c)Neck – Feature Aggregation

The next step is called the feature aggregation phase, and this combines the features formed in the convolutional network backbone. YOLOv3 uses an FPN for the feature aggregation, as shown in Lin et al. (2017). The authors of YOLOv4, (Liu et al., 2018), chose the PANet as the primary feature aggregator. PAN is an enhancement of FPN. The authors added a bottom-up path as shown in Figure 2.8 In section (a), the red link shows how the FPN system works for fine-grained features. The detailed features in FPN have to travel longer to reach the higher-level layers. PANnet introduces a new shortcut that feeds the fine features directly to the top augmented layer, as shown by the green path in Fig. 2.8

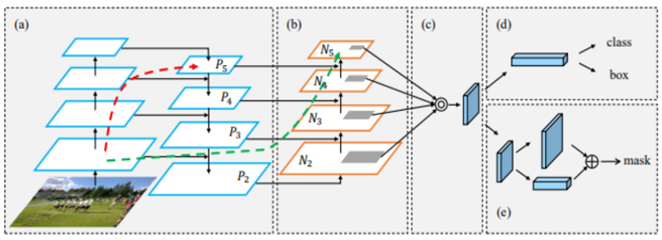


Figure 2.8: PANet architecture (a) FPN backbone. (b) Bottom-up path augmentation. (c) Adaptive feature pooling. (d) Box branch. (e) Fully connected fusion - reproduced from Liu et al. (2018)

d)Head

The Head's main functionality is to perform the dense prediction. A vector containing the predicted bounding boxes coordinates, probability classes and confidence score. The authors of YOLOv4 used the same head structure as YOLOv3.

e)Bag of Freebies.

A "Bag of Freebies" was introduced in YOLOv4. The authors introduced these added features to augment the data and improve the performance without jeopardising the accuracy.

2.5.10 EfficientDet

Tan, Pang and Le presented the EfficientDet single-shot family model in July 2020 as part of the Google Research Team AI initiative. They built EfficientDet on Tan et al. (2020) architecture.

The authors set forth two major contributions:

1. BiFPN - A weighted bidirectional feature pyramid network. They based it on the ideas of FPN, PANet and NAS-FPN, which enable information to flow in both the top-down and bottom-up directions.

2. Compound scaling: A new concept, which improves the backbone, feature network, box/class network, and resolution.

2.5.11 Faster R-CNN

Faster R-CNN forms part of the region-based CNN family. Ren et al. (2017) released the algorithm. In contrast with the YOLO family and EfficientDet, Faster R-CNN is a two-stage detector.

Both the previous R-CNN and Fast R-CNN algorithms used selective search to locate the region proposals. Selective searches are a slow and time-consuming process. The authors of the Faster R-CNN algorithm eliminated the bottleneck and created a new network for region proposals or RPN for short, as shown in Fig. 2.9.

The authors have also introduced the concept of anchors. Anchors are the central points of the sliding windows. These anchors assign labels based on two factors:

- The anchors with highest Intersection-over-Union overlap with a ground truth box

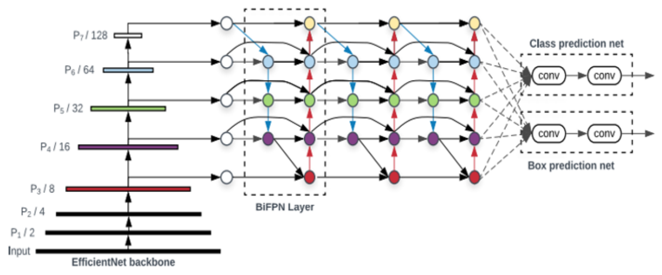


Figure 2.9: EfficientDet Architecture - reproduced from Ren et al. (2016)

- The anchors with Intersection-Over-Union Overlap higher than 0.7

2.5.12 MobileNetV2

Sandler et al. (2018) released MobileNet V2 in April 2018. The V2 implementation has several improvements to the previous V1 system, and it is suitable for Mobile devices with low hardware resources.

The major difference between version one and version two is that v2 has two types of convolution blocks, as shown in Figure 2.10. In contrast, version one has only one block. The first is a residual block with a stride of one. The second block has a stride of two, which is used for downsizing.

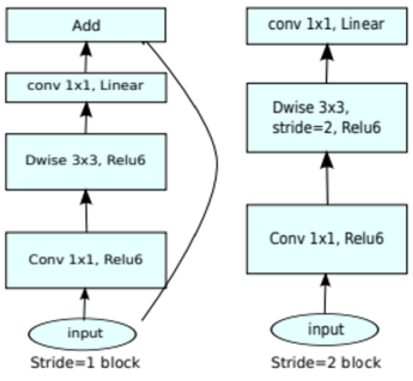
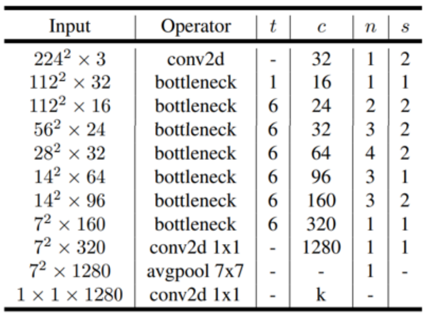


Figure 2.10: MobileNetV2 - reproduced from Sandler et al. (2019)

Both blocks comprise three layers. A ReLU6 one by one convolution layer is the first layer. The second is a depth-wise convolution (different strides). The third is a linear one by one convolution layer. Figure 2.11 shows the overall architecture of MobileNetV2.



Input	Operator	t	c	n	s
$224^2 \times 3$	conv2d	-	32	1	2
$112^2 \times 32$	bottleneck	1	16	1	1
$112^2 \times 16$	bottleneck	6	24	2	2
$56^2 \times 24$	bottleneck	6	32	3	2
$28^2 \times 32$	bottleneck	6	64	4	2
$14^2 \times 64$	bottleneck	6	96	3	1
$14^2 \times 96$	bottleneck	6	160	3	2
$7^2 \times 160$	bottleneck	6	320	1	1
$7^2 \times 320$	conv2d 1x1	-	1280	1	1
$7^2 \times 1280$	avgpool 7x7	-	-	1	-
$1 \times 1 \times 1280$	conv2d 1x1	-	k	-	

Figure 2.11: MobileNetV2 architecture - reproduced from Sandler et al. (2019)

2.5.13 Classification process

One important process in machine learning life cycles is the evaluation of the model. Different machine learning models perform differently on the same data set. To measure the suitability of the model against a particular task, we use different metrics to quantify the success rate.

To evaluate the model's performance, we compare every prediction made in the test with the record's label and the predicted class value. There are mainly four outcomes:

- True Positive (TP) = A correct detection and classification
- False Positive (FP) = An incorrect detection and classification
- True Negative (TN) = The object was not in the image and was not detected by the model
- False Negative (FN) = The object was in the image and the model did not detect it

We consider a TP prediction if the predicted value belongs to the same label class. If the value prediction is not the same, then we mark it as TN. A FP proves that we wrongly classified the test record as the same label class, but in reality, it does not form part of the label class. When we classify the test record wrongly, we mark it as FN.

2.5.13.1 Accuracy, Precision, Recall and F1 Score

In Chapter 4, we used Accuracy, Precision, Recall and F1-score to assess the model's classification performance, as shown below:

Accuracy shows the percentage of correct forecast in respect to all classes. This is presented as equation (2.2).

$$Accuracy = \frac{TP + TN}{TP + FN + FP + FN} \tag{2.2}$$

Precision is a crucial metric that highlights the proportion of relevant features predicted by the model, which are genuinely relevant. The precision formula is presented as equation (2.3).

$$Precision = \frac{TP}{TP + FP} \tag{2.3}$$

Recall is a vital metric that assesses the model's proficiency in capturing all relevant features present in the dataset. The formula for recall, denoted as (2.4), quantifies the model's ability to identify true positives correctly.

$$Recall = \frac{TP}{TP + FN} \tag{2.4}$$

F1-score is a valuable metric that combines the precision and recall measures into a single score, mitigating the influence of imbalanced datasets. The formula for calculating the F1-score, denoted as (2.5), captures the balance between precision and recall, providing a more robust evaluation of the model's overall performance.

$$F1Score = \frac{2TP}{2TP + FP + FN} \tag{2.5}$$

2.5.13.2 mAP

The computer vision community has converged on the mAP metric to compare the performance of object detection systems.

To calculate mAP, we first must find the Intersection over Union (IoU). This is the overlap between the object detection bounding box and the ground truth box. Figure Figure 2.12 shows the ground truth box, and the predicted bounding box of a pothole. The lime and the magenta boxes represent the ground truth and predicted boxes, respectively.

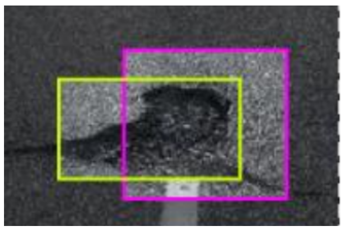


Figure 2.12: Visualizing Model Accuracy: Intersection over Union (IoU)

The True Positive, False Positive and False Negative are determined by the IoU formula as indicated in equation (2.6).

$$IoU = \frac{area(Bp \cap Bgt)}{area(Bp \cup Bgt)} \tag{2.6}$$

where the B_p is the bounding box and B_{gt} represent the ground truth box.

For a mAP of .50, we mark it as a true positive if the IoU is greater than 0.5. Similarly, we register it as a false positive if the IoU is smaller than 0.5. No detection is a false negative. We do not evaluate true negatives because we assume an object is always present in an image. Then we calculated the mAP by taking the area under the Precision/Recall Curve for every class. We calculate this by taking the average of the calculated AP for all the classes.

A model with 95% mAP yields better results than a model with a 60% mAP, but there are cases where the precision and recall of a 60% mAP model performs better in real-life scenarios.

2.6 Summary

This chapter has evaluated the state-of-the-art IMU and vision machine-learning techniques for road anomaly detection. First, we briefly introduced the different machine learning methods. In particular, supervised and unsupervised methods. Next, we presented smartphones in the road anomaly detection domain, including their advantages and disadvantages. We also gave a brief introduction to the various smartphone-embedded sensors. Later, we introduced the various smartphone solutions based on a comprehensive literature review presented to understand the use of IMU and computer vision models.

Chapter 3

Materials & Methods

3.1 Introduction

In chapter 2, we presented our review of the existing research on road anomaly detection. These include threshold-based solutions and machine learning approaches, including deep learning and convolutional neural networks for imagery data. The vibration sensory review shows that machine learning techniques are more accurate and versatile than threshold techniques. However, it is clear from previous studies and experiments that detecting road anomalies just from inertial readings is way harder than it looks. Notably, IMU methods require a direct anomaly impact to sense an anomaly. As a result, an avoidance manoeuvre will not detect a road defect.

Previous deep-learning imagery studies on road anomalies show that vision solutions deliver excellent results and can categorise multiple object recognition and classifications. Although there are numerous studies in this field, research into real-time road anomaly detection is still ongoing. Particularly, distinguishing between an oil spill and a genuine road anomaly, such as a pothole or crack, remains difficult. Furthermore, distinguishing between a pothole and a depression remains difficult. To overcome these weaknesses, this dissertation will fuse both the IMU and vision data to improve anomaly detection.

The rest of this chapter is organized as follows. We provide the solution overview in section 3.2. In this section, we give a detailed overview of the entire solution, including all the individual components and software used in this dissertation. In section 3.4, we cover the IMU capturing application, including the methodology used to carry out the experiments. Similarly, in section 3.5, we explain how we addressed the vision application requirements. We then cover the vision application in section 3.5 followed by the smartphone placement to capture the road imagery. In the section 3.6 we cover the back-end processes for both applications, including the data collection and processing techniques.

Finally, in section 3.7 we discuss the data fusion methods proposals.

3.2 Proposed Solution

To the best of our knowledge, there are no publicly available road anomaly data sets available for the Maltese islands. At the same time, the available foreign imagery data sets lacked the timestamp and corresponding IMU data. With this in mind, the sensory and MobileNetSSD images were both collected using our Android smartphone device. We will inspect the data set collection and pre-processing stages later in this chapter.

The proposed road anomaly detection solution comprises three core modules, as shown in Figure 3.1.

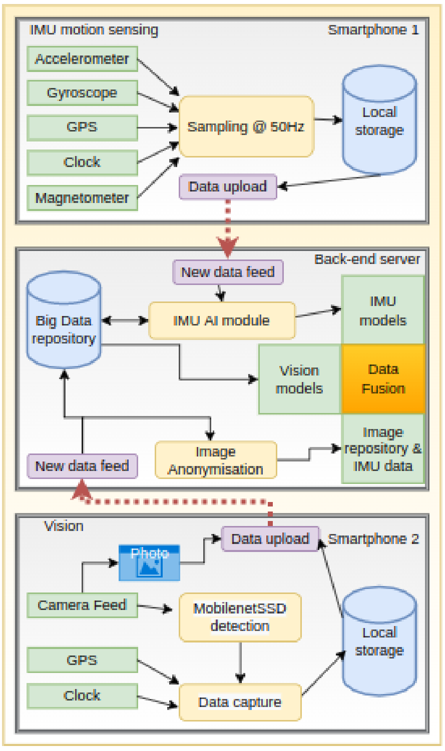


Figure 3.1: Comprehensive System Overview: A Tripartite Breakdown

Both the IMU motion sensing and vision detection modules run on two separate smartphone devices. The GPS data is providing the backbone for synchronising the applications and providing the vehicle speed and geolocation coordinates. We used a back-end server for data cleansing, re-sampling, analysis, and reporting. As explained earlier, the primary aim of this research is to detect road anomalies and produce a data-rich platform to help the authorities to pinpoint road anomaly locations. As a result, we have included a data fusion platform to further enhance the prediction and also provide a rich data set for further analysis. Additionally, an image feed, including the geolocation and magnetometer data, is also being captured and stored for future studies. Equally important, the stored imagery data is being anonymised using a third-party product for privacy protection.

Figure 3.2 illustrates how we conducted our experiments. We started our first phase with the development of the IMU data-capturing application, followed by the experiments stage. During the second phase, we addressed the vision module, where we trained a MobileNetSSD model to detect road anomalies. Finally, we experimented with the fusing of both data streams to further enhance the reporting.

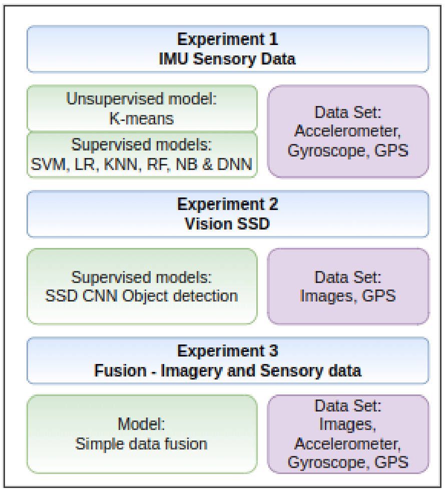

Experiment 1
IMU Sensory Data
Unsupervised model: K-means
Supervised models: SVM, LR, KNN, RF, NB & DNN
Data Set: Accelerometer, Gyroscope, GPS
Experiment 2
Vision SSD
Supervised models: SSD CNN Object detection
Data Set: Images, GPS
Experiment 3
Fusion - Imagery and Sensory data
Model: Simple data fusion
Data Set: Images, Accelerometer, Gyroscope, GPS

Figure 3.2: Illustration of Experimental Methodology: Unveiling the Stages and Processes in our Study

3.3 Software used

We can find a list of software used for this study in Table 3.1

Table 3.1: Comprehensive List of Utilized Software: Tools and Programs Employed in the Study.

Software	Version	Usage
Flutter[1]	2.4.0	Mobile applications
Dart[2]	2.1.2	Mobile applications
Python[3]	3.8.5	Backend, IMU and Vision models, data fusion
PostgreSQL[4]	13.4	Mobile applications and reporting
understand-ai/anonymizer[5]	n/a	Vision applications
Go[6]	1.18	Reporting
JavaScript	ES6	Reporting
Leaflet[7]	1.8	Reporting

3.4 IMU application overview

The fundamental concept behind the IMU application was to create a native smartphone application that would capture the accelerometer, gyroscope, GPS coordinates, and clock data at a predefined sampling rate. The application uses the smartphone's operating system programming interfaces to read data from the device's sensors. We created software functions that stored this data in an embedded smartphone RDBMS database, which was then transferred to the back-end server for analysis.

3.4.1 Methodology

Figure 3.3 shows the methodology used for the IMU sensory experiment. To record the vibration, we mounted one smartphone on the car's dashboard. The device recorded accelerometer and gyroscope readings, as well as GPS data and magnetometer readings. Following that, we proceeded with the data preparation and feature extraction stages. This process examined several iterations to determine the best sampling rates.

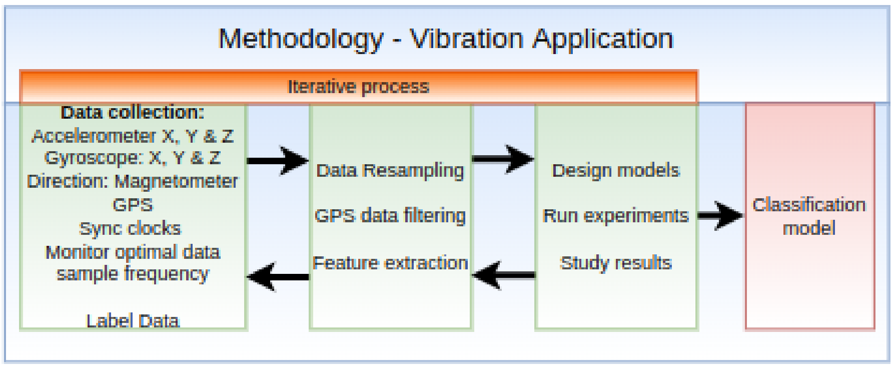


Figure 3.3: Illustration of the Adopted System Methodology

3.4.2 IMU sensing

As previously stated, we collected the sensory data using a custom native application developed in Flutter to capture the accelerometer, gyroscope and GPS data. Particularly, the design process included the GPS timestamp with every sample. This was required to synchronise and align the readings from the two smartphone applications. Literature review shows that GPS time synchronisation is accurate and can synchronise devices where GPS coverage is available. This topic is further discussed in 3.6.5. The accelerometer (X, Y and Z) and gyroscope (X, Y and Z) data were sampled at 50 Hz. We considered sampling at higher rates, but due to hardware constraints, our experiments revealed that 50 Hz was the best sampling frequency.

We also set the sampling rate for the GPS data at one Hz because of hardware constraints. Therefore, we collected 50 sensor samplings for every GPS coordinate. To be more specific, each GPS location generates 300 unique sensor readings consisting of 50 samples of three accelerometer axes and three gyroscope axes. We collected the samples while driving on arterial, distributor, and secondary roads. We do not include tertiary roads in this study.

3.4.3 Smartphone placement

We secured the smartphone to the centre console of the car. As shown in Figure 3.4, the smartphone's Z axis was tightly positioned against the back of the entertainment console, and its Y axis was mounted horizontally on the dashboard. Placing the smartphone in full contact with the car's console guarantees that we always maintain the same mounting angle. Likewise, the smartphone is always in full contact with the vehicle's body, unlike windscreen rubber-mounted stands.

Figure 3.5 shows the vehicle's axis together with the mobile's axis alignment. Our setup is based on Basavaraju et al. (2020)'s work where the authors used a fixed smartphone position to gather the data without the need to realign the smartphone's accelerometer and gyroscope axes.

3.4.4 Kalman filter

Accurate GPS coordinates are of the utmost importance for this research. Research by Raghunath et al. (2013) shows the advantages of using Kalman filters to enhance GPS positioning by correcting false GPS readings and signal interference. In our experiments, we included a simple Kalman filter in the native smartphone application to reduce the GPS position errors in areas of bad GPS reception.

3.4.5 Data Annotation

Precise data labelling is crucial for supervised machine learning algorithms. The accuracy of the models depends on accurate data and the corresponding labels. For this task, both smartphones were used, one for reading the sensory data, as shown in Figure 3.6 (a) and the other to annotate the anomalies, as shown in Figure 3.6 Figure (b).

Figure 3.7 shows the route used for our training data set. The data-collection experiment was carried out on various roads in Malta, primarily in the northern and central

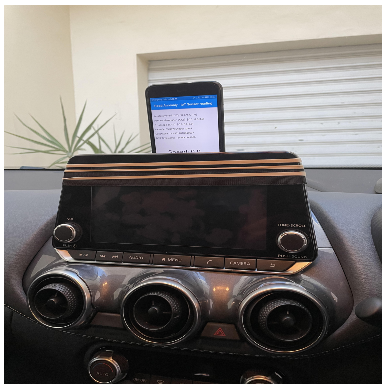


Figure 3.4: Illustration of Smartphone Mounting on the Vehicle Dashboard for Data Collection

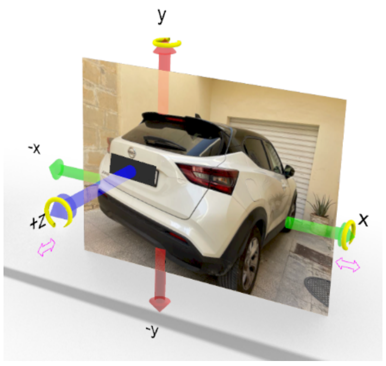


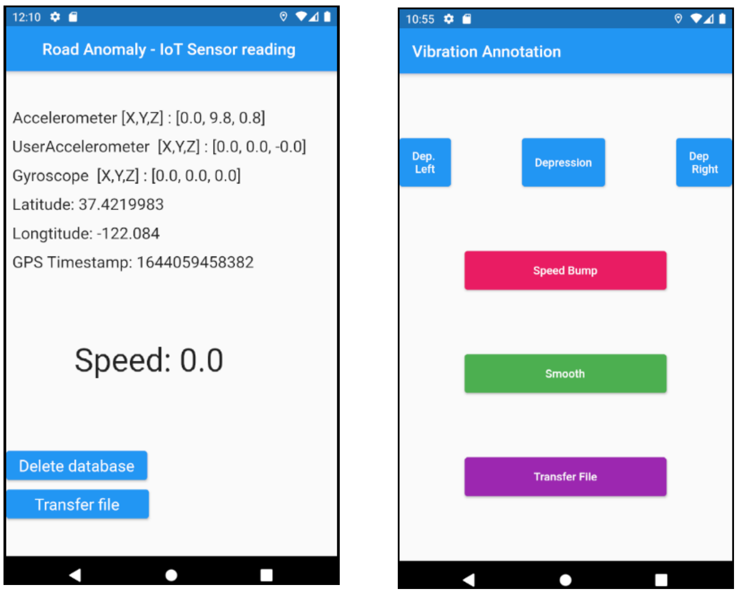
Figure 3.5: Alignment of Vehicle Axis, Gyroscope, and Accelerometer: Illustrated Configuration

regions. These data-gathering events occurred during a relatively low traffic period, at a constant driving speed of 25 to 40 km/h.

This exercise covered approximately 70 km in total distance. The arterial roads were in relatively good condition. However, some urban and main roads were significantly worse, with numerous potholes and depressions. Furthermore, we observed that speed bumps were not as common as expected.

An assistant was given the job of labelling the road anomalies. To label the road anomalies, the application provided five labelling options. These included:

1. Left Depression
2. Depression
3. Right Depression
4. Speed Bump


12:10

Road Anomaly - IoT Sensor reading

Accelerometer [X,Y,Z] : [0.0, 9.8, 0.8]
UserAccelerometer [X,Y,Z] : [0.0, 0.0, -0.0]
Gyroscope [X,Y,Z] : [0.0, 0.0, 0.0]
Latitude: 37.4219983
Longtitude: -122.084
GPS Timestamp: 1644059458382

Speed: 0.0

a IMU Sensing b Vibration Annotation

Figure 3.6: Snapshot of IMU Sensing and Annotation Applications in Action

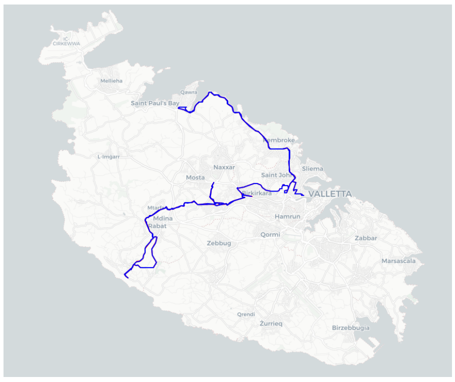


Figure 3.7: Illustration of Training Route: Sampling and Labeling Process Employed in Our Test

5. Smooth Road

In the IMU tests, we grouped potholes, cracks, and other forms of road surface cavities under one depression category. Table 3.2 shows a sample label data set consisting of three different categories:

- Time Element (GPS time, smartphone time and time drift). As explained earlier, the variance is used to measure the time difference between the smartphone clock and

Table 3.2: Dataset Snapshot: Labeled IMU Samples for Training and Validation

gpstime	mobiletime	time drift	long	lat	speed	label
1641628262000	1641628250772	114	35.8901005	14.4644139	9.43468475341797	1
1641628266000	1641628254610	114	35.8899563	14.4648398	10.7666940689087	1
1641628894000	1641628882850	114	35.8963245	14.4578413	7.38304233551025	1
1641628902000	1641628891213	114	35.8965452	14.4581358	7.70915126800537	1
1641629015000	1641629004576	114	35.9009803	14.4529821	10.722484588623	1
1641630269000	1641630258280	114	35.9142195	14.4482561	9.865159034729	1
1641630541000	1641630529997	114	35.8986568	14.4563057	7.81180143356323	1

the GPS atomic clock. We performed a similar reading on the sensory data capturing application (Figure 3.6)(a) to synchronise both smartphones.

- GPS Element: (longitude and latitude). To capture the GPS coordinates
- Label Element (Label). The captured anomaly type.

Following several tests to determine the optimal sampling window, we concluded that a two-second sampling window was satisfactory to capture the road anomaly. This time window comprises a one-second sampling window before and after the user clicks on an anomaly button. Therefore, for every anomaly label, we are annotating both the prior and post 50 sensory readings. We consider this time window adequate for a user to locate and press the correct label on the annotation smartphone, even during bumpy rides.

A clear limitation of this method is that, during the two-second interval, we might have collected more than one anomaly and distort the sample readings. As we will see in subsubsection 3.6.5.1, speed dependency is an important factor when monitoring road anomalies. A vehicle travelling at 40 km/h covers roughly 22 meters of road in two seconds. However, this solution is common in road anomaly detection and we are basing our assumption that there is only one anomaly per time window. In addition, we will also capture smooth road sampling in the process and this might negatively impact the ANN model, since we will label the smooth sampling records as an anomaly during training.

On transferring both the sensory and annotation data sets to the back-end server, a process aligned both data sets by utilising the GPS atomic clock variance method described earlier.

3.4.6 Hyper-parameters

In our experiments, we used the Sklearn grid search[8] function to search for the optimal hyper-parameter values. This is the simplest approach to optimisation to increase the classification score. It is extremely easy to implement and use, but can be very time-consuming depending on the size of the search grid. Table 3.3 shows the hyper-parameter used for the above tests. Furthermore, the tables also show the parameters used in our grid search test to find the optimal values.

Table 3.3: Configured Hyperparameters: Overview of Experimental Settings in Focus

Model	Hyper-parameter	Grid Search parameters
LR	solver='newton-cg',multi_class='auto',C=1	'C': [1,5,10]
SVM	kernel='rbf', gamma=1, C=1	kernel: ['rbf','linear'], 'C': [1,10,20]
KNN	n_neighbors = 3	n_neighbors: [1, 3, 5, 7, 9, 11, 13, 15]
RF	n_estimators=11	n_estimators: [1,5,10,11,13]
NB	var_smoothing = 1e-09	var_smoothing = 1e-09

[8]https://scikit-learn.org/stable/modules/generated/sklearn.model$_selection.GridSearchCV.html$

3.5 Vision app overview

The vision application hosts the MobilenetSSD detection model for detecting road anomalies instantaneously through an image feed using the smartphone's camera. Once the model detected an anomaly, we extracted the GPS coordinates and timestamps and stored them in a database. The corresponding anomaly class and GPS data were also being sent to the back-end server for processing.

3.5.1 Vision Methodology

Figure 3.8 shows the method of the vision module for the second experiment. We conducted several tests from different mounting points to improve the visibility and model accuracy. Furthermore, we used Roboflow [9], a third-party annotation and machine learning platform. Moreover, we trained the model on Colab[10].

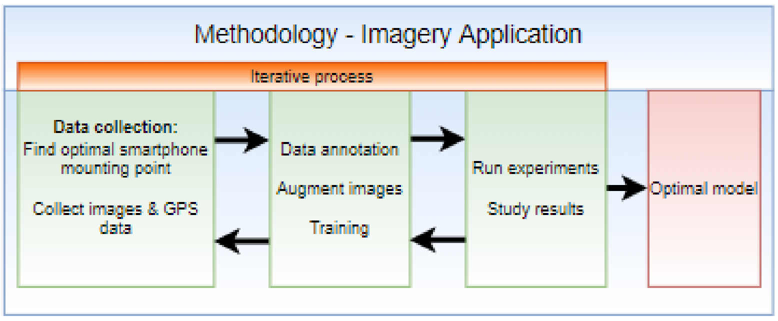
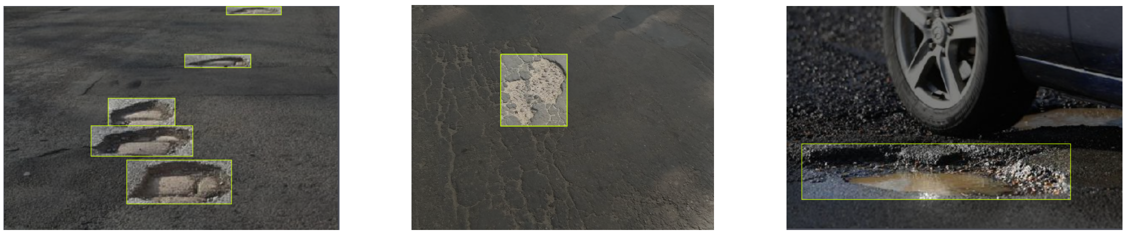


Figure 3.8: Visual Module System Overview: Illustrating the Functionality

In our preliminary experiments, we tested five advanced computer vision models to compare their performance and object detection accuracy. We carried out the experiments using the Roboflow pre-configured computer vision model libraries and code. The code runs on Jupyter notebooks using Google's Colab Pro cloud services. One major disadvantage of using Colab is that there is no guarantee that the cloud service will assign the same hardware resources for every consequent run, therefore, the model's frames per second could not be correctly evaluated. We implemented three YOLO implementations using the PyTorch machine learning framework. We executed the EfficientDet and TensorFlow V2 models using a pre-trained COCO 2017 data set. Faster R-CNN and MobileNetSSD both used the TensorFlow V1.5 models with a pre-trained COCO 2018 data set.

[9] For more information see: `https://roboflow.com/`
[10] For more information see: `https://colab.research.google.com/`

For this experiment, we used a Roboflow public data set[11] with 665 pothole images in a 70-20-10 ratio for training, validation and testing. The data set contained various potholes from different angles and distances as shown in Figure 3.9. We selected the Roboflow pothole data set following several online searches. Atikur Rahman Chitholian[12] originally released the data set as part of his undergraduate thesis. The data set is now included in Roboflow's public data set under the developers' section to promote their machine learning toolsets. Furthermore, the data set comes furnished with pre-annotated road potholes. MobilenetSSD produced encouraging results, as can be seen in Table 4.17.


(a) Roboflow data set sample 1

(b) Roboflow data set sample 2

(c) Roboflow data set sample 3

Figure 3.9: Snapshot of Roboflow Dataset: Sample Images Illustrating the Variety of Data

To conduct live testing, we migrated the MobileNetSSD model to TensorFlow Lite to run the model directly on our mobile applications. However, during our field testing, we noticed the model was performing poorly in our setup. We were getting a high percentage of false positives and negatives. Therefore, we agreed to remove closeup images of potholes and distant ones, and leave average-sized potholes. We also removed images of potholes that were taken at 90 degree angles. To keep the same number of images, we added new images taken from the new camera position to keep the data set size consistent. We also added images of speed bumps to be consistent with the IMU experiments. Following these changes, we trained a new model and ran additional tests. In addition, according to our analysis, the new model and camera position were not yielding optimal results.

3.5.2 Smartphone placement

In the course of our vision experiments, the smartphone placement played an important role. We conducted numerous experiments to determine the optimal mounting location.

[11]Accessible from `https://public.roboflow.com/object-detection/pothole`

[12]Atikur Rahman Chitholian `https://www.kaggle.com/datasets/chitholian/annotated-potholes-dataset`

Our primary aim was to capture road anomalies that were within the two-second time window used for IMU machine sampling tests. This was required so that we could fuse both results and augment the reporting dashboards. As a result, our primary objective was to configure the smartphone camera so that we only capture anomalies within 20 metres of the vehicle's front wheels. Therefore, to ensure optimal performance, we strategically positioned the smartphone at the highest available location on the vehicle's windscreen, as depicted in Figure 3.10. This placement methodology draws inspiration from the successful approaches implemented by Camilleri and Gatt (2020), Tithi et al. (2021) and Rani et al. (2020).

However, upon implementation, we encountered an unforeseen issue: the camera captured not only the intended elements but also the pavements and other unrelated objects. Regrettably, this unintended inclusion of irrelevant data rendered this solution less effective for our specific application.

Despite drawing inspiration from previous works, the captured extraneous content impacted the accuracy of our data collection. As a result, we explored alternative solutions to address this challenge and refine our methodology for improved results.

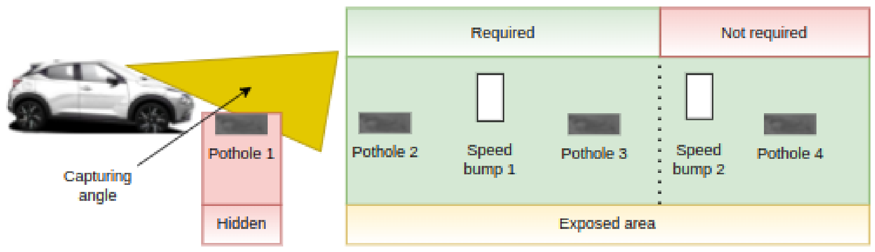


Figure 3.10: Initial Test: Camera Position Configuration for Observation

Our next attempt was to mount the camera onto the front bumper closer to the front vehicle wheels as shown in figure 3.11. This experimental design was based on the assumption that the camera would be closer to the vehicle's front wheel with unobstructed views. Based on this assumption, we reintroduced closeup images of potholes and speed bumps in our training data set. However, during live testing, there were trends in our data suggesting that the smartphone camera could not maintain the focus when driving at speeds of over 5 km/h. Therefore, we discarded this option.

We, therefore, investigated another method based on mounting the smartphone in the vehicle's cabin and setting the camera's zoom to 300%. The main reason behind this experiment was to capture as much road surface as possible and exclude sidewalks. As a result, we focused our camera on the approaching road anomalies. Figure 3.12 shows the capturing area in more detail. Once again, we removed all closeup images from the

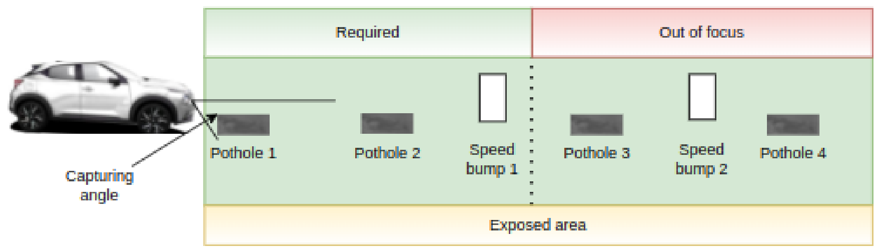


Figure 3.11: Second Test: Camera Position Configuration for Observation

data set and augmented the images by adding image flipping, shearing and brightness.

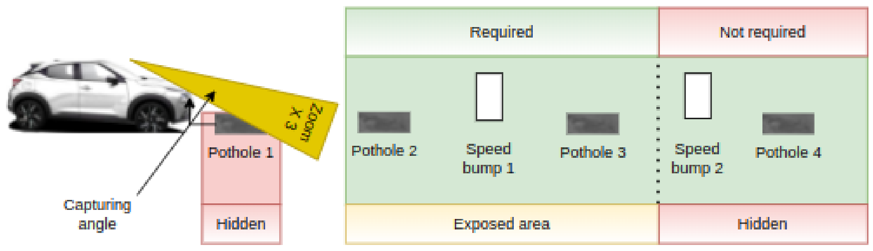


Figure 3.12: Third Test: Camera Position Configuration for Observation

We carried out the tests during various hours of the day and we could observe that tests carried out around and during sunset and sunrise yielded better results. The low angle of light softens the shadows, creating a flattering visual effect. In contrast, daytime photos contained shadows, which increased the false positive rate.

3.6 Back-end system overview

The back-end system consists of a Linux server running on a cloud virtual machine. We used a PostgreSQL database as a central repository where several Python scripts were used to process the data and merge the IMU sensory and vision data sets to augment and enrich the reporting data, as shown in Figure 3.13. The IMU, vision annotation and vision applications files are all transferred to the server for processing.

3.6.1 Data Collection and Pre-processing

Data collection and pre-processing are a pivotal part of machine learning and a debated subject in various communities. According to Ramírez-Gallego et al. (2017), data pre-processing is a major feature in machine learning processes which normally accounts for approximately 50% of the total effort during data analysis. As a result, we often under-

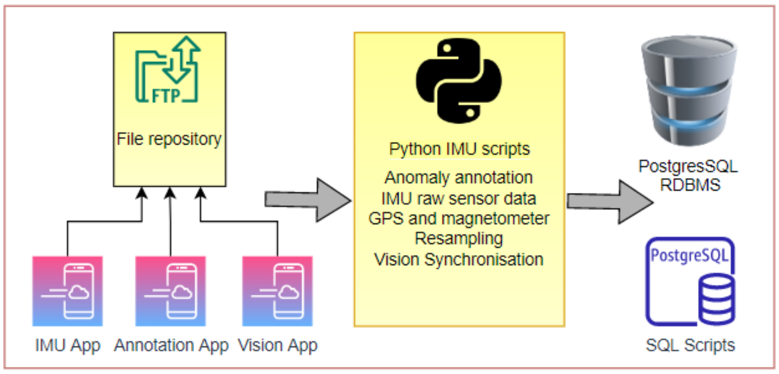


Figure 3.13: Visualizing the Infrastructure: Overview of the Back-End Server System

estimate this process during the planning stages. Data pre-processing mainly consists of data acquisition, data labelling, and data preparation.

Raw data is prone to several irregularities, such as missing data, noise and irregular sampling, which can drastically affect the performance and accuracy of the subsequent learning and prediction steps. Therefore, it is imperative to perform a proper data pre-processing exercise on the raw data sets to improve the machine learning model accuracy.

Another important task of data analysis is called feature engineering. This process involves the designing of the features for the machine learning models. According to Oyamada (2019), the ideal features must capture the characteristics of the data being analysed to improve the predictive and performance of the models.

Roh et al. (2021) recognised the challenges associated with feature engineering and argue that users need to become familiar with the problem domain to provide the best features to train the models. In their survey, they also highlight that deep learning is now changing the backdrop since deep learning models can now auto generate features, which saves the user considerable time. Similarly, Bach et al. (2017) confirm that traditional supervised machine learning techniques require access to labelled data to work, and this is usually a major bottleneck in developing new methods and applications. Furthermore, they also underline that deep learning requires larger amounts of training data to perform.

3.6.1.1 Resampling

For this dissertation, two identical Huawei P10 Lite smartphones were used to host both the IMU and vision applications. We have carried out a thorough sensor sampling testing exercise to find the optimal sampling frequency without overloading the smartphone device. One has to remember that the number and types of sensors directly affected the sampling rate accuracy. For instance, apart from the accelerometer sensors, the application also captured the gyroscope movements and GPS data. For this reason, we analysed sampling rates ranging from 40Hz to 100Hz for sampling periods of 30 minutes each. Sampling rates exceeding 55 Hz produced inaccurate sensing results and uneven sampling batches.

Sampling ranging from 48 Hz to 52 Hz produced stable readings with minimal sampling variances per second. We mainly attributed these variances to the sampling mechanism, since we were using a software-based sampling thread to read the sensors. Ultimately, we settled on a 50 Hz sampling frequency.

We, therefore, used a simple linear interpolation re-sampling technique to correct the sampling deficiency, as shown by He (2018). The author combined decimation and interpolation to re-sample raw accelerometer data to extract a constant sampling rate.

3.6.2 Dimensionality reduction

3.6.2.1 Sequential Forward Selection and Sequential Backward Selection

Sequential Forward Selection (SFS) and Sequential Backward Selection (SBS) are feature selection techniques used in machine learning and statistics. Both methods aim to improve model performance and reduce complexity by selecting the most relevant subset of features from a larger set.

In Sequential Forward Selection, the algorithm starts with an empty feature set and iteratively adds one feature at a time, evaluating the performance of the model after adding each feature. The feature that provides the most improvement in model performance is selected in each step until a predefined number of features is reached.

On the other hand, Sequential Backward Selection begins with the entire feature set and removes one feature at a time in each iteration. The feature whose removal causes the least drop in model performance is eliminated until the desired number of features is achieved.

These techniques are useful when dealing with high-dimensional data and can help prevent overfitting and improve model interpretability by selecting only the most relevant features.

3.6.2.2 PCA

Principal Component Analysis (PCA) is a method used to make complex data simpler. It takes a big set of information and turns it into a smaller one while keeping the important parts.

PCA does this by finding new directions that are at right angles to each other in the original data. The first direction is the one with the most variation in the data. The next ones are chosen to be perpendicular to the ones before and have the most variation left.

This way, PCA changes the data to be easier to understand and work with. It helps researchers or analysts see the main patterns and connections in the data. PCA is used before applying machine learning techniques to make the process faster and more accurate.

PCA is particularly helpful when there is a large amount of data to handle, as it can be challenging to manage such extensive information. It helps users avoid problems that may arise from having too much data to process, such as encountering technical difficulties or making mistakes in the analysis.

3.6.3 Cross-validation

Since normally only a limited amount of data is available for developing and testing the models, a data shuffling technique called cross-validation is used to recycle the data between the learning and testing phases. Cross-validation splits the data several times to build a randomly built statistical training and test model to maximise the entire data set. K-fold cross-validation evaluates the performance of classification algorithms as shown by Wong and Yeh (2020). The K-fold technique divides the data set into a K subset and performs the training and testing phases K times. At the end of the process, all K-fold results get statistically analysed and presented for evaluation. Consequently, we used a K-fold of five folds in our tests.

3.6.4 Feature Extraction

Our feature extraction is based on a similar work conducted by Hemminki et al. (2013). The study is based on accelerometer-based techniques for accurate and fine-grained detection of transportation modes on smartphones. In total, we extracted 204 distinct features based on our research. We equally divided these between the accelerometer and gyroscope data. Table 3.4 shows the final group of selected features used in our IMU experiments. The table includes the feature, the feature's time and frequency domain usage, and the reference source extracted from the research phase.

3.6.5 GPS Time synchronisation

Smartphone manufacturers rely on oscillator components to keep the clock running when disconnected from the internet. The frequency rate of the oscillator determines the rate

Table 3.4: Utilised IMU Features: Comprehensive Overview in the Feature used in our Experiments

Feature	Time Domain	Frequency Domain	Source
Mean	✓	✓	Cabral et al. (2018)
Variance	✓	✓	Cabral et al. (2018)
Std dev	✓	✓	Cabral et al. (2018)
Min	✓	✓	Cabral et al. (2018)
Max	✓	✓	Cabral et al. (2018)
Max-min diff	✓	✓	Carlos et al. (2018)
Median	✓	✓	Cabral et al. (2018)
Median abs dev	✓	✓	Cabral et al. (2018)
Interquartile range	✓	✓	Cabral et al. (2018)
Skewness	✓	✓	Cabral et al. (2018)
Energy	✓	✓	Cabral et al. (2018)
Signal magnitude area	✓	✓	Cabral et al. (2018)
Kurtosis	✓	✓	Cabral et al. (2018)
Negative count	✓	✓	Fox et al. (2015)
Positive count	✓	✓	Fox et al. (2015)
Positive less Negative	✓	✓	Fox et al. (2015)
Values above mean	✓	✓	Fox et al. (2015)
Number of peaks	✓	✓	Fox et al. (2015)
Avg resultant	✓	✓	Fox et al. (2015)

at which the clock runs, as shown by Sivrikaya and Yener (2004). In their survey, the authors stressed on the importance of the trade-offs between precision and energy efficiency. Additionally, the authors argue that in some applications, the synchronisation accuracy may be in the order of a few milliseconds. However, the on-board clocks may become inaccurate over time because of frequency shifts, resulting in clock drifts.

One major contribution by Mazur et al. (2017) proposed a simple solution for synchronising smartphone clocks in GPS. Manufacturers furnish smartphones with GPS receivers. GPS provides accurate longitude, latitude, and altitude data. Equally important, it also provides an important fourth dimension needed to synchronise time. In particular, GPS transmitters send out multiple atomic clock feeds with precise timing data, which, according to the authors, are much more accurate than other synchronisation methods such as Network Transport Protocol. Therefore, this simple technique of synchronising smartphone clocks can yield high-precision time synchronisation between devices by using standard onboard smartphone components.

Moreover, Benndorf and Haenselmann (2016) recognised the benefits of using GPS data to synchronise mobile devices. In their study, they determine the best options for synchronising multiple Android devices. The authors confirmed that the GPS accuracy surpasses all tested techniques where the GPS data was accessible.

In our experiments, we calibrated our applications by checking the clock variance between the smartphone internal clock and the GPS clock and included this offset variable in our data sets as shown in Figure. 3.14. The road anomaly sampling periods were less than 30 minutes each and the evaluation results indicate minimal time drifting between both clocks during these short data-gathering exercises.

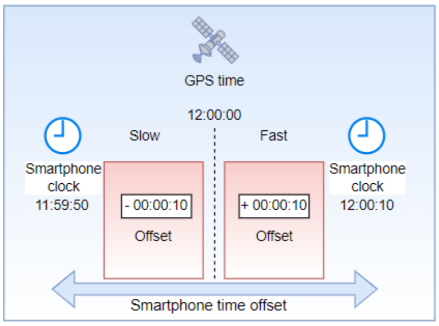


Figure 3.14: Visualisation of Clock Offset and GPS Timestamp Calibration Process

3.6.5.1 Speed Dependency

Driving at 60 km/h means a vehicle covers a distance of 16.6 meters per second under normal driving conditions. With a sampling rate of 50 Hz, every sensor sampling cycle covers approximately 0.33 meters of road. This means that at a 50 Hz sampling rate, we will capture road anomalies larger than 33 cm. At lower speeds, the system will capture more detailed readings. This shows that driving speed is an important factor in our experiments. As a result, there is a direct relationship between the sensor data and the driving speed. In another observation reported by Lindfors et al. (2016), the authors show that there is a strong correlation between vehicle speed and chassis vibrations. Therefore, the accelerometer's reading depends on the vehicle's speed and not just on the road anomaly. In our experiments, the driving speed during the data collection hovered around 25 km/h to 35 km/h.

3.7 Data Fusion

As discussed earlier in this section, the primary aim of this research is to develop a fusion model for the IMU and vision machine learning models. Figure 3.15 shows the methodology used to combine the IMU and vision results. Our solution is based on the GPS atomic time feeds and smartphone synchronisation methods mentioned earlier. The method that we use is based on three distinct processes. In the first process, we stored the images from the vision application and later transferred them to the back-end server for anonymisation. Furthermore, we stored the image timestamp with the image. Next, we stored the MoblenetSSD object detection results. Finally, we extracted the first 100 IMU readings captured after the imagery timestamp to use in the IMU model.

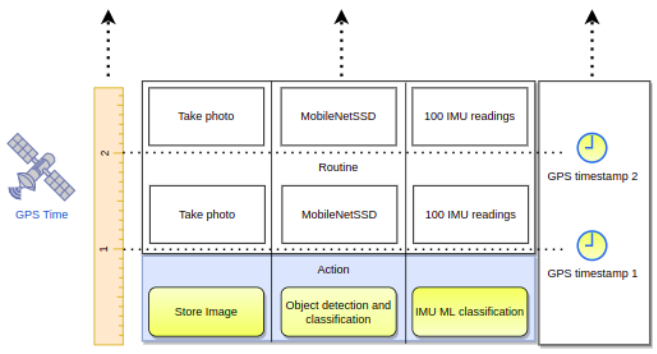


Figure 3.15: Illustration of Fusion Methodology: Integrating Multiple Data Sources for Enhanced Insights

The proposed solution allows us to set boundaries on the captured data and make sure that the models process only the corresponding data from the two models. In Figure 3.16, we demonstrate how this is being achieved. The 100 readings represent a two-second time window at a 50 Hz sampling rate. This translates to roughly ten meters per second when driving at 35 km/h. Therefore, a stretch of 20 meters is being covered for every IMU window. The vision method, however, requires a fair share of calibration to make sure that we only capture roughly 20 meters of the road surface.

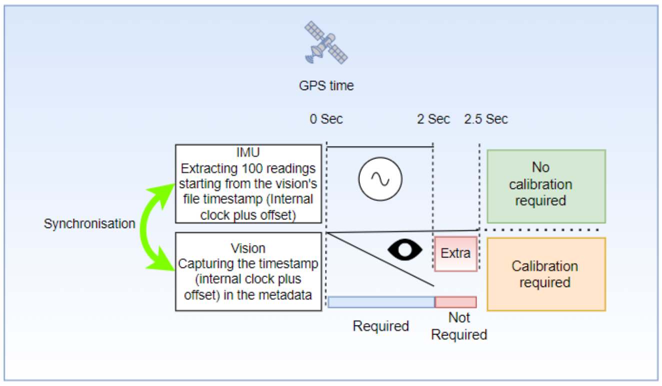

GPS time
0 Sec
2 Sec
2.5 Sec
IMU
Extracting 100 readings starting from the vision's file timestamp (Internal clock plus offset)
Synchronisation
Vision
Capturing the timestamp (internal clock plus offset) in the metadata
Extra
No calibration required
Calibration required
Required
Not Required

Figure 3.16: Illustration of Fusion Methodology: IMU and Vision data capturing techniques

Finally, we package all the information into a simple reporting dashboard where we display all the fused information onto an interactive visual map using Geo-tagging and markers. Figure 3.17 shows the reporting methods used to produce the interactive Geo-location map.

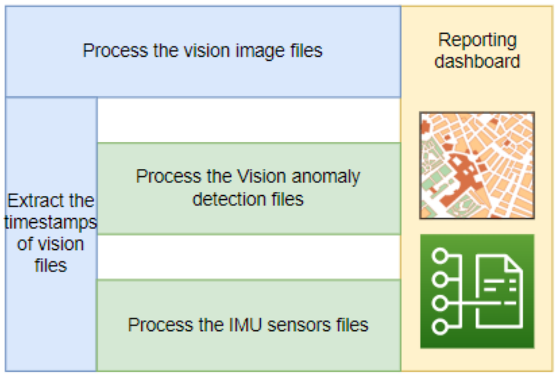

Process the vision image files
Reporting dashboard
Extract the timestamps of vision files
Process the Vision anomaly detection files
Process the IMU sensors files

Figure 3.17: Visualised Reporting Dashboard: A Comprehensive Overview of Key Metrics and Insights

3.8 Summary

In this chapter, we discussed the important attributes which affect the classification performance of our anomaly detection solution. These mainly include the IMU smartphone mounting points, speed, feature extraction and GPS data filtering. We also covered the pre-processing techniques and filters used to improve the learning techniques in the chapter. Next, we covered the three different components of our study, namely: IMU, vision and back-end processes. Consequently, we explained the methodology of all three elements. Also, in this chapter, we explained the different smartphone mounting points for the vision application and their implications. More precisely, the capturing angle and how these impacted the vision models. This chapter also covered another key component in our dissertation, GPS synchronisation. We wrapped up this chapter by providing the methods used to fuse both the IMU and vision data sets to improve and augment the reporting data set.

Chapter 4

Results & Discussion

4.1 Introduction

In this chapter, we will summarise our achieved results, and the experiment processes used to solve the research questions. We begin by summarising the training data set, feature extraction process and a brief description of the machine learning algorithms used. Next, we present the results of our core modules. In section 4.2, we present our IMU data set experiments and results. Because we want to identify the most relevant factors that influence anomalies, we also included in this analysis the top ten features used in the IMU models.

We then move to our vision experiments where in section 4.3, we present our results. Finally, in section 4.4, we outline the data fusion model that blends the prediction of the two models in one reporting dashboard, where we can assess the effectiveness of our models. We conclude this chapter by discussing the data fusion results. In this section, we present the combined results of the model's prediction.

4.2 IMU - Accelerometer and gyroscope machine learning results

As part of our thorough investigation into road irregularities, we collected a comprehensive dataset during the empirical phase of our study. This data set encompassed 85,000 inertia readings, each obtained across five distinct categories, as visualised in Figure 4.1. These categories are structured into classes, namely: depression left (1), depression right (2), depression (3), speed bump (4), and smooth (5). To ensure precision and consistency, we established a uniform sampling rate of 50 Hz, with each annotation encompassing a

2-second interval, yielding a set of 100 inertial readings for each anomaly. In Table 4.1 a summary of the counts per anomaly category is presented, which were aggregated during the initial collection phase.

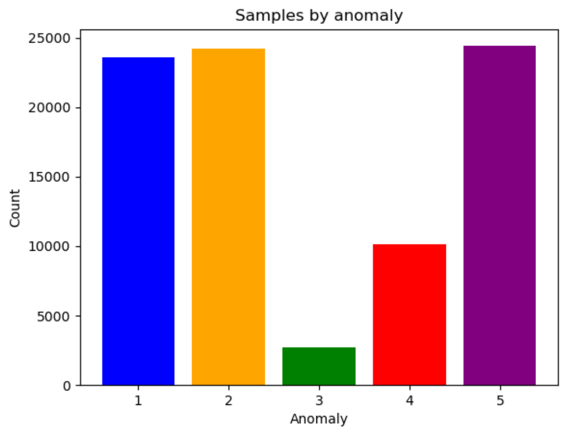


Figure 4.1: Distribution of Anomaly Instances Across Five Distinct Classes

Table 4.1: Anomaly-Specific Sample Counts: Distribution of Samples per Anomaly Category

Anomaly	Sensor readings	Label count
1 - Left Depression	23,600	236
2 - Right Depression	24,200	242
3 - Depression	2,700	27
4 - Speed bump	10,100	101
5 - Smooth	24,400	244

Notably, the depression (3) category, characterised by depressions exceeding the vehicle's wheel track, contained 27 samples, considerably fewer than other classes. This inherent imbalance, coupled with the more substantial sample size of 101 in the speed bump category, led to the emergence of an uneven dataset. Establishing a harmonious equilibrium among the dataset classes holds paramount importance in the construction of precise machine learning models. This undertaking not only enriches data quality and rectifies class imbalances but also improves the model's overall performance and precision. The presence of imbalanced datasets, characterised by minority classes, presents a challenge for widely employed classification techniques, ultimately leading to diminished accuracy.

In order to rectify this disparity, a substantial overhaul was implemented for the depression category. This involved substituting the depression class with an equal distribution of both left and right depression readings. Initially, our experimental approach involved categorising depressions into three distinct classes based on their locations: left, right, and centre of a vehicle. However, upon further consideration, we recognised that the crucial factor lay in accurately identifying potholes themselves rather than their specific locations. Remarkably, empirical testing highlighted an absence of notable distinctions between left and right depressions in our modelling efforts. This revelation prompted the consolidation of all depression instances into a singular class. This consolidation not only achieved a more cohesive data set but also adeptly tackled the challenges arising from class imbalance. It's worth emphasising that with each alteration in classes, we diligently executed a complete machine learning cycle, reconstructing the models based on the freshly curated labelled data sets.

Conversely, the speed bump class unveiled a more complex challenge. Owing to its unique characteristics, augmenting this category to match the smooth class presented inherent difficulties. Consequently, an under-sampling technique emerged as the chosen solution for the smooth class. This technique involved the prudent reduction of the smooth surface training data set while meticulously preserving class representation. By doing so, we re-established an equilibrium, ensuring that each category, including the speed bump class, retained proportional representation.

It is imperative to recognise that the dimensions of our training data set need not be a limiting factor in achieving noteworthy outcomes. Even when dealing with a small training data set, the potential for yielding favourable results remains substantial. It's important to highlight that the quality of data, accompanied by its alignment with pertinent statistical properties, holds far greater significance than the sheer size of the data set. The effectiveness of a machine learning model hinges on the richness, accuracy, and relevance of the information it processes. Therefore, if one pays careful attention to selecting high-quality data with the appropriate statistical attributes, it is possible to achieve exceptional model performance, surpassing the limitations of data set size.

Moreover, our sampling figures are consistent with the work carried out by Chen et al. (2019), who used two small data sets. The first data set included 82 potholes and 17 speed bumps. Consequently, they later added a second set of data, consisting of 236 potholes and 138 speed bumps. Similarly, Silvister et al. (2019) used a data set of 125 labelled potholes in their work to detect potholes using a deep-learning model.

To investigate the impact of the training data set size further, we ran two additional full model run cycles with under-sampling data sets of 50 and 75 records per class. When we compared the results of the three experiments, our findings revealed that there was little difference between the three data sets. However, we could see that the data set with 75 records per class produced a lower score for the multi-class classification. Despite a positive overall result, we were still concerned with the size of our test data and how this will impact our overall test cases. Consequently, we will address this limitation in our future work. We can find the full test results in section D.7.

Figure 4.2 shows the IMU testing phases conducted during our testing cycles. The process included four distinct cycles. First, we planned the test scripts based on our research questions. Next, we focused on selecting the best features to use in our models. The machine learning model's fine tuning followed this, including the result analysis to improve the model's accuracy.

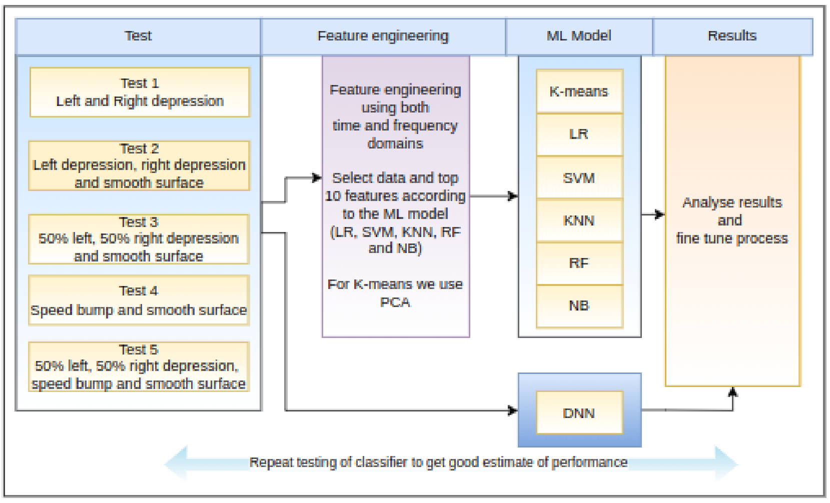

Test
Feature engineering
ML Model
Results
Test 1 Left and Right depression
Test 2 Left depression, right depression and smooth surface
Test 3 50% left, 50% right depression and smooth surface
Test 4 Speed bump and smooth surface
Test 5 50% left, 50% right depression, speed bump and smooth surface
Feature engineering using both time and frequency domains
Select data and top 10 features according to the ML model (LR, SVM, KNN, RF and NB)
For K-means we use PCA
K-means
LR
SVM
KNN
RF
NB
DNN
Analyse results and fine tune process
Repeat testing of classifier to get good estimate of performance

Figure 4.2: Stages of Testing Illustrated: Sequential Phases Of Our Tests

In our testing phases, we used seven ML techniques for our road anomaly detection process. These included:

1. K-means
2. LR
3. SVM
4. KNN
5. RF
6. NB
7. DNN

Items one through six are all based on the feature engineering techniques. We describe these techniques in 4.2.1. The process entailed converting the three accelerometer and gyroscope axes into complex features to aid the machine learning models in their predictions.

Our final IMU sensor test is based on the DNN machine learning technique, which works with raw data. That is, the model uses only the raw accelerometer and gyroscope (X, Y and Z axes) readings. In the final phase, we analysed the results, and adjusted the model structure to improve the accuracy.

4.2.1 Feature extraction

Following the procedures outlined in subsection 3.6.4, we extracted 204 different features from the original six IMU axes.

In their review of machine learning feature selection, El Aboudi and Benhlima (2016) emphasise the importance of minimising the amount of features in the pre-processing phase to improve the classification outcomes. Besides delivering superior outcomes, the feature reduction procedure also lowers the computational cost. Equally important, the authors also highlighted that dimensionality reduction is also critical for reducing overfitting.

Our feature reduction technique is based on the work carried out by Liu et al. (2020). The authors used the SFS to select the significant features to improve the classification accuracy on their time domain accelerometer data.

In our dissertation experiments, we also evaluated the Sequential Backward Selection (SBS) technique, to meticulously evaluate our feature set. The detailed mechanism of these two process is explained upon in the Materials & Method section, precisely under paragraph 3.6.2.1.

Throughout the complex journey of feature selection, we meticulously evaluated the model's performance, harnessing the power of the multi-class accuracy metric. While trying to distinguish between our different classes, including Left and Right depressions, Speed bumps, and Smooth surfaces, we found that the multi-class accuracy metric was the perfect fit. This metric not only encapsulated the complexity of our multi-class classification but also provided a comprehensive framework to gauge the model's effectiveness across the spectrum of our categories.

Table 4.2 presents the sequential feature results based on our collected data and our selected ML algorithms. The results demonstrated that SFS outperformed SBS when considering the specific machine learning techniques and data set utilised.

Table 4.2: Sequential Feature Selector Results: Unveiling Optimal Feature Subsets

Algorithm	**Selector**	**Accuracy score**
LR	**SFS**	**66%**
	SBS	60%
SVM	**SFS**	**64%**
	SBS	54%
KNN	**SFS**	**67%**
	SBS	55%
RF	**SFS**	**65%**
	SBS	56%
NB	**SFS**	**68%**
	SBS	62%

4.2.2 High-pass Butterworth filter

Several studies (Brisimi et al. (2016), Carlos et al. (2018), Cabral et al. (2018)) highlighted the importance of using Butterworth high-pass filters to sanitise the vehicle's IMU data. Results have shown that these filters remove unwanted noise. We carried out a series of tests on our data to determine the optimal high-pass settings. Table 4.3 displays the confusion matrix results that were carried out in this experiment. The frequencies ranged between one hertz and three hertz, whilst the filter order varied between one and five. The tests show that the optimal Butterworth filter parameters are a cutoff frequency of two hertz and with an order of four for LR, SVM, RF and NB. KNN performed better with a frequency of two hertz and an order of five. It was also observed that the high-pass filter had a great impact on the accuracy of the machine learning algorithms.

Table 4.3: Confusion Matrix Results for High-Pass Filter in Our Road Anomaly Detection Experiments

Parameters		LR		SVM		KNN		RF		NB	
		False	True	False	True	False	True	False	True	False	True
Frequency: 2 Hz	False	63	37	61	39	58	42	56	44	41	59
Order: 1	True	37	63	33	67	41	59	37	63	22	78
Frequency: 2 Hz	False	**67**	**33**	59	41	52	48	53	47	34	66
Order: 2	True	**31**	**69**	36	64	42	58	34	66	23	77
Frequency: 2 Hz	False	63	37	53	47	57	43	53	47	45	55
Order: 3	True	33	67	32	68	43	57	32	68	30	70
Frequency: 2 Hz	False	**66**	**34**	**62**	**38**	56	44	**60**	**40**	**58**	**42**
Order: 4	True	**30**	**70**	**29**	**71**	39	61	**33**	**67**	**26**	**74**
Frequency: 2 Hz	False	62	38	57	43	**61**	**39**	58	42	50	50
Order: 5	True	39	61	30	70	**29**	**71**	38	62	31	69
Frequency: 3 Hz	False	62	38	56	44	56	44	**58**	**42**	40	60
Order: 1	True	30	70	31	69	38	62	**32**	**68**	23	77
Frequency: 3 Hz	False	64	36	56	44	59	41	53	47	48	52
Order: 2	True	31	69	36	64	48	52	37	63	30	70
Frequency: 3 Hz	False	58	42	55	45	59	41	55	45	**53**	**48**
Order: 3	True	40	60	44	56	44	56	35	65	**31**	**69**
Frequency: 3 Hz	False	60	40	57	43	45	55	**60**	**40**	59	41
Order: 4	True	38	62	36	64	45	55	**35**	**65**	35	65
Frequency: 3 Hz	False	63	37	55	45	**54**	**46**	**60**	**40**	51	49
Order: 5	True	35	65	26	74	**33**	**67**	**35**	**65**	28	72
No filter	False	*65*	*35*	*59*	*41*	*54*	*46*	*57*	*43*	*52*	*48*
	True	*32*	*68*	*29*	*71*	*35*	*65*	*36*	*64*	*33*	*67*

4.2.3 Test one: K-means clustering - Understanding our data

The unsupervised K-means clustering model serves as the foundation for our first machine learning exam. The K-means clustering algorithm is useful when attempting to understand the relationships and similarities among categorical data. Based on how the categories perform on a set of variables, the algorithm generates a set of groups known as clusters.

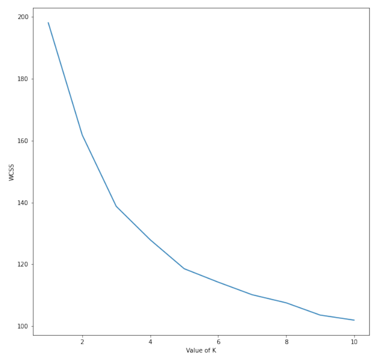
To reduce the 204 manually engineered features and improve the accuracy of the task, we used a Principal component analysis (PCA) dimensionality reduction technique similar to the work conducted by Mishra et al. (2011) to minimise loss of information.

The choice of the ideal number of clusters to map the complete data set is a vital step in developing a K-means clustering model. The elbow method is a straightforward method employed to calculate the value of K. In this method, we calculate the distortion score of each K parameter by applying the K-means clustering algorithm to a set of K values. However, the elbow results in our testing to determine the ideal value were inconclusive.

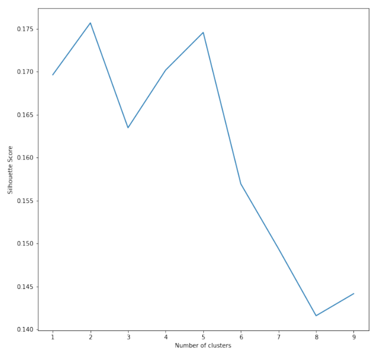
On the other hand, a more systematic approach used to determine the k value is the silhouette method approach, as shown in Shahapure and Nicholas (2020). When the silhouette coefficient value is close to a positive one, it shows that the data point belongs to the right cluster. A coefficient close to a negative one shows the data point is in the incorrect cluster, while values near zero suggest that it may belong to another cluster.

In Figure 4.3, we can see the two clusters based on the left and right road anomalies. As we can see in Figure 4.3 (a), the elbow output does not clearly define the optimal number of K clusters for our data. However, we can observe that the silhouette method shown in Figure 4.3 (b) identified two clusters for the left and right road depressions. Nonetheless, the coefficient value of the highest point hovered around 0.175, which is considered too low. It also reflected the same pattern in the k-means data plot, as shown in figure 4.3 (c). This test shows that it is highly challenging to distinguish between the left and right depressions.

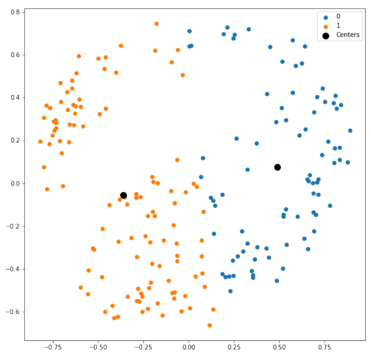
We then combined the smooth data with the left and right depressions for the subsequent test. The elbow result produced modest results, as shown in figure 4.4 (a). In the silhouette exercise (figure 4.4 (b)), we could observe that the smooth data impacted the exercise outcome. The highest coefficient value is 0.24 for two clusters, followed by four clusters with a coefficient of 0.228 and three clusters with a value of 0.22. This is an improvement on the previous left and right depression test, which returned a coefficient of 0.175. However, the exercise reveals that the optimal number of clusters is two and not three, as we were expecting. Figure 4.4 (c) illustrates the final k-means plot with three clusters representing left and right depression, and smooth data.



(a) Elbow

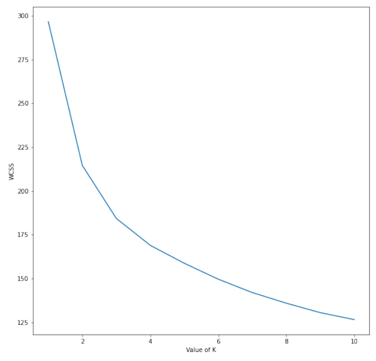


(b) Silhouette

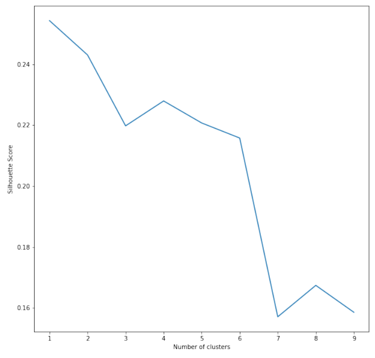


(c) K-means Clustering

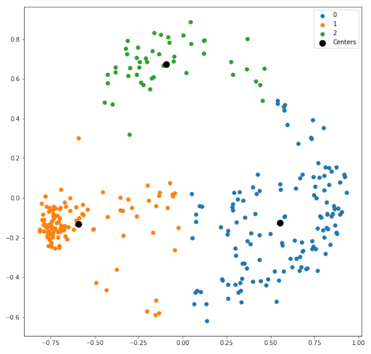
Figure 4.3: K-means - Left and Right potholes



(a) Elbow



(b) Silhouette



(c) K-means Clustering

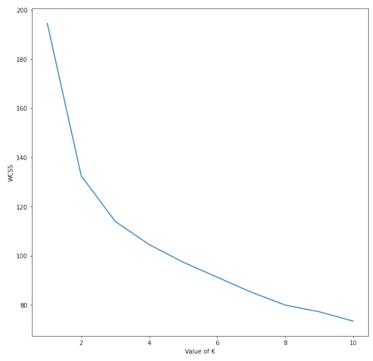
Figure 4.4: K-means - Left potholes, right potholes and smooth

We compared category three road depression to smooth data in the following experiment. We only had 2,400 inertial readings in this category, as described in Section 4.1. To keep the data set balanced, we replaced the depression data with an equal number of left and right depression records. Once again, the silhouette provided crucial information as the elbow method failed to determine the optimal number of clusters, as shown in Figure 4.5 (a). Figure 4.5 (b) reveals a coefficient of nearly 0.27 for two clusters and a coefficient of 0.26 for three clusters. Figure 4.5 (c) illustrates the final K-mean plot which resembles the previous left, right depression and smooth figure (Figure 4.4 (c)).

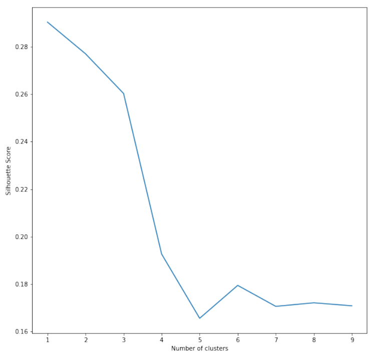
Even though the data is a subset of the left and right depression, we could observe that the silhouette coefficient was higher in this test than in the prior depression test.

Our next investigation covered the speed bump category. The speed bump anomaly is a distinct irregularity, as automobiles frequently strike speed bumps with both front wheels at the same time.

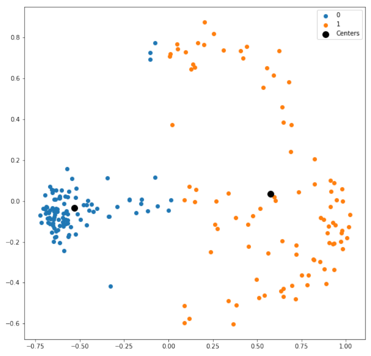
In this test, we could observe that the k-means elbow test performed achieved supe-



(a) Elbow



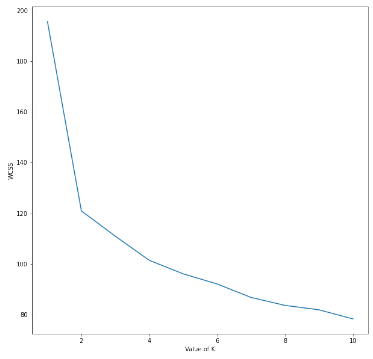
(b) Silhouette



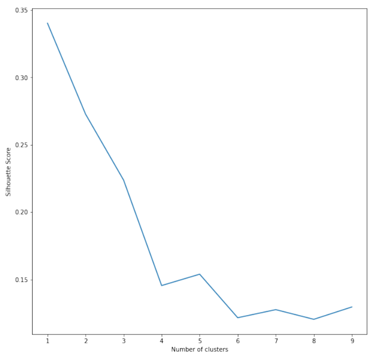
(c) K-means Clustering

Figure 4.5: K-means - Left and right grouped potholes - Smooth

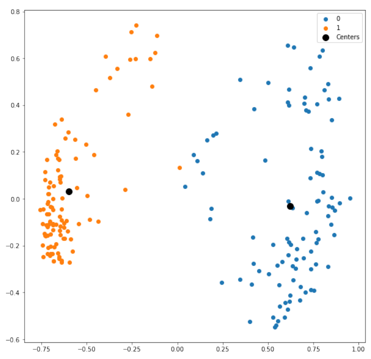
rior result by identifying two clusters, as shown in Figure 4.6 (a). The silhouette method returned a coefficient of 0.275 for two clusters (Figure 4.6 (b)). As shown in Figure 4.6 (c), the k-means plot exhibits an almost perfect two-node cluster for this data set.



(a) Elbow



(b) Silhouette

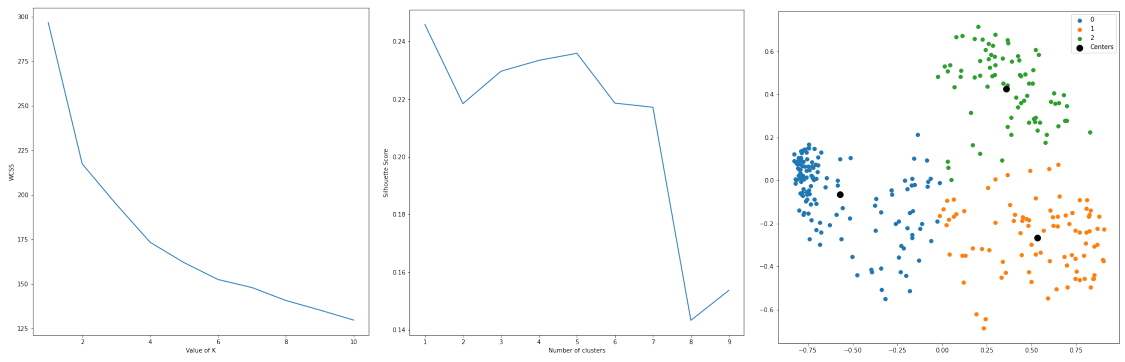


(c) K-means Clustering

Figure 4.6: K-means - Speed bumps and Smooth

In our final K-means test, we tested the road depressions, speed bumps and smooth categories to evaluate the clustering capabilities and also better understand the data set structure by visualising the data points.

Once again, we could observe that the elbow method did not perform so well under this clustering scenario (figure 4.7 (a)). Figure 4.7 (b) illustrates that the silhouette method produced coefficients ranging between 0.23 and 0.235 for clusters of '3', '4' and '5' (depression, speed bump and smooth). Finally, the k-means plot proves that there is a strong clustering formation for the three selected classes as shown in figure 4.7 (c)



(a) Elbow (b) Silhouette (c) K-means Clustering

Figure 4.7: K-means - Left and right potholes grouped, speed bump and Smooth

4.2.4 Supervised IMU machine learning techniques

In this section, we present the test results for the supervised machine learning techniques for the accelerometer and gyroscope data.

As mentioned in subsection 2.3.3, the three gyroscope axes measure the following vehicle rotation characteristics:

- The Z (roll) gyroscope senses the vehicle rolling left and right.
- The X (pitch) gyroscope senses the vehicle going up and down.
- The Y (yaw), veering left and right.

4.2.5 Test case two: Depression left VS Depression right

Description

In our first supervised test, we retested the left and right depression categories. Our primary aim is to determine whether supervised techniques can distinguish between both left and right depressions. We conducted this experiment using binary classification. We achieved this by substituting the right depression with a ’0’ class and the left data set with a binary ’1’ class.

Results and discussions

Table 4.4 shows that the findings support a similar conclusion for the K-means clustering outcome, with no observable distinction between the left and right depressions. With an accuracy of 0.69 and a precision, recall, and F1 score that ranged between 0.66 and 0.69, LR produced the best results. We provide the information in a confusion matrix

Table 4.4: Test case two. Left VS Right test - (1) left, (0) Right

Algorithm	Accuracy	Class	Precision	Recall	F1 score
LR	0.69	0	0.69	0.66	0.68
		1	0.68	0.71	0.69
SVM	0.47	0	0.46	0.39	0.42
		1	0.47	0.55	0.51
KNN	0.47	0	0.47	0.49	0.48
		1	0.46	0.44	0.45
RF	0.49	0	0.49	0.50	0.50
		1	0.49	0.48	0.48
NB	0.48	0	0.48	0.41	0.44
		1	0.49	0.56	0.52

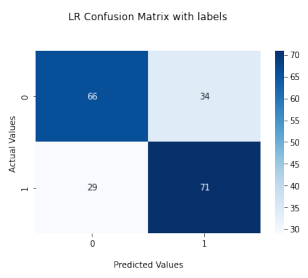


Figure 4.8:
Test case two
Left vs right depression
confusion matrix

format, as shown in Table 4.8 The top ten features chosen by the sequential feature selector for each of the five machine learning models are one intriguing finding from these tests. Based on the cross-validation procedure, the estimator selected the optimal feature to include. The ten leading features, including the sensors utilised, the axis, and the frequency domain, are featured in Table 4.5. Our results at least provide a strong indication that the gyroscope sensor could detect roll motions on the axes, especially the X and Z axes, which in our situation correspond to the roll and pitch axes.

Test conclusion

The findings reveal that the proposed solution does not distinguish between left and right depression.

Table 4.5: Left (1) and right (2) depression - LR features used

Feature name	Sensor	Axis	Feature description	Domain
xg_maxmin_diff	G	X	Max-Min difference	Time
zg_maxmin_diff	G	Z	Max-Min difference	Time
zg_mad	G	Z	Median absolute deviation	Time
za_IQR	A	Z	Interquartile range	Time
yg_peak_count	G	Y	Number of Peaks	Time
za_kortosis	A	Z	Kurtosis	Time
zg_kortosis	G	Z	Kurtosis	Time
zg_max_fft	G	Z	Maximum	Freq.
ya_IQR_fft	A	y	Interquartile range	Freq.
xg_peak_count_fft	G	x	Number of peaks	Freq.

Appendix D.1 presents all the features used in the SVM (Table D.1), KNN (Table D.2), RF (Table D.3) and NB (Table D.4) tests. Furthermore, we present the corresponding confusion matrices in Figure D.1 and Figure D.2

4.2.6 Test case three: Left Depression, right Depression and Smooth surface

4.2.6.1 Description

In our next test, we conducted an experiment on the left and right depression and smooth surfaces. The aim of this evaluation is to test the depression data set against the smooth surface data.

4.2.6.2 Results and discussions

Our findings are summarized in Table 4.6, where logistic regression demonstrated the highest accuracy score of 0.76. Notably, this represents a significant improvement of seven points compared to the previous test. Table 4.8 provides the data in a confusion matrix format. Label ‘0’ in the matrix represents class ‘1’ (left depression), label ‘1’ represents class ‘2’ (right depression), and label ‘3’ represents class ‘5’ (smooth).

There were minor changes in precision, recall, and F1 score of the left and right depression. The smooth data set improved the final accuracy score. We expected this result since the smooth data set has minimal accelerometer and gyroscope movements. The k-means result also confirmed this data trend. Therefore, we can conclude that it is possible to detect depression and a smooth surface.

Table 4.6: Test case three.
Left (1), Right (2) and Smooth (5)

Algorithm	Class	Precision	Recall	F1 Score	Accuracy
LR	1	0.65	0.68	0.67	0.76
	2	0.65	0.62	0.64	
	5	0.98	0.99	0.99	
SVM	1	0.53	0.59	0.56	0.68
	2	0.53	0.46	0.49	
	5	0.99	1.00	1.00	
KNN	1	0.54	0.53	0.53	0.67
	2	0.53	0.49	0.51	
	5	0.93	1.00	0.96	
RF	1	0.54	0.44	0.48	0.67
	2	0.52	0.61	0.56	
	5	0.96	0.97	0.97	
NB	1	0.48	0.53	0.50	0.65
	2	0.49	0.45	0.47	
	5	1.00	0.98	0.99	

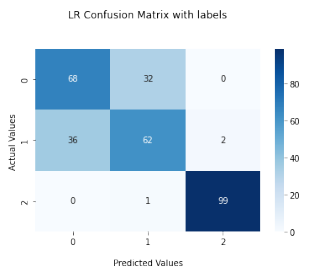


Figure 4.9: Test case three. Left, right depression and smooth confusion matrix

From the LR features used in Table 4.7, we can observe that there were some notable changes. The gyroscope was once again the primary element that affected the results. However, we also noticed that the top ten features differed from the previous left and right depression test. We could also notice that the frequency domain was more prevailing in this test.

Table 4.7: LR Features used

Feature	Sensor	Axis	Feature	Domain
yg_mad	G	Y	Median absolute deviation	Time
xa_above_mean	A	X	Values above mean	Time
avg_result_accl	G	X, Y and Z	Average resultant	Time
smaa	G	Z	Signal magnitude area	Time
xg_var_fft	G	X	Variance	Freq.
zg_var_fft	G	Z	Variance	Freq.
ya_peak_count_fft	A	Y	Number of peaks	Freq.
xg_peak_count_fft	G	X	Number of peaks	Freq.
zg_peak_count_fft	G	Z	Number of peaks	Freq.
zyZscoreRatio	A	Z and Y	ZScore	Freq.

Appendix D.2 presents all the features used in the SVM (Table D.5), KNN (Table D.6), RF (Table D.7) and NB (Table D.8) tests. Furthermore, we present the corresponding confusion matrices in Figure D.3 and Figure D.4

4.2.6.3 Test conclusion

The findings show the same patterns as in test case two, with no noticeable difference between left and right depression.

4.2.7 Test case four: 50% left and 50% right depression grouped and smooth surface

4.2.7.1 Description

We test a combination of 50% left and 50% right depression against a smooth surface. In this test, we used a combination of left and right depression transactions to replace the depression anomaly larger than the vehicle's wheel track. As a result, we kept the data balanced between all classes. We used a binary classification, relabelling the depression to '0' and smooth as '1'.

4.2.7.2 Results and discussions

We present the results in Table 4.8. All five techniques returned high accuracy, ranging from 0.97 to 1.00, with NB scoring the best accuracy. This test demonstrates we can identify road depressions in a simple binary environment. Table 4.12 provides the data in a confusion matrix format.

Table 4.8: Test case four. Left & Right (0) and smooth (1)

Algorithm	Class	Precision	Recall	F1 score	Accuracy
LR	0	0.94	1.00	0.97	0.97
	1	1.00	0.94	0.97	
SVM	0	0.99	1.00	1.00	0.99
	1	1.00	0.99	0.99	
KNN	0	0.94	1.00	0.97	0.97
	1	1.00	0.94	0.97	
RF	0	0.99	0.99	0.99	0.99
	1	0.99	0.99	0.99	
NB	0	1.00	1.00	1.00	1.00
	1	1.00	1.00	1.00	

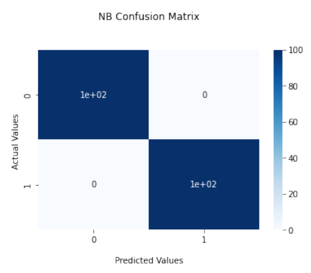


Figure 4.10:
Test case four.
Combined left, right depression, and smooth
confusion matrix

From the NB's top ten selected features shown in Table 4.9, we could observe that the sequential forward selection selected both the accelerometer and gyroscope sensors for all the three axes. Furthermore, the process selected the features from the time domain.

Table 4.9: Test case four. Naive Bayes Features used

Feature	Sensor	Axis	Feature	Domain
xa_mean	A	X	Mean	Time
ya_mean	A	Y	Mean	Time
za_mean	A	Z	Mean	Time
xg_mean	G	X	Mean	Time
yg_mean	G	Y	Mean	Time
zg_mean	G	Z	Mean	Time
xa_var	A	X	Variance	Time
ya_var	A	Y	Variance	Time
za_var	A	Z	Variance	Time
smaa	A	X and Y	Signal magnitude area	Time

Appendix D.3 presents all the features used in the LR (Table D.9), SVM (Table D.10), KNN (Table D.11) and RF (Table D.12) tests. Furthermore, we present the corresponding confusion matrices in Figure D.5 and Figure D.6

4.2.7.3 Test conclusion

The test results revealed that the proposed model can correctly identify depressions. Furthermore, we could not distinguish between left and right depression in the preceding two tests, indicating a fault in our method.

4.2.8 Test case five: Speed bump and Smooth surface

4.2.8.1 Description

With the depression data test completed, we then moved to analyse the speed bump data set. For this test, we compared the speed bumps with smooth surfaces. We once again used a binary classification to compare both data sets with the speed bump labelled as ‘0’ and the smooth data set as ‘1’.

4.2.8.2 Results and discussions

Table 4.10 displays the achieved results, where all five algorithms have returned high accuracy scores. We expected this level of accuracy since we were comparing two discreet sets of data sets with minimal sensor movements on the smooth side and high sensor activity on the accelerometer device. Table 4.12 provides the data in a confusion matrix format.

Table 4.10: Test case five. Speed bump and smooth

Algorithm	Class	Precision	Recall	F1 score	Accuracy
LR	0	0.97	1.00	0.99	0.98
	1	1.00	0.97	0.98	
SVM	0	0.99	1.00	1.00	0.99
	1	1.00	0.99	0.99	
KNN	0	0.94	1.00	0.97	0.97
	1	1.00	0.94	0.97	
RF	0	1.00	1.00	1.00	1.00
	1	1.00	1.00	1.00	
NB	0	1.00	1.00	1.00	1.00
	1	1.00	1.00	1.00	

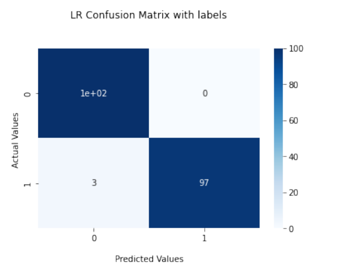


Figure 4.11:
Test case five.
Speed bump and smooth confusion matrix

One intriguing finding that we observed in this experiment was that the features selected for this speed bump test resembled the previous depression versus a smooth surface test. Tables 4.11 and 4.12 list the ten features used for the speed bump vs smooth test for RF and NB, respectively. One can see this resemblance in tables D.13 and 4.9

Appendix D.4 presents all the features used in the LR (Table D.13), SVM (Table D.14) and KNN (Table D.15) tests. Furthermore, we present the corresponding confusion matrices in Figure D.7 and Figure D.8

Table 4.11: Test case five. Random Forest Features used

Feature	Sensor	Axis	Feature	Domain
xa_mean	A	X	Mean	Time
ya_mean	A	Y	Mean	Time
za_mean	A	Z	Mean	Time
xg_mean	G	X	Mean	Time
yg_mean	G	Y	Mean	Time
zg_mean	G	Z	Mean	Time
xa_var	A	X	Variance	Time
zg_var	G	Z	Variance	Time
za_std	A	Z	Standard deviation	Time
xa_aad	A	X	Average absolute difference	Time

Table 4.12: Test case five. Naive Bayes Features used

Feature	Sensor	Axis	Feature	Domain
xa_mean	A	X	Mean	Time
ya_mean	A	Y	Mean	Time
za_mean	A	Z	Mean	Time
xg_mean	G	X	Mean	Time
yg_mean	G	Y	Mean	Time
zg_mean	G	Z	Min	Time
xa_var	A	X	Variance	Time
ya_var	A	Y	Variance	Time
za_var	A	Z	Variance	Time
smaa	G	X and Y	Signal magnitude area	Time

4.2.8.3 Test conclusion

The results show that our proposed model can detect speed bumps with high accuracy. Furthermore, we can see a strong similarity between the selected features and the depression model in the previous test.

4.2.9 Test case six: Depression, speed bump and smooth surface

4.2.9.1 Description

In our final IMU supervised machine learning experiment, we merged the depression, speed bump, and smooth surface data sets into one test to simulate a real-life driving journey.

4.2.9.2 Results and discussions

Table 4.13 shows the result of our tests.

LR returned the best accuracy rate with a score of 0.88. We observed that the precision rate of the depression and speed bump were high too at 0.86 and 0.83 respectively, with a recall value of 0.79 and 0.85. The F1 score reached 0.82 and 0.84, respectively. The smooth surface's precision, recall, and F1 score once again returned high scores: 0.95, 1.00 and 0.98.

SVM also returned a satisfying score of 0.82. However, the precision score for the depression and speed bump was much lower. The high accuracy score was mainly the result of a high-precision result for the smooth class.

Table 4.12 provides the data in a confusion matrix format. Label '0' in the matrix represents class '3' (depression), label '1' represents class '4' (speed bump) and label '3' represents class '5' (smooth).

Table 4.13: Test case six. Depression, speed bump and smooth surface

Algorithm	Class	Precision	Recall	F1 Score	Accuracy
LR	3	0.86	0.79	0.82	0.88
	4	0.83	0.85	0.84	
	5	0.95	1.00	0.98	
SVM	3	0.71	0.77	0.74	0.82
	4	0.76	0.68	0.72	
	5	0.99	1.00	1.00	
KNN	3	0.72	0.69	0.70	0.80
	4	0.72	0.71	0.71	
	5	0.95	1.00	0.98	
RF	3	0.62	0.64	0.63	0.75
	4	0.65	0.65	0.65	
	5	0.99	0.96	0.97	
NB	3	0.68	0.69	0.69	0.79
	4	0.69	0.70	0.70	
	5	1.00	0.98	0.99	

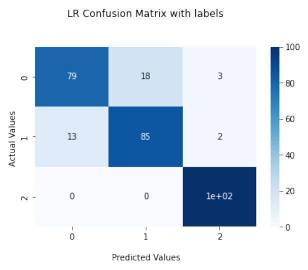


Figure 4.12: Test case six. Depression, speed bump and smooth confusion matrix

Table 4.14 illustrates the features selected for the LR algorithm. This test revealed one noteworthy finding that applies to the features chosen for the categorisation exercise. Instead of the accelerometer data, the top three attributes were based on the gyroscope sensor.

We also detected similar patterns in the other algorithms, as shown in tables: D.16, D.17, D.18, and D.19 in Appendix D.

This shows that the gyroscope sensor played an important role in detecting depressions and speed bumps.

Appendix D.6 presents all the features used in the SVM (Table D.16), KNN (Table D.17), RF (Table D.18) and NB (Table D.19) tests. Furthermore, we present the corresponding confusion matrices in Figure D.9 and Figure D.10

Table 4.14: Test case six. LR Features used

Feature	Sensor	Axis	Feature	Domain
zg_mean	G	Z	Mean	Time
yg_median	G	Y	Median	Time
zg_median	G	Z	Median	Time
xa_IQR	A	X	Interquartile range	Time
ya_neg_count	A	Y	Negative count	Time
xa_above_mean	A	X	Values above mean	Time
avg_result_accl	A	X, Y and Z	Average resultant	Time
smaa	G	X, Y and Z	Signal magnitude area	Time
za_mean_fft	A	Z	Variance	Freq.
xg_maxmin_diff_fft	G	X	Max-Min difference	Freq.

4.2.9.3 Test conclusion

The results show our model can distinguish between depression, speed bump, and smooth with high accuracy, precision, and recall.

4.2.10 Test case seven: Testing the DNN algorithm

4.2.10.1 Description

The final test we carried out on the IMU data sets is based on a deep neural network model similar to the one used by Silvister et al. (2019)

The DNN architecture consisted of the first visible layer for the inputs, four hidden layers and, an output layer. The visible layer has six neurons for the six sensor axes, respectively. For the first hidden layer, we added 128 neurons. Following various test iterations, we added two more hidden layers of 196 and 64 neurons, respectively. For the final hidden layer, we applied a layer of 32 neurons. Table 4.15 presents the DNN architecture in a table format.

We used a Rectified Linear Unit Activation (ReLU) function at the hidden layers and a Sigmoid function at the output layer as the activation function.

4.2.10.2 Results and discussions

We present the five test results for the ANN model in Table 4.16. In our first test, we could observe a slight improvement in the left and right depression classification. However, the accuracy was still trailing behind the traditional machine learning techniques.

As expected, the third and fourth tests performed well with both the grouped left and right depressions and speed bump, and a 100% accuracy against a smooth surface.

Table 4.15: DNN architecture

Layer	Type	Output Shape	Activation	Param
$\text{dense}_1 2$	Dense	6	ReLU	42
$\text{dense}_1 3$	Dense	64	ReLU	448
Batch Normalization	Batch Normalization	64		256
$\text{dense}_1 4$	Dense	128	ReLU	8,320
$\text{dense}_1 5$	Dense	196	ReLU	25,284
$\text{dense}_1 6$	Dense	64	ReLU	12,608
$\text{dense}_1 7$	Dense	6	ReLU	390

In our depression, speed bump and smooth test, our neural network got an accuracy of 0.82. This result is comparable to the previous SVM result shown in Table 4.13. Furthermore, precision, recall, and F1 Score are all comparable to the previous test.

Table 4.16: Test case seven. Depression, speed bump, Smooth

Class	Precision	Recall	F1 Score	Accuracy
1	0.62	0.67	0.64	0.62
2	0.63	0.58	0.61	
1	0.55	0.70	0.61	0.71
2	0.60	0.43	0.48	
5	0.99	1.00	1.00	
3	1.00	1.00	1.00	1.00
5	0.99	1.00	1.00	
4	1.00	1.00	1.00	1.00
5	1.00	1.00	1.00	
3	0.73	0.75	0.74	0.82
4	0.74	0.72	0.72	
5	0.99	1.00	1.00	

4.2.10.3 Test conclusion

Surprisingly, the DNN model came in second place for our assessment of depression, speed bumps, and smoothness. The result matches the SVM feature engineered model.

4.3 Vision results

4.3.1 Test case eight: Testing various vision models

4.3.1.1 Description

In this test, we compare the smartphone enabled MobilenetSSD model to traditional object detection models that require more computer resources, such as YOLOv3, YOLOv4, EfficientDet D0, and Faster R-CNN.

4.3.1.2 Results and discussions

Table 4.17 shows how the six implementations perform against the raw single class data set, multi class (MC) data set, and single class data set excluding large pothole images (SC-EVL) as mentioned in subsection 3.5.1. Both the YOLOv5 and Mobilenet v2 performed better on the raw dataset with 4.9% and 7.5% difference, respectively. YOLOv3, YOLOv4 and EfficientDet D0 produced better accuracy results on the SC-EVL configuration. The results also show that, for YOLOv3 and YOLOv4, there was a minimal improvement between the accuracy of the Raw data and SC-EVL. In contrast, there was an increase of 16.5% accuracy for the EfficientDet D0 - SC-EVL technique.

On the other hand, Faster R-CNN produced poor results under all three tests. The table sorting is based on the experiment order, with the bold readings represent the highest precision for the three runs per method. The poor performance multi-class test results can be attributed to the class annotation process. It is difficult to label and classify the potholes because of the different photo angles and perspectives. We could easily have misclassified small, medium, large, and oversized potholes during the labelling process, affecting overall accuracy.

Table 4.17: Vision test results - Single class

Model	Dataset	Raw mAP.50	MC mAP.50	SC -EVL mAP.50
YOLOv3 Pytorch	RoboFlow DS	70	39.7	**73**
YOLOv4 Pytorch	RoboFlow DS	77.3	31.97	**77.35**
EfficientDet D0 TF2	RoboFlow DS	52.17	54.49	**68.69**
Faster R-CNN TF1.5	RoboFlow DS	21.8	**26.6**	21.05
Mobilenet SSD v2 TF1.5	RoboFlow DS	**60.3**	45.04	52.77

Table 4.18 exhibits the results sorted by the accuracy performance per test. Our findings on the MC data show that the object detection accuracy is much lower compared with the other data sets. We speculate that this might be due to incorrectly annotated data. As a result, it impacted the anomaly detection. YOLOv4 and YOLOv3 produced the

best results for both the raw and SC-EVL images, with YOLOv4 scoring 7.3 points more than YOLOv3 on the raw data and 4.35 points more SC-EVL.

Remarkably, YOLOv4 produced the same results for both the raw and SC-EVL configurations. On the other hand, YOLOv3 achieved a 3% better accuracy on the SC-EVL dataset compared to the raw dataset.

Table 4.18: Vision test results - Multi-class

Model	Raw mAP.50	Model	MC mAP.50	Model	SC -EVL mAP.50
YOLOv4	77.30	EfficientDet D0	54.49	YOLOv4	77.35
YOLOv3	70.00	Mobilenet SSD v2	45.04	YOLOv3	73.00
Mobilenet SSD v2	60.30	YOLOv3	39.70	EfficientDet D0	68.69
EfficientDet D0	52.17	YOLOv4	31.97	Mobilenet SSD v2	52.77
Faster R-CNN	21.80	Faster R-CNN	26.60	Faster R-CNN	21.05

4.3.1.3 Test conclusion

When testing raw and SC-EVL labelled data, our tests show that the YOLO family outperformed the MobileNetSSD model. In the MC data set, however, we found the MobileNetSSD model outperformed the YOLO models. Furthermore, we discover that MobileNetSSD surpasses Faster R-CNN in all tests. Finally, we observed that MobileNetSSD had a mAP of only 52.77% in the SC-EVL class. Despite the lower performance in the SC-EVL classification, we believe MobileNetSSD has potential given the limited computer power required to detect objects.

4.3.2 Test case nine: MobileNetSSD evaluation using a custom anomaly data set

4.3.2.1 Description

As described in the penultimate paragraph in subsection 3.5.1, we revised the training data set. This was required to match the image set to our new camera position. Finally, we created a new data set with 1891 images. We took the images during good weather condition with no shadow from surrounding objects. The data set contained no waterlogged potholes. Furthermore, we added speed bumps and cracks, which included depressions. We asked a third party to annotate the images. As a result, we acknowledge that there might be some bias in the data set. Finally, we divided these into 1,500 training images, 257 validation images, and 134 training images.

4.3.2.2 Results and discussions

Table 4.19 displays the mAP and recall results for the second data set used for the live testing after 120k training steps.

Table 4.19: Pothole, crack and speed bump

Model	Metric	Percentage	Number of training steps
Mobilenet SSD v2	mAP.50	52.30	120,000
	Recall	36.50	120,000

4.3.2.3 Test conclusion

Notably, the new data set's findings back up the SC-EVL's findings in test case seven. We believe that the 0.523 mAP and 0.365 recall scoring make the object detection procedure an interesting solution to our research topic because we may execute the model directly on smartphone devices.

4.4 Data fusion - Field testing results

4.4.1 Test case ten: Data fusion and field testing result

4.4.1.1 Description

We linked the IMU and Vision data streams via a custom-built web application that forms the core of the data fusion module, as shown in Figure 4.13. We can find the web page at: http://maltaroadanomaly.info:8080/map.

The GPS atomic clock is the primary structural link between the two data sets. The primary features of the application are the independent IMU and vision results after applying our anomaly detection machine learning algorithms. Moreover, we included the fused IMU and vision results to augment the reporting data set. We can customise this capability using the legend menu at the bottom of the HTML page, as shown in Figure 4.14

The markers connected to the legend will appear when a user selects the components in the legend. In addition, when a user clicks on a marker, a pop-up window containing a picture of the detected anomaly appears. We divided the IMU part into three sections:

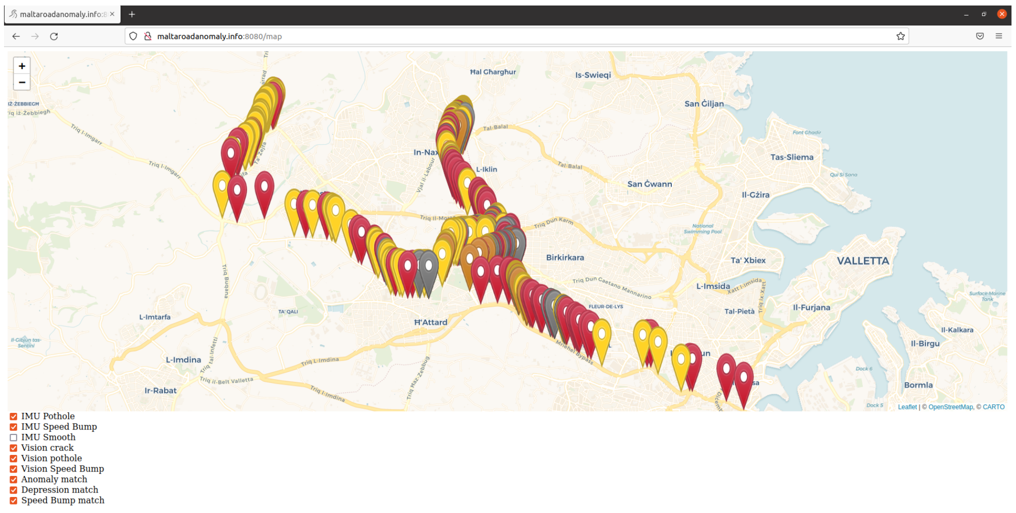


Figure 4.13: Visualising Anomaly Maps: Experimental Results and Anomaly Distribution

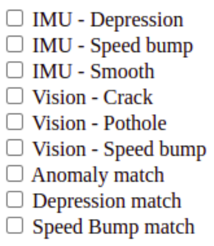

☐ IMU - Depression
☐ IMU - Speed bump
☐ IMU - Smooth
☐ Vision - Crack
☐ Vision - Pothole
☐ Vision - Speed bump
☐ Anomaly match
☐ Depression match
☐ Speed Bump match

Figure 4.14: Legend for the Data Fusion Website Interface: Visual Guide to Iconography and Features

depression, speed bump, and smooth surface. In Figure 4.15, we display the three major road anomaly categories used in our study.

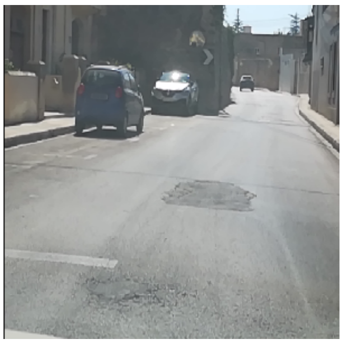
Table 4.20 summarises the IMU sensing anomalies according to the anomaly classes. During our field testing exercise, we captured 1,003 IMU data points and corresponding images.

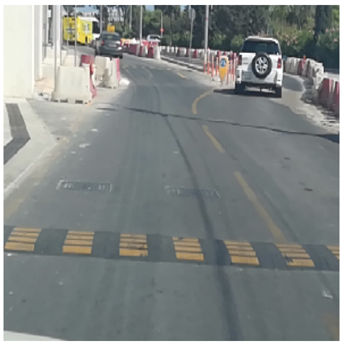
4.4.1.2 Results and discussions

We observed some false positive detection in the IMU classification during our first observation of the classification process. We present samples of these incorrectly classified data points in Figure 4.16.

Correspondingly, there are three subsections in the vision section: crack, pothole, and speed bump. The markers are colour coded to make navigation easier for the user.

Despite the relatively high numbers of IMU depression and speed bump anomalies,


(a) Depression


(b) Speed Bump

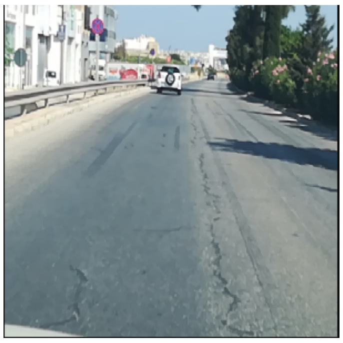

(c) Crack - Vision classification

Figure 4.15: Field Testing Results: Visual Examples of Speed Bump, Depression, and Crack Anomalies

Table 4.20: Fusion IMU data

Anomaly	Count
Depression	281
Speed bump	158
Smooth	564

the vision module returned conservative results. Table 4.21 reveals the deteriorated anomaly detection results. The vision model detected 115 anomalies.

Table 4.21: Fusion Vision test results

Anomaly	Count
Crack	37
Pothole	40
Speed bump	38

Navigating through the vision section, we could observe some false positives for the speed bump classification. In Figure 4.17, we display three images that were incorrectly predicted as speed bumps.

We speculate that this might be because of the close resemblance of the speed bump training data. The pedestrian crossing, water culvert, and the tree shadow all resemble some of the training speed bump images as seen in Figure 4.18.

We dedicate the final section of the legend to IMU and vision recognition matching. In other words, we compare the prediction of the IMU and vision models based on the same

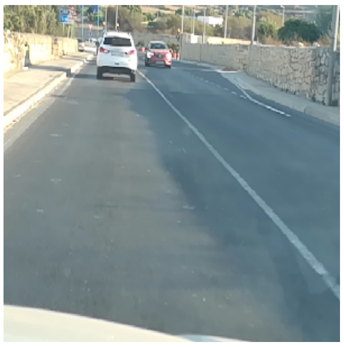
(a) Road trenching sample 1

(b) Road trenching sample 2

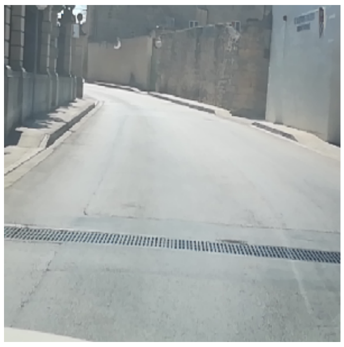
(c) Road trenching sample 3

(d) Road trenching sample 4

(e) Road trenching sample 5


(f) Asphalt joint

(g) Aggressive breaking


(h) Water culvert

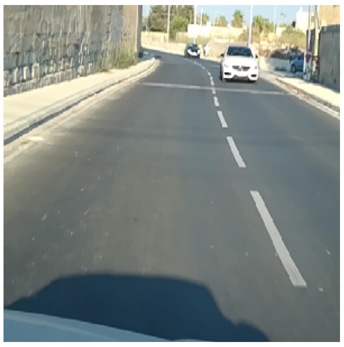
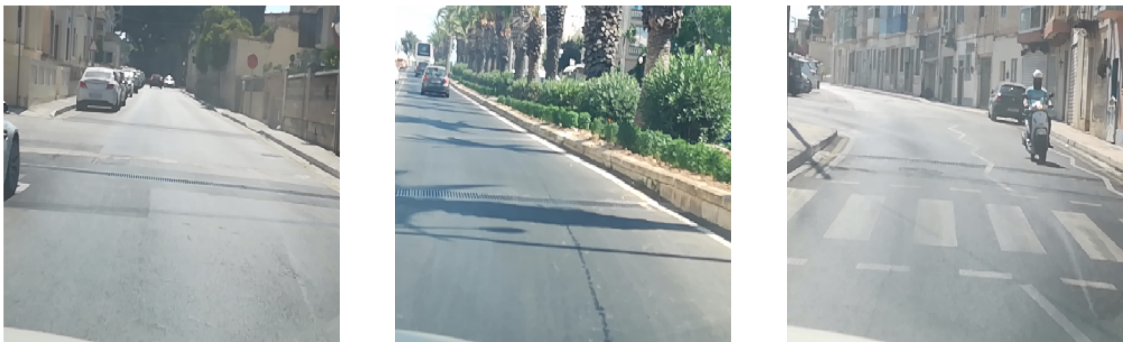

(i) Road trenching sample 6

Figure 4.16: Illustrating Instances of False Positive Speed Bump Detection

timestamp. This means the output of IMU after a two-second sensing time window, and the vision module prediction of the image captured at the beginning of the IMU sensing. There are three options in this section: Anomaly, Depression, and Speed Bump match.

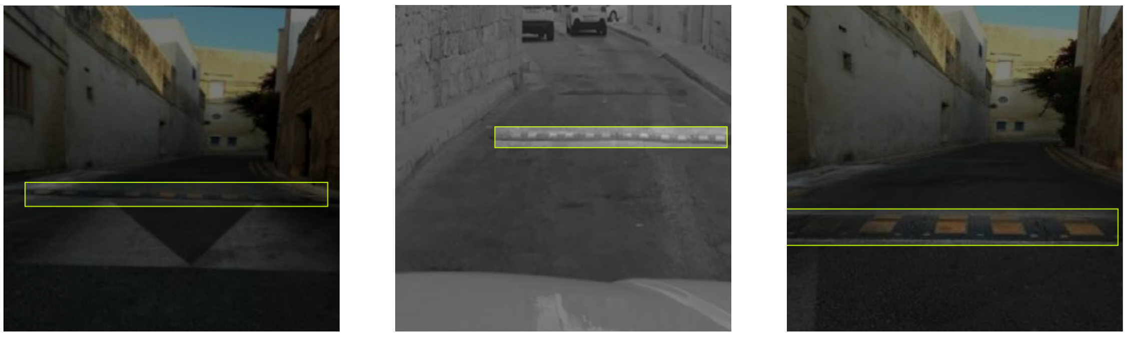
The anomaly match legend displays all points where the IMU and vision detected a dual anomaly. In other words, depression, speed bump, or crack. In the depression match section, we display the matched IMU and vision depressions. Finally, in the speed bump match, we display the speed bumps detected by both techniques.

Table 4.22 presents the combined anomaly detection and ratio for the depression and



(a) Water culvert (b) Water culvert and shadow (c) Water culvert and pedestrian crossing

Figure 4.17: raining Samples for Vision-Based Speed Bump Detection Illustrated



(a) Speed bump exhibit 1 (b) Speed bump exhibit 2 (c) Speed bump exhibit 3

Figure 4.18: nstances of False Positive Detection in Vision Speed Bump Identification

speed bump anomalies. For the vision section, we grouped both the crack and pothole classes as these and compared them with the IMU depression classification. The vision represents 27.7% of the IMU classified data points. Similarly, the vision detected speed bumps makes up 24% of the total IMU speed bumps.

Table 4.22: Fusion Anomaly detected per class and ratio

Anomaly	IMU	Vision	Percentage
Depression	281	77	27.4
Speed bump	158	38	24.0

Furthermore, Table 4.23 lists the complete match of the same anomaly classes for both models. The data reveals that the fusion module correctly matched 48% of the IMU and vision depression anomalies. Meanwhile, we observe a 21% match for the speed bump anomalies.

Table 4.23: Fusion Anomaly dual detection

Anomaly	Amount
Depression	37
Speed bump	8

4.4.1.3 Test conclusion

Test cases two and three concluded we cannot distinguish between left and right depressions. On the other hand, test cases four, five and six confirm we can detect depressions, speed bumps and smoothness. Furthermore, test case seven confirms that DNN techniques are on par with traditional IMU systems for predicting road anomalies.

Test eight confirms that vision techniques can detect road anomalies with various degrees of accuracy. This test also confirmed that the MobileNetSSD performs well compared to other resource-hungry vision solutions. Moreover, the results of test case nine also kept consistent with test eight. This confirms that we can tune vision techniques to detect objects at different angles without losing accuracy.

Test ten also confirmed that we can fuse IMU and vision systems together to augment and improve the reporting capabilities. Consequently, we demonstrated that smartphone devices can deliver multi-technique solutions for road anomaly detection.

4.4.1.4 Final remarks

We listed the research questions at the start of the project. This section summarises the answers to these questions.

Question one: Can we develop a concise automated road anomaly detection system?

In this dissertation, we demonstrated that road anomaly detection automation is a viable solution. Furthermore, since the solution does not require any specialised hardware and is easy to configure, we believe that the local authorities can use the proposed solution to monitor road anomalies. Moreover, the fusion system can provide brief but comprehensive results.

Question two: Can we use inexpensive hardware to collect satisfactory sensory and image data to detect road anomalies?

This research has revealed that inexpensive hardware can collect and process sensor data to detect road anomalies. The results show that inexpensive hardware, combined with machine learning techniques, can differentiate between depressions, speed bumps, and smooth surfaces. However, as exhibited in test cases two and three, we conclude that inexpensive hardware sensory reading does not collect enough granularity to distinguish between left and right depressions.

Question three: Can smartphone devices provide a simple and cost-effective solution to detect road anomalies?

The results achieved in this study show that smartphones are a cost-effective solution for identifying road anomalies. Test cases four, five, and six show our ability to recognise depressions, speed bumps, and smoothness. Additionally, test case seven shows that DNN methods are effective at predicting road anomalies as conventional IMU systems.

Furthermore, tests eight and nine demonstrate that vision techniques can detect road irregularities with varying degrees of accuracy. Additionally, these tests confirmed that the MobileNetSSD's performance was comparable to that of other resource-intensive vision solutions.

Question four: Can we combine IMU, imagery, and GPS data to create a more meaningful road anomaly data set?

The data fusion assessment reveals that we can combine IMU and imagery machine learning techniques to augment the road anomaly result set. Test case ten, including the online dashboard solution, confirms this. As a result, we can combine both data stream to produce a more meaningful reporting tool.

4.5 Summary

This chapter has discussed the results of the IMU models for detecting road anomalies. In this research, we are detecting depressions, speed bumps and smooth surfaces. We compared five different models with manually extracted features. Furthermore, we compared the results with a neural network model based on raw data. Logistic regression produced the best results with an accuracy of 0.88. The precision of the depression class achieved a 0.86 and a recall rate of 0.79. Comparatively, the speed bump model produced a score of 0.83 and 0.85. The DNN model yielded an accuracy of 0.82 with a precision and recall rate of 0.73 and 0.74 for depression. Likewise, the speed bump returned similar ratings of 0.74 and 0.72.

This chapter has also discussed the results of the vision anomaly detection module. We tested five vision state-of-the-art object detection models to compare the accuracy.

The MobileNetSSDv2 implementation, which is targeted towards resource-limited architectures, exhibited good results with an accuracy of 52.17%. Additionally, the model returned a mAP.50 of 52.30% and a recall rate of 36.50% when used with a locally compiled image set.

Finally, this chapter has examined the development of a fusion model to process IMU and imagery data for road anomaly detection. We developed the fusion model by synchronising the IMU and imagery classification results and project this information onto a simple to use HTML dashboard. The fusion mechanism is based on the GPS atomic clock.

In the chapter that follows, we will discuss the evaluation process and provide examples of how designing machine learning models with evaluation in mind can lead to efficient and ideal outcomes.

Chapter 5

Evaluation

5.1 Introduction

This dissertation has discussed the usefulness of detecting road anomalies in real-world scenarios. The research has addressed depression and speed bump detection by running a sequence of experiments using IMU and vision approaches. Then we applied the IMU approach to process vibration and rotation sensory data. Moreover, we applied a vision model to confirm the IMU anomalies and also to detect anomalies that were not captured by the vehicle's vibration.

We organise the rest of the chapter as follows.

In subsection 5.1.1, we conduct a comprehensive evaluation of the field testing exercise. To ensure accuracy, all the images were meticulously tagged by a third party, resulting in a reliable ground truth data set. Through this meticulous process, we were able to draw meaningful conclusions about the performance of our approach. Moving forward to section 5.2, we extend our evaluation by benchmarking our results against state-of-the-art work in the field. This enables us to ascertain the level of advancement achieved by our approach, highlighting its contributions to the existing body of knowledge.

Overall, our thorough evaluation in autorefFusion Evaluation and the subsequent comparison in section 5.2 demonstrate the robustness and efficacy of our methodology while acknowledging the significance of our contributions within the broader research landscape.

5.1.1 Fusion Evaluation

In this section, we will evaluate the field testing results to assess the machine learning models' performance. By comparing the results from the field testing and the model training, we hope to determine the accuracy of our models.

5.1.1.1 IMU Depression

We appointed a third party to evaluate the 1,118 images and create a ground truth labelled data set. The process involved visually inspecting all the collected images and tagging them under 12 categories. The classification included smooth, crack, not clear, shadow, depression, road trench, pothole, turning, asphalt joint, gutter, excessive speeding and speed bump. This image multi-tagging evaluation produced a feature-rich ground truth data set linking images to multiple categories. Finally, we then compared the corresponding IMU and vision field testing classification results to this benchmark. We removed images that were marked "not clear" from the analysis. In addition, we also removed images taken at speeds over 50 km/h for the analysis. As a result, we ended up with 895 images for the IMU section and 993 files for both the IMU and vision analysis.

In our first appraisal, we assessed the IMU depression performance. For this test, we compared the predicted depression outcomes with the categorised data set for images tagged as depression, road trenches, and potholes. Figure 5.1 demonstrates the confusion matrix for this category. The analysis shows that the process wrongly predicted 58 false positive entries and 72 false negative anomalies. To put it differently, the false positives resulted from data collection points that were wrongly predicted as depression. In contrast, the false negatives reveal the depression entries that were not predicted as depression. The model returned an accuracy of 0.854 and a recall of 0.681.

Next, we added the cracks, gutters and surface joint categories to the equation, as shown in Figure 5.2.

Adding the three additional categories to the depression analysis improved the true positive and false positive scores. We observed that by adding these additional categories, it correctly transferred 36 false positive observations to the true positive score. In summary, gutters and surface joints improved the accuracy rating of the depression anomaly. The model's accuracy was 0.870, with a recall of 0.669.

Likewise, we conducted similar testing on the surface joint and gutter categories. We performed this to further analyse how these categories affect depression prediction, as shown in Figure 5.3 and Figure 5.4. These two categories can impact the classification of the depression and speed bump categories since they can have identical physical characteristics of depressions and bumps.

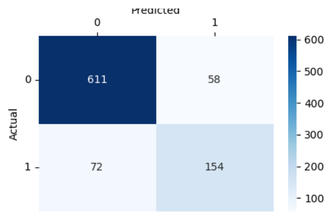


Figure 5.1: IMU Test one: Depression, road trench and pothole

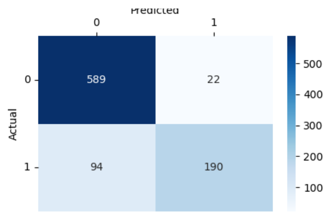


Figure 5.2: IMU Test two: Crack, depression, road trench, pothole, gutter and asphalt joint

The ground truth data has shown that there were 26 surface joints and three gutters that were classified under depression. Likewise, we could observe that, under the speed bump category, we had 13 surface joints and eight gutters. The accuracy reduced to 0.77 in both evaluations, while the recall score returned 0.666 and 0.278, respectively.

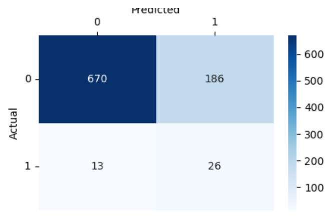


Figure 5.3: IMU depression test three : Asphalt joint

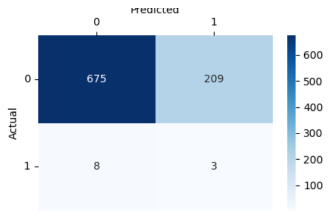


Figure 5.4: IMU depression test four: Gutter

5.1.1.2 IMU Speed bump

We relate one interesting observation that comes out of the next evaluation exercise. This is in connection with the high amount of false positives in the speed bump category review. We analysed three different confusion matrices to probe our field testing speed bump classification. Data from Figure 5.5 demonstrates that the false positives outnumbered the true positives by a ratio of 3:1. The model's accuracy is 0.885, and the recall stands at 0.870. In contrast, the precision for the test returned a score of 0.167.

Even after adding the asphalt joint and gutter classifications, we noted that the false positives and negatives remained high, as shown in Figure 5.6 and Figure 5.7. The accuracy for both tests is 0.891 and 0.874, with a recall of 0.824 and 0.556, respectively. Likewise, the corresponding precision score stood at 0.233 and 0.333.

The evidence appears to show that the small number of depression samples in the learning stage could cause this low rating. As a result, in future work, we will increase

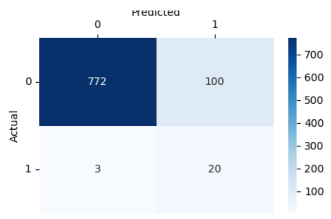


Figure 5.5: IMU speed bump test one:
Speed bump

the number of depression samples in the learning stage and reevaluate the classification accuracy.

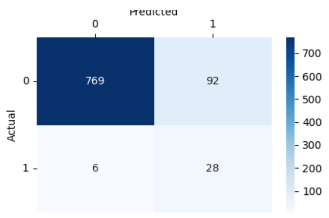


Figure 5.6: IMU Speed bump test two:
Speed bump and gutter

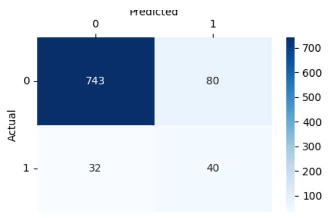


Figure 5.7: IMU Speed bump test three:
Asphalt joint, speed bump and gutter

5.1.1.3 IMU Smooth

In our final IMU performance evaluation, we analyse the smooth classification. As shown in Figure 5.8, we had 35 wrongly classified entries comprising 20 false positive and 15 false negative data points. The evaluation returned an accuracy of 0.961 and a recall of 0.973.

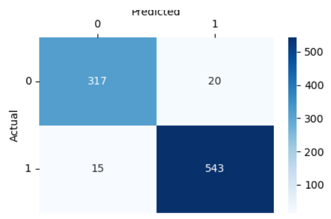


Figure 5.8: IMU Smooth test one:
Smooth

5.1.2 Vision

In the following three sections, we evaluate the vision model performance of our field tests. Our ground truth data set for the vision evaluation contained 993 images, with 895 IMU and 98 imagery classified data points.

A closer look at the vision data indicated the model was less responsive when driving at speeds higher than 20 km/h. In particular, we observed this lack of responsiveness during our testing phases. Whilst driving at speeds lower than 15 km/h, we could observe that the model was more accurate. This is contrary to the findings of Camilleri and Gatt (2020). However, we acknowledge that the camera zooming was a major source of limitations. Digital zoom helps to crop the entire image, and then digitally enlarges the size. As a result, there is a major reduction in the image's quality.

5.1.2.1 Vision Crack

The classification process correctly identified 905 true negatives and 25 true positives from the 993 manually classified images, as shown in Figure 5.9. Conversely, we also observed that the model missed 59 cracks, including four images that were incorrectly classified as road cracks. Given the speed issue, we believe the overall score is within acceptable limits. This evaluation returned an accuracy of 0.937 and a recall of 0.297.

In addition, we added the road trench-tagged images to measure how these affected the classification score. We present the results for the vision crack classification with both the crack and road trenching results in Figure 5.10. The data suggest that road tranches have no bearing on the results, with all the added images classified as false negatives. This evaluation returned an accuracy of 0.866 and a recall of 0.162.

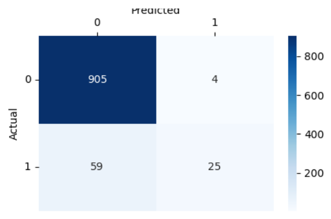


Figure 5.9: Vision Crack test one: Crack

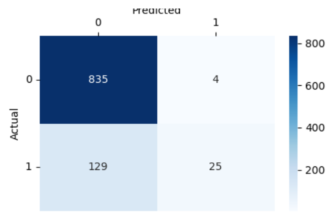


Figure 5.10: Vision Crack test two: Crack and road trench

To evaluate this segment, we also added the asphalt joint tagged images under road cracks. Once more, the classification as displayed in Figure 5.11 was unaffected by this category. In our fourth test, we tested the gutter data set, and once again, we did not notice any impact. We display the results in Figure 5.12. These evaluations returned an accuracy of 0.825 and a recall of 0.128 for the added asphalt join and 0.811 and 0.120 respectively, for the added gutter data.

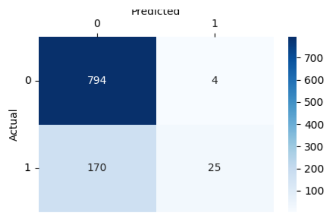


Figure 5.11: Vision Crack test three:
Crack, Road trench and Surface joint

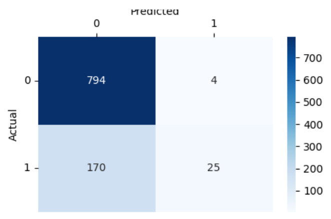


Figure 5.12: Vision Crack test four:
Crack, road trench, asphalt joint and Gutter

Finally, we added the depression-tagged images to the evaluation data set. Figure 5.13 shows the results for the added depression class. This evaluation returned an accuracy of 0.667 and a recall of 0.076.

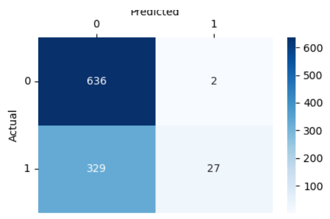


Figure 5.13: Vision Crack test five:
Crack, road trench, surface joint, Gutter and depression

5.1.2.2 Vision Pothole

Next, we evaluated the vision pothole prediction results. In this evaluation, we compared all the data points that were tagged as potholes in our ground truth data set. As shown in Figure 5.14, the model correctly recognised 27 images that contained potholes. In contrast, the model missed 30 instances of pothole images while mistakenly identifying ten images as containing a pothole anomaly. This evaluation returned an accuracy of 0.960 and a recall of 0.474.

Our second evaluation included road trenching images in the pothole category. We carried out this exercise to investigate whether the model could distinguish between both categories. As we can see from Figure 5.15, there was no impact on the true positive and false positive scoring. From the depicted results, we could notice that the vision model correctly distinguished the potholes from road trenching. This evaluation returned an accuracy of 0.894 and a recall of 0.205.

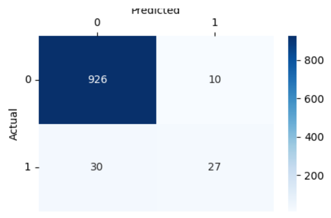


Figure 5.14: Vision Pothole test one:
Pothole

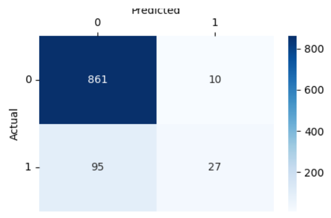


Figure 5.15: Vision Pothole test two:
Pothole and road trench

Forthwith, we added the road cracks tagged images to the pothole and road trenching tagged images to determine the correlation between the three classes. The figure in Figure 5.16 confirmed that the model did not misclassify the two distinct classes. This evaluation returned an accuracy of 0.818 and a recall of 0.144.

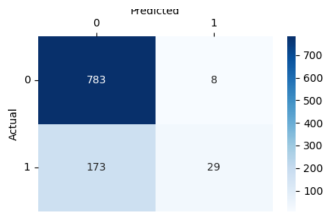


Figure 5.16: Vision Pothole test three:
Pothole, road trench and crack

Similarly, as we can see in Figure 5.17 and Figure 5.18, we established that neither the asphalt joint nor the water gutter images impacted the model's ability to distinguish potholes. The evaluation for the fourth test returned an accuracy of 0.776 and a recall of 0.119. Likewise, the gutter addition test returned an accuracy of 0.762 and a recall of 0.112.

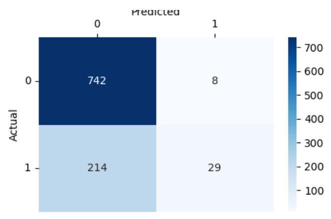


Figure 5.17: Vision Pothole test four:
Pothole, road trench, crack and asphalt joint

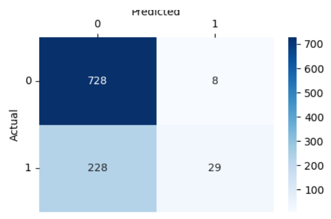


Figure 5.18: Vision Pothole test five:
Pothole, road trench, crack, asphalt joint and gutter

In our final analysis in this section, we added the depression tagged images to the pothole class. Figure 5.19 shows our analysis results. This evaluation returned an accuracy of 0.622 and a recall of 0.075.

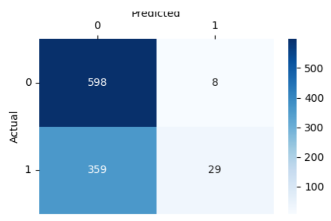


Figure 5.19: Vision pothole test five:
Pothole, road trench, crack, asphalt joint and gutter

5.1.2.3 Vision Speed bump

The speed bump anomaly classification is the subject of our final evaluation for the visual field testing exercise.

To determine the functionality and behaviour of the vision speed bump anomaly detection, we ran five tests.

The first study involved analysing all the data points that the vision module had identified as speed bumps. Figure 5.20 illustrates the performance based on the whole test data set. The algorithm correctly identified 12 images. However, the model missed 23 images containing potholes whilst it mistook 20 objects for potholes. This evaluation returned an accuracy of 0.957 and a recall of 0.343.

From the evaluation of the images, we could observe that the vision model wrongly classified surface joints, gutters, long shadows and road trenches as speed bumps.

To further evaluate this inconsistency, we added the asphalt joint-tagged images to the pothole subset. The results show this category had little effect on the true positives, with almost all surface joint tagged items being added to the false negative section, as shown in Figure 5.20. This evaluation returned an accuracy of 0.920 and a recall of 0.187.

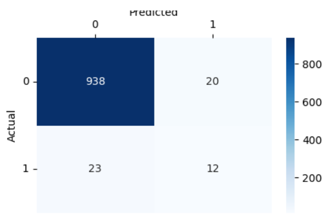


Figure 5.20: Vision speed bump test one:
speed bump

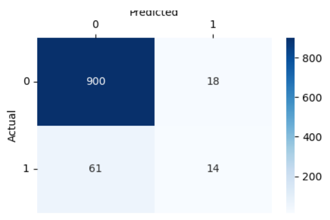


Figure 5.21: Vision speed bump test two:
speed bump or asphalt joint

In a similar vein, we included gutter-tagged images in the evaluation procedure. We display the result of the added class to the speed bump category in Figure 5.22. This impacted the false negative rating but also slightly improved the true positive. This evaluation returned an accuracy of 0.912 and a recall of 0.191.

We could see an interesting observation by introducing shadow-tagged images. We could also observe that the vision module mistakenly classified six shadows as speed bumps, as shown in Figure 5.23. This evaluation returned an accuracy of 0.912 and a recall of 0.228.

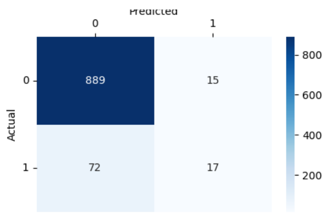


Figure 5.22: Vision speed bump test three: Speed bump, surface joint or gutter

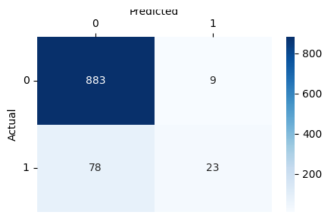


Figure 5.23: Vision speed bump test four: Speed bump, surface joint, gutter or shadow

Another observation reveals that when we added road trenching-tagged images to the speed bump group, there was an increase in the true positives. Data from Figure 5.23 indicates an increase of five images. This evaluation returned an accuracy of 0.850 and a recall of 0.121.

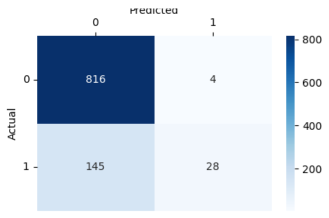


Figure 5.24: Vision speed bump test five: Speed bump, surface joint, gutter, shadow or road trench

5.1.2.4 Speed dependency

We introduced the speed dependency phenomenon in subsubsection 3.6.5.1. During our field testing, we found it challenging to maintain the same constant speed as the training exercise, and the outcomes impacted the classification. Furthermore, Perttunen et al.

(2011) attempted to approximate the relationship between speed and inertial data as a trend in the value of calculated features by using linear regression and then subtracting that trend from the features used for classification. Furthermore, when evaluating road anomalies, Seraj et al. (2015) considered the vehicle's speed and how this affected the vehicle's vibrations. This is a significant limitation in our solution and we will investigate ways to reduce this limitation in our future work to improve our models.

5.2 Evaluation with other papers

We compare our results with previous studies' state-of-the-art work in Table 5.1 and Table 5.2. Moreover, we also include our field test results after the third-party annotated results data set and subsequent analysis.

5.2.1 IMU results

5.2.1.1 Overall accuracy

Our accuracy results demonstrated a notable discrepancy, ranging from 0.006 to 0.087, across the three state-of-the-art research papers, as illustrated in Table 5.1. This variation highlights the diversity in the performance of the various models or methods presented in those papers.

5.2.1.2 Pothole detection

Our pothole precision finished in second place after Wu et al. (2020)'s work with a difference of 0.048 points from their best-performing model, which scored 0.908. From a recall perspective. Our model produced the best result with a score of 0.790. Basavaraju et al. (2020) registered the second-best score.

5.2.1.3 Smooth surface

Similar to the pothole result, our smooth precision trailed Basavaraju et al. (2020) and Wu et al. (2020) by a score margin of 0.019 from the best score. On the other hand, our recall score finished in first place with a margin of 0.008.

5.2.1.4 Silva et al. (2017)'s speed bump results

Silva et al. (2017) scoring is based on the confusion matrix's percentage of correct classification. They obtained a score of 0.778 and 0.639 for the Gradient boost and decision tree. We secured a result of 0.850 through our LR model, therefore, our speed bump classification performed better results.

5.2.2 Vision results

As shown in Table 5.2, we could not match Camilleri and Gatt (2020) and Rani et al. (2020)'s work in our vision model. Our pothole and depression mAP finished 0.165 behind Camilleri and Gatt (2020)'s work. Furthermore, there was a difference of 0.065 between our results and Rani et al. (2020)'s score.

Table 5.1: Final evaluation - IMU

Source	Algorithm	Acc	Crack		Pothole / Depression		Speed bump		Smooth	
			Precision	Recall	Precision	Recall	Precision	Recall	Precision	Recall
Basavaraju et al. (2020)	SVM	0.886	0.403	0.438	0.722	0.678			0.944	0.947
	Decision tree	0.883	0.435	0.412	0.666	0.671			0.950	0.947
	NN	0.921	**0.559**	**0.611**	0.769	0.781			**0.969**	0.963
Wu et al. (2020)	SVM	0.948			**0.908**	0.642			0.952	0.992
	LR	0.952			0.851	0.734			0.965	0.984
	RF	0.957			0.885	0.750			0.965	0.988
Silvister et al. (2019)	DNN	**0.967**			n/a	n/a			n/a	n/a
	SVM	0.929			n/a	n/a			n/a	n/a
Our results	LR	0.880			0.860	**0.790**	**0.830**	**0.850**	0.950	**1.000**
	SVM	0.820			0.710	0.770	0.760	0.760	0.680	**1.000**
	KNN	0.800			0.720	0.690	0.720	0.710	0.950	**1.000**
	RF	0.750			0.620	0.640	0.650	0.650	0.990	0.960
	NB	0.790			0.680	0.690	0.690	0.700	**1.000**	0.980
	DNN	0.820			0.730	0.750	0.740	0.720	0.990	**1.000**
Field testing	LR	0.854			0.726	0.669			0.964	0.973

Table 5.2: Final evaluation - Vision

Source	Algorithm	Prediction accuracy		Pothole / Depression		Speed bump	
		Pothole	Speed bump	mAP	Recall	mAP	Recall
Camilleri and Gatt (2020)	Yolov3 SPP			**0.688**			
Tithi et al. (2021)	SSDMobileNetV2	0.670 - 0.920	0.790 - 0.930				
Rani et al. (2020)	SSDMobileNetV2			0.600		**0.700**	
Our results	SSDMobileNetV2			0.523	0.365	0.635	0.514
Field testing	SSDMobileNetV2			0.726	0.669	0.910	0.191

5.3 Summary

In this section, we discussed the evaluation techniques and offered suggestions for improving our models. We demonstrated how designing machine learning models with evaluation in mind can lead to successful and optimised solutions.

We started this section by discussing the classification process to evaluate our model's performance. Next, we described how to calculate accuracy, precision, recall, and F1 scores. In addition, we discuss why such metrics are important in our classification processes. We then reviewed our field testing exercise together with the fusion module. We enlisted the assistance of a third party to evaluate the collected imagery data and extract the ground truth from the images. Consequently, we divided the information into 12 categories. We then compared the prediction of our model to the ground truth data set. According to the evidence, gutters, road tranches, and water gutters in the IMU model influenced the accuracy of the depression and speed bump classes. Furthermore, speeds above 50 km/h also influenced the accuracy. At these speeds, we noticed a significant increase in speed bump false positives. As a result, speed dependency will play an important role in future work. In addition, evidence in the vision model suggested a link between speed bumps, water gutters, and road trenches. We also demonstrated how the surrounding shadows influenced the detection process.

We also illustrated how laboratory-controlled experiments vary from real-life usage. Testing a model's accuracy in a laboratory setting can produce better results. However, this technique has some limitations. To illustrate, we narrowed the types of road anomalies and also reduced our driving speed in our work. Therefore, we did not simulate real-world usage. As a result, our laboratory experiments did not capture all real-life anomalies. This was demonstrated in our field-testing result analysis.

However, despite the size of the IMU training data and the smartphone's digital zoom resolution reduction, our models produced respectable results. Most importantly, we showed that we can fuse both techniques to better track road anomalies.

Chapter 6

Conclusion

6.1 Revisiting the Aims and objectives

The aim of this research was to create a solution for detecting and classifying road anomalies. By combining two concurrent machine-learning techniques, we aimed at enhancing the overall accuracy. More specifically, we created a simple but effective platform for detecting and authenticating road anomalies. We achieved this by combining traditional IMU techniques with innovative and robust visualisation models. In other words, our goal was to combine the two techniques and create an augmented reporting data set. As a result, we demonstrated how both systems can complement one another, with each technique highlighting different but important, aspects of the detection process.

The first objective of this study was to use inexpensive smartphone devices to assess road conditions. We target both the IMU and camera sensors to achieve the desired results. Additionally, our second objective was to merge both pipelines and augment the final reporting data set.

6.2 Experiments Conducted

We presented a general overview of the proposed solution in Figure 6.1. For this research, we proposed a dual smartphone method to detect anomalies. The IMU and vision systems operated independently of one another.

We strategically placed the IMU smartphone on the vehicle's dashboard to capture the vehicle's vibration and rotation. To capture these movements, we employed the smartphone's accelerometer and gyroscope sensors. In particular, we captured data from all three axes, namely X, Y and Z. To pinpoint the anomaly's geolocation, we employed the

GPS sensor to extract the GPS coordinates. In addition, we based our synchronisation methods on the GPS atomic clock data.

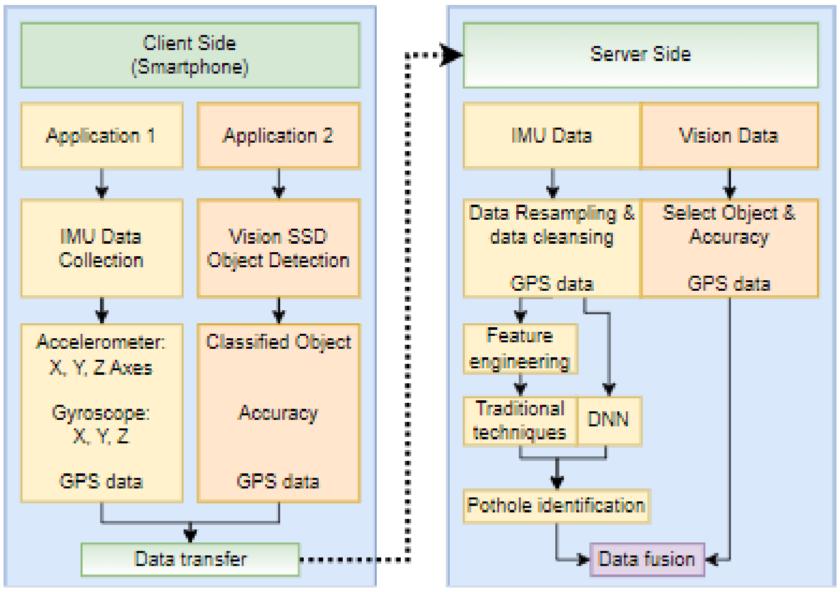


Figure 6.1: Illustration of the Proposed Road Anomaly Detection Solution

Likewise, we hosted the vision object detection module on the second smartphone device. We employed an embedded MobileNetSSDv2 on this device. The model captured images from the smartphone camera and returned the classification results and accuracy of the detected anomalies. Similarly, we extracted the GPS data from the second smartphone and packaged the GPS reading with the vision data stream. Additionally, we calibrated the timing by comparing the smartphone's clock with the GPS atomic clock. We also embedded the variance in both data sets.

In the final stage of the routine, we sent both data sets to the back-end server for analysis. As can be seen in the solution diagram, we supplied the vision module data to the fusion component. To emphasise, we used the vision classification process that runs on the smartphone application. In contrast, the IMU anomaly prediction is being processed on the back-end server. The DNN model works with raw IMU sensor data. On the other hand, the traditional IMU machine-learning techniques work on engineered features. In summary, we merge the IMU results with the vision predictions and fuse both data sets under one augmented reporting portal.

This study collected 85,000 anomaly inertia readings at a frequency of 50 Hz. Every anomaly consisted of a two-second sampling period. Therefore, every IMU anomaly contained 100 sensor readings per label. For this reason, we built an application to label the anomalies in real-time. In our study, we addressed the following five road anomaly classes: left, right depression, depression wider than the wheel track, speed bump and smooth surface. Because of an imbalanced data set problem, we down-sampled the data to simplify the machine-learning techniques. Most machine learning algorithms for classification were designed to work with an equal number of examples for each class. Therefore, unbalanced data sets pose a challenge for predictive modelling. Following a series of data cleansing and filtering routines, we proceeded with the machine learning model development. We used six supervised machine learning techniques for our tests: LR, SVM, KNN, RF, NB and DNN. For the first five traditional techniques, we manually engineered the features. However, this step was not required for the DNN model, as neural networks work with raw sensor data. This is one of the main advantages of this technique. The results show that the anomaly prediction for DNN is on par with the traditional feature-engineered models.

We trained the vision module using a Maltese anomaly data set of 1,891 images. For the labelling task, we labelled the ground truth under three categories: crack, pothole, and speed bump. We evaluated various smartphone mounting points for capturing the road. Based on the results and observations, we changed the training data set by using various capturing angles to improve accuracy. Considering the limited resources, the MobilenetSSD model returned satisfactory mAP and recall results.

In our data fusion solution, we produced a convenient dashboard for linking the IMU and vision predictions. The main aim of our reporting tool is to reinforce the prediction score and enhance the reporting data.

We also carried out a field texting exercise to test the effectiveness of the system. In total, we collected 993 images and corresponding IMU readings. The field testing yielded mixed results. Most notably, the number of false positives in the speed bump class and depression. We analysed these issues in the evaluation section. We attributed these misclassifications to excessive speeding and the similarity between speed bumps and water gutters and road tranches.

6.3 Summary of Results

In the IMU solution, we collected 85,000 inertia readings under five different road anomalies: Depression left, depression right, depression, speed bump and smooth surface. Each anomaly consisted of a two-second sensor reading. Additionally, we set the sensor frequency rate at 50 Hz. As a result, we had 100 time-series readings per anomaly. To keep the data set balanced, we used the downsampling technique. As a result, we replaced the depression class that was bigger than the wheel track with equal parts of the left and right depression records. During the initial testing, we observed that the proposed solution was not able to distinguish between left and right depression. Therefore, we excluded these classes from our tests. Logistic regression produced the best results, with an accuracy of 88%. The depression class had a recall rate of 79% and a precision of 0.86. On the other hand, the speed bump category produced scores of 83% and 85%. Similarly, the DNN model produced accuracy and recall rates of 82%, 73%, and a precision of 74% for depressions. The speed bump tests returned scores of 74% and 72%. Given that the DNN works on raw data, the results are noteworthy. Next, we covered the outcomes of the vision anomaly detection module. To compare the accuracy, we tested five innovative object detection models for vision. The MobileNetSSDv2 implementation produced positive results with an accuracy of 52.17%. We then used the MobileNetSSDv2 with a locally compiled image set. The model produced similar scores with mAP.50 of 52.30% and a recall rate of 36.50%. Finally, we analysed the data fusion model to process IMU and imagery data for the detection of road anomalies. We built the fusion model by synchronising the IMU and imagery classification results. Finally, we demonstrated how the fused data can simplify the anomaly verification process by presenting the two results in an easily operated dashboard.

6.4 Research question

In conclusion, the research undertaken to address the research question has demonstrated the feasibility of developing an innovative smartphone-based system for detecting road anomalies, with a primary focus on potholes. The results of extensive testing have underscored the potential of this approach, indicating its viability for further exploration and refinement.

The integration of various technologies, including inexpensive smartphone hardware, IMU sensing, and vision-based anomaly detection, has proven to be successful in enhancing the accuracy and reliability of the system. By utilizing both IMU sensing and vision-based techniques, a notable improvement in accuracy has been achieved. These two approaches harmoniously complement each other, creating a more robust and comprehensive detection mechanism that enhances the overall reliability of anomaly identification.

One of the significant strengths of this solution lies in its utilisation of widely available and affordable smartphone hardware, making it accessible to a broad user base. This accessibility not only contributes to the proliferation of anomaly detection but also aligns with the goal of creating a cost-effective and user-friendly solution.

Furthermore, the system's user interface provides an intuitive and easy-to-use portal for accessing and interpreting the results. This feature allows for manual checks and verification of detected anomalies through a user-friendly interface, thereby adding an extra layer of validation to the automated detection process.

In summary, this research has successfully demonstrated the potential of a smartphone-based road anomaly detection system, particularly focused on potholes. The combination of IMU sensing and vision anomaly detection techniques, along with the utilisation of inexpensive smartphone hardware, has showcased encouraging results. Moving forward, there exists substantial room for further exploration, refinement, and optimization, paving the way for a more efficient and comprehensive solution to address the challenges posed by road anomalies.

6.5 Critique and Limitations

In this section, we list the limitations of our study and discuss how they may affect the validity of our findings.

Test vehicle: The first limitation in our study is related to the number of participating vehicles used to capture the training data. The IMU data set used for this study was captured using only one vehicle. Therefore, the proposed model will not yield the same results on other vehicles. This is a major limitation. In future releases, the data set will have data from various vehicle makes and models to widen the user base. Furthermore, the vision training data set model is based on a predefined angle and zoom size. As explained in Chapter 3.5.2, the windscreen mounting angle and vehicle bonnet restricted the smartphone's camera visibility. As a result, this visual restriction and the camera's digital zoom hampered our vision model.

Speed issue: As mentioned in subsubsection 5.1.2.4, one major limitation of this study is that the model only works with limited and predefined speeds. In the future, we will address this issue by including speed as a feature in our models similar to the work conducted by Chen et al. (2019).

IMU training data set size: As described in Chapter 4, the IMU data set used was too small. A larger training data set will increase the accuracy of the models.

Limited anomaly classes: During the field testing, we noticed we were missing several anomaly classifications. In this study, we excluded the asphalt joints, road trenching, and water gutters from the training data. As a result, this shortcoming impacted the overall accuracy.

Vision training set: The vision training data set contains no waterlogged potholes. This creates an operational constraint during winter. Likewise, the training images contained no shadows from the surroundings.

6.6 Future research

Our solution used standard smartphones and custom applications to collect data. We then transferred the compiled data to a back-end server in batches. In future releases, the IMU data can be collected and processed in real time using cloud services.

Future research could include running traditional and DNN algorithms directly on smartphone devices to minimise the back-end computing power required to classify the anomalies. This will also provide instant results similar to our MobileNetSSD solution.

Future research could include investigating other lightweight smartphone-enabled SSD solutions such as tiny Yolo for real-time object detection.

Future research can address the speed limitation constraints during road testing. This will allow for greater driving flexibility during road analysis.

6.7 Summary and conclusion

In this dissertation, we embarked on a journey to develop a solution for the detection and classification of road anomalies, with a specific emphasis on enhancing accuracy through the synergy of two concurrent machine-learning techniques. By harmonising traditional Inertial Measurement Unit (IMU) methods with robust visual models, we succeeded in creating an effective platform for detecting and verifying road anomalies. Our overarching objective was to combine these two techniques into an augmented reporting data set, showcasing how each method brings its unique strengths to the anomaly detection process.

We effectively harnessed the power of inexpensive smartphone devices to assess road conditions, leveraging both IMU and camera sensors to gather crucial data. Our approach focused on merging these two data pipelines to enrich the final reporting data set.

Our journey was not without its share of challenges, but we successfully navigated them. An imbalanced data set posed a unique problem, but we addressed it through careful data pre-processing and downsampling, ensuring a level playing field for our machine-learning models. Furthermore, we employed a variety of supervised machine learning techniques, including Logistic Regression (LR), Support Vector Machines (SVM), K-Nearest Neighbours (KNN), Random Forests (RF), Naive Bayes (NB), and Deep Neural Networks (DNN), each tailored to our specific data set needs.

In the realm of vision anomaly detection, we trained our model using a diverse image data set. We adapted our approach based on experimentation, fine-tuning the model to achieve satisfactory mean Average Precision (mAP) and recall results.

Our findings hold significant implications for the future of road anomaly detection in similar settings. Notably, the integration of diverse technologies, encompassing cost-effective smartphone hardware, IMU sensing, and vision-based anomaly detection, demonstrated remarkable success, promising substantial enhancements in accuracy and reliability. This approach aligns with the goal of creating an accessible and cost-effective solution, bridging the gap to a broader user base. Moreover, our user-friendly interface empowers users to interpret results seamlessly, facilitating manual verification of detected anomalies and reinforcing the automated detection process's reliability. Additionally, future research avenues may delve into real-time IMU data processing via cloud services, potentially delivering instant results akin to the MobileNetSSD solution we presented.

In conclusion, this dissertation has not only demonstrated the feasibility of a smartphone-based road anomaly detection system but also laid the groundwork for future research and innovation in this field. The fusion of IMU sensing and vision-based detection, combined with the accessibility of smartphone hardware, paves the way for safer and more efficient road analysis and maintenance, benefiting society as a whole.

References

Abualsaud, K., Elfouly, T. M., Khattab, T., Yaacoub, E., Ismail, L. S., Ahmed, M. H., and Guizani, M. A Survey on Mobile Crowd-Sensing and Its Applications in the IoT Era. *IEEE Access*, 7:3855–3881, 2019. Conference Name: IEEE Access.

Angrisano, A. and Gaglione, S. Smartphone GNSS Performance in an Urban Scenario with RAIM Application. *Sensors*, 22(3), 2022.

Bach, S. H., He, B., Ratner, A., and Ré, C. Learning the Structure of Generative Models without Labeled Data. *Proceedings of machine learning research*, 70:273–82, August 2017.

Baek, J.-W. and Chung, K. Pothole Classification Model Using Edge Detection in Road Image. *Applied Sciences*, 10(19):6662, January 2020. Number: 19 Publisher: Multidisciplinary Digital Publishing Institute.

Basavaraju, A., Du, J., Zhou, F., and Ji, J. A Machine Learning Approach to Road Surface Anomaly Assessment Using Smartphone Sensors. *IEEE Sensors Journal*, 20(5):2635–2647, March 2020. Conference Name: IEEE Sensors Journal.

Bella, F., Calvi, A., and D'Amico, F. Impact of Pavement Defects on Motorcycles' Road Safety. *Procedia - Social and Behavioral Sciences*, 53:942–951, October 2012.

Benndorf, M. and Haenselmann, T. Time Synchronization on Android Devices for Mobile Construction Assessment. 2016.

Bhoraskar, R., Vankadhara, N., Raman, B., and Kulkarni, P. Wolverine: Traffic and road condition estimation using smartphone sensors. In *2012 Fourth International Conference on Communication Systems and Networks (COMSNETS 2012)*, pages 1–6, January 2012.

Breiman, L. Random Forests. *Machine Learning*, 45(1):5–32, October 2001.

Brisimi, T. S., Cassandras, C. G., Osgood, C., Paschalidis, I. C., and Zhang, Y. Sensing and Classifying Roadway Obstacles in Smart Cities: The Street Bump System. *IEEE Access*, 4:1301–1312, 2016. Conference Name: IEEE Access.

Cabral, F. S., Pinto, M., Mouzinho, F. A. L. N., Fukai, H., and Tamura, S. An Automatic Survey System for Paved and Unpaved Road Classification and Road Anomaly Detection using Smartphone Sensor. In *2018 IEEE International Conference on Service Operations and Logistics, and Informatics (SOLI)*, pages 65–70, July 2018.

Camilleri, N. and Gatt, T. Detecting road potholes using computer vision techniques. In *2020 IEEE 16th International Conference on Intelligent Computer Communication and Processing (ICCP)*, pages 343–350, September 2020.

Carlos, M. R., Aragón, M. E., González, L. C., Escalante, H. J., and Martínez, F. Evaluation of Detection Approaches for Road Anomalies Based on Accelerometer Readings—Addressing Who's Who. *IEEE Transactions on Intelligent Transportation Systems*, 19(10):3334–3343, October 2018. Conference Name: IEEE Transactions on Intelligent Transportation Systems.

Chen, Y., Zhou, M., Zheng, Z., and Huo, M. Toward Practical Crowdsourcing-Based Road Anomaly Detection With Scale-Invariant Feature. *IEEE Access*, 7:67666–67678, 2019. Conference Name: IEEE Access.

Chitale, P. A., Kekre, K. Y., Shenai, H. R., Karani, R., and Gala, J. P. Pothole Detection and Dimension Estimation System using Deep Learning (YOLO) and Image Processing. *2020 35th International Conference on Image and Vision Computing New Zealand (IVCNZ)*, pages 1–6, 2020.

Darapaneni, N., Reddy, N. S., Urkude, A., Paduri, A. R., Satpute, A. A., Yogi, A., Natesan, D. K., Surve, S., and Srivastava, U. Pothole Detection Using Advanced Neural Networks. In *2021 IEEE 12th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, pages 0567–0572, October 2021.

Dib, J., Sirlantzis, K., and Howells, G. A Review on Negative Road Anomaly Detection Methods. *IEEE Access*, 8:57298–57316, 2020. Conference Name: IEEE Access.

Douangphachanh, V. and Oneyama, H. Formulation of a simple model to estimate road surface roughness condition from Android smartphone sensors. In *2014 IEEE Ninth International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP)*, pages 1–6, April 2014.

El Aboudi, N. and Benhlima, L. Review on wrapper feature selection approaches. In *2016 International Conference on Engineering MIS (ICEMIS)*, pages 1–5, September 2016.

Engelbrecht, J., Booysen, M. J., van Rooyen, G.-J., and Bruwer, F. J. Survey of smartphone-based sensing in vehicles for intelligent transportation system applications. *IET Intelligent Transport Systems*, 9(10):924–935, 2015.

Eriksson, J., Girod, L., Hull, B., Newton, R., Madden, S., and Balakrishnan, H. The pothole patrol: using a mobile sensor network for road surface monitoring. In *Proceeding of the 6th international conference on Mobile systems, applications, and services - MobiSys '08*, page 29, Breckenridge, CO, USA, 2008. ACM Press. ISBN 978-1-60558-139-2.

Fazeen, M., Gozick, B., Dantu, R., Bhukhiya, M., and González, M. C. Safe Driving Using Mobile Phones. *IEEE Transactions on Intelligent Transportation Systems*, 13(3):1462–1468, September 2012. Conference Name: IEEE Transactions on Intelligent Transportation Systems.

Fox, A., Kumar, B. V., Chen, J., and Bai, F. Crowdsourcing undersampled vehicular sensor data for pothole detection. In *2015 12th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*, pages 515–523, June 2015.

Girshick, R., Donahue, J., Darrell, T., and Malik, J. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 580–587, June 2014.

He, K., Zhang, X., Ren, S., and Sun, J. Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(9):1904–1916, September 2015. Conference Name: IEEE Transactions on Pattern Analysis and Machine Intelligence.

He, Z. Gesture Recognition Based on Tri-Axis Accelerometer Using 1D Gabor Filters. In *2018 9th International Symposium on Parallel Architectures, Algorithms and Programming (PAAP)*, pages 146–151, December 2018.

Hemminki, S., Nurmi, P., and Tarkoma, S. Accelerometer-based transportation mode detection on smartphones. In *Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems - SenSys '13*, pages 1–14, Roma, Italy, 2013. ACM Press.

Huang, G., Liu, Z., Van Der Maaten, L., and Weinberger, K. Q. Densely Connected Convolutional Networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2261–2269, July 2017.

Ioffe, S. and Szegedy, C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. page 9.

Lin, T.-Y., Dollár, P., Girshick, R., He, K., Hariharan, B., and Belongie, S. Feature pyramid networks for object detection. pages 936–944, 2017. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

Lindfors, M., Hendeby, G., Gustafsson, F., and Karlsson, R. Vehicle speed tracking using chassis vibrations. In *2016 IEEE Intelligent Vehicles Symposium (IV)*, pages 214–219, June 2016.

Liu, M.-K., Lin, Y.-T., Qiu, Z.-W., Kuo, C.-K., and Wu, C.-K. Hand Gesture Recognition by a MMG-Based Wearable Device. *IEEE Sensors Journal*, 20(24):14703–14712, December 2020. Conference Name: IEEE Sensors Journal.

Liu, S., Qi, L., Qin, H., Shi, J., and Jia, J. Path aggregation network for instance segmentation. pages 8759–8768, 2018. 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition.

Maeda, H., Sekimoto, Y., Seto, T., Kashiyama, T., and Omata, H. Road Damage Detection and Classification Using Deep Neural Networks with Smartphone Images. *Computer-Aided Civil and Infrastructure Engineering*, 33(12):1127–1141, 2018.

Mazur, D. C., Entzminger, R. A., Kay, J. A., and Morell, P. A. Time Synchronization Mechanisms for the Industrial Marketplace. *IEEE Transactions on Industry Applications*, 53(1): 39–46, January 2017. Conference Name: IEEE Transactions on Industry Applications.

Mednis, A., Strazdins, G., Zviedris, R., Kanonirs, G., and Selavo, L. Real time pothole detection using Android smartphones with accelerometers. In *2011 International Conference on Distributed Computing in Sensor Systems and Workshops (DCOSS)*, pages 1–6, June 2011.

Merry, K. and Bettinger, P. Smartphone GPS accuracy study in an urban environment. *PLOS ONE*, 14(7):e0219890, July 2019.

Mishra, S., Mishra, D., Das, S., and Rath, A. K. Feature reduction using principal component analysis for agricultural data set. In *2011 3rd International Conference on Electronics Computer Technology*, volume 2, pages 209–213, April 2011.

Mohamed, A., Fouad, M. M. M., Elhariri, E., El-Bendary, N., Zawbaa, H. M., Tahoun, M., and Hassanien, A. E. RoadMonitor: An Intelligent Road Surface Condition Monitoring System. In Filev, D., Jab\lkowski, J., Kacprzyk, J., Krawczak, M., Popchev, I., Rutkowski, L., Sgurev, V., Sotirova, E., Szynkarczyk, P., and Zadrozny, S., editors, *Intelligent Systems'2014*, pages 377–387, Cham, 2015. Springer International Publishing.

Mohan, P., Padmanabhan, V. N., and Ramjee, R. Nericell: rich monitoring of road and traffic conditions using mobile smartphones. In *Proceedings of the 6th ACM conference on Embedded network sensor systems - SenSys '08*, page 323, Raleigh, NC, USA, 2008.

ACM Press.

Orhan, F. and Eren, P. E. Road Hazard Detection and Sharing with Multimodal Sensor Analysis on Smartphones. In *2013 Seventh International Conference on Next Generation Mobile Apps, Services and Technologies*, pages 56–61, September 2013.

Oyamada, M. Extracting Feature Engineering Knowledge from Data Science Notebooks. In *2019 IEEE International Conference on Big Data (Big Data)*, pages 6172–6173, December 2019.

Perttunen, M., Mazhelis, O., Cong, F., Kauppila, M., Leppänen, T., Kantola, J., Collin, J., Pirttikangas, S., Haverinen, J., and Riekki, J. Distributed Road Surface Condition Monitoring Using Mobile Phones. pages 64–78, September 2011.

Raghunath, S., Barooru, D. L. M., and Karnam, S. Comparative Analysis Of Kalman And Extended Kalman Filters In Improving GPS Accuracy. *International Journal of Engineering Research*, 2(6):9, 2013.

Ramírez-Gallego, S., Krawczyk, B., García, S., Woźniak, M., and Herrera, F. A survey on data preprocessing for data stream mining: Current status and future directions. *Neurocomputing*, 239:39–57, 2017. Publisher: Elsevier.

Rani, M. R., Mustafar, M. Z. C., Ismail, N. H. F., and Mansor, M. S. F. Road Peculiarities Detection using Deep Learning for Vehicle Vision System. *Materials Science and Engineering*, page 11, 2020.

Rao, C. V., Pavani, M., Ahammed, R., Varma, D. S., and Reddy, P. B. Pothole Detection System using Artificial Intelligence. 07(04):3, 2020.

Redmon, J. and Farhadi, A. YOLO9000: Better, Faster, Stronger. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Honolulu, HI, July 2017. IEEE.

Redmon, J. and Farhadi, A. YOLOv3: An Incremental Improvement. *arXiv:1804.02767 [cs]*, April 2018. arXiv: 1804.02767.

Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. You Only Look Once: Unified, Real-Time Object Detection. pages 779–788, 2016.

Ren, S., He, K., Girshick, R., and Sun, J. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6):1137–1149, 2017.

Roh, Y., Heo, G., and Whang, S. E. A Survey on Data Collection for Machine Learning: A Big Data - AI Integration Perspective. *IEEE Transactions on Knowledge and Data Engineering*, 33(4):1328–1347, April 2021. Conference Name: IEEE Transactions on Knowledge and Data Engineering.

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., and Chen, L.-C. MobileNetV2: Inverted Residuals and Linear Bottlenecks. pages 4510–4520, June 2018. 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition.

Seraj, F., Zhang, K., Turkes, O., Meratnia, N., and Havinga, P. J. M. A smartphone based method to enhance road pavement anomaly detection by analyzing the driver behavior. In *Adjunct Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2015 ACM International Symposium on Wearable Computers*, UbiComp/ISWC'15 Adjunct, pages 1169–1177, New York, NY, USA, September 2015. Association for Computing Machinery.

Shahapure, K. R. and Nicholas, C. Cluster Quality Analysis Using Silhouette Score. In *2020 IEEE 7th International Conference on Data Science and Advanced Analytics (DSAA)*, pages 747–748, October 2020.

Silva, N., Soares, J., Shah, V., Santos, M. Y., and Rodrigues, H. Anomaly Detection in Roads with a Data Mining Approach. *Procedia Computer Science*, 121:415–422, 2017.

Silvister, S., Komandur, D., Kokate, S., Khochare, A., More, U., Musale, V., and Joshi, A. Deep Learning Approach to Detect Potholes in Real-Time using Smartphone. In *2019 IEEE Pune Section International Conference (PuneCon)*, pages 1–4, December 2019.

Sivrikaya, F. and Yener, B. Time synchronization in sensor networks: a survey. *IEEE Network*, 18(4):45–50, July 2004. Conference Name: IEEE Network.

Stisen, A., Blunck, H., Bhattacharya, S., Prentow, T. S., Kjærgaard, M. B., Dey, A., Sonne, T., and Jensen, M. M. Smart Devices are Different: Assessing and MitigatingMobile Sensing Heterogeneities for Activity Recognition. In *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems*, pages 127–140, Seoul South Korea, November 2015. ACM.

Tai, Y.-c., Chan, C.-w., and Hsu, J. Automatic road anomaly detection using smart mobile device. 15th Conference on Artificial Intelligence and Applications (TAAI), January 2010.

Tan, M., Pang, R., and Le, Q. V. Efficientdet: Scalable and efficient object detection. pages 10778–10787, 2020. 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR).

Tithi, A., Ali, F., and Azrof, S. Speed Bump amp; Pothole Detection with Single Shot Multi-

Box Detector Algorithm amp; Speed Control for Autonomous Vehicle. In *2021 International Conference on Automation, Control and Mechatronics for Industry 4.0 (ACMI)*, pages 1–5, July 2021.

Wong, T.-T. and Yeh, P.-Y. Reliable Accuracy Estimates from k-Fold Cross Validation. *IEEE Transactions on Knowledge and Data Engineering*, 32(8):1586–1594, August 2020. Conference Name: IEEE Transactions on Knowledge and Data Engineering.

Wu, C., Wang, Z., Hu, S., Lepine, J., Na, X., Ainalis, D., and Stettler, M. An Automated Machine-Learning Approach for Road Pothole Detection Using Smartphone Sensor Data. *Sensors (Basel, Switzerland)*, 20(19), September 2020.

Xu, C., Chai, D., He, J., Zhang, X., and Duan, S. InnoHAR: A Deep Neural Network for Complex Human Activity Recognition. *IEEE Access*, 7:9893–9902, 2019. Conference Name: IEEE Access.

Index

Appendix A

Media Content

A.0.1 Source code

The dissertation code is stored on a public GitHub account. We have six different repositories.

Two for the IMU sensing part consisting of the IMU vibration annotation mobile application and the second for the actual IMU sensing. We developed these using Flutter. The third repository holds the vision MobileNetSSD application. We also developed this application in Flutter, containing the TensorFlow Lite model. The fourth repository contains all the IMU experiments and the testing data set. The fifth stores the fusion reporting dashboard. We developed this application in Go and JavaScript. The sixth repository contains the Roboflow Colab vision training model .Table A.1 shows the applications and repository locations.

Table A.1: GitHub repositories

Application	Repository
Vibration annotation	https://github.com/anthonyscerri/IMUVibrationAnnotation
IMU sensing	https://github.com/anthonyscerri/IMUSensing
Vision MobileNetSSD	https://github.com/anthonyscerri/VisionMobilenetSSD
IMU Experiments	https://github.com/anthonyscerri/RoadAnomaly_IMU
Fusion reporting website	https://github.com/anthonyscerri/ModelFusionDashboard
Roboflow Colab vision model	https://github.com/anthonyscerri/RoboflowColabRoadAnomaly

A.0.2 Pre-compiled APKs

Additionally, the pre-compiled Android APK files for the smartphone applications can be downloaded from:

https://www.dropbox.com/sh/r9lkji37nb58vct/AACOj__hehO4sQ0pXI_jriBQa?dl=0

Appendix B

Installation Instructions

B.0.1 Installing the Roboflow Colab Jupyter file

To install the Roboflow Colab Jupyter file, please follow the below instructions:

1. Download the Jupyter file from:
   https://github.com/anthonyscerri/RoboflowColabRoadAnomaly

2. Upload the file to Colab

3. Run the notebook

B.0.2 Installing the Android APKs

To install the Android APKs on your Android smartphone, please follow the below instructions:

1. Download the desired application from the GitHub repository

2. Open the Settings app on your Android device.

3. In the Settings menu, tap Apps.

4. Tap Special app access (or Advanced > Special app access).

5. Tap Install unknown apps.

6. Select an app to use to install an APK file.

7. Tap the Allow from this source slider to allow APK files to be installed via that app.

B.0.3 Installing IMU experiments files

To install the IMU experiments files, please follow the below instructions:

1. Download the IMU experiments folder from:
   https://github.com/anthonyscerri/RoadAnomaly_IMU
2. Open the folder using your Jupyter editor
3. Run the notebooks

B.0.4 Installing Fusion reporting website

To install the Fusion reporting website, please follow the below instructions:

1. Download the Fusion reporting folder from:
   https://github.com/anthonyscerri/ModelFusionDashboard
2. Follow the Go installation notes: https://go.dev/doc/install
3. Run the Go application
4. A pre-compiled version is available online http://maltaroadanomaly.info:8080/map

Appendix C

User Manuals

C.0.1 Roboflow Colab Jupyter file user manual

1. Upload the Jupyter file to Colab
2. Run all the cells
3. The python script will automatically download the dissertation image files and annotations for the training

C.0.2 Android APKs file user manual

C.0.2.1 MobileNetSSD application

1. Follow the installation notes in subsection B.0.2
2. Run the application
3. Place the smartphone in a horizontal position
4. Note - There is a ten-second delay for the application to calibrate the onboard clock with the GPS atomic clocks.
5. Adjust the camera position as required.

C.0.2.2 MobileNetSSD application

1. Follow the installation notes in subsection B.0.2
2. Run the application

3. Affix the smartphone to the dashboard in a vertical position.
4. Note - There is a ten-second delay for the application to calibrate the onboard clock with the GPS atomic clocks.

C.0.2.3 Vibration annotation application

1. Follow the installation notes in subsection B.0.2 to download the APK file
2. Run the application

C.0.3 IMU experiments files file user manual

1. Follow the installation notes in subsection B.0.2 to download the RoadAnomaly_IMU repository
2. Run the Jupyter files
   a) PHML_1_featureSelection.ipynb contains the sequential feature selector code as described in subsection 4.2.1
   b) PHML_2_MLCodeBinary.ipynb contains the code of the LR, SVM, KNN, RF and NB binary classification
   c) PHML_2_MLCodeMultiClass.ipynb contains the code of the LR, SVM, KNN, RF and NB multiclass classification
   d) PHML_4_ANN.ipynb contains the code of the neural network model for binary classification
   e) PHML_4_ANN_multiclass.ipynb contains the code of the neural network model for multiclass classification
   f) PHML_5_KMEANS.ipynb contains the code of the K-means experiments
   g) Live_PHML_2_MLCodeMultiClass.ipynb and Live_PHML_3_MLCodeMultiClass.ipynb contain the field testing anomaly detection and fusion code to build the reporting dashboard
3. Additional instructions are included in the Jupyter notebook files

C.0.4 Fusion reporting website file user manual

1. Follow the installation notes in subsection B.0.2

2. Open static/app.js with your preferred text editor and replace the URL strings 'http://maltaroadanomaly.info' with your local IP address and save the file
3. From the root folder
4. To compile the application:
   a) $ go build fusion.go
   b) $ ./fusion
5. Point your browser to HTTP://<your IP address>:8080/map
6. A pre-compiled version is available online http://maltaroadanomaly.info:8080/map

Appendix D

Additional Test results

D.1 Test case two: Depression Left VS Depression Left

Table D.1: Test case two - SVM Features used

Feature	Sensor	Axis	Feature	Domain
xa_std	A	X	Standard deviation	Time
yg_max	G	Y	Maximum	Time
yg_maxmin_diff	G	Y	Max-Min difference	Time
ya_neg_count	A	Y	Negtive count	Time
xa_skewness	A	X	Skewness	Time
yg_mean_fft	G	Y	Mean	Freq.
ya_min_fft	A	Y	Minimum	Freq.
yg_median_fft	G	Y	Median	Freq.
yg_mad_fft	G	Y	Median absolute deviation	Freq.
yg_IQR_fft	G	Y	Interquartile range	Freq.

Table D.2: Test case two - KNN Features used

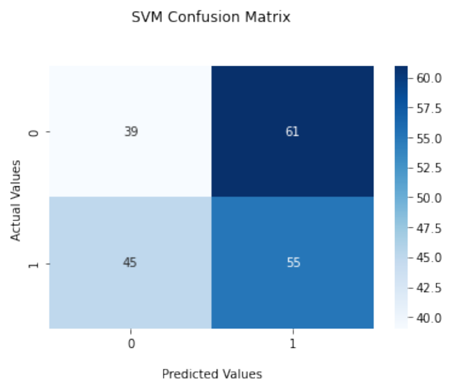
Feature	Sensor	Axis	Feature	Domain
xa_mean	A	X	Mean	Time
ya_var	A	Y	Variance	Time
yg_max	G	Y	Maximum	Time
xg_median	G	X	Median	Time
ya_neg_count	A	Y	Negtive count	Time
ya_pos_count	A	Y	Positive count	Time
yg_mad_fft	G	Y	Median absolute deviation	Freq.
yg_IQR_fft	A	Y	Interquartile range	Freq.
yg_peak_count_fft	G	Y	Number of peaks	Freq.
za_kurtosis_fft	A	Z	Kurtosis	Freq.

Table D.3: Test case two - RF Features used

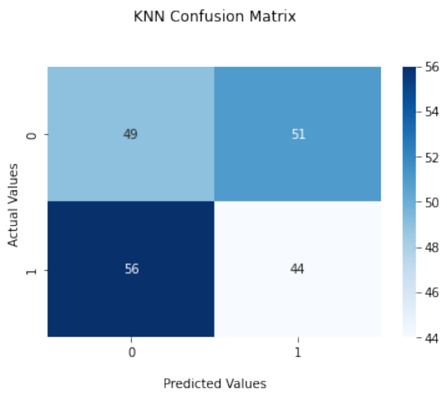
Feature	Sensor	Axis	Feature	Domain
yg_var	G	Y	Variance	Time
xg_mad	G	Y	Median absolute deviation	Time
xg_above_mean	G	X	Values above mean	Time
yg_above_mean	G	Y	Values above mean	Time
yg_peak_count	G	Y	Number of peaks	Time
ya_energy	A	Y	Energy	Time
xg_mean_fft	G	X	Mean	Freq.
ya_min_fft	A	Y	Minimum	Freq.
zg_maxmin_diff_fft	G	Z	Max-Min difference	Freq.
xg_median_fft	G	X	Median	Freq.

Table D.4: Test case two - NB Features used:

Feature	Sensor	Axis	Feature	Domain
ya_mad	A	Y	Median absolute deviation	Time
za_above_mean	A	Z	Values above mean	Time
yg_above_mean	G	Y	Values above mean	Time
yg_peak_count	G	Y	Number of peaks	Time
zg_skewness	G	Z	Skewness	Time
za_kurtosis	A	Z	Kurtosis	Time
avg_result_gyro	G	X, Y and Z	Average resultant	Time
zg_min_fft	G	Z	Minimum	Frequency
xg_peak_count_fft	G	X	Number of peaks	Freq.
xg_skewness_fft	G	X	Skewness	Freq.

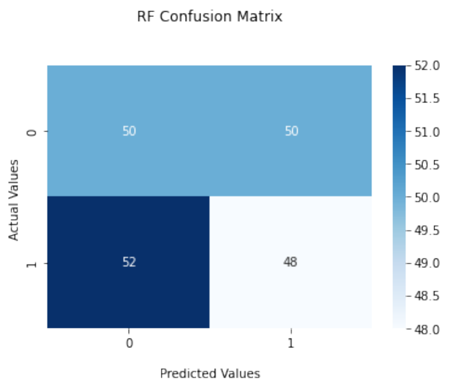


(a) SVM Left and Right confusion matrix

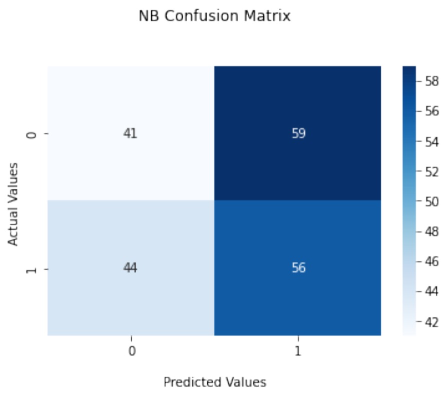


(b) KNN Left and Right confusion matrix

Figure D.1: Test case one: LR, SVM and KNN confusion matrix



(a) RF Left and Right confusion matrix



(b) NB Left and Right confusion matrix

Figure D.2: Test case two: RF and NB confusion matrix

D.2 Test case three: Depression right, depression left and smooth surface

Table D.5: Test case three - SVM features used

Feature	Sensor	Axis	Feature	Domain
za_var	A	Z	Variance	Time
xa_aad	A	X	Average absolute differance	Time
ya_aad	A	Y	Average absolute differance	Time
xg_pos_count	G	X	Positive count	Time
xa_above_mean	A	X	Values above mean	Time
xa_std_fft	A	x	Standard deviation	Freq.
za_aad_fft	A	Z	Average absolute differance	Freq.
zg_min_fft	G	z	Minimum	Freq.
xa_mad_fft	A	x	Median absolute deviation	Freq.
xg_peak_count_fft	G	X	Number of peaks	Freq.

Table D.6: Test case three - KNN features used

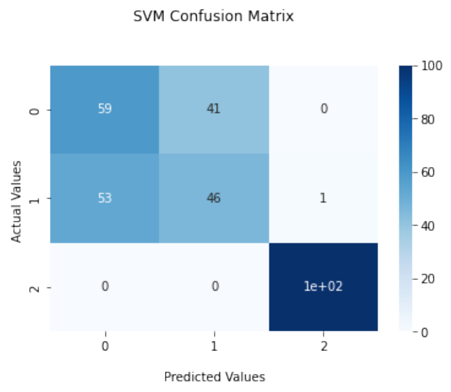
Feature	Sensor	Axis	Feature	Domain
zg_mean	G	Z	Mean	Time
zg_var	G	Z	Variance	Time
za_max	A	Z	Maximum	Time
ya_neg_count	A	Y	Negative count	Time
zg_neg_count	G	Z	Negative count	Time
zg_pos_count	G	Z	Positive count	Time
yg_peak_count	G	Y	Number of peaks	Time
xa_skewness	A	X	Skewness	Time
smaa	A	X and Y	Signal magnitude area	Time
ya_mad_fft	A	Y	Median absolute deviation	Freq.

Table D.7: Test case three - Random Forest features used

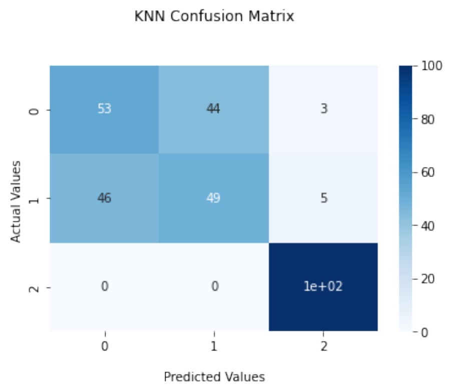
Feature	Sensor	Axis	Feature	Domain
xg_median	G	X	Median	Time
xa_mad	A	X	Median absolute deviation	Time
xg_pos_count	G	X	Positive count	Time
za_peak_count	A	Z	Number of peaks	Time
za_skewness	A	Z	Skewness	Time
za_kurtosis	A	Z	Kurtosis	Time
za_energy	A	Z	Energy	Time
smaa	G	X and Y	Signal magnitude area	Time
yg_peak_count_fft	G	Y	Number of peaks	Freq.
smaa_fft	A	Z	Signal magnitude area	Freq.

Table D.8: Test case three - Naive Bayes features used

Feature	Sensor	Axis	Feature	Domain
zg_mean	G	Z	Mean	Time
yg_max	G	Y	Maximum	Time
xg_median	G	X	Median	Time
xa_IQR	A	X	Interquartile range	Time
zg_IQR	G	Z	Interquartile range	Time
xa_above_mean	A	X	Values above mean	Time
za_above_mean	A	Z	Values above mean	Time
zg_skewness	G	Z	Skewness	Time
yg_std_fft	G	Y	Standard deviation	Freq.
xa_IQR_fft	A	X	Interquartile range	Freq.

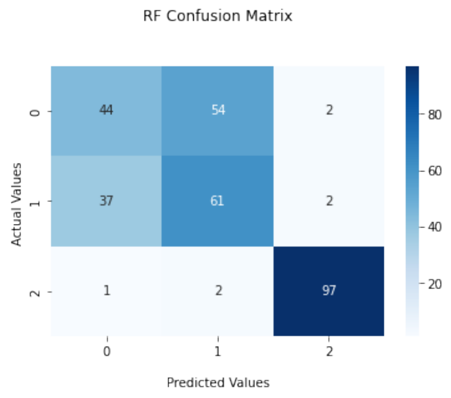


(a) SVM Depression - Smooth confusion matrix

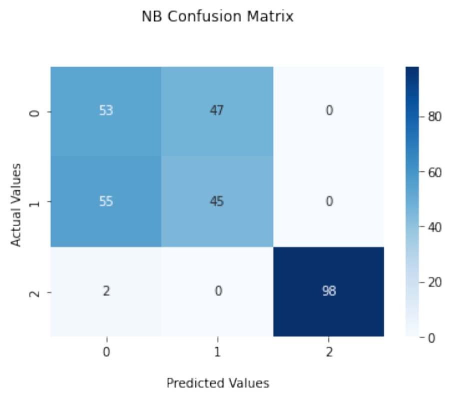


(b) KNN Depression - Smooth confusion matrix

Figure D.3: Test case two: LR, SVM and KNN confusion matrix



(a) RF Depression - Smooth confusion matrix



(b) NB Left and Right confusion matrix

Figure D.4: Test case two: RF and NB confusion matrix

D.3 Test case four: 50% left and 50% right depression grouped and Smooth surface

Table D.9: Test case four - LR features used

Feature	Sensor	Axis	Feature	Domain
yg_mean	G	Y	Mean	Time
zg_mean	G	Z	Mean	Time
ya_var	A	Y	Variance	Time
xa_aad	A	X	Average absolute difference	Time
zg_median	G	Z	Median	Time
zg_neg_count	G	Z	Negative count	Time
zg_pos_count	G	Z	Positive count	Time
ya_energy	A	Y	Energy	Time
smaa	G	X and Y	Signal magnitude area	Time
zg_peak_count_fft	G	Z	Number of peaks	Freq.

Table D.10: Test case four - SVM features used

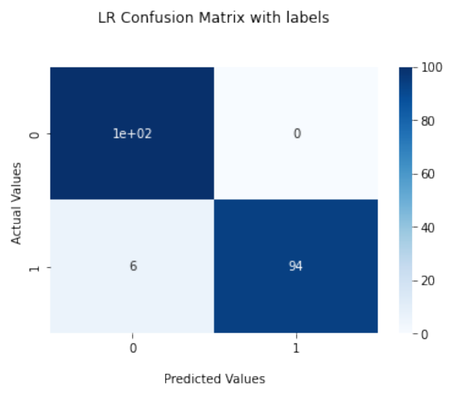
Feature	Sensor	Axis	Feature	Domain
xa_mean	A	X	Mean	Time
ya_mean	A	Y	Mean	Time
yg_mean	G	Y	Mean	Time
zg_mean	G	Z	Mean	Time
ya_var	A	Y	Variance	Time
za_var	A	Z	Variance	Time
xg_var	G	X	Variance	Time
yg_var	G	Y	Variance	Time
zg_var	G	Z	Variance	Time
xa_IQR	A	X	Interquartile range	Time

Table D.11: Test case four - KNN features used

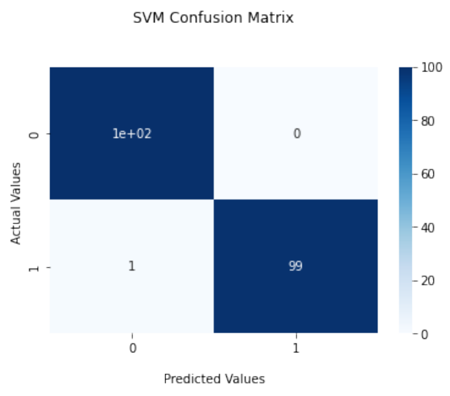
Feature	Sensor	Axis	Feature	Domain
xa_mean	A	X	Mean	Time
zg_mean	G	Z	Mean	Time
xa_var	A	X	Variance	Time
ya_var	A	Y	Variance	Time
za_var	A	Z	Variance	Time
xg_var	G	X	Variance	Time
zg_var	G	Z	Variance	Time
xa_std	A	X	Standard deviation	Time
xg_std	G	X	Standard deviation	Time
xa_mad	A	X	Median absolute deviation	Time

Table D.12: Test case four - Random forest features used

Feature	Sensor	Axis	Feature	Domain
xa_mean	A	X	Mean	Time
ya_mean	A	Y	Mean	Time
za_mean	A	Z	Mean	Time
xg_mean	G	X	Mean	Time
yg_mean	G	Y	Mean	Time
zg_mean	G	Z	Mean	Time
xa_var	A	X	Variance	Time
ya_var	A	Y	Variance	Freq.
yg_var	G	Y	Variance	Freq.
xa_mad	A	X	Median absolute deviation	Freq.

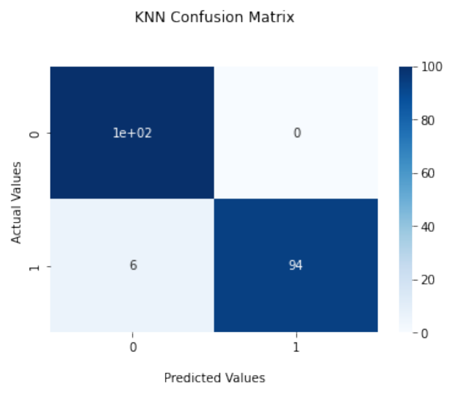


(a) LR Speed 50 % left and 50% right depression grouped and Smooth surface confusion matrix

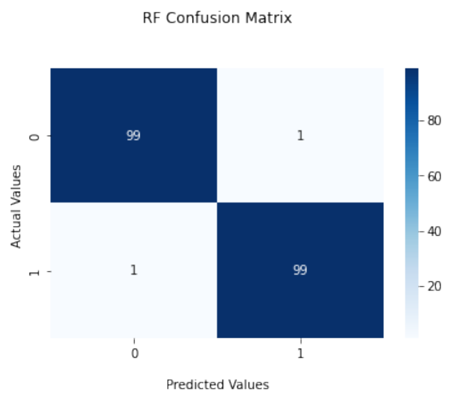


(b) SVM Speed 50 % left and 50% right depression grouped and Smooth surface confusion matrix

Figure D.5: Test case three: LR, SVM and KNN confusion matrix



(a) KNN Speed 50 % left and 50% right depression grouped and Smooth surface confusion matrix



(b) RF 50 % left and 50% right depression grouped and Smooth surface confusion matrix

Figure D.6: Test case three: RF and NB confusion matrix

D.4 Test case five: Speed bump and Smooth surface

Table D.13: Test case five - LR Features used

Feature	Sensor	Axis	Feature	Domain
xg_mean	G	X	Mean	Time
yg_mean	G	Y	Mean	Time
zg_mean	G	Z	Mean	Time
xa_var	A	X	Variance	Time
ya_var	A	Y	Variance	Time
xg_var	G	X	Variance	Time
yg_var	G	Y	Variance	Time
xa_std	A	X	Standard deviation	Time
za_std	A	Z	Standard deviation	Time
smaa	A	X and Y	Signal magnitude area	Time

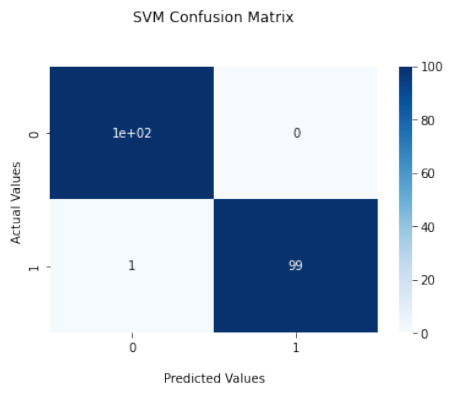
Table D.14: Test case five - SVM Features used

Feature	Sensor	Axis	Feature	Domain
xa_mean	A	X	Mean	Time
za_mean	A	Z	Mean	Time
xg_mean	G	X	Mean	Time
yg_mean	G	Y	Mean	Time
zg_mean	G	Z	Mean	Time
xa_var	A	X	Variance	Time
ya_var	A	Y	Variance	Time
yg_var	G	Y	Variance	Time
zg_var	G	Z	Variance	Time
xa_aad	A	X	Average absolute difference	Time

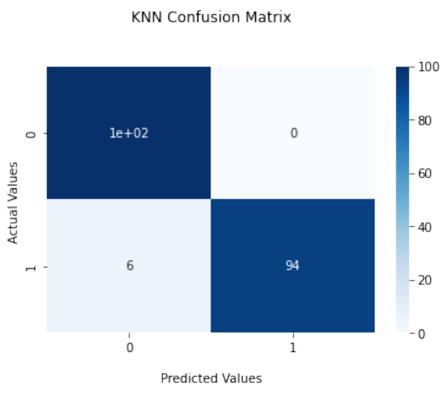
Table D.15: Test case five - KNN Features used

Feature	Sensor	Axis	Feature	Domain
xa_mean	A	X	Mean	Time
xg_mean	G	X	Mean	Time
yg_mean	G	Y	Mean	Time
zg_mean	G	Z	Mean	Time
xa_var	A	X	Variance	Time
ya_var	A	Y	Variance	Time
za_var	A	Z	Variance	Time
xg_var	G	X	Variance	Time
yg_var	G	Y	Variance	Time
xa_aad	A	X	Average absolute difference	Time

D.5 Confusion matrices

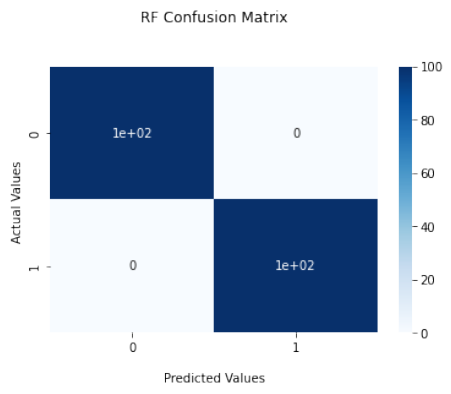


(a) SVM Speed bump - smooth confusion matrix

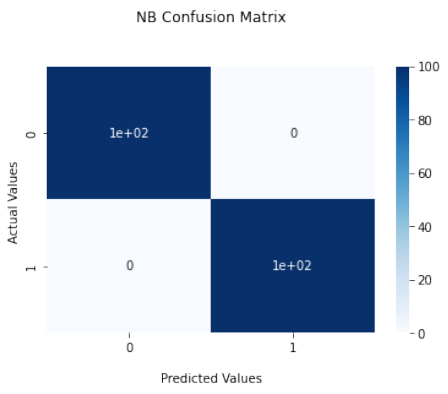


(b) KNN Speed bump - smooth confusion matrix

Figure D.7: Test case four: LR, SVM and KNN confusion matrix



(a) RF Speed bump - smooth confusion matrix



(b) NB Speed bump - smooth confusion matrix

Figure D.8: Test case three: RF and NB confusion matrix

D.6 Test case six: Depression, speed bump and smooth

Table D.16: Test case six - SVM Features used

Feature	Sensor	Axis	Feature	Domain
xg_mean	G	X	Mean	Time
zg_var	G	Z	Variance	Time
xa_std	A	X	Standard deviation	Time
xg_aad	G	X	Average absolute difference	Time
za_min	A	Z	Minimum	Time
yg_mad	G	Y	Median absolute deviation	Time
yg_energy	G	Y	Energy	Time
smaa	A	X, Y and Z	Signal magnitude area	Time
zg_mean_fft	G	Z	Mean	Freq.
xg_peak_count_fft	G	X	Number of peaks	Freq.

Table D.17: Test case six - KNN Features used

Feature	Sensor	Axis	Feature	Domain
za_var	A	Z	Variance	Time
zg_var	G	Z	Variance	Time
za_std	A	Z	Standard deviation	Time
xa_max	A	X	Maximum	Time
xa_mad	A	X	Median absolute deviation	Time
zg_IQR	G	Z	Interquartile range	Time
avg_result_accl	A	X, Y and Z	Average resultant	Time
xg_max_fft	G	X	Maximum	Freq.
xg_maxmin_diff_fft	G	X	Max-Min difference	Freq.
yg_kurtosis_fft	G	Y	Kurtosis	Freq.

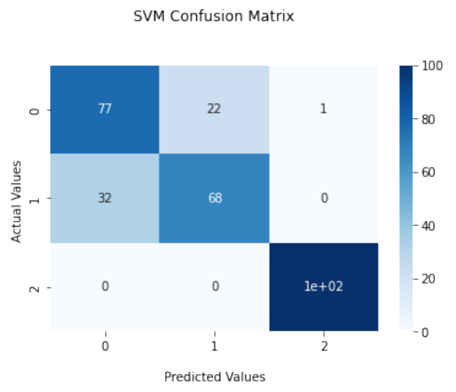
Random Forest Features used:

Table D.18: Test case six - Random Forest Features used

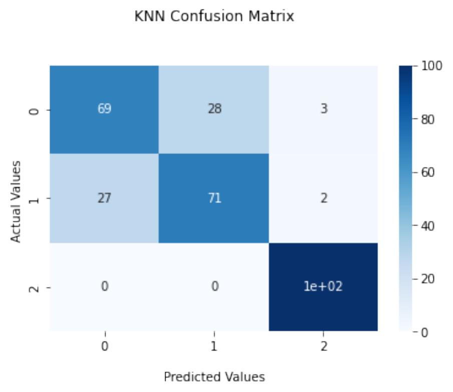
Feature	Sensor	Axis	Feature	Domain
za_mean	A	Z	Mean	Time
xg_var	G	X	Variance	Time
xa_mad	A	X	Median absolute deviation	Time
yg_mad	G	Y	Median absolute deviation	Time
ya_peak_count	A	Y	Number of peaks	Time
xa_skewness	A	X	Skewness	Time
xa_max_ftt	A	X	Maximum	Freq.
xg_max_fft	G	X	Maximum	Freq.
zg_median_fft	G	Z	Median	Freq.
za_peak_count_fft	A	Z	Number of peaks	Freq.

Table D.19: Test case six - Naive Bayes Features used

Feature	Sensor	Axis	Feature	Domain
xg_mean	G	X	Mean	Time
xa_aad	A	X	Average absolute difference	Time
zg_median	G	Z	Median	Time
ya_IQR	A	Y	Interquartile range	Time
yg_std_fft	G	Y	Standard deviation	Freq.
xg_maxmin_diff_fft	G	X	Max-Min difference	Freq.
za_median_fft	A	Z	Median	Freq.
xg_peak_count_fft	G	X	Number of peaks	Freq.
yg_peak_count_fft	G	Y	Number of peaks	Freq.
za_skewness_fft	A	Z	Skewness	Freq.

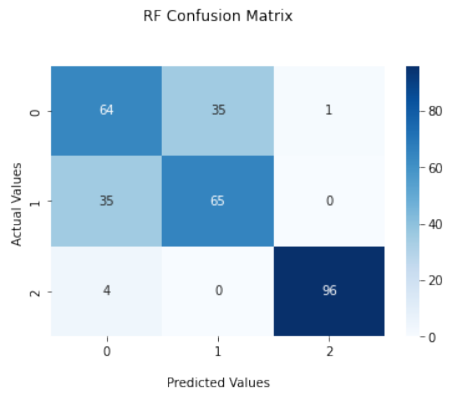


(a) SVM Speed bump - smooth confusion matrix

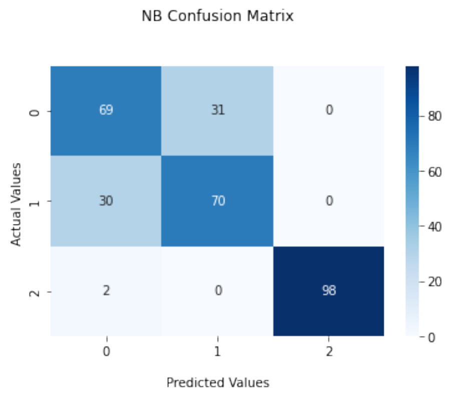


(b) SVM Speed bump - smooth confusion matrix

Figure D.9: Test case four: LR, SVM and KNN confusion matrix



(a) RF Speed bump - smooth confusion matrix



(b) NB Speed bump - smooth confusion matrix

Figure D.10: Test case three: RF and NB confusion matrix

D.7 IMU Number of samples testing - 50 VS 75 VS 100 records

Alg	*Class*	*Precision*	*Recall*	*F1score*	*Accuracy*
LR	0	0.66	0.58	0.62	0.64
	1	0.62	0.70	0.66	
SVM	0	0.54	0.42	0.47	0.53
	1	0.52	0.64	0.58	
KNN	0	0.59	0.54	0.56	0.58
	1	0.57	0.62	0.60	
RF	0	0.66	0.66	0.66	0.50
	1	0.66	0.66	0.66	
NB	0	0.64	0.46	0.53	0.60
	1	0.61	0.60	0.59	

(a) 50 records per class

Alg	*Class*	*Precision*	*Recall*	*F1score*	*Accuracy*
LR	0	0.65	0.65	0.65	0.65
	1	0.65	0.65	0.65	
SVM	0	0.61	0.56	0.58	0.60
	1	0.59	0.64	0.62	
KNN	0	0.57	0.57	0.57	0.57
	1	0.57	0.57	0.57	
RF	0	0.55	0.53	0.54	0.55
	1	0.55	0.56	0.55	
NB	0	0.58	0.51	0.54	0.57
	1	0.56	0.63	0.59	

(b) 75 records per class

Alg	*Class*	*Precision*	*Recall*	*F1score*	*Accuracy*
LR	0	0.69	0.66	0.68	0.69
	1	0.68	0.71	0.69	
SVM	0	0.46	0.39	0.42	0.47
	1	0.47	0.55	0.51	
KNN	0	0.47	0.49	0.48	0.47
	1	0.46	0.44	0.45	
RF	0	0.49	0.50	0.50	0.49
	1	0.49	0.48	0.48	
NB	0	0.48	0.41	0.44	0.48
	1	0.49	0.56	0.52	

(c) 100 records per class

Table D.20: IMU data set size test - Left (1) VS Right (0)

Alg	*Class*	*Precision*	*Recall*	*F1Score*	*Accuracy*
LR	1	0.71	0.74	0.73	0.80
	2	0.72	0.66	0.69	
	5	0.96	1.00	0.98	
SVM	1	0.56	0.70	0.62	0.70
	2	0.59	0.40	0.48	
	5	0.93	1.00	0.96	
KNN	1	0.55	0.58	0.56	0.67
	2	0.54	0.44	0.48	
	5	0.89	1.00	0.94	
RF	1	0.59	0.72	0.65	0.73
	2	0.62	0.48	0.54	
	5	0.98	0.98	0.98	
NB	1	0.61	0.70	0.65	0.75
	2	0.64	0.56	0.60	
	5	1.00	0.98	0.99	

(a) 50 records per class

Alg	*Class*	*Precision*	*Recall*	*F1Score*	*Accuracy*
LR	1	0.60	0.65	0.63	0.74
	2	0.62	0.56	0.59	
	5	0.99	1.00	0.99	
SVM	1	0.56	0.61	0.59	0.70
	2	0.58	0.49	0.53	
	5	0.95	1.00	0.97	
KNN	1	0.57	0.53	0.55	0.70
	2	0.56	0.56	0.56	
	5	0.94	1.00	0.97	
RF	1	0.59	0.56	0.48	0.71
	2	0.57	0.59	0.58	
	5	0.95	0.97	0.96	
NB	1	0.56	0.60	0.58	0.71
	2	0.57	0.53	0.55	
	5	1.00	0.99	0.99	

(b) 75 records per class

Alg	*Class*	*Precision*	*Recall*	*F1Score*	*Accuracy*
LR	1	0.65	0.68	0.67	0.76
	2	0.65	0.62	0.64	
	5	0.98	0.99	0.99	
SVM	1	0.53	0.59	0.56	0.68
	2	0.53	0.46	0.49	
	5	0.99	1.00	1.00	
KNN	1	0.54	0.53	0.53	0.67
	2	0.53	0.49	0.51	
	5	0.93	1.00	0.96	
RF	1	0.54	0.44	0.48	0.67
	2	0.52	0.61	0.56	
	5	0.96	0.97	0.97	
NB	1	0.48	0.53	0.50	0.65
	2	0.49	0.45	0.47	
	5	1.00	0.98	0.99	

(c) 100 records per class

Table D.21: IMU data set size test - Left (1), right (2) and smooth (5)

Alg	*Class*	*Precision*	*Recall*	*F1score*	*Accuracy*
LR	0	0.93	1.00	0.96	0.96
	1	1.00	0.92	0.96	
SVM	0	1.00	1.00	1.00	1.00
	1	1.00	1.00	1.00	
KNN	0	0.93	1.00	0.96	0.97
	1	1.00	0.92	0.96	
RF	0	1.00	1.00	1.00	1.00
	1	1.00	1.00	1.00	
NB	0	1.00	0.98	0.99	0.99
	1	0.98	1.00	0.99	

(a) 50 records per class

Alg	*Class*	*Precision*	*Recall*	*F1score*	*Accuracy*
LR	0	0.95	0.99	0.97	0.97
	1	0.99	0.95	0.97	
SVM	0	0.99	1.00	0.99	0.99
	1	1.00	0.99	0.99	
KNN	0	0.93	1.00	0.96	0.96
	1	1.00	0.92	0.96	
RF	0	0.99	0.99	0.99	0.99
	1	0.99	0.99	0.99	
NB	0	1.00	0.99	0.99	0.99
	1	0.99	1.00	0.99	

(b) 75 records per class

Alg	*Class*	*Precision*	*Recall*	*F1score*	*Accuracy*
LR	0	0.94	1.00	0.97	0.97
	1	1.00	0.94	0.97	
SVM	0	0.99	1.00	1.00	0.99
	1	1.00	0.99	0.99	
KNN	0	0.94	1.00	0.97	0.97
	1	1.00	0.94	0.97	
RF	0	0.99	0.99	0.99	0.99
	1	0.99	0.99	0.99	
NB	0	1.00	1.00	1.00	1.00
	1	1.00	1.00	1.00	

(c) 100 records per class

Table D.22: IMU data set size test - Left & right (0) and smooth (1)

Alg	*Class*	*Precision*	*Recall*	*F1score*	*Accuracy*
LR	0	0.96	1.00	0.98	0.98
	1	1.00	0.96	0.98	
SVM	0	0.98	1.00	0.99	0.99
	1	1.00	0.98	0.99	
KNN	0	0.89	1.00	0.94	0.94
	1	1.00	0.88	0.94	
RF	0	0.98	1.00	0.99	0.99
	1	1.00	0.98	0.99	
NB	0	1.00	0.98	0.99	0.99
	1	0.98	1.00	0.99	

(a) 50 records per class

Alg	*Class*	*Precision*	*Recall*	*F1score*	*Accuracy*
LR	0	0.97	1.00	0.99	0.99
	1	1.00	0.97	0.99	
SVM	0	0.99	1.00	0.99	0.99
	1	1.00	0.99	0.99	
KNN	0	0.91	1.00	0.96	0.95
	1	1.00	0.91	0.95	
RF	0	0.99	0.99	0.99	0.99
	1	0.99	0.99	0.99	
NB	0	1.00	1.00	1.00	1.00
	1	1.00	1.00	1.00	

(b) 75 records per class

Alg	*Class*	*Precision*	*Recall*	*F1score*	*Accuracy*
LR	0	0.97	1.00	0.99	0.98
	1	1.00	0.97	0.98	
SVM	0	0.99	1.00	1.00	0.99
	1	1.00	0.99	0.99	
KNN	0	0.94	1.00	0.97	0.97
	1	1.00	0.94	0.97	
RF	0	1.00	1.00	1.00	1.00
	1	1.00	1.00	1.00	
NB	0	1.00	1.00	1.00	1.00
	1	1.00	1.00	1.00	

(c) 100 records per class

Table D.23: IMU data set size test - Speed bump and smooth

Alg	*Class*	*Precision*	*Recall*	*F1Score*	*Accuracy*
LR	3	0.85	0.92	0.88	0.89
	4	0.90	0.76	0.83	
	5	0.93	1.00	0.96	
SVM	3	0.81	0.70	0.75	0.84
	4	0.75	0.82	0.78	
	5	0.96	1.00	0.98	
KNN	3	0.82	0.74	0.78	0.85
	4	0.80	0.80	0.80	
	5	0.91	1.00	0.95	
RF	3	0.72	0.84	0.78	0.84
	4	0.82	0.72	0.77	
	5	1.00	0.96	0.98	
NB	3	0.75	0.80	0.78	0.85
	4	0.79	0.76	0.78	
	5	1.00	0.98	0.99	

(a) 50 records per class

Alg	*Class*	*Precision*	*Recall*	*F1Score*	*Accuracy*
LR	3	0.75	0.71	0.73	0.82
	4	0.78	0.75	0.76	
	5	0.90	1.00	0.95	
SVM	3	0.64	0.71	0.67	0.76
	4	0.68	0.59	0.63	
	5	0.97	1.00	0.99	
KNN	3	0.62	0.57	0.60	0.73
	4	0.61	0.63	0.62	
	5	0.95	1.00	0.97	
RF	3	0.52	0.47	0.49	0.68
	4	0.52	0.59	0.55	
	5	0.99	0.97	0.98	
NB	3	0.58	0.59	0.58	0.72
	4	0.59	0.59	0.59	
	5	1.00	0.99	0.99	

(b) 75 records per class

Alg	*Class*	*Precision*	*Recall*	*F1Score*	*Accuracy*
LR	3	0.86	0.79	0.82	0.88
	4	0.83	0.85	0.84	
	5	0.95	1.00	0.98	
SVM	3	0.71	0.77	0.74	0.82
	4	0.76	0.68	0.72	
	5	0.99	1.00	1.00	
KNN	3	0.72	0.69	0.70	0.80
	4	0.72	0.71	0.71	
	5	0.95	1.00	0.98	
RF	3	0.62	0.64	0.63	0.75
	4	0.65	0.65	0.65	
	5	0.99	0.96	0.97	
NB	3	0.68	0.69	0.69	0.79
	4	0.69	0.70	0.70	
	5	1.00	0.98	0.99	

(c) IMU data set size test - 100 records per class

Table D.24: IMU data set size test - Depression, speed bump, smooth