MACHINE LEARNING APPROACHES TO ROAD SURFACE ANOMALY ASSESSMENT USING SMARTPHONE SENSORS

A Thesis

by

AKANKSH BASAVARAJU

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

August 2018

Major Subject: Electrical Engineering

ABSTRACT

Road surface quality is an essential component of roadway infrastructure that leads to better driving standards and reduces risk of traffic accident. Traditional road condition monitoring systems fall short of current need for quick responses to maintain road quality. Several alternative systems have been proposed that utilize sensors mounted on vehicles and with the ubiquitous use of smartphone for personal use and navigation, smartphone based road condition assessment has gained prominence.

We propose to analyze different multiclass supervised machine learning techniques to effectively classify road surface conditions using accelerometer, gyroscope and GPS data collected from smartphones. Our work focusses on classification of three main class labels- smooth road, pothole and deep transverse cracks. We investigate our conjecture that using features from all three axes of the sensors provide more accurate results as compared to using features from only one axis. We also investigate the performance of deep neural networks to classify road conditions with and without explicit manual feature extraction. Our results consistently show that models trained with features from all axes of the smartphone sensors perform better than models that use only one axis. This shows that there is information in the vibration signals along all three axis for road anomalies. We also observe that the use of neural networks provide significantly accurate data classification. The approaches discussed here can be implemented on a larger scale to monitor road for defects that present a safety risk to commuters as well as provide maintenance information to relevant authorities.

DEDICATION

I take great pleasure in dedicating this thesis to my family, for the immense love and support they have provided me every single day of my life. They have been by my side through not only my times of triumph and joy but also my times of failure and sadness. To my dad, who has worked hard his whole life and made numerous sacrifices to ensure that we could live a better life. He has been a source of guidance and inspiration that makes me a better person everyday. To my mom, who taught me to be compassionate, respectful and responsible, qualities that I am forever grateful for. To my little brother, who I have loved from the moment he was born. They are the people in my life I want to make the most proud of me.

ACKNOWLEDGEMENTS

This thesis is not just the result of my own effort but also the result of the love and support of numerous people in my life. I want to take this opportunity to show them the gratitude that they deserve and to show them how much their support has meant to me.

I would like to thank the committee chair, Dr. Ji, the committee co-chair, Dr. Wright, and the committee members, Dr. Serpedin, Dr. Du, and Dr. Huang for their guidance and support throughout the course of this research. I extend my gratitude to Kun Tan, Michael Saulnier and Dillon Knox for their support in gathering and organizing the data that was essential for this thesis.

I want to thank my friends, Lekhya, Sakshi, Deepashree, Sachin, Lohith, Manosha and many more, for being amazing people and providing me with support and encouragement throughout my Masters journey. I want to especially thank Prithvi and Aishwarya, for being there for me when I probably needed them the most. I also want to thank my roommate, Shubham, for being the kind, supportive and understanding person he is.

Finally, I want to make a very special mention of immense gratitude and love for Yerania Hernandez, who has been a source of light whenever I was in a dark place. She has shown me unconditional kindness, encouragement and belief. She has been by my side whenever I needed her and has helped me achieve what I have achieved today. Thank you, for making me work when I had to and for making me laugh and smile even in times when I thought I didn’t want to.

CONTRIBUTORS AND FUNDING SOURCES

This work was supervised by a thesis committee consisting of my advisor, Dr. Jim Ji, and committee members, Dr. Steven Wright, Dr. Erchin Serpedin, Dr. Jing Du, and Dr. Jeff Huang. All work for the thesis was completed independently by the student with support from Michael Saulnier, Dillon Knox and Kun Tan, students at Texas A&M University.

There are no outside funding contributions to acknowledge related to the research and compilation of this document.

NOMENCLATURE

FPS	Frames per Second
SVM	Support Vector Machine
NN	Neural Network
TxDOT	Texas Department of Transportation
Tanh	Tan Hyperbolic
ReLU	Rectified Linear Unit
GIS	Geographic Information System

TABLE OF CONTENTS

Page

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

Road condition monitoring is a challenging worldwide problem in the field of transportation infrastructure [1] [2]. Poor road surface conditions lead to high repair and maintenance costs, creates a risk of damage to vehicles and increases chances of traffic accidents. Thousands of people are hurt or killed each year on roads and highways due to poor road quality [3]. In 2015, the United States Congress passed the Surface Transportation Reauthorization and Reform Act for the maintenance of federal highways over a five year period with a budget of $46 billion per year. In their survey they noted that nearly 10,000 traffic fatalities each year involve poor road conditions [4].

Maintaining good road quality is therefore essential not only to support an efficient road network but also to reduce traffic accidents. However, road surface maintenance is challenging due to weather conditions, heavy traffic and high costs of manpower. Due to the need of frequent repairs to prevent road quality from deteriorating, a reliable and low-latency road condition monitoring system is required. Traditional monitoring systems collect data from vehicles equipped with expensive road monitoring sensors such as LIDAR and Ground Penetrating Radar (GPR) [5]. Various equipment used for measuring road conditions is surveyed in [6]. However these equipment are expensive, costing between $8,000 and $220,000. Due to limitations in cost, such systems cannot be effectively deployed on a large scale road network to regularly check for need for repairs. Another approach to road maintenance is to visually inspect road conditions and record data concerning the condition of pavements. This can be used by itself or in combination with ride quality, structural adequacy, skid resistance, climate, and traffic data to assist in describing the overall condition of the state-maintained highway system [7].

According to the TxDOT's Pavement Management Information System Rater's Manual, pavement distress for asphalt flexible pavement sections are mainly categorized into eight types:

1) Rutting
2) Patching
3) Block Cracking
4) Alligator Cracking
5) Longitudinal Cracking
6) Transverse Cracking
7) Raveling
8) Potholes

Rutting is the road condition that is a longitudinal surface depression in a wheel path caused by consolidation or lateral movement of the pavement material due to traffic loads as shown in Figure 1. Rutting is classified from shallow rutting to Failure rutting that ranges from a depression of 0.25 inches to 2.0 inches or greater respectively. Patches are repairs made to pavement distress as shown in Figure 2.

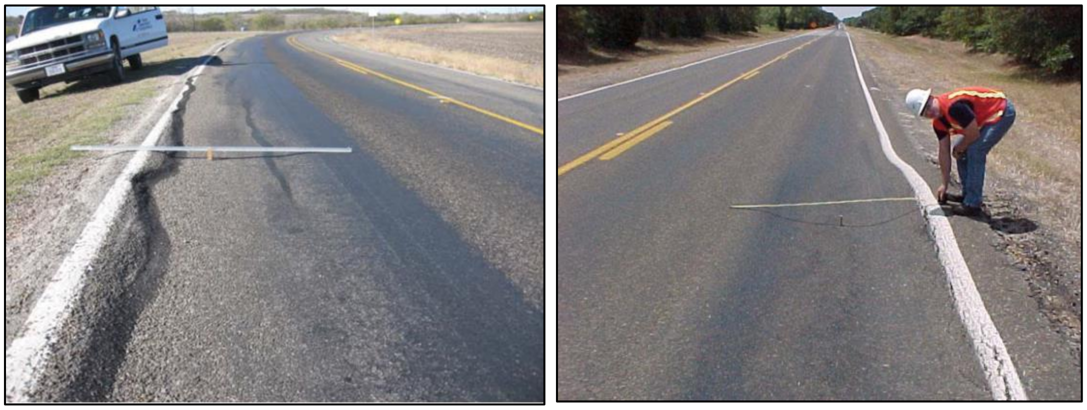


Figure 1. Instance of Deep Rutting and Severe Rutting. Reprinted from [7].

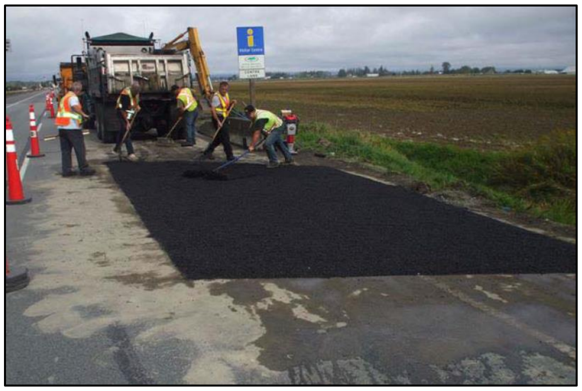


Figure 2. Patching of Road to Repair Pavement Distress. Reprinted from [7].

Block cracks form irregular blocks and are a result of age hardening of the asphalt coupled with shrinkage of asphalt concrete during cold weather. They consist of interconnecting cracks that divide the pavement surface into approximately rectangular pieces. Alligator cracks also known as fatigue cracks, is formed whenever the pavement is repeatedly flexed under traffic load. Although they resemble block cracks, they are smaller in shape and resemble patterns found on an alligator’s skin. Instances of block cracking and Alligator cracking is shown in Figure 3.

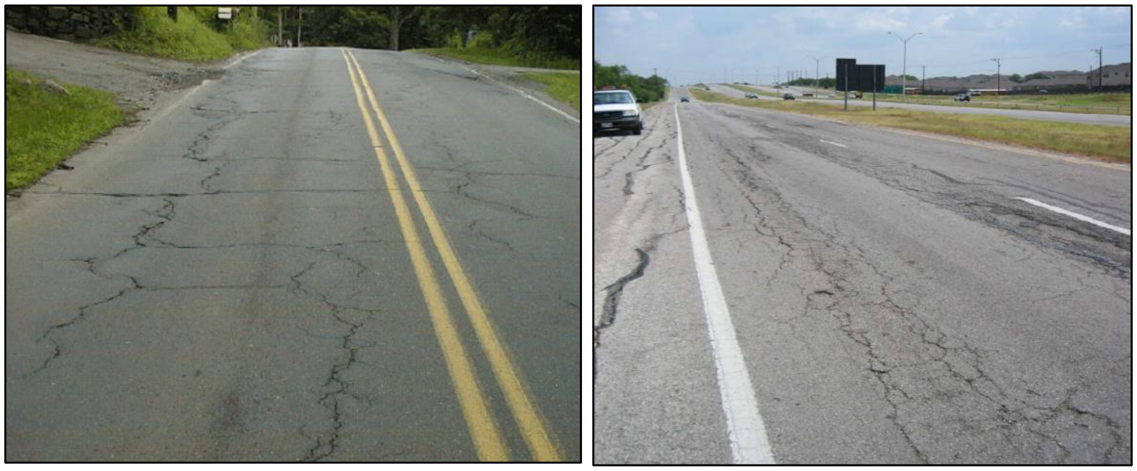


Figure 3. Instance of Block Cracking and Alligator Cracking. Reprinted from [7].

Longitudinal cracks occur as a result of poorly constructed paving lane joints, thermal shrinkage, inadequate support or reflection from underlying layers. In contrast, Transverse cracking consists of cracks or breaks that travel at right angles to the pavement centerline. They usually create unevenness in the pavement surface that causes faster pavement deterioration.

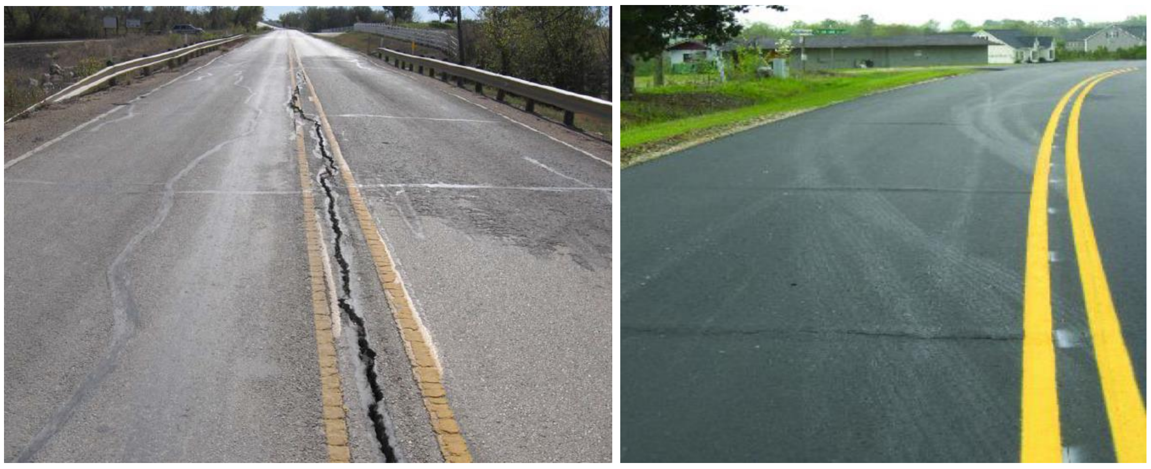


Figure 4. Instance of Longitudinal Cracking and Transverse Cracking. Reprinted from [7].

Raveling is the progressive disintegration of the surface due to dislodgement of aggregate particles. Potholes are usually formed due to fatigue of road surface that have a width/length of 4 to 12 inches and 0 to 4 inches.

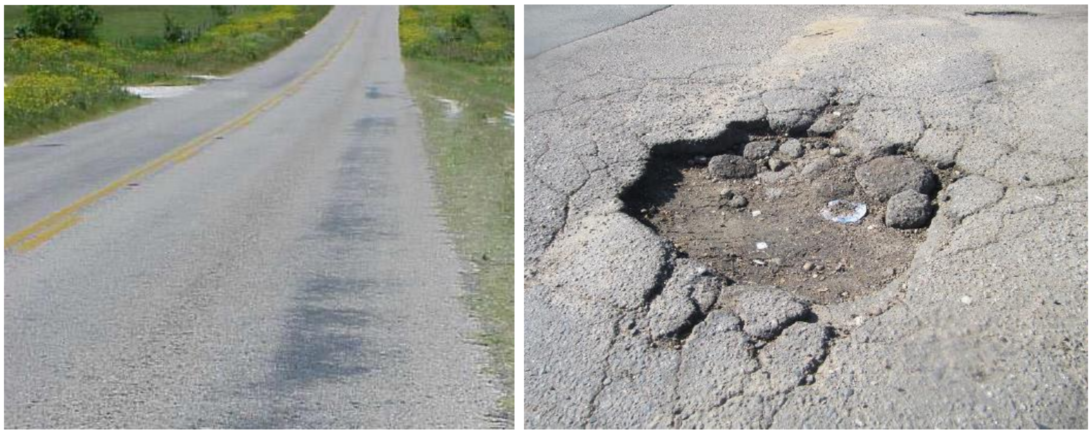


Figure 5. Instances of Ravelling and Pothole. Reprinted from [7].

As an alternative to traditional road condition monitoring systems, several vehicular sensing systems have been proposed and evaluated to determine road conditions and identify certain road defects. The following section provides a comprehensive assessment of exiting literature regarding alternative system developed or proposed to monitor road condition.

1.1 Literature Survey

‘Potholes Marker And More’ [8] and ‘Fill That Hole’ [9] are applications developed where users take photographs of potholes and submit them to a central server. They are not practical on a large scale as users are reluctant to stop and record the pothole locations. PAVEMON [10] is a GIS web-based pavement monitoring system by VOTERS that uses data from multiple sensors such as accelerometer, microphone, tire pressure sensor, imaging, Radar etc. to evaluate different road distress parameters. Despite its capabilities, the need for a specialized vehicular setup restricts the use of this system as a platform for mass data collection platform. Nericell [11] and TrafficSense [12] are systems developed by Microsoft Research India that use Windows smartphone sensors such as accelerometer, microphone and GPS to detect potholes using simple threshold-based heuristics. However, these thresholds are arrived at based on observation and hence remain very subjective. Wolverine [13] detects road bumps based on change in accelerometer readings along the direction of gravity. However, the thresholds for Wolverine are determined using mean and standard deviation only, ignoring higher order statistics. Pothole Patrol (P^2) [1], developed at MIT, is based on a simple machine learning approach to analyze patterns in accelerometer data using X–Z ratio and Speed-Z ratio. The system reads accelerometer data from three different locations and requires an embedded PC that records the data.

Another approach to assessing road conditions is to measure the International Roughness Index (IRI) using a quarter-car vehicle math model [14]. IRI is used to define a characteristic of the longitudinal profile of the traveled wheel path and constitutes a standardized roughness measurement. It is most commonly expressed in units of meters per kilometer (m/km) or millimeters per meter (mm/m). The IRI is based on the average rectified slope (ARS), which is a filtered ratio of a standard vehicle's accumulated suspension motion (in mm, inches, etc.) divided by the distance traveled by the vehicle during the measurement (km, mi, etc.). Roadroid [15] is a commercially available, Android smartphone application that is based on smartphone sensors for monitoring road conditions and classifying them as good, satisfactory, unsatisfactory and poor based on estimated IRI. It offers two solutions for roughness data calculation- the estimated IRI (eIRI) is based on Peak and RMS vibration analysis and the calculated IRI (cIRI) is based on the quarter car simulation. Li et al. [16] calculates a proxy-IRI value that is linearly related to IRI. Although IRI is a common road roughness index measure worldwide, it sometimes fails to recognize isolated faults on smooth roads as it is calculated for a stretch of road using the road's profile. Lepine et al. [17] [18] uses machine learning to identify shocks present in acceleration signals measured on road vehicles. He concludes that machine learning algorithms can be optimized and tuned to achieve high accuracy in detecting road vehicle vibration shocks. However, the road vehicle vibration signals he used were artificially generated using non-stationary random vibration and shock impulses that reproduce typical vehicle dynamic behavior. Perttunen et al. [50] uses acceleration signal to extract road features using Spectral analysis. SVM was used to predict three categories of transient event- speed bump, bump, and large pothole. Allouch et al. [19] uses machine learning techniques such as C4.5 Decision Tree, SVM and Naïve Bayes to label road conditions as 'Smooth' or 'Potholed'. To optimize the feature selection process, a correlation-based Feature selection technique was applied to the data. Bhoraskar et al. [13] introduces a traffic monitoring system that uses braking and acceleration events along with k-means clustering and SVM to label road conditions as 'Smooth' or 'Bumpy'. Nuno Silva et al. [20] approached the problem with data mining using Scikit-learn and Weka to detect unlevelled manholes, short bumps and long bumps. Singh et al. [21] proposed using five separate filter and Dynamic Time Warping (DTW) techniques to detect bumps and potholes. The accelerometer data was collected by an Android app called 'Smart Patrolling' that are placed in the car at different locations. Several other works use similar techniques and focus on either estimating a roughness metric or detecting potholes only [22] [23] [24]. However, effective road lifecycle management requires timely maintenance in stages prior to pothole formation such as cracking, shoving, delamination etc. Crack detection using accelerometers is challenging due to its subtle vibration pattern and vehicle vibration noise. Several video image processing techniques have been suggested [25] [26], but these techniques are memory and computation intensive. With significant advancements in big data analytics and a push for smart cars in recent years, high volume of driving data can be collected from users and processed to obtain useful information. Crowd-sensing approaches to collect road surface vibration data can be very useful in continuously monitoring changes to road condition for relevant authorities. Li et al. [16] and Masino et al. [27] proposed the use of crowd-sensing to obtain data and classify road conditions. Chen et al. [24] uses crowd sensed data acquisition in their system called CRSM to identify potholes. However, instead of using smartphones, they use low cost hardware equipment like accelerometers and gyroscopes that are fixed on vehicles.

1.2 Literature Survey Summary

To summarize the literature review, there are certain areas that can be explored or improved for the purpose of road anomaly detection using smartphone sensors. While there has been research focused on detection of potholes using smartphone collected data, road defects such as deep transverse cracks, rutting, shoving etc. that help in monitoring the lifecycle of the road have not been explored. Works that estimate road roughness indexes do so for a stretch of road and can miss individual faults in the road surface. The majority of current literature focuses on binary classifications using simple machine learning techniques or threshold-based heuristics. However, multiclass classification has not been explored for different stages of road deterioration. Accelerometer data collected from vehicle vibrations give a good marker to distinguish road conditions. However, data used for anomaly detection only focusses on vibration signals collected from acceleration data in the direction of gravity. Information and relationships between data samples that may be present in other directions orthogonal to the direction of gravity are not taken into consideration. Among different machine learning algorithms investigated, neural networks have scare implementation for the purpose of Road Condition monitoring due to the requirement of high quantity of good training data. As neural networks are gaining more prominence in the age of AI and big data mining, further investigation into neural networks is needed. Finally, very few studies utilize crowd-sensing as an approach to collect data and has high potential in the field of road surface condition monitoring.

1.3 Key Focus

This thesis aims to analyze different machine learning techniques for multiclass classification to classify smooth roads, potholes and deep transverse cracks. Transverse cracks were chosen as they pose a higher risk to vehicle safety and have the highest potential to develop into bigger faults or potholes. We utilize features extracted from time domain, frequency domain and wavelet domain from all three Cartesian coordinate axes of sensor data to train our classifiers. The effectiveness of neural networks in road condition monitoring is explored and compared with results from other machine learning techniques. The thesis shall serve as a proof of concept for a large scale crowd-sensing based approach to road condition monitoring using machine learning.

2. METHODS

Our goal is to analyze different multiclass supervised machine learning techniques to effectively classify road surface conditions using data collected from smartphones. We investigate our conjecture that using features from all three axes of the sensor provides more accurate results as compared to using only one axis. We also investigate the performance of deep neural networks to classify road conditions without explicit manual feature extraction.

Our general methodology consists of five stages whose system block diagram is shown in Figure 6. The data acquisition stage deals with obtaining and recording data required for our system. The data is acquired using the accelerometer, gyroscope and GPS sensors present in smartphones. The data collected is then passed through a pre-processing stage where the raw data collected is labelled with appropriate road conditions and then filtered prior to extracting required features. The features extracted are then passed to the training stage of various machine learning algorithms like SVM, Decision Tree and Neural Networks to obtain a trained machine learning model. These models are then evaluated with various performance metrics and finally, the classification stage classifies unlabeled data to determine the appropriate road condition label. We then compare the performance of the various algorithms used and provide a conclusion to our assumptions and hypothesis.

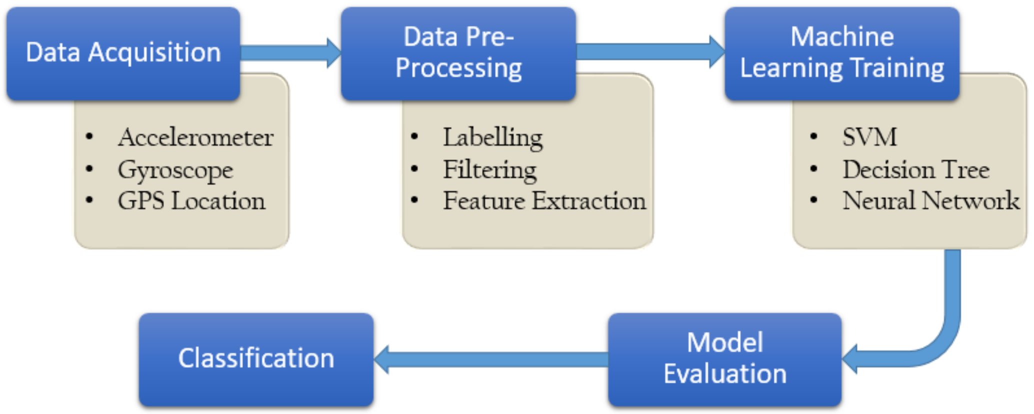
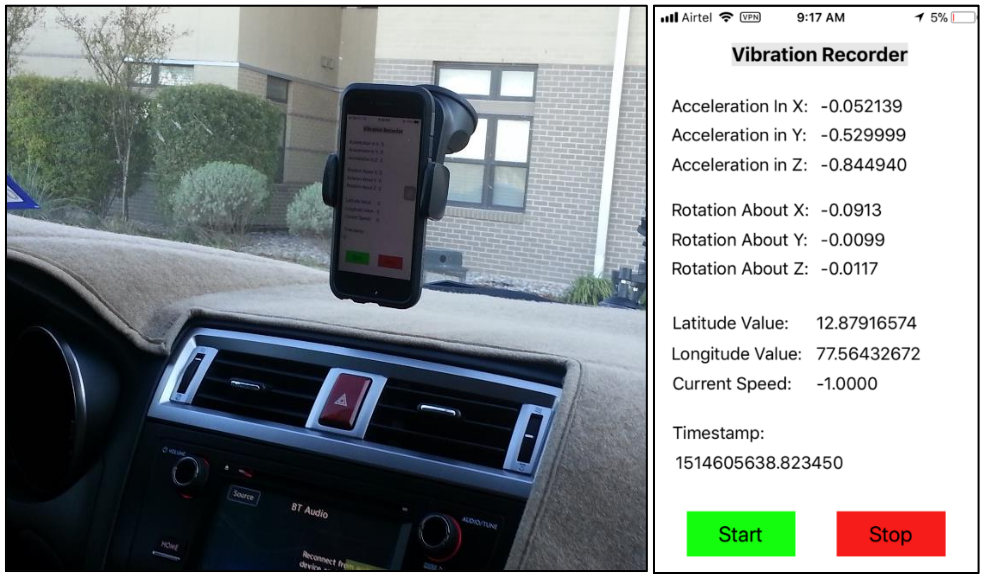


Figure 6. Overall block diagram of the proposed system

2.1 Data Acquisition

To realize the system described above, an Apple iPhone 6 was chosen to collect accelerometer, gyroscope and GPS data. The iPhone contains two separate accelerometer chips- Bosch BMA280 and InvenSense MPU-6700 [28]. The InvenSense MPU-6700 sensor operates as a six-axis combination gyroscope-accelerometer, whose specifications are comparable to the InvenSense MPU-6500. The chip has a specified output data rate of 4,000 samples per second which is common for accelerometers in most smartphones available in the market today. Although they are capable of sampling at 4,000 samples per second, operating systems such as Android and iOS restrict the output data rate to reduce power consumption. Our initial analysis with Apple iOS version 11.2.6 shows that the operating system restricts the maximum sampling frequency of the accelerometer and gyroscope that is available to app developers through Xcode to approximately 100Hz.

Three different cars were used as data collection vehicles to take into consideration the differences in the suspension quality of different types of cars. A Ford Focus sedan, a Ford Focus hatchback and a Subaru Outback SUV were used to represent compact, mid-size and SUV car types. Generally, the type and condition of the car affects the vibration recorded [22] [29]. An iOS app called 'Vibration Recorder' was developed to record Accelerometer and Gyroscope data at 100Hz, GPS longitude and latitude data at 1Hz and their corresponding Epoch/UNIX timestamps. The iPhone running the Vibration Recorder app was mounted to the windshield of the car with a phone mount as shown in Figure 7.



Vibration Recorder

Acceleration In X:	-0.052139
Acceleration in Y:	-0.529999
Acceleration in Z:	-0.844940
Rotation About X:	-0.0913
Rotation About Y:	-0.0099
Rotation About Z:	-0.0117
Latitude Value:	12.87916574
Longitude Value:	77.56432672
Current Speed:	-1.0000

Timestamp:
1514605638.823450

Figure 7. Set up of the iPhone and Screenshot of the Vibration Recorder App

A DJI Osmo was used to record video of the road surface to label particular road conditions in the pre-processing stage. The Osmo was mounted on the front of the vehicle's hood and angled towards the road using a DJI Osmo Vehicle Mount. The video was recorded at 720p and 60FPS to obtain clear videos of the road ahead. In order to sync the video with the data recorded by the Vibration Recorder, another smartphone that also displayed the Epoch/UNIX timestamp was placed in the field of view of the Osmo. The Osmo setup is shown in Figure 8 and a screenshot of its view is shown in Figure 9.

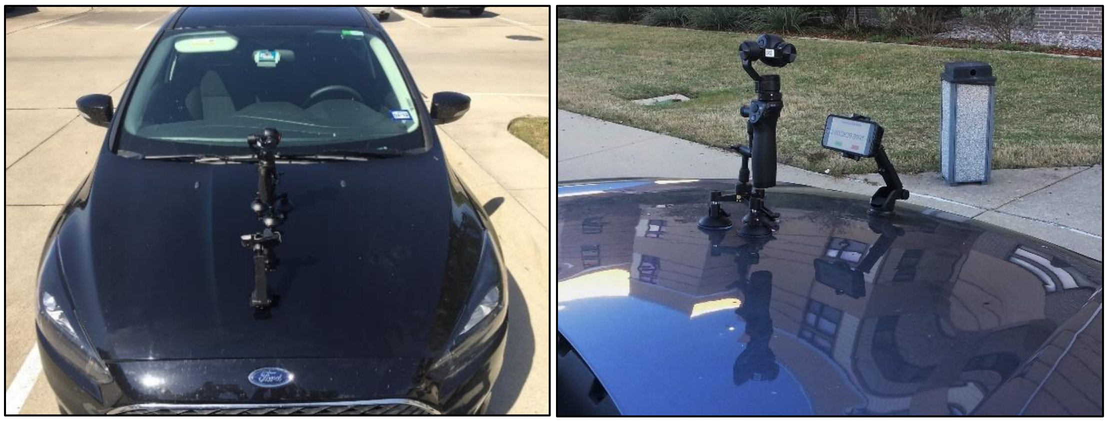


Figure 8. Setup of the Osmo Camera

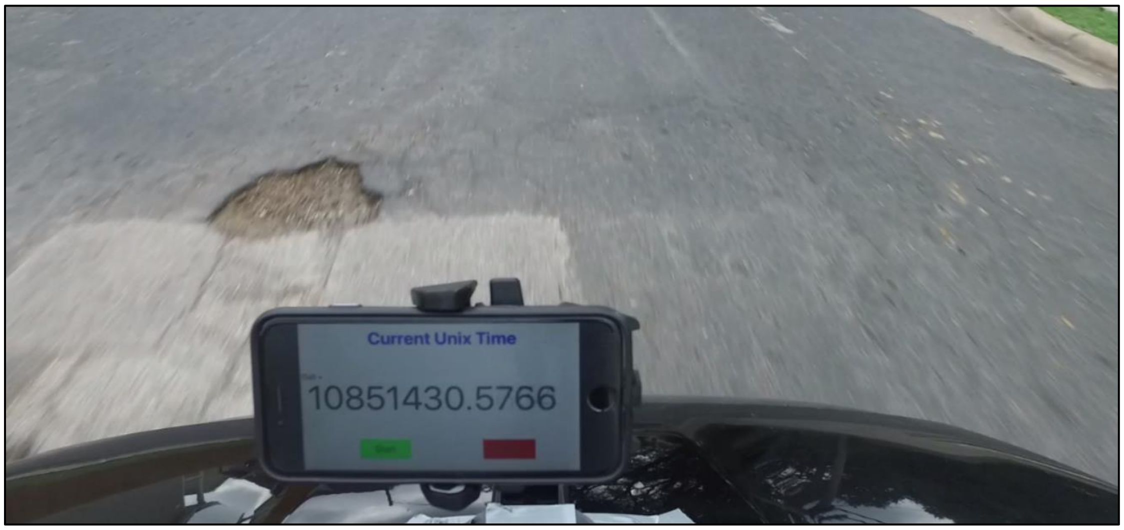


Figure 9. Screenshot of the Osmo Setup’s Field of View

The Osmo's position and angle of elevation was measured and recorded to estimate the distance between the road ahead and the car tires using trigonometric relationships. A total of four data collection runs were conducted in and around College Station, Texas covering road surfaces with asphalt pavements.

2.2 Data Pre-Processing

Data acquired was pre-processed in several stages to make it more coherent and pragmatic. First, the acceleration data collected needed to be virtually reoriented to a global frame of reference to remove variations due to the phone's position and orientation. The acceleration and gyroscope measurements are recorded in a three-dimensional Cartesian coordinate system with respect to the phone's frame of reference as shown in Figure 10. To maintain uniformity and integrity of the data collected from multiple data runs, the phone's frame of reference was transformed to a global frame of reference with respect to the ground as shown in Figure 11.

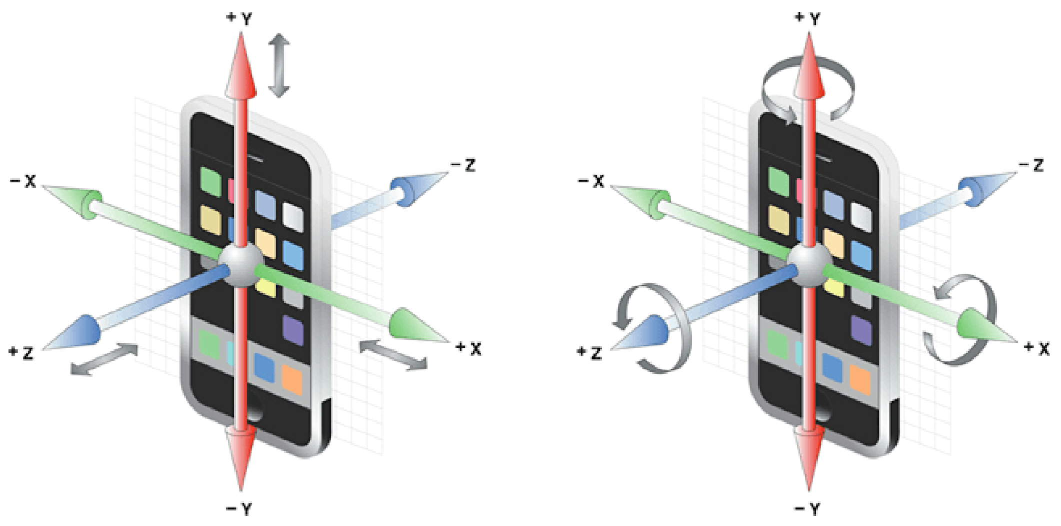


Figure 10. Cartesian Co-ordinate Axes for iPhone Accelerometer and Gyroscope

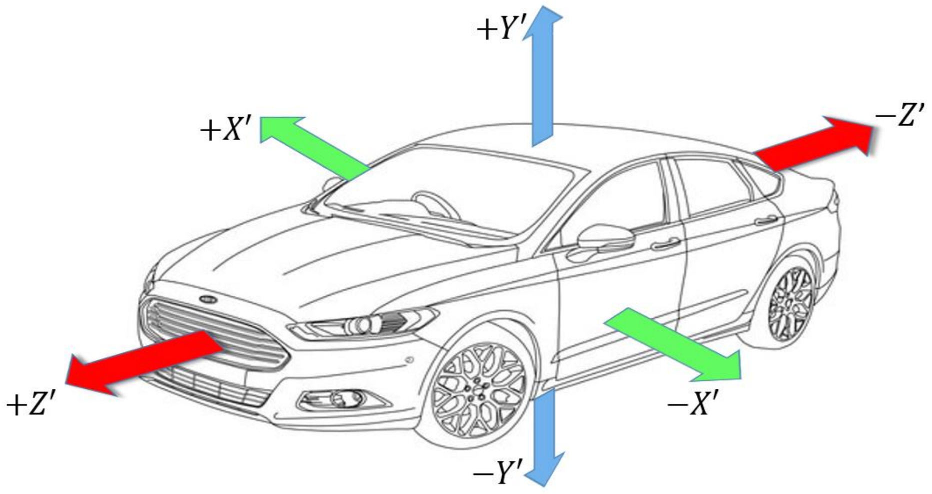


Figure 11. Global Frame of Reference Cartesian Coordinate axes w.r.t. Car

The reorientation algorithm performs accelerometer data reorientation using Euler's angles, which form a representation of the spatial orientation of a certain reference frame as a combination of three orthogonal elemental rotations. Ideally, when a car is at rest on a flat surface, the acceleration values would be:

a_x=0 m/s^2, a_y=9.81 m/s^2 and a_z=0 m/s^2

Equations (1) to (4) are used to calculate two of the three Euler angles and reorient acceleration values to the global frame of reference [30]. $a'_x, a'_y\ a'_z$ are the acceleration values with respect to the global reference frame while α and β are the roll and pitch angles respectively. Figure 12 shows the plot of the acceleration data of a 1.5s window before and after reorientation.

$$\alpha = \tan^{-1}\left(\frac{a_y}{a_z}\right) \qquad \beta = \tan^{-1}\left(\frac{-a_x}{\sqrt{(a_y)^2+(a_z)^2}}\right) \tag{1}$$

$$a'_x = \cos(\beta)\, a_x + \sin(\beta)\sin(\alpha)\, a_y + \cos(\alpha)\sin(\beta) a_z \tag{2}$$

$$a'_y = \cos(\alpha)\, a_y - \sin(\alpha)\, a_z \tag{3}$$

$$a'_z = -\sin(\beta)\, a_x + \cos(\beta)\sin(\alpha)\, a_y + \cos(\alpha)\cos(\beta) a_z \tag{4}$$

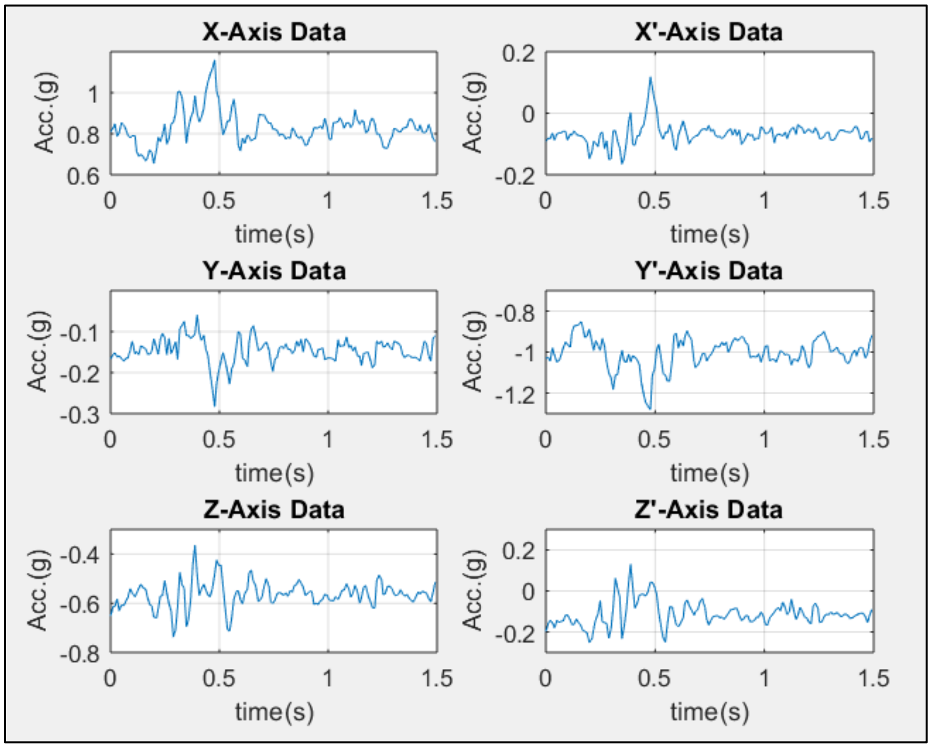


Figure 12. Reorientation of Acceleration Data to Global Frame of Reference

The next stage of pre-processing requires the road surface condition to be labelled in order to obtain the ground truth for our supervised machine learning algorithms. Road pavement surface was classified as Potholes, Deep Transverse Cracks or Smooth Road by following guidelines and descriptions provided in pavement maintenance manuals from the Texas Department of Transportation [7] [31]. Transverse cracks that created a pavement elevation or depression of over 0.5 inches at the position of the crack were considered to be Deep Transverse Cracks.

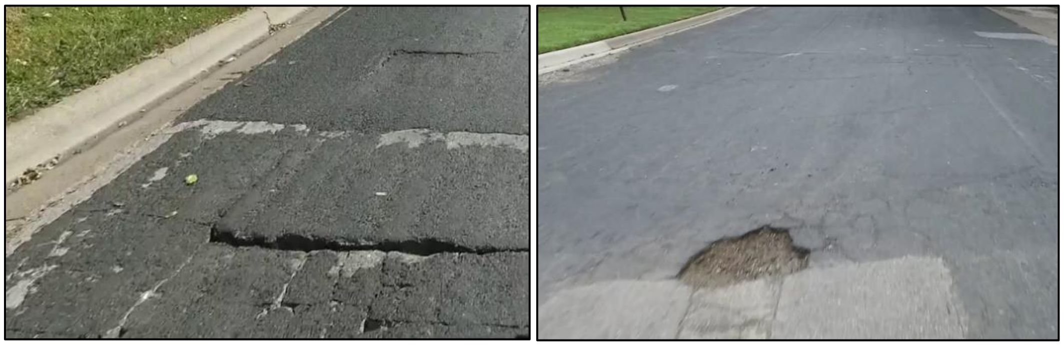
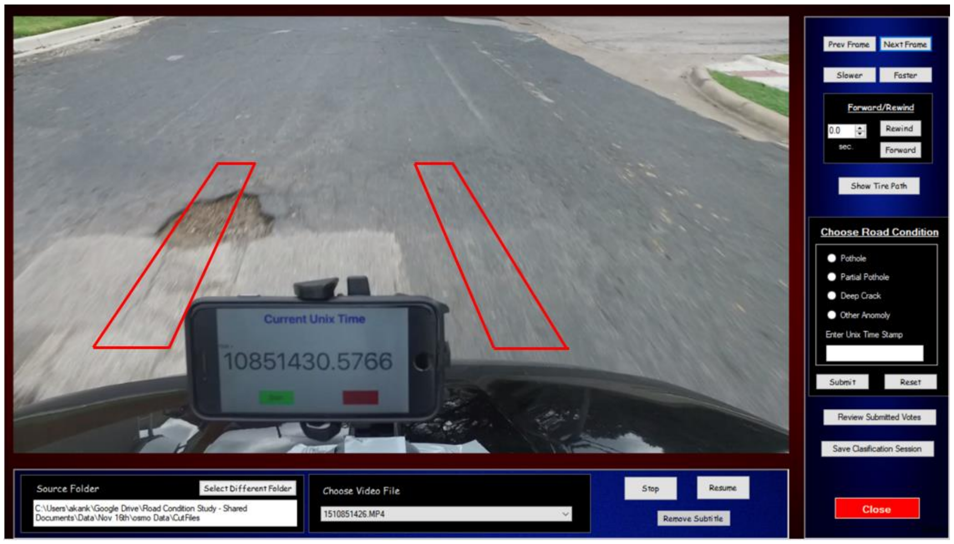


Figure 13. Instances of Road Anomalies: Deep Crack and Pothole

A custom software application was developed to label the video data that was recorded. It enabled the user to perform standard video playback operations such as play, pause, fast-forward, rewind and view frame-by-frame. Since our interest lies only in the section of the road that the car tires travel over, it also provided a feature to overlay the projected tire-trajectory onto the video frames as shown in Figure 14. Instances where the car tires partially travel over a road anomaly was labelled as an anomaly if it covered at least 60% of the tire width. Finally, the user assigns a label to the road segment by selecting a certain frame and specifying the anomaly and the current timestamp displayed.



Source Folder: C:\Users\akank\Google Drive\Road Condition Study - Shared Documents\Data\Nov 16th\osmo Data\CutFiles

Choose Video File: 1510851426.MP4

Forward/Rewind: 0.0 sec.

Figure 14. Road Condition Classifier Software with Tire-Trajectory Overlay

Next, in order to geographically localize the instances of road conditions recorded, the recorded GPS data was synced with the vibration data collected using the timestamps. The speed of the vehicle was calculated based on the rate of change of GPS coordinates. However, due to difference in sampling rate of the Accelerometer/Gyroscope and the GPS sensor, the GPS data and the vehicle speed was interpolated using a spline transformation. This provided a reasonably accurate estimation of the location and speed at a higher sampling rate. Furthermore, to remove certain driving conditions that are not related to the quality of road surface such as acceleration, stopping, braking, lane changing, turning etc., the acceleration data in the X' and Z' axis was filtered with a Butterworth high-pass filter of order 11, cut-off frequency of 3Hz and attenuation of 80dB. The filter removes low frequency components related to these events while preserving any high frequency changes due to road anomalies as shown in Figure 15. To analyze the information contained in higher frequency bands due to the anomalies, a low pass filter or smoothing filter was not applied.

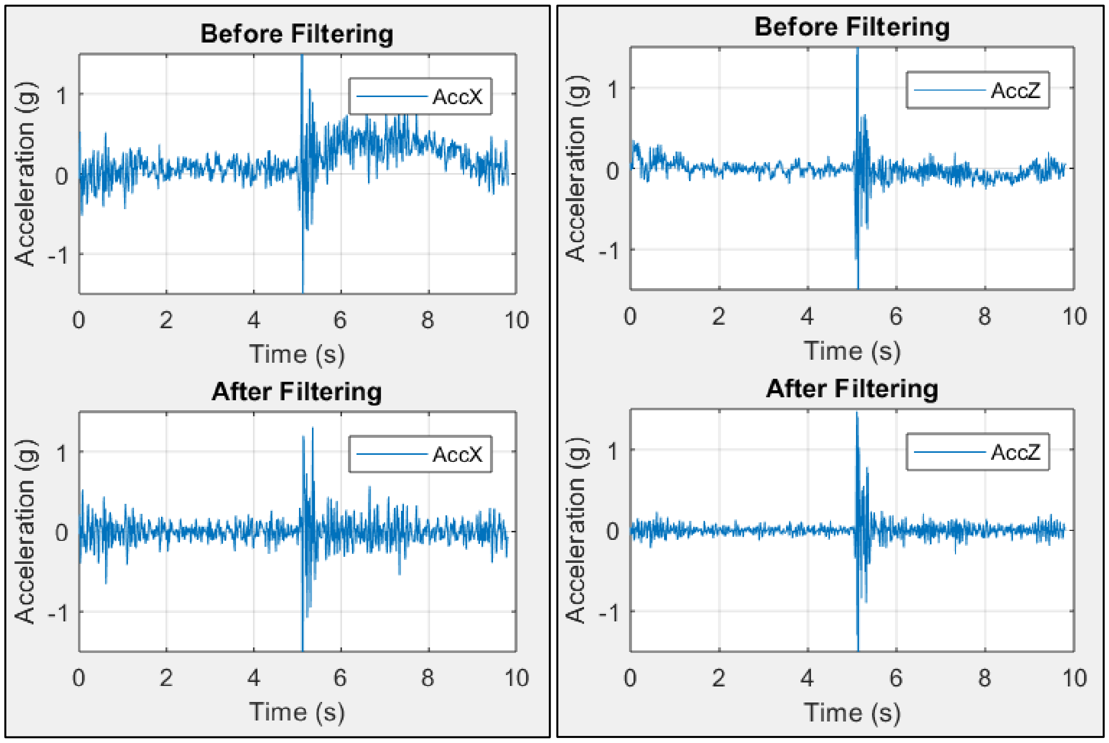


Figure 15. Acceleration signal in X’ and Z’ axis before and after filtering

The continuous filtered data was then converted into segments of data windows of length 100 data samples with a 50% overlap of windows. Labeled anomalies and smooth road segments were extracted and stored separately for further processing described in the feature extraction section. From all data collected, a dataset of 1010 window segments was taken into consideration which contained 149 pothole instances, 45 deep crack instances and 817 smooth road window segments.

2.3 Feature Extraction

There are different types of features used for the purpose of road vibration analysis. We consider three broad categories namely, time domain features, frequency domain features and wavelet domain features.

Previous works in literature only used a few selected features that were considered to provide good distinction between road conditions. However, we wanted to comprehensively explore various possible features to extract any useful information provided by them. Gadelmawla et al. [32] discusses 59 different surface roughness parameters. After reviewing various possible parameters mentioned by Gadelmawla et al. and previous literature, various time domain measures such as Maximum Value, Minimum Value, Mean Value, RMS Value, Peak-to-Peak Value and Ten-Point Average Value were calculated from the time domain signal, its peaks, troughs and signal envelopes. In the frequency domain, the power spectral density of vibration signals provide very useful information that could be used to distinguish different road conditions [33] [34]. The power spectral density was calculated for the windowed signals and the entire bandwidth was divided into smaller bands of 5Hz each. For each of these bands, average band power, RMS band value and maximum band value were considered as frequency domain features. In the wavelet domain, Mortlet wavelets and Daubechies wavelets were deemed suitable to analyze vibration patterns due to road conditions following a review of literature [35][36][37]. Griffiths [36] conducted an extensive study to determine suitable mother wavelets by comparing Haar, Mortlet, Mexican Hat and Daubechies 6 and 10. She concluded that the Mortlet wavelet as well as Daubechies 6 and

10 wavelets could be used to effectively analyze road vehicle vibrations. Upon preliminary study, scales 4 and 5 for each of the three wavelets showed the most distinguishable characteristics for different road conditions. RMS values and ten-point averages of these scales were considered as wavelet domain features.

In previous literature, as mentioned in the introduction section, acceleration in the Y' direction was considered to contain most of the features needed to adequately distinguish road anomalies. Accelerations in X' and Z' directions were considered for driving events only. However, we believe that more information regarding road anomalies are present in the X' and Z' directions. For example, when a car hits a pothole with its left front wheel, there is a sudden deceleration in the Z' direction as well as a sudden tilt in the X' direction. Such information may contribute to distinguishing cracks and potholes, considering cracks tend to span the entire width of the road whereas potholes are more localized. In total, 54 features were extracted from the accelerometer data for each of the three axes. Hence, each feature vector consisted of 162 feature values that were saved as a .MAT file.

2.4 Machine Learning Approaches

Machine Learning is an application of Artificial Intelligence (AI) that provides computer systems the ability to learn and improve from experience without explicit programming. Once a computer algorithm is trained, the algorithm can apply the relationship learnt during training to solve similar problems. For example, the retail industry such as Amazon utilize machine learning algorithms to provide highly personalized services. Data collected from prior purchases or searches is used as training data to classify online recommendations to specific users. This type of machine learning that divides analyzed data into discrete clusters or classes is referred to as the "classification problem". Another kind of machine learning problem, known as "regression problem", finds continuous relationships between data variables instead of clustering data into different classes.

Figure 16 shows the general workflow for both classification and regression type of machine learning approaches. It begins with a dataset of raw data whose class labels are previously known. For the case of road vehicle vibration, this is the acceleration signals recorded by the smartphone which are labelled with different road conditions. This input dataset is processed to obtain various attributes of the data called features that are compatible with machine learning algorithms. Once the features and class labels are extracted, the features list and corresponding class labels are partitioned into three sets, the training dataset, validation dataset and testing dataset. All three sets have the same distribution of classes in terms of proportion. The training set is used to train the algorithm and develop the classifier model. The validation dataset is then used to validate the performance of the trained classifier. If there is not enough data to create a validation set, there are several other approaches for validation of models such as cross validation where the entire data is used for both training and validation. The validation phase is useful to compare and correlate the performance of different models and choose the best one that fits the problem. To test the model on new data, the testing dataset is used as input to the final model to predict output data labels.

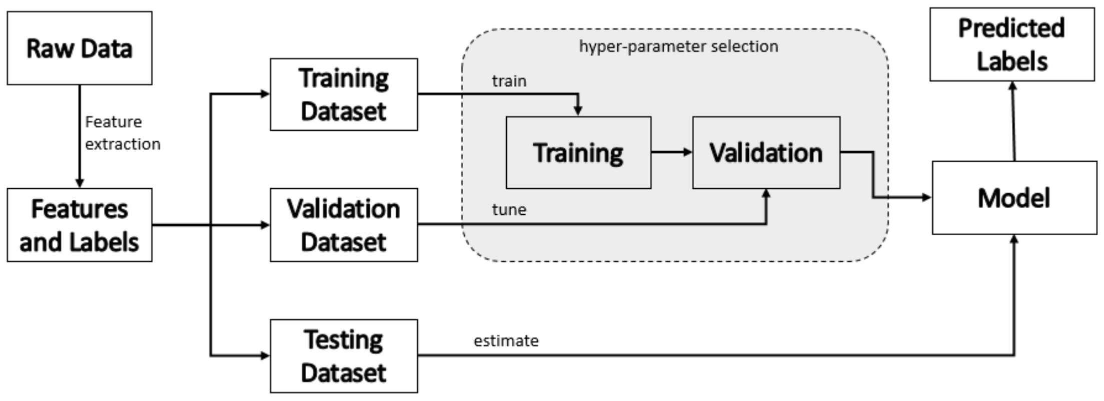


Figure 16. General Workflow Diagram of Machine Learning Algorithms

Various machine learning classification algorithms have been developed, which makes the selection of a classifier a difficult task. Since there are no standardized nomenclature in machine learning, similar classification algorithms may be expressed with different names. MATLAB® incorporates the Statistics and Machine Learning Toolbox, which included implementations of various machine learning classifiers [38]. These classifiers can be primarily divided into seven categories- Naive Bayes Classification, Discriminant Analysis, Ensembles, Decision Trees, Nearest Neighbors, Support Vector Machines (SVM) and Neural Networks [39] [40]. For our study, SVM, Decision Trees and Neural Networks were chosen as they are popular and reliable techniques used for classification of road vibration data. The complete dataset is randomized and divided into training and testing dataset with an 80:20 ratio, keeping the proportion of the classes in both datasets constant. To investigate whether the models trained with input features extracted from all three axes perform better than using features from Y' axis only, two datasets containing 162 features and 54 features were created for each case respectively. The parameters used to analyze and quantify results are discussed in Section 2.5.

2.4.1 Support Vector Machines

Support Vector Machine is a supervised machine learning model that evaluates input data and recognizes patterns for classification and regression analysis. SVM performs classification by finding the hyperplane that maximizes the margin between data point clusters corresponding to different classes. SVMs are versatile, memory-efficient and effective in high-dimensional spaces. Generally, SVM is used to classify data that have two distinct labels. The SVM hyperplane is defined by (5), where $(\boldsymbol{x}_i, y_i)$ for $i = 1,2, \dots n$ are the feature matrix and class vector for the n training data points and $(\boldsymbol{x} \cdot \boldsymbol{x}')$ is the matrix feature inner product. The parameters $\widehat{\alpha_i}$ are found by maximizing the function given in (6) with the constraint given in (7), where C is a regulation parameter.

$$D(\boldsymbol{x}) = \sum_{i=1}^{n} \hat{\alpha}_i y_i (\boldsymbol{x} \cdot \boldsymbol{x}') + \hat{b} \tag{5}$$

$$L(\alpha) = \sum_{i=1}^{n} \alpha - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j \left(\boldsymbol{x_i} \cdot \boldsymbol{x_j}\right) \tag{6}$$

$$\sum_{i=1}^{n} \alpha_i y_i = 0 \tag{7}$$

$$where\ 0 < \alpha_i < \frac{C}{n}$$

The parameter $\hat{b}$ is used to scale the support vector $(\boldsymbol{x_s} \cdot y_s)$ such as only the first class exits in $D(\boldsymbol{x}) \geq 1$ and only the second class exits in $D(\boldsymbol{x}) \leq -1$:

$$\hat{b} = y_s - \sum_{i=1}^{n} \hat{\alpha}_i y_i((\boldsymbol{x_i} \cdot \boldsymbol{x_s})) = 0$$

In order to execute multi-class SVM, MATLAB® incorporated the 'ClassificationECOC' class in their Statistics and Machine Learning Toolbox. ClassificationECOC is an Error Correcting Output Code (ECOC) classifier used to perform multiclass learning by reducing the classifier to simple binary classifiers such as SVMs. An ECOC model reduces a classification problem involving atleast three classes into a set of binary classifiers. If M is the coding design matrix with elements m_{kl} and s_l is the predicted classification score for the positive class of learner l, a new observation is assigned to the class $\hat{k}$ that minimizes the aggregation of losses for the L binary learners given by (8) [41].

$$\hat{k} = \underset{k}{\operatorname{argmin}} \frac{\sum_{l=1}^{L} |m_{kl}| g(m_{kl}, s_l)}{\sum_{l=1}^{L} |m_{kl}|} \tag{8}$$

For our study, SVM was implemented in two ways- the Simple SVM and Cross Validated SVM. The Simple SVM implementation uses the default SVM binary learners and one-versus-one coding design to train the SVM model. However, this type of model tends to have the problem of over-fitting. In order to try and overcome this problem, a subset of data called validation set is used to test the model during the training phase. Cross validation techniques such as 5-fold cross-validation, 7-fold cross validation, 10-fold cross-validation and Leave One Out cross-validation are implemented for our analysis.

2.4.2 Decision Tree

Decision trees, also known as classification trees and regression trees, predict output responses based on input data. Following the decisions in the tree from the root node to the leaf node gives the output response to that particular input data [42]. The decision tree is an algorithm that classifies data through a cascade of statistical tests as shown in Figure 17. These tests compare the value that is input to a node with a threshold value that splits the tree's path. Tests can have multiple results and different tree paths can follow to the same output class label. The complexity of the tree is defined by the number of branch splits and depending on its complexity, they have quick training and prediction speeds, moderate predictive accuracy and low computational memory requirements.

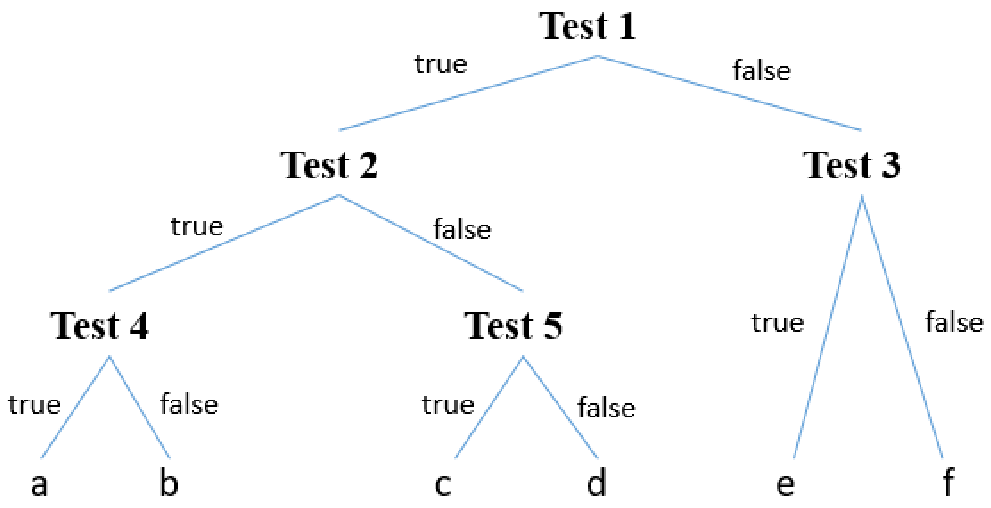


Figure 17. Decision Tree Structure

The MATLAB® Statistics and Machine Learning Toolbox was used to train a binary classification decision tree for multiclass classification. Allouch et al. used a C4.5 Decision Tree model for pothole detection and concluded that it is an accurate classifier [19]. Similar to our approach to SVM, we develop a simple classification decision tree and a cross validated tree to reduce over-fitting.

2.4.3 Neural Networks

Neural networks are a popular machine learning framework that attempt to imitate the learning pattern of natural biological neural networks in the brain. A typical neural network consists of inter-connected arithmetic processors called neurons which produce a sequence of real valued activation outputs. Neurons present in the input layer of the neural network gets activated through sensor data perceiving the environment, while neurons present in other layers get activated through weighted connections from previously active neurons. Neural network algorithms link the feature vectors (input layer) to the class labels (output layer) using multilayered networks called hidden layers as shown in Figure 18. The complexity of the classification problem determines the number of hidden layers needed. Although neural networks are powerful, high accuracy algorithms, training them requires a large dataset. The size of the required dataset also increases as the number of hidden layers increases.

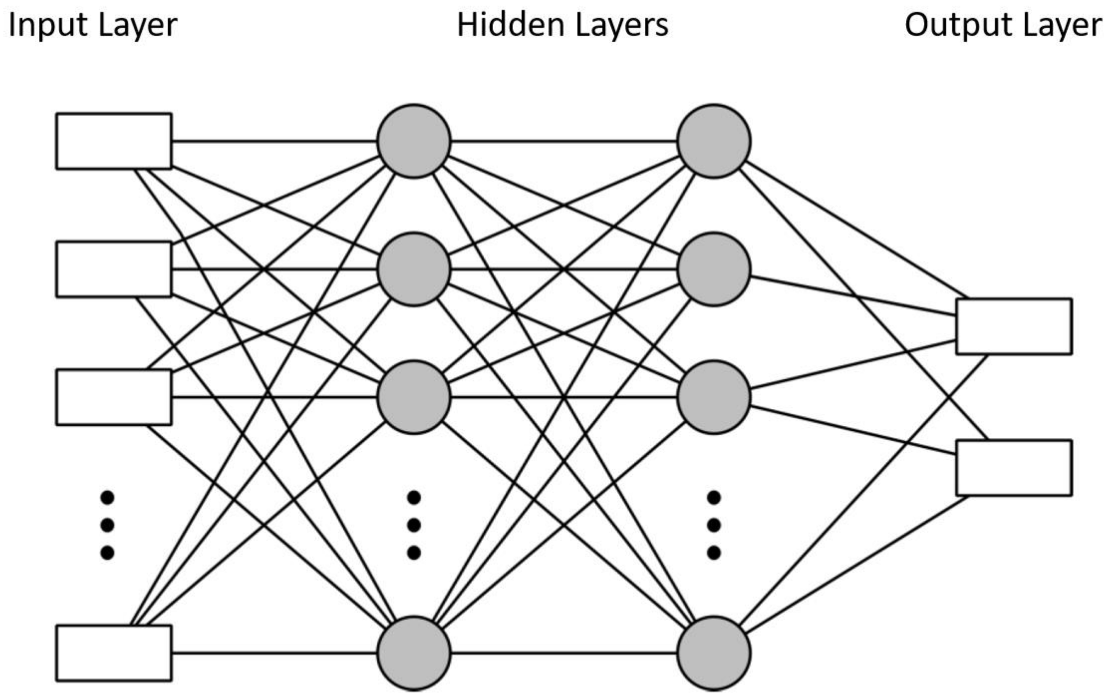


Figure 18. Structure of a Neural Network with Two Hidden Layers

A multilayer perceptron (MLP) is a class of feedforward neural networks that comprises of at least one hidden layer and uses backpropagation for training its models [43] [44]. Each neuron in the hidden layers use a nonlinear activation function which distinguishes it from a linear perceptron. Each neuron inputs values from neurons in the previous layer and outputs the result of a weighted linear summation followed by a non-linear activation function. The output layer receives the values from the final hidden layer and outputs the class that is predicted for that input data. To realize MLP networks, the scikit-learn library for supervised Neural Network was used [45]. The training data and the testing data goes through additional pre-processing where the features are standardized by removing the mean and scaling to unit variance. This standardization step is a common requirement for various machine learning algorithms including MLPs as they may perform poorly if the individual features do not resemble a standard normal distribution. The MLPClassifier class available in scikit-learn creates a model that optimizes the log-loss function using LBFGS or stochastic gradient descent. It includes various parameters such as activation function, hidden layer size, weight optimization solver, regularization factor, weight update learning rate, etc. to tune the model to the specific problem [46]. After evaluating the performance of the classifier for all permutations of parameters, the MLP classifier that provided reliable results with high accuracy consisted of 7 to 10 hidden layers, an LBFGS weight optimization solver, a constant learning rate for weight update and an activation function of ‘Tanh’ or ‘ReLU’.

LBFGS is a limited memory optimizer in the family of quasi-Newton methods that approximates the Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm [47] [48]. For MLPs, the LBFGS solver can converge faster and performs well when dealing with small datasets. Adam, a stochastic gradient-based optimizer proposed by Diederik Kingma and Jimmy Ba was also used in comparison [49], however, Adam works best in terms of training time and validation scores for larger datasets with thousands of training samples. A comparison of activation functions ReLU, Tanh and logistic sigmoid discussed in the Results Section showed that ReLU and Tanh perform better.

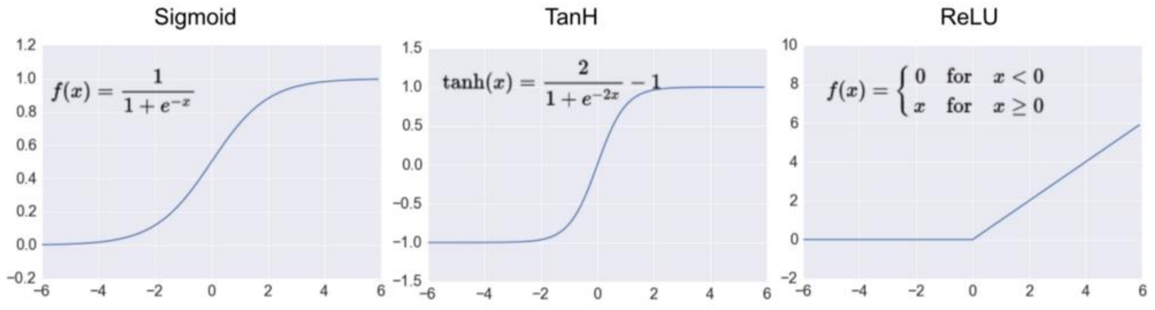


Figure 19. Activation Function Plots for Sigmoid, Tanh and ReLU functions

Since deep neural networks can be used with raw data and performs feature extraction implicitly, similar MLP classifiers were designed by providing the raw acceleration data as the input instead of the extracted features. As a window size of 100 data points was considered, each input vector had a length of 100 for the single Y' axis and 300 when all three axes were considered. Providing direct data to a neural network eliminates the process of manual feature extraction and hence saves time and memory in the training stage. However, such networks require a very large dataset in order to extract useful features and may not give high accuracy for the limited dataset we possess. Therefore, we not only explore the use of neural network classifiers in classifying feature vectors but also classifiers that can classify raw data directly. The results are provided in the Results section using the performance evaluation parameters discussed in the next section.

2.5 Model Evaluation Parameters

To evaluate the performance of the classifiers described in the previous section, various performance evaluation metrics are used for machine learning models. For each of the classifiers, we consider relevant and important parameters which best enable us to derive a conclusion on its performance.

A confusion matrix is a specific tabular representation of the performance of a supervised machine learning algorithm. Each column represents the number of instances of the predicted class while each row represents the number of instances of an actual class. Most classification metrics are derived from the confusion matrix based on the number of true positives (TP), true negatives (TN), false positives (FP) and false negatives (FN). A classifier's accuracy, precision and recall are described in (9), (10) and (11).

$$Accuracy = \frac{(TP+TN)}{(TP+TN+FP+FN)} \tag{9}$$

$$Precision = \frac{TP}{(TP+FP)} \tag{10}$$

$$Recall = \frac{TP}{(TP+FN)} \tag{11}$$

To evaluate the performance of the Simple SVM and Simple Decision Tree classifier, the average training loss and average test accuracy for the trained classifier are recorded. The average training loss is the average in-sample loss of the trained classifier model using the training dataset while the average test accuracy is the average classification accuracy using the testing dataset for *n* iterations. The average precision and average recall for the three distinct classes predicted by the model are also recorded to analyze what proportion of positive identifications were correct and what proportion of actual positives were correctly identified. For cross validated SVM and cross validated Decision Tree, average training loss and cross validation error rates for k-fold and leave-p-out cross validations are recorded. Graphs of these parameters give an intuitive understanding of their reproducibility.

Similarly, for the MLP classifier, the average training accuracy and average test accuracy is recorded for each of the selected combination of parameters. This provides an overview of the classifier’s performance while the Precision and Recall rates for each of the three classes provide more specific insights into the classifier’s performance.

3. RESULTS

In this section, we analyze and discuss the obtained results and evaluate each of the machine learning model's capability for detecting road anomalies. The parameters used to measure and quantify performance are described in the previous section. The algorithms were run on an HP ENVY x360 convertible notebook running on Microsoft Windows 10 Home OS with an Intel® core™ i5-6200 processor, 2.30GHz CPU and 8GB RAM. SVM and Decision tree algorithms were implemented using the Statistics and Machine Learning Toolbox on MATLAB® 2017, while Neural Network MLP implementation was carried out using Scikit-learn on Python 3.6 environment.

To analyze the time requirement significance of extracting features using all three axes as compared to using only one axis, a comparison of time required to extract the features was performed and results tabulated in Table 1. These times correspond to the average time taken over 200 trial runs. It can be noted that even though extracting one axis features is faster, the feature extraction process for both cases are fast enough to realize the entire process in real time. An analysis for time taken to classify the data is discussed later in this section. Since sliding windows of 1 second with 50% overlap are used, the worst-case time requirement to realize the system in real-time is 500ms.

TABLE 1
FEATURE EXTRACTION AVERAGE TIME REQUIREMENTS

Parameter	**Using Features from all Axes (ms)**	**Using Features from Y' Axis (ms)**
High Pass Filtering	0.0157	0.0157
Time Domain Feature Extraction	15.257	5.135
Frequency Domain Feature Extraction	1.674	1.084
Wavelet Domain Feature Extraction	52.877	19.01
Total	69.823	25.245

3.1 Support Vector Machines

The Simple SVM was implemented with one hundred iterations, each using distinct combinations of instances for the training and testing datasets while maintaining the same proportion of classes. Average values of evaluation parameters for these iterations were considered to evaluate the generalized performance of the algorithm. As discussed earlier, the SVM was trained separately using features from all three axes as well as features from only Y' axis to conduct a comparative analysis of performance. The simple SVM models trained were one-vs-one classifiers with equal misclassification cost and a linear kernel function. The training loss, testing accuracy, precision and recall rates are tabulated in Table 2. The precision and recall rates are displayed for each of the three classes to analyze bias.

TABLE 2

SIMPLE SVM IMPLEMENTATION RESULTS

Parameter	Using Features from all Axes			Using Features from Y' Axis Only		
Avg. Training Loss	**0.0279**			0.0773		
Avg. Test Accuracy	0.8855			**0.9015**		
	Crack	Pothole	Smooth	Crack	Pothole	Smooth
Avg. Precision	0.4025	0.7221	0.9442	0.3862	0.7479	0.9417
Avg. Recall	0.4375	0.6776	0.9471	0.2100	0.6568	0.9823

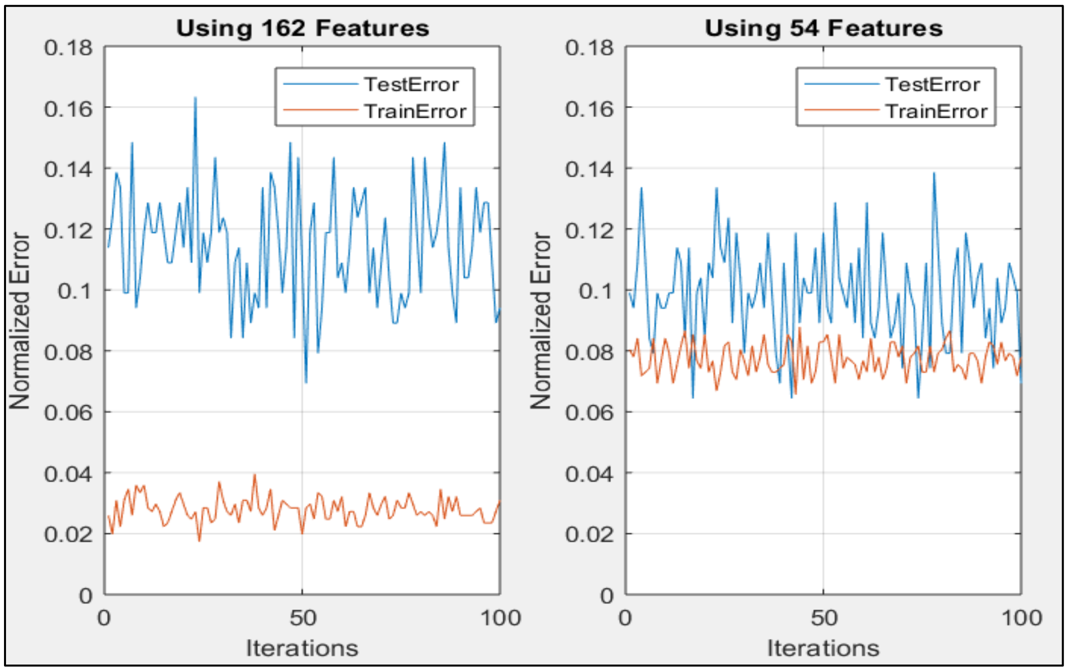


Figure 20. Simple SVM: Training and Testing Error Rates using 162 features from 3 axes and 54 features from 1 axis

The cross validated SVM model also implements a one-vs-one classifier with a linear kernel function and measures performance using different cross validation methods. The results of the cross validation ECOC classifier for SVM is tabulated in Table 3.

TABLE 3
CROSS VALIDATED SVM IMPLEMENTATION RESULTS

Parameter	Using Features from all Axes	Using Features from Y' Axis Only
Avg. Training Loss	**0.0149**	0.0663
Avg. 5-fold Loss	**0.0822**	0.0990
Avg. 7-fold Loss	**0.0851**	0.0990
Avg. 10-fold Loss	**0.0842**	0.0941
Avg. Leave One Out Loss	**0.0812**	0.0931

From Table 2, it is observed that the classifier trained with features from all three axes has lower loss and performs much better than the classifier trained with features from Y' axis only. The average training loss is lower and the average testing accuracy is higher for the former case. The precision and recall rates for the individual classes are also higher when all three axes are used. The recall rate for cracks show the most significant improvement, going up by over 20%, while the recall for smooth road reduces by about 3.5%. The precision and recall rates for potholes remains very comparable. Table 3 shows that the cross validated classifier with features from all three axes has a lower training loss and lower cross validated errors as well.

3.2 Decision Tree

The decision trees are implemented in a similar manner to SVM, with five hundred iterations of the simple decision tree being implemented with unique sets of training and testing data for each iteration. Decision trees are faster to train, however, they create a highly varying set of hyperparameters with each iteration such as number of nodes and node thresholds. There exists a tradeoff between speed and reproducibility. The training loss, testing accuracy, precision and recall rates of the simple decision tree implementation are tabulated in Table 4. The results of the cross validated ECOC classifier for Decision Tree is tabulated in Table 5.

TABLE 4
SIMPLE DECISION TREE IMPLEMENTATION RESULTS

Parameter	**Using Features from all Axes**			**Using Features from Y' Axis Only**		
Avg. Training Loss	**0.0199**			0.0248		
Avg. Test Accuracy	**0.8835**			0.8734		
	Crack	Pothole	Smooth	Crack	Pothole	Smooth
Avg. Precision	0.4348	0.6663	0.9497	0.2925	0.6581	0.9442
Avg. Recall	0.4121	0.6716	0.9470	0.3080	0.6462	0.9471

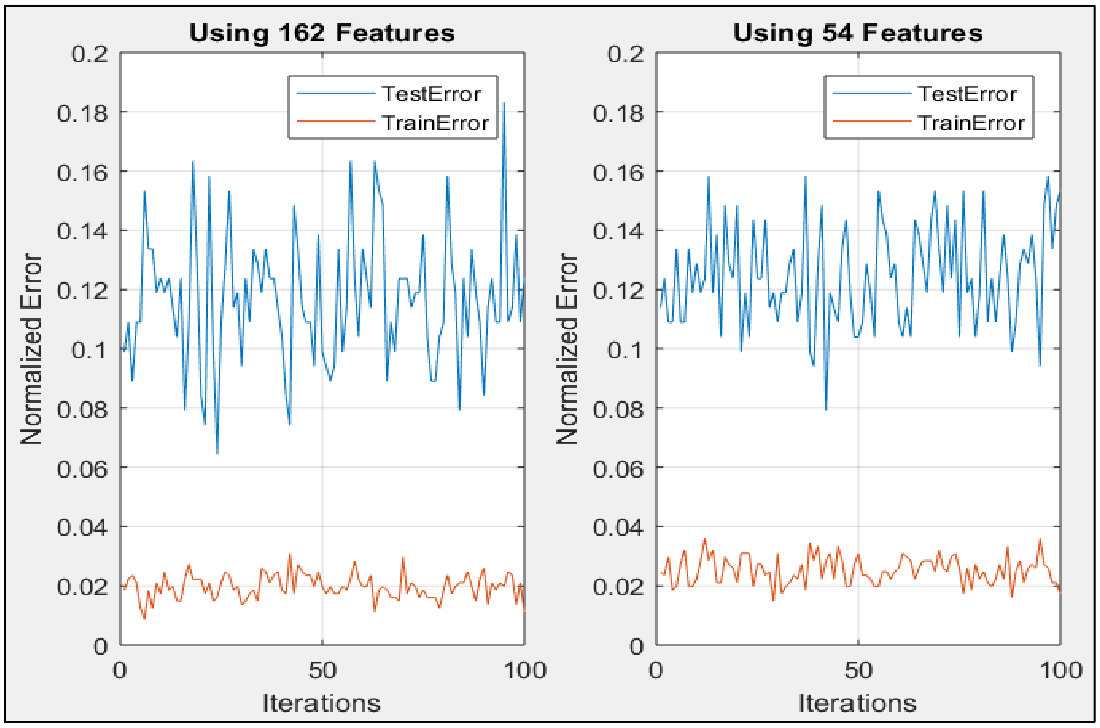


Figure 21. Simple Decision Tree: Training and Testing Error Rates using 162 features from 3 axes and 54 features using 1 axis

TABLE 5
CROSS VALIDATED DECISION TREE IMPLEMENTATION RESULTS

Parameter	**Using Features from all Axes**	**Using Features from Y' Axis Only**
Avg. Training Loss	**0.0188**	0.0267
Avg. 5-fold Loss	**0.1178**	0.1257
Avg. 7-fold Loss	0.1208	**0.1109**
Avg. 10-fold Loss	**0.1010**	0.1218
Avg. Leave One Out Loss	**0.0970**	0.1317

From Table 4, it is observed again that the classifiers trained with features from all three axes performs better than the classifiers trained with only Y' axis. Precision and

Recall for cracks increases by over 10% each and training loss and testing accuracy shows slight improvements. However, from Figure 21, it is observed that the test accuracy is not very consistent across different iterations. This is expected as each iteration is trained well to a particular set of training data and may not perform as well with the testing data. Table 5 shows that the cross validated classifier with all axes also performs better and shows lower training loss and cross validation errors. However, when compared to SVM performance, the cross validation errors are higher.

3.3 Neural Networks

The preliminary analysis stage of implementing an MLP neural network classifier involved comparison of test accuracy, precision and recall for the various combinations of parameters that were chosen. Twenty iterations of each set of parameters was implemented and on inspection of the output performance metrics, the following conclusions were made: classifiers that implemented the Adam weight optimization solver gave a slightly better overall test accuracy than the LBFGS when used with activation function ReLU and comparable accuracy when used with activation function Tanh. However, the individual precision and recall rates for Crack and Pothole was much lower for Adam as compared to LBFGS. Classifiers that implemented LBFGS converged faster than Adam when the number of hidden layers was small but increases as the neural network grows deeper with more hidden layers. Comparison of precision and recall rates showed that the Tanh activation function gave poor precision and recall for cracks which was compensated in overall accuracy by high precision and recall for smooth road. After the analysis, it was concluded that a classifier that implements LBFGS solver and hidden layer size 8 and 9 gave the most optimal results. However, there was a trade-off existed between ReLU, which yielded better precision for cracks and Tanh, which yielded better precision for smooth road but gave very poor precision rates for cracks.

The final analysis stage compared the performance of the MLP neural networks for input feature vector lengths 162 and 54 while implementing ReLU and Tanh with LBFGS. In order to account for variability with the number of hidden layers, results were compared for these parameters using hidden layer count from 7 to 10. The test accuracy, precision and recall for these models are tabulated in Table 6 and Table 7.

TABLE 6
MLP IMPLEMENTATION USING RELU- RESULTS

MLP Hidden Layer Count	**Using Features from all Axes**			**Using Features from Y' Axis Only**		
TEST ACCURACY						
7	0.9212			0.8921		
8	0.9190			0.8919		
9	0.9132			0.8917		
10	0.9031			0.8832		
PRECISION RATES						
	Crack	Pothole	Smooth	Crack	Pothole	Smooth
7	0.559	0.769	0.969	0.350	0.674	0.962
8	0.550	0.769	0.964	0.345	0.688	0.959
9	0.481	0.768	0.966	0.377	0.685	0.958
10	0.418	0.730	0.967	0.323	0.647	0.957
RECALL RATES						
	Crack	Pothole	Smooth	Crack	Pothole	Smooth
7	0.611	0.799	0.962	0.342	0.723	0.953
8	0.585	0.781	0.963	0.365	0.708	0.952
9	0.481	0.768	0.966	0.377	0.685	0.958
10	0.418	0.730	0.967	0.323	0.647	0.957

TABLE 7
MLP IMPLEMENTATION USING TANH- RESULTS

MLP Hidden Layer Count	**Using Features from all Axes**			**Using Features from Y' Axis Only**		
TEST ACCURACY						
7	0.9122			0.8978		
8	0.9149			0.8950		
9	0.9132			0.8952		
10	0.9132			0.8978		
PRECISION RATES						
	Crack	Pothole	Smooth	Crack	Pothole	Smooth
7	0.486	0.757	0.964	0.395	0.705	0.961
8	0.490	0.754	0.967	0.364	0.708	0.958
9	0.532	0.731	0.967	0.404	0.695	0.959
10	0.482	0.763	0.965	0.395	0.712	0.959
RECALL RATES						
	Crack	Pothole	Smooth	Crack	Pothole	Smooth
7	0.498	0.774	0.959	0.409	0.738	0.956
8	0.529	0.782	0.958	0.416	0.718	0.955
9	0.490	0.794	0.959	0.417	0.726	0.955
10	0.510	0.783	0.958	0.408	0.727	0.957

Based on Table 6 and Table 7 it can be observed that the average test accuracy, precision and recall rates are higher for the MLP models using features from all three axes compared to only a single axis. Considering, models trained using features from only one axis, using Tanh as the activation function yields higher precision and recall rates among the three classes. However, when considering features from all three axes, ReLU stands out in its high precision and recall rates for cracks. The precision and recall rates for pothole and smooth remains quite similar between the two activation functions.

In order to test performance of the MLP Neural Network classifiers in classifying road vibration data without manually performing feature extraction prior to training, the acceleration data is directly used as the input to the neural network and is evaluated over 20 iterations. The single axis input vector has length 100 and input vector with all axes has length 300 with data each axes concatenated end to end. An initial analysis regarding choice of activation function showed that Tanh activation function failed to produce significant precision and recall rates for cracks. Therefore, only ReLU was considered for the purpose of analyzing MLP classifiers using direct data. The average test accuracy, precision and recall for direct data input using ReLU activation function is tabulated in Table 8.

TABLE 8
MLP USING DIRECT DATA FOR RELU- RESULTS

MLP Hidden Layer Count	**Using Features from all Axes**			**Using Features from Y' Axis Only**		
	TEST ACCURACY					
7	0.8027			0.8157		
8	0.7946			0.8112		
9	0.8031			0.8140		
10	0.7903			0.7998		
	PRECISION RATES					
	Crack	Pothole	Smooth	Crack	Pothole	Smooth
7	0.283	0.329	0.918	0.271	0.423	0.918
8	0.408	0.301	0.906	0.258	0.469	0.905
9	0.412	0.346	0.908	0.267	0.420	0.910
10	0.396	0.351	0.893	0.263	0.451	0.894
	RECALL RATES					
	Crack	Pothole	Smooth	Crack	Pothole	Smooth
7	0.139	0.672	0.9118	0.156	0.607	0.911
8	0.142	0.673	0.9193	0.141	0.621	0.921
9	0.154	0.711	0.9203	0.149	0.563	0.915
10	0.135	0.656	0.9266	0.124	0.549	0.924

It is observed that the average test accuracy of MLP models using direct data is lower compared to MLP models trained using extracted features as input. The average precision and recall rates for the case of cracks and potholes are also lower. However, it was already anticipated that training neural networks without features would require a large dataset and we are limited by the size and composition of our data.

The main advantage of using Neural Networks without feature extraction is the time saved in feature extraction when realizing real time systems. Earlier, we saw that on average, the feature extraction requires approximately 70ms and 25ms for the case of 3 axes and 1 axis respectively. Table 9 shows the average time required to classify a single data window using data from all three axes for different trained machine learning algorithms discussed in this thesis. Since each of the classifiers take classification times in the order of microseconds, using MLP with direct data as the input would save computation time for feature extraction. When realizing such a system in real time, this saves significant computation time.

TABLE 9
CLASSIFIER PERFORMANCE: TESTING TIME

Classifier	**Avg. Time to Classify One Window (μs)**
SVM	29.372
Decision Tree	4.8032
MLP	36.0142
MLP (Direct Data)	72.078

4. DISCUSSION AND FUTURE WORK

Based on the results of the study, we observe that the machine learning approaches implemented are quite effective in classifying road anomalies such as cracks and potholes. Classifiers trained using features from all axes proved to be more accurate when compared to features from only one axis. Since our approach of extracting large number of features from all three axes to train multiclass machine learning classifiers was a novel approach, our results are independent from those in current literature.

There are certain limitations in our current work that can be overcome in future works by addressing certain challenges. The relatively small size of our training dataset can cause loss of accuracy and precision. The disproportional distribution of instances of cracks, potholes and smooth road conditions introduces a bias and may have affected the individual precision and recall rates. Since neural networks generally require a very large data set to accurately train itself using direct data, results can be improved by addressing our shortage of data. For our study, we implemented a fully connected MLP network with equal number of neurons in each hidden layer. Exploring different neural network architectures could help improve results. A separate study was conducted to analyze the influence of different data acquisition conditions such as the type of car, quality of car suspension, position of smartphone, use of high sampling rate accelerometers etc. It was seen that these factors significantly impact the quality of signal captured and are important factors to be considered in future works. We also observed that the machine learning algorithms discussed in this thesis can be used to classify road vibration data very quickly after the classifiers have been trained. This encourages the possibility of implementing these approaches on a large scale in real time using crowdsensing to collect data.

5. CONCLUSION

Based on the results and discussions presented in this thesis, it can be concluded that the use of machine learning techniques to classify road anomalies based on sensor data collected from smartphones is a viable and cost effective way of monitoring road conditions. Machine learning models trained with features extracted from all three coordinate axes give significantly higher accuracy, precision and recall rates as compared to models trained with features from only the axis perpendicular to ground. This trend is observed in all three machine learning techniques explored in this thesis. It justifies our initial hypothesis that useful and relevant information regarding the road condition is present in data collected with respect to all three coordinate axes. MLP neural networks perform particularly well at classifying potholes, cracks and smooth road when trained with features extracted from raw data. The use of neural networks trained using direct input data has immense potential in road surface anomaly assessment using sensors. They provide scope for scalability and real time system realizations as big data analytics gains more importance. With the increase in appeal of smart cars and self-driving cars that possess multiple sensors, data collected from them could be used to provide road surface assessment to improve safety and infrastructure quality.

REFERENCES

[1] J. Eriksson, L. Girod and B. Hull, etc. "The Pothole Patrol: Using a Mobile Sensor Network for Road Surface Monitoring", 6th International Conference on Mobile Systems, Application, and Services (MobiSys 2008), Breckenridge, USA, June, 2008, pp. 29-39.

[2] R. Bishop, "A Survey of Intelligent Vehicle Application Worldwide", IEEE Intelligent Vehicles Symposium (IV 2000), Dearborn, MI, USA, May 2000, pp. 25-30.

[3] U.S. Department of Transportation, Traffic Safety Facts – Crash Stats, June 2015. <http://www-nrd.nhtsa.dot.gov/Pubs/812160.pdf>

[4] Pothole (2002). http://www.pothole.info

[5] Roadscanners. Roadscanners oy (1998). http://www.roadscnners.fi

[6] J. Budras, A synopsis on the current equipment used for measuring pavement smoothness. http://www.fhwa.dot.gov/pavement/smoothness/rough.cfm, August, 2001.

[7] Texas DOT Pavement Management Information System (PMIS) Rater's Manual. Retrieved from Texas State Department of Transporation: ftp://ftp.dot.state.tx.us/pub/txdot-info/cst/raters_manual.pdf

[8] "Pothole Marker And More - Apps on Google Play," *Google Play*. [Online]. https://play.google.com/store/apps/details?id=com.phonegap.pointpotholes.

[9] Fill That Hole, 2015. http://www.fillthathole.org.uk/iphone.

[10] Shamsabadi, S.S.; Wang, M.L.; Birken, R. Pavemon: A GIS-based pavement monitoring system using large amounts of near-surface geophysical sensor data. In Proceedings of the 27th Annual Symposium on the Application of Geophysics to Engineering and Environmental Problems (SAGEEP), Boston, MA, USA, 16 March 2014.

[11] Mohan, P.; Padmanabhan, V.N.; 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, Raleigh, NC, USA, 4–7 November 2008.

[12] "Trafficsense: Rich monitoring of road and traffic conditions using mobile smartphones," Microsoft Research, Tech. Rep. MSR-TR-2008-59, April 2008. [Online]. Available: http://research.microsoft.com/pubs/70573/tr-2008-59.pdf

[13] Bhoraskar, R.; Vankadhara, N.; Raman, B.; Kulkarni, P. Wolverine: Traffic and road condition estimation using smartphone sensors. In Proceedings of the 2012 Fourth International Conference on Communication Systems and Networks (COMSNETS 2012), Bangalore, India, 3–7 January 2012.

[14] Sayers, M.W.; Karamihas, S.M. (1998). "Little Book of Profiling" (PDF). University of Michigan Transportation Research Institute. Retrieved 2010-03-07.

[15] L. Forslöf and H. Jones, "Roadroid: Continuous Road Condition Monitoring with Smart Phones," *Journal of Civil Engineering and Architecture*, vol. 9, no. 4, 2015.

[16] X. Li and D. W. Goldberg, "Toward a mobile crowdsensing system for road surface assessment," *Computers, Environment and Urban Systems*, vol. 69, pp. 51–62, 2018.

[17] J. Lepine, V. Rouillard, and M. Sek, "On the Use of Machine Learning to Detect Shocks in Road Vehicle Vibration Signals," *Packaging Technology and Science*, vol. 30, no. 8, pp. 387–398, Apr. 2016.

[18] J. Lepine, "An Optimised Machine Learning Algorithm for Detecting Shocks in Road Vehicle Vibration," Ph.D. Thesis, College of Engineering and Science, Victoria Univ., Melbourne, Australia. 2016.

[19] Allouch, A., Koubaa, A., Abbes, T., & Ammar, A. (2017). RoadSense: smartphone application to estimate road conditions using accelerometer and gyroscope. IEEE Sensors Journal, 17(13), 4231–4238. http://dx.doi.org/10.1109/jsen.2017.2702739.

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

[21] G. Singh, D. Bansal, S. Sofat, and N. Aggarwal, "Smart patrolling: An efficient road surface monitoring using smartphone sensors and crowdsourcing," *Pervasive and Mobile Computing*, vol. 40, pp. 71–88, 2017.

[22] V. Douangphachanh and H. Oneyama, "A Model For The Estimation Of Road Roughness Condition From Sensor Data Collected By Android Smartphones," *Journal of Japan Society of Civil Engineers, Ser. D3 (Infrastructure Planning and Management)*, vol. 70, no. 5, 2014.

[23] F. Seraj, B. J. V. D. Zwaag, A. Dilo, T. Luarasi, and P. Havinga, “RoADS: A Road Pavement Monitoring System for Anomaly Detection Using Smart Phones,” *Lecture Notes in Computer Science Big Data Analytics in the Social and Ubiquitous Context*, pp. 128–146, 2016.

[24] K. Chen, M. Lu, X. Fan, M. Wei, and J. Wu, “Road condition monitoring using on-board Three-axis Accelerometer and GPS Sensor,” *2011 6th International ICST Conference on Communications and Networking in China (CHINACOM)*, 2011.

[25] A. Mohan and S. Poobal, “Crack detection using image processing: A critical review and analysis,” *Alexandria Engineering Journal*, 2017.

[26] A. Tedeschi and F. Benedetto, “A real-time automatic pavement crack and pothole recognition system for mobile Android-based devices,” *Advanced Engineering Informatics*, vol. 32, pp. 11–25, 2017.

[27] J. Masino, J. Thumm, M. Frey, and F. Gauterin, “Learning from the crowd: Road infrastructure monitoring system,” *Journal of Traffic and Transportation Engineering (English Edition)*, vol. 4, no. 5, pp. 451–463, 2017.

[28] J. Dixon-Warren, “Comparing the InvenSense and Bosch Accelerometers Found in the iPhone 6,” Chipworks, 25-Sep-2014. [Online]. Available: https://www.chipworks.com/about-chipworks/overview/blog/comparing-invensense-and-bosch-accelerometers-found-iphone-6.

[29] V. Douangphachanh and H. Oneyama, “Formulation of a simple model to estimate road surface roughness condition from Android smartphone sensors,” 2014 IEEE Ninth International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP), 2014.

[30] V. Astarita, M. V. Caruso, G. Danieli, D. C. Festa, V. P. Giofrè, T. Iuele, and R. Vaiana, “A Mobile Application for Road Surface Quality Control: UNIquALroad,” Procedia - Social and Behavioral Sciences, vol. 54, pp. 1135–1144, 2012.

[31] Pavement Manual, Texas Department of Transportation. (online) http://onlinemanuals.txdot.gov/txdotmanuals/pdm/pdm.pdf

[32] E. Gadelmawla, M. Koura, T. Maksoud, I. Elewa, and H. Soliman, “Roughness parameters,” Journal of Materials Processing Technology, vol. 123, no. 1, pp. 133–145, 2002

[33] L. Sun, “Simulation of Pavement Roughness and IRI Based on Power Spectral Density”, Mathematics and Computers in Simulation, 2003, vol. 61, pp. 77-88.

[34] D. Xu, A. Mohamed, and R. Yong, etc. “Development of a criterion for road surface roughness based on power spectral density function”, Journal of Terramechanics, 1992, vol. 29, pp. 477-486.

[35] W. Staszewski, K. Giacomin, “Application of the Wavelet based FRFs to the Analysis of Nonstationary Vehicle Data,” Proceedings- SPIE the International Society for Optical Engineering, pp. 425-431, 1997

[36] K.R. Griffiths, “An Improved Method for Simulation of Vehicle Vibration Using a Journey Database and Wavelet Analysis for the Pre-Distribution Testing of Packaging,” Ph.D. Dissertation, Univ. of Bath, Bath, UK, 2012

[37] L. Wei, T. F. Fwa, and Z. Zhe, “Wavelet Analysis and Interpretation of Road Roughness,” Journal of Transportation Engineering, vol. 131, no. 2, pp. 120–130, 2005.

[38] “Statistics and Machine Learning Toolbox,” *MATLAB*. [Online]. Available: https://www.mathworks.com/products/statistics.html.

[39] S. Shalev-Shwartz and S. Ben-David, *Understanding machine learning: from theory to algorithms*. New York: Cambridge University Press, 2016.

[40] V. S. Cherkassky and F. Mulier, *Learning from data: concepts, theory, and methods*. Hoboken, NJ: IEEE Press, 2007.

[41] Multiclass Model for Support Vector Machines or Other Classifiers - MATLAB. https://www.mathworks.com/help/stats/classificationecoc-class.html.

[42] Gordon, A., Breiman, L., Friedman, J., Olshen, R. and Stone, C. (1984). Classification and Regression Trees. Biometrics, 40(3), p.874.

[43] F. Rosenblatt, Principles of neurodynamics; perceptrons and the theory of brain mechanisms. Washington, D.C.: Spartan Books, 1962.

[44] Rumelhart, David E., Geoffrey E. Hinton, and R. J. Williams. "Learning Internal Representations by Error Propagation". David E. Rumelhart, James L. McClelland, and the PDP research group. (editors), Parallel distributed processing: Explorations in the microstructure of cognition, Volume 1: Foundation. MIT Press, 1986.

[45] Fabian Pedregosa; Gaël Varoquaux; Alexandre Gramfort; Vincent Michel; Bertrand Thirion; Olivier Grisel; Mathieu Blondel; Peter Prettenhofer; Ron Weiss; Vincent Dubourg; Jake Vanderplas; Alexandre Passos; David Cournapeau; Matthieu Perrot; Édouard Duchesnay (2011). "Scikit-learn: Machine Learning in Python". Journal of Machine Learning Research. 12: 2825–2830

[46] “sklearn.neural_network.MLPClassifier,” *sklearn.neural_network.MLPClassifier - scikit-learn 0.19.1 documentation*. [Online]. Available: http://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html.

[47] Malouf, Robert (2002). A comparison of algorithms for maximum entropy parameter estimation (PDF). Proc. Sixth Conf. on Natural Language Learning (CoNLL). pp. 49–55. Archived from the original (PDF) on 2013-11-01.

[48] Andrew, Galen; Gao, Jianfeng (2007). "Scalable training of L_1-regularized log-linear models". Proceedings of the 24th International Conference on Machine Learning. doi:10.1145/1273496.1273501. ISBN 9781595937933.

[49] D.P. Kingma and J. Ba., “Adam: A Method for Stochastic Optimization,” 3rd International Conference for Learning Representations, 2015. [online]. https://arxiv.org/abs/1412.6890v9.

[50] Perttunen, M., Mazhelis, O., Cong, F., Kauppila, M., Leppänen, T., Kantola, J., Riekki, J. (2011). Distributed road surface condition monitoring using mobile phones. Ubiquitous intelligence and computing (pp. 64–78)Berlin Heidelberg: Springer. http://dx.doi.org/10.1007/978-3-642-23641-9_8.