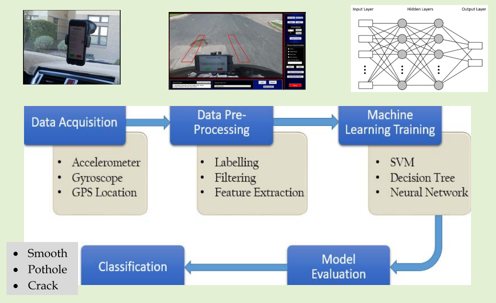
A Machine Learning Approach to Road Surface Anomaly Assessment Using Smartphone Sensors

Akanksh Basavaraju, Jing Du, Fujie Zhou, and Jim Ji

***Abstract*—Road surface quality is essential for improving driving experience and reducing traffic accidents. Traditional road condition monitoring systems are limited in their temporal (speed) and spatial (coverage) responses needed for maintaining overall road quality. Several alternative systems have been proposed that utilize sensors mounted on vehicles. In particular, with the ubiquitous use of smartphones for navigation, smartphone-based road condition assessment has emerged as a promising new approach. In this paper, 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 focuses on classification of three main class labels- smooth road, potholes, and deep transverse cracks. We hypothesize that using features from all three axes of the sensors provides 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 indicate that models trained with features from all axes of the smartphone sensors outperform models that use only one axis. We also observe that the use of neural networks provides a significantly improved data classification. The machine learning approach discussed here can be implemented on a larger scale to monitor roads for defects that present a safety risk to commuters as well as to provide maintenance information to relevant authorities.**



Assessing road surface anomaly using smartphone sensors and machine learning

***Index Terms*—Support vector machines, neural network, multilayer perceptron, decision tree, road condition, pavement condition, pothole, crack, smartphone sensor, accelerometer.**

I. Introduction

ROAD condition monitoring is a challenging worldwide problem in the field of transportation and road infrastructure [1], [2]. Poor road surface conditions create a risk of damage to vehicles and increase chances of traffic accidents. Each year, thousands of people are injured or killed on roadways due to poor road quality [3]. Significant resources are spent on regular road repair and maintenance. 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 the 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 minimize risk of traffic accidents. However, maintaining roads regularly is a challenging task due to the heavy traffic, weather conditions and high costs of manpower. Since frequent repairs are needed to prevent road quality from deteriorating, a reliable and low-latency road condition monitoring system is highly desirable to identify critical road segments and to optimally allocate limited maintenance resources. At present, road inspection is usually done with a manual process [5]. The manual approach brings two potential problems: variation in inspection results due to inspectors' personal bias, and the difficulty of a high-frequent inspection and coverage. To overcome the limitations of manual inspection methods, a variety of automated road surface inspection methods have been proposed [6], [7]. Representative technologies include vision based methods [5], [8]–[11], LiDAR scanning [12] ground penetration radar (GPR) [5], [13], [14], natural lighting methods [15] and a combination of multiple sensors [16].

Manuscript received September 19, 2019; accepted October 23, 2019. Date of publication November 11, 2019; date of current version February 5, 2020. The associate editor coordinating the review of this article and approving it for publication was Prof. Meribout Mahmoud. *(Corresponding author: Jim Ji.)*

A. Basavaraju and J. Ji are with the Department of Electrical Engineering, Texas A&M University, College Station, TX 77840 USA (e-mail: akanksh.b@tamu.edu; jimji@tamu.edu).

J. Du is with the Department of Civil and Coastal Engineering, University of Florida, Gainesville, FL 32611 USA (e-mail: eric.du@essie.ufl.edu).

F. Zhou is with the Transportation Institute, Texas A&M University, College Station, TX 77840 USA (e-mail: f-zhou@tti.tamu.edu).

Digital Object Identifier 10.1109/JSEN.2019.2952857

However, these sensor technologies can be expensive, costing between $8,000 and $220,000 [17]. Due to the scarce public funds, such systems cannot be effectively deployed on a large-scale road network to regularly check for repairs.

To fill the gap, this paper tests an automated, crowdsourcing based approach that predicts road conditions based on running vehicles' vibration data via sensors built in most smartphones. We recognize that the vehicle vibration data as an instrument for road damage detection is challenged by the low quality and variability of the collected data, as it can be affected by the different models, ages, conditions of the vehicles, as well as the varying driving behaviors of the drivers. Therefore, this paper aims to contribute in the following areas:

1. Providing a state of art evaluation of the road inspection methods;
2. Providing empirical evidence about the feasibility of using vehicle vibrations for the automated road inspection;
3. Applying and analyzing different machine learning techniques for multiclass classification for vehicle-vibration enabled road inspection. Specially, a benchmarking study is performed to identify the potentials and challenges of multiple machine learning techniques for road damage recognition.

In this study, the classified road damage types include 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 the time domain, the frequency domain and the wavelet domain from all three Cartesian coordinate axes of sensor data to train our classifiers. The remainder of this manuscript provides a review of the state of art in automated road inspection, the benchmarking study and findings for future investigations.

II. Related Works

At present, road inspection is usually done with a manual process [5]. State Departments of Transportation have published standard procedures for manual inspection [18], such as Texas Pavement Management Information System Rater's Manual [19]. The manual approach is to visually inspect road conditions and record data regarding the condition of road surfaces. This can be combined with factors like ride quality, structural adequacy, skid resistance, climate, and traffic data to help describe the quality of road networks such as the state-maintained highway system [20]. Although easier to implement, the manual inspection is always costly and inaccurate. As an alternative to the manual approach, sensor-based approaches have been proposed and tested. A representative method is via the use of radars. For example, Benedetto *et al.* [21] tested the inspection procedure with GPR. Their data indicates a good performance of the numerical algorithm and electronic equipment assessing the reliability of the procedure and shows a 20 percent or lower false alarm [3]. Following the previous findings, Benedetto *et al.* [22] further tested an algorithm based on GPR data to significantly lower the amount of data to be processed and the time required for data processing. The method has also been successfully tested in mobile platforms [23]–[25]. Another track of methods relies on imagery data of pavement damages. For example, Pu *et al.* [26] presented a framework for road damage recognition from mobile laser scanned point clouds, and achieved an 87% accuracy in pothole recognitions. Su *et al.* [15] proposed a dual-light inspection (DLI) method to predict road damages based on pixel differences under at least two different lighting conditions. The results based on 212 samples indicate that the DLI method can significantly reduce false alarms.

Recently a crowdsourcing approach has been tested to reduce the data collection cost. Potholes Marker [12] and Fill That Hole [27] are applications developed where users take photos of potholes and submit them to a central server. They are less practical on a large scale as users are reluctant to stop and record the pothole locations. PAVEMON [28] is a GIS web-based pavement monitoring system by VOTERS that uses data from multiple sensors such as accelerometer, microscope, tire pressure sensor, imaging, radar etc. to evaluate different road distress parameters. However, the need for specialized vehicular setup restricts the use of this system as a platform for mass data collection. Nericell [29] and TrafficSense [30], systems developed by Microsoft Research India, uses sensors such as accelerometers, microphones and GPS on Windows smartphones to detect potholes using simple threshold-based heuristics. These thresholds are assigned based on observation and hence remain subjective. Wolverine [31] detects road bumps or potholes based on variations in accelerometer data perpendicular to the ground. The thresholds are determined using mean and standard deviation only, ignoring higher order statistics. Pothole Patrol (P^2) [2], developed by Massachusetts Institute of Technology, is based on a simple machine-learning approach to analyze patterns in accelerometer data using X–Z ratio and speed-Z ratio.

Another approach is to measure the International Roughness Index (IRI) using a quarter-car vehicle math model [32]. Forslöf *et al.* [33] is an Android smartphone application that surveys road condition and classifies it as good, satisfactory, unsatisfactory and poor based on calculated and estimated IRI. Li and Goldberg [34] calculated 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.* [35], [36] implemented different machine learning algorithms to separate non-stationary vibrations and transient shocks in road vehicle vibration (RVV) signals using accelerometer data. He concluded that machine learning algorithms could be optimized and tuned to achieve a 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 reproduced typical vehicle dynamic behavior. Allouch *et al.* [37] used machine learning techniques such as C4.5 Decision Tree, SVM and Naïve Bayes to label road conditions as 'Smooth' or 'Potholed'. Bhoraskar *et al.* [31] used k-means clustering and SVM to label road conditions

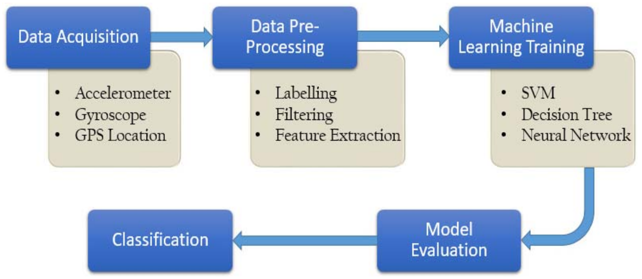


Fig. 1. Overall block diagram of the proposed system.

as 'Smooth' or 'Bumpy'. Silva *et al.* [38] approached the problem with data mining using Scikit-learn and Weka to detect unlevelled manholes, short bumps and long bumps. Several other works used similar techniques and focused on either estimating a roughness metric or detecting potholes only [39]–[42]. 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], [43], but these techniques are memory and computation intensive. With significant advancements in big data analytics and a push for smart cars in recent years, a high volume of driving data can be collected from users and processed to obtain useful information. Li and Goldberg [34] and Masino *et al.* [44] proposed the use of crowd-sensing to obtain data and classify road conditions.

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. The majority of current literature focuses on binary classifications using simple machine learning techniques or threshold-based heuristics. Multiclass classification has not been explored for different stages of road deterioration. Data used for anomaly detection only focuses on vibration signals collected from acceleration data in the direction of gravity. Information and relationships between data that may be present in other directions orthogonal to the direction of gravity are not taken into consideration. Finally, the use of neural networks for multiclass classification has not been explored.

III. 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.

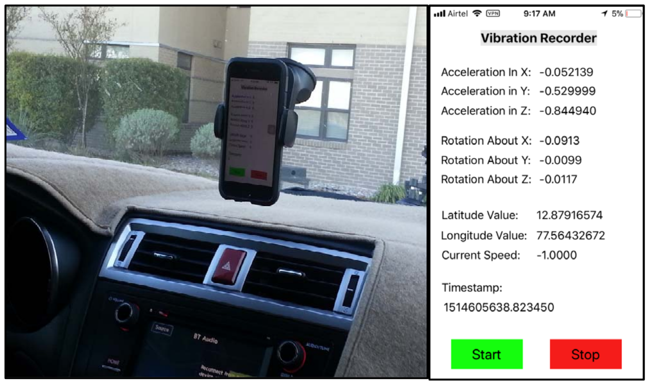
Our general methodology consists of five stages whose system block diagram is shown in Fig. 1. The data acquisition stage deals with obtaining and recording data required for our system. This data is acquired using the accelerometer, gyroscope and GPS sensors present in smartphones. The data collected 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 to test our hypothesis. In our study, all data processing, machine learning training, and model evaluation were performed offline. For real-world applications, only filtering and feature extraction and classification will be needed after an optimal processing pipeline is selected.

A. Data Acquisition

To build the system described above, an Apple iPhone 6 was chosen to collect accelerometer, gyroscope and GPS data in this pilot study. The device was chosen due to our familiarity with programming on iOS devices. We do not anticipate difficulty implementing the data acquisition system in Android devices. The iPhone 6 contains two separate accelerometer chips- Bosch BMA280 and InvenSense MPU-6700 [45]. 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 [40], [46]. 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 Fig. 2.

A DJI Osmo was used to record video of the road surface to facilitate the road condition labelling in the pre-processing stage. The Osmo was mounted on 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 views of the road ahead. In order to synchronize the video frames with the data recorded by the Vibration Recorder, another smartphone that displayed the Epoch/UNIX timestamp was placed in the field of view of the Osmo. The Osmo setup and a snapshot of its field of view are shown in Fig. 3.


Airtel 9:17 AM 5%

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

Fig. 2. Setup of the iPhone and screenshot of the vibration recorder app.

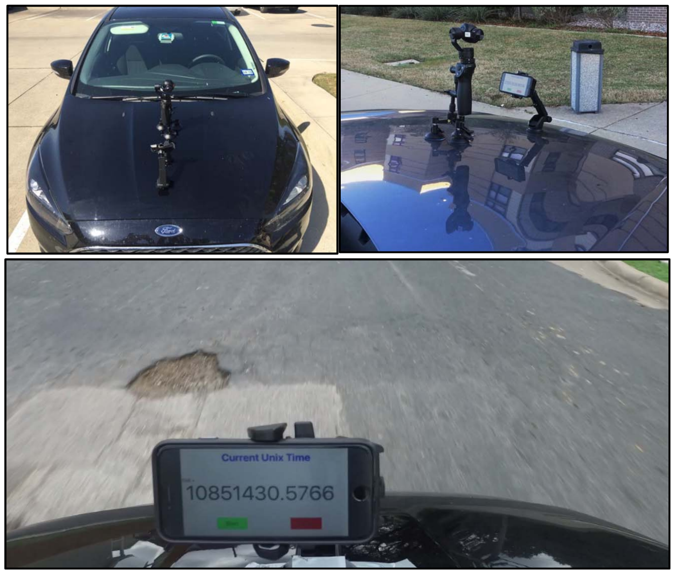


Fig. 3. Osmo setup and snapshot of the Osmo setup's field of view.

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 4 data collection runs were conducted in and around College Station, Texas, covering road surfaces with asphalt pavements.

B. Data Pre-Processing

Data acquired was pre-processed in several stages to make it more coherent and pragmatic. First, the acceleration data collected was virtually reoriented to a global frame of reference to remove variations due to the phone's position and orientation. The acceleration and gyroscope measurements were recorded in a three-dimensional Cartesian coordinate system with respect to the phone's frame of reference as shown in Fig. 4. 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 Fig. 5.

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

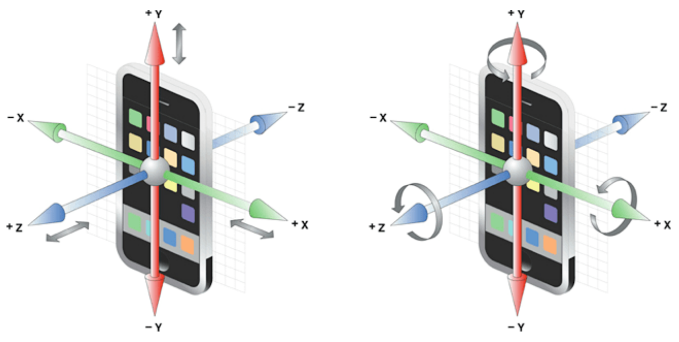


Fig. 4. Cartesian coordinate axes of iPhone accelerometer and gyroscope (adapted from [47]).

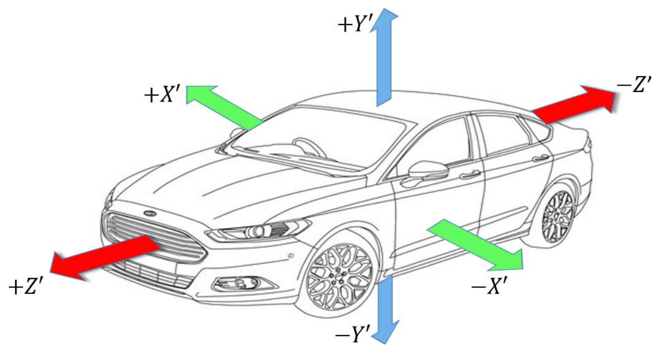


Fig. 5. Global frame of reference: Cartesian coordinate axes w.r.t. ground.

a car is at rest on a flat surface, the acceleration values would be:

$$a_x = 0\text{m/s}^2, \quad a_y = 9.81\text{m/s}^2 \text{ and } a_z = 0\text{m/s}^2$$

Equations (1) to (4) are used to calculate two of the three Euler angles and reorient acceleration values to the global frame of [48]. $a'_x, a'_y a'_z$ are the acceleration values with respect to the global reference frame while α and β are the roll and pitch angles, respectively. Fig. 6. shows the plot of the acceleration data of a 1.5s window before and after reorientation.

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

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

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

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

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 based on guidelines and descriptions provided in pavement maintenance manuals from the Texas Department of Transportation [20], [49]. 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.

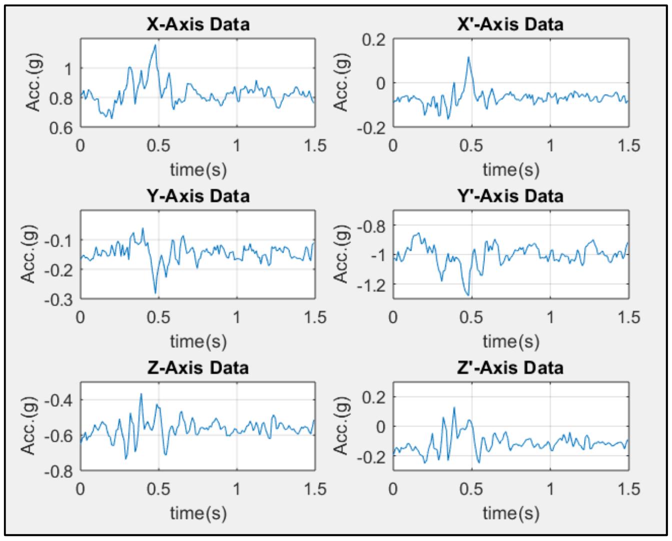


Fig. 6. Reorientation of acceleration data to global frame of reference.

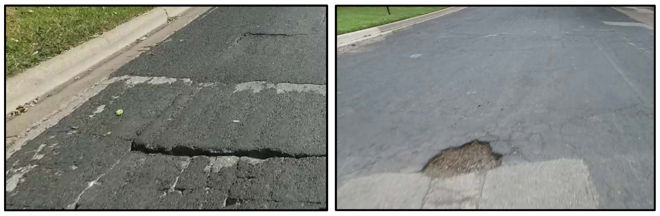


Fig. 7. Instances of road anomalies: deep transverse crack and pothole.

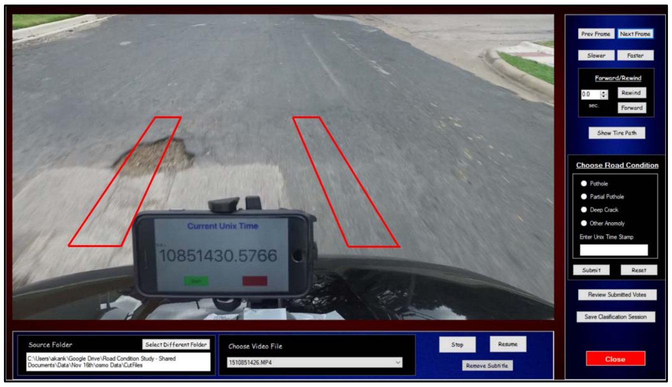


Fig. 8. Road condition classifier software with tire-trajectory overlay.

A custom software application was developed in Java platform to help label the recorded video data [50]. In its current form, the software runs as a stand-alone program, even though it can be potentially extended to serving as a web-based portal for crowdsourcing. It enables 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 Fig. 8. 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.

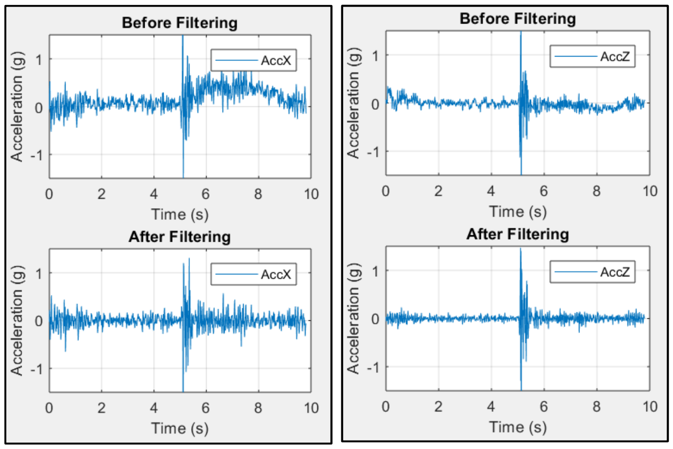


Fig. 9. Acceleration signal in X' and Z' axis before and after filtering.

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 differences in sampling rates of the Accelerometer/Gyroscope and the GPS sensor, the GPS data and the vehicle speed was interpolated using a spline function. 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 Fig. 9. To analyze the information contained in higher frequency bands due to the anomalies, a low pass filter or smoothing filter was not applied.

The continuously filtered data was then converted into segments of data windows, each with 100 data samples and a 50% overlap with the previous window. 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. The filtered data segments and their associated labels (smooth road, potholes, and deep transverse cracks) were then saved for next-step processing, which is feature extraction.

C. 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.* [51] discussed 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 and its peaks, troughs and signal envelopes.

In the frequency domain, the power spectral density of vibration signals provides very useful information that could be used to distinguish different road conditions [52], [53]. 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 [54]–[56]. Griffiths [54] 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 classify road anomalies. Accelerations in X' and Z' directions were considered for driving events only. However, we believe that more information regarding road anomalies presents 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 between cracks and potholes, considering that 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. These features include Maximum Value, Minimum Value, Mean Value, RMS Value, Peak-to-Peak Value and Ten-Point Average Value calculated from time-domain signals; average band power, RMS band value and maximum band value from power spectral density; and RMS values and ten-point averages of three Wavelets at scales 4 and 5 (Mortlet, Daubechies 6 and 10 Daubechies wavelets) in the Wavelet domain. Note that signals from all three axes were included. These features along with the manually labelled road conditions for the 1011 data sets were then used in the training, validating, and testing of the machine-learning algorithms.

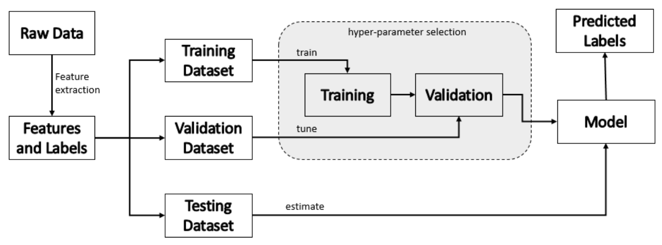


Fig. 10. General workflow diagram of machine learning algorithms.

D. Machine Learning Approaches

Machine Learning (ML) is an application of Artificial Intelligence that provides computer systems with the ability to learn and improve from experience without explicit programming. Once a computer algorithm is trained in ML, it can apply the relationship learnt during training to solve similar problems. In this work, we apply and analyze different ML algorithms for road surface assessment collected with sensors on smartphones.

Fig. 10. 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. Vehicle vibrations are acceleration signals recorded by the smartphone that 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. Note that we used supervised ML models because they will provide a direct outcome for the road conditions based on the collected signals. 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 includes implementations of various machine learning classifiers [57]. 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 [58], [59]. 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 according to the literature.

For ML training and evaluation, the complete dataset of 1010 samples, each consisting of a feature vector and road condition label, was randomized and divided into training and testing dataset with an 80:20 ratio, keeping the proportion of the classes in each dataset constant. All data was collected using the device described in Section III.A and processed using the procedures described in Section III.B. 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 the Model Evaluation Parameters section.

1) Support Vector Machines (SVM): 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. 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 at least 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 (5) [60].

$$\hat{k} = \underset{k}{\operatorname{argmin}} \frac{\sum_{l=1}^{L} |m_{kl}| g(m_{kl}, s_l)}{\sum_{l=1}^{L} |m_{kl}|} \tag{5}$$

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) Decision Trees: Decision trees, also known as classification trees or 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 [61]. The decision tree is an algorithm that classifies data through a cascade of statistical tests as shown in Fig. 11. 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. Decision trees have quick training and prediction speeds, a moderate predictive accuracy and low computational memory requirements.

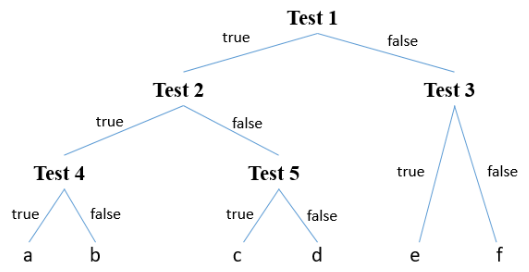


Fig. 11. Decision tree structure used in the work.

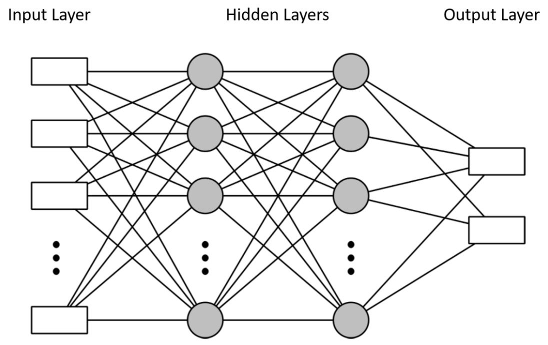


Fig. 12. Structure of a neural network classifier with two hidden layers. In this work, we use 7 to 8 hidden layers.

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 [37]. Similar to our approach to SVM, we develop a simple classification decision tree and a cross validated tree to reduce over-fitting.

3) Neural Networks: Neural networks represent a popular machine learning framework that attempts 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 are 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 multi-layered networks called hidden layers, as shown in Fig.12. 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.

A multilayer perceptron (MLP) is a class of feedforward neural networks comprised of at least one hidden layer and uses backpropagation for training its models [62], [63]. Each neuron in the hidden layers uses a nonlinear activation function which distinguishes it from a linear perceptron. Each neuron

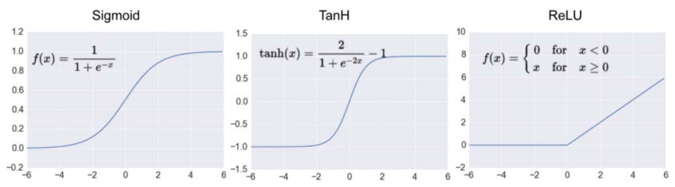


Fig. 13. Activation function plots for Sigmoid, Tanh and ReLU.

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. We chose this NN architecture because the signal/feature vectors used in the study are 1-D in nature. For 2-D or higher dimensional data, other NN architectures such as convolutional NN (CNN) might be more powerful and appropriate.

To realize MLP networks, the Scikit-learn library for supervised Neural Network was used [64]. MLP can be implemented similarly in Matlab and Java platforms. 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 MLP Classifier 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 [65]. 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 [66], [67]. 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 [68], 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.

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 explore not only 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.

E. 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 (6).

$$Accuracy = \frac{(TP + TN)}{(TP + TN + FP + FN)} \tag{6}$$

$$Loss = \frac{(FP + FN)}{(TP + TN + FP + FN)} \tag{7}$$

$$Precision = \frac{TP}{(TP + FP)} \tag{8}$$

$$Recall = \frac{TP}{(TP + FN)} \tag{9}$$

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 are 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 insight into the classifier's performance.

All algorithms were implemented and tested 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

TABLE I
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

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.

IV. 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.

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 I. These durations correspond to the average time taken over 200 trial runs. It can be noted that even though extracting one axis features was faster, the feature extraction process for both cases were fast enough to complete 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.

A. 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 II. The precision and recall rates are displayed for each of the three classes to analyze bias.

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 III.

TABLE II
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

TABLE III
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 II, it is observed that the classifier trained with features from all three axes had a lower loss and outperformed the classifier trained with features from Y' axis only. The average test accuracy was slightly lower when all three axes are used against only one axis since the number of dependency variables was greater. The precision and recall rates for the individual classes were also higher when all three axes are used. The recall rate for cracks showed the most significant improvement, increasing by over 20%, while the recall for smooth road decreased by about 3.5%. The precision and recall rates for potholes remined very comparable. Table III shows that the cross validated classifier with features from all three axes has a lower training loss and lower cross validated errors as well.

B. Decision Tree

The decision trees were 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 usually faster to train. However, they create a highly varying set of hyperparameters with each iteration such as number of nodes and node thresholds. There is 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 IV. The results of the cross validated ECOC classifier for Decision Tree is tabulated in Table V.

TABLE IV
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

TABLE V
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 IV, it is observed again that the classifiers trained with features from all three axes outperformed the classifiers trained with only Y' axis. Precision and Recall for cracks increased by over 10% each and training loss and testing accuracy shows slight improvements. Table V 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.

C. 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 were 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 grew 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 7 and 8 gave the most optimal results. However, 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.

TABLE VI
MLP IMPLEMENTATION USING RELU- RESULTS

MLP Hidden Layer Count	Using Features from all Axes			Using Features from Y' Axis Only		
AVG. TEST ACCURACY						
7	**0.9212**			0.8921		
8	**0.9190**			0.8919		
AVG. 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
AVG. 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

TABLE VII
MLP IMPLEMENTATION USING TANH- RESULTS

MLP Hidden Layer Count	Using Features from all Axes			Using Features from Y' Axis Only		
AVG. TEST ACCURACY						
7	**0.9122**			0.8978		
8	**0.9149**			0.8950		
AVG. PRECISION RATES						
	Crack	Pothole	Smooth	Crack	Pothole	Smooth
7	**0.486**	**0.757**	**0.964**	0.395	0.705	0.961
8	**0.491**	**0.754**	**0.967**	0.364	0.708	0.958
AVG. 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

The final analysis compared the performance of the MLP neural networks for input feature vector lengths 162 and 54 while implementing ReLU and Tanh with LBFGS. The test accuracy, precision and recall for these models are tabulated in Table VI and Table VII.

Based on Table VI and Table VII it can be observed that the average test accuracy, precision and recall rates were higher for the MLP models using features from all three axes compared to only a single axis. In models trained using features from only one axis, using Tanh as the activation function yielded higher precision and recall rates among the three classes. However, when utilizing features from all three axes, ReLU standed out in its high precision and recall rates for cracks.

TABLE VIII
MLP CLASSIFICATION USING DIRECT DATA FOR RELU- RESULTS

MLP Hidden Layer Count	Using Features from all Axes			Using Features from Y' Axis Only		
AVG. TEST ACCURACY						
7	0.8027			0.8157		
8	0.7946			0.8112		
AVG. PRECISION RATES						
	Crack	Pothole	Smooth	Crack	Pothole	Smooth
7	0.283	0.329	0.918	0.271	0.423	0.911
8	0.408	0.301	0.906	0.258	0.469	0.905
AVG. RECALL RATES						
	Crack	Pothole	Smooth	Crack	Pothole	Smooth
7	0.139	0.672	0.912	0.156	0.607	0.911
8	0.142	0.673	0.919	0.141	0.621	0.921

TABLE IX
CLASSIFIER PERFORMANCE: TESTING TIME

Classifier	Avg. Time to Classify One Window (µs)
SVM	29.4
Decision Tree	4.8
MLP1	36.0
MLP2 (Direct Data)	72.1

The precision and recall rates for potholes and smooth roads remained 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 was directly used as the input to the neural network and was evaluated over 20 iterations. The single axis input vector contained length 100 and input vector with all axes had 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 VIII.

It is observed that the average test accuracy of MLP models using direct data was lower compared to MLP models trained using extracted features as input. The average precision and recall rates for the case of cracks and potholes were also lower. However, it was already anticipated that training neural networks without features would require a larger dataset and we were 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 required approximately 70ms and 25ms for the case of 3 axes and 1 axis respectively. Table IX 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 paper. 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.

V. DISCUSSION

Based on the results of the study, we observed that the machine learning approaches implemented were 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 a large number of features from all three axes to train multiclass machine learning classifiers was a novel approach, our results extended in current literature.

There are certain limitations in our current work that shall be addressed in future works. 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 impacted the quality of signal captured and were important factors to be considered in future works. We also observed that the machine learning algorithms discussed in this paper could be used to classify road vibration data very quickly after the classifiers were trained. This indicates the possibility of scaling up the implementation of these approaches using crowd-sensed data for real time detections.

VI. CONCLUSION

Based on the results and discussions presented in this paper, 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 paper. It supports our initial hypothesis that useful and relevant information regarding the road condition presents 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 show potential for real time big data analytics. With the expected increase in autonomous vehicles that possess multiple sensors, more data would become available for road surface assessment to improve safety and infrastructure quality.

Acknowledgment

The authors would like to thank M. Saulnier, D. Knox and K. Tan, students at Texas A&M University, for their support and contribution in collecting and labeling data for this project.

References

[1] R. Bishop, "A survey of intelligent vehicle applications worldwide," in *Proc. IEEE Intell. Vehicles Symp.*, Oct. 2000, pp. 25–30.

[2] J. Eriksson, L. Girod, B. T. Hull, R. Newton, S. Madden, and H. Balakrishnan, "The pothole patrol: Using a mobile sensor network for road surface monitoring," in *Proc. 6th Int. Conf. Mobile Syst., Appl., Services*, Jul. 2008, pp. 29–39.

[3] *Traffic Safety Facts—Crash Stats*, U.S. Department of Transportation, Washington, DC, USA, Jun. 2015.

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

[5] R. Medina, J. Gómez-García-Bermejo, and J. E. Zalama, "Automated visual inspection of road surface cracks," in *Proc. Int. Symp. Automat. Robot. Construct. (ISARC)*, 2010, pp. 14–20.

[6] J. D. Achenbach, "On the road from schedule-based nondestructive inspection to structural health monitoring," in *Proc. 6th Int. Workshop Struct. Health Monit., Quantification, Validation, Implement. (IWSHM)*. Stanford, CA, USA: DEStech, 2007, pp. 16–28.

[7] S. Cafiso *et al.*, "Tools for road inspection and safety management," in *Proc. 3rd Int. Conf. Trans. Infrastruct.* Pisa, Italy, 2014.

[8] R. A. Ferguson *et al.*, "Road pavement deterioration inspection system," Google Patents 6 615 648 B1, Sep. 9, 2003.

[9] Y. Huang and B. Xu, "Automatic inspection of pavement cracking distress," *J. Electron. Imag.*, vol. 15, no. 1, Jan. 2006, Art. no. 013017.

[10] E. Salari and G. Bao, "Automated pavement distress inspection based on 2D and 3D information," in *Proc. IEEE Int. Conf. Electro/Inf. Technol.*, May 2011, pp. 1–4.

[11] S. Varadharajan, S. Jose, K. Sharma, L. Wander, and C. Mertz, "Vision for road inspection," in *Proc. IEEE Winter Conf. Appl. Comput. Vis.*, Mar. 2014, pp. 115–122.

[12] *Pothole Marker And More APK*, Mountain View, CA, USA, Google Play, 2017.

[13] Roadscanners oy. (1998). *Roadscanners*. [Online]. Available: http://www.roadscnners.fi

[14] M. R. Mahmoudzadeh, J. B. Got, S. Lambot, and C. Grégoire, "Road inspection using full-wave inversion of far-field ground-penetrating radar data," in *Proc. 7th Int. Workshop Adv. Ground Penetrating Radar*, Jul. 2013, pp. 1–6.

[15] Y.-S. Su, S. R. Sukumar, A. F. Koschan, D. L. Page, and M. A. Abidi, "Dual-light inspection method for automatic pavement surveys," *J. Comput. Civil Eng.*, vol. 27, no. 5, pp. 534–543, Jul. 2012.

[16] S.-J. Yu, S. R. Sukumar, A. F. Koschan, D. L. Page, and M. A. Abidi, "3D reconstruction of road surfaces using an integrated multi-sensory approach," *Opt. Lasers Eng.*, vol. 45, no. 7, pp. 808–818, Jul. 2007.

[17] J. Budras. (Aug. 2011). *A Synopsis on the Current Equipment Used for Measuring Pavement Smoothness*. [Online]. Available: http://www.fhwa.dot.gov/pavement/smoothness/rough.cfm

[18] L. M. Pierce, G. McGovern, and K. A. Zimmerman, "Practical guide for quality management of pavement condition data collection," U.S. Dept. Transp., Federal Highway Admin., Tech. Rep., Feb. 2013.

[19] *Pavement Management Information System—Rater's Manual*, Texas Dept. Transp., Austin, TX, USA, 2016.

[20] *Texas DOT Pavement Management Information System (PMIS) Rater's Manual*, Texas State Dept. Transp., Austin, TX, USA, 2018.

[21] A. Benedetto, F. Benedetto, F. de Blasiis, and M. R. Giunta, "Reliability of Radar Inspection for Detection of Pavement Damage," *Road Mater. Pavement Des.*, vol. 5, no. 1, pp. 93–110, 2004.

[22] F. Benedetto and F. A. M. Tosti Alani, "An entropy-based analysis of GPR data for the assessment of railway ballast conditions," *IEEE Trans. Geosci. Remote Sens.*, vol. 55, no. 7, pp. 3900–3908, Jul. 2017.

[23] F. Benedetto and A. A. Benedetto Tedeschi, "A mobile Android application for road and pavement inspection by GPR data processing," in *Proc. 15th Int. Conf. Ground Penetrating Radar*, Jun./Jul. 2014, pp. 842–846.

[24] F. Benedetto and A. A. Benedetto Tedeschi, "GPR image and signal processing for pavement and road monitoring on Android smartphones and tablets," in *Proc. EGU Gen. Assem. Conf. Abstr.*, May 2014.

[25] A. Tedeschi and F. Benedetto, "A real-time automatic pavement crack and pothole recognition system for mobile Android-based devices," *Adv. Eng. Inform.*, vol. 32, pp. 11–25, Apr. 2017.

[26] S. Pu, M. Rutzinger, G. Vosselman, and S. O. Elberink, "Recognizing basic structures from mobile laser scanning data for road inventory studies," *ISPRS J. Photogramm. Remote Sens.*, vol. 66, no. 6, pp. S28–S39, Dec. 2011.

[27] (2015). *Fill That Hol*. Accessed: Dec. 2015. [Online]. Available: https://www.fillthathole.org.uk/iphone

[28] S. S. Shamsabadi and M. R. Wang Birken, "PAVEMON: A GIS-based data management system for pavement monitoring based on large amounts of near-surface geophysical sensor data," in *Proc. 27th Annu. Symp. Appl. Geophys. Eng. Environ. Problems (SAGEEP)*, Mar. 2014.

[29] P. Mohan and V. N. Padmanabhan, "Nericell: Rich monitoring of road and traffic conditions using mobile smartphones," in *Proc. 6th ACM Conf. Embedded Netw. Sensor Syst.*, Nov. 2008, pp. 323–336.

[30] P. Mohan, V. N. Padmanabhan, R. Ramjee, and V. Padmanabhan, "Trafficsense: Rich monitoring of road and traffic conditions using mobile smartphones," Microsoft Res., Washington, DC, USA, Tech. Rep. MSR-TR-2008-59. Apr. 2008.

[31] R. Bhoraskar, N. Vankadhara, B. Raman, and P. Kulkarni, "Wolverine: Traffic and road condition estimation using smartphone sensors," in *Proc. 4th Int. Conf. Commun. Syst. Netw. (COMSNETS)*, Jan. 2012, pp. 1–6.

[32] M. W. Sayers, "The little book of profiling: Basic information about measuring and interpreting road profiles," Univ. Michigan, Ann Arbor, MI, USA, Tech. Rep., 1998.

[33] L. Forslöf and H. Jones, "Roadroid: Continuous road condition monitoring with smart phones," *J. Civil Eng. Archit.*, vol. 9, no. 4, pp. 485–496, Apr. 2015.

[34] X. Li and D. W. Goldberg, "Toward a mobile crowdsensing system for road surface assessment," *Comput., Environ. Urban Syst.*, vol. 69, pp. 51–62, May 2018.

[35] J. Lepine, "An optimised machine learning algorithm for detecting shocks in road vehicle vibration," Ph.D. thesis, Victoria Univ., Footscray VIC, Australia, 2016.

[36] J. Lepine, V. Rouillard, and M. Sek, "On the use of machine learning to detect shocks in road vehicle vibration signals," *Packaging Technol. Sci.*, vol. 30, no. 8, pp. 387–398, Aug. 2017.

[37] A. Allouch, A. Koubâa, T. Abbes, and A. Ammar, "Roadsense: Smartphone application to estimate road conditions using accelerometer and gyroscope," *IEEE Sensors J.*, vol. 17, no. 13, pp. 4231–4238, Jun. 2017.

[38] N. Silva, J. Soares, V. Shah, M. Y. Santos, and H. Rodrigues, "Anomaly detection in roads with a data mining approach," *Procedia Comput. Sci.*, vol. 121, pp. 415–422, Jan. 2017.

[39] K. Chen, M. Lu, X. Fan, M. Wei, and J. Wu, "Road condition monitoring using on-board three-axis accelerometer and GPS sensor," in *Proc. 6th Int. ICST Conf. Commun. Netw. China (CHINACOM)*, Aug. 2011, pp. 1032–1037.

[40] V. Douangphachanh and H. Oneyama, "A model for the estimation of road roughness condition from sensor data collected by Android smartphones," *J. Jpn. Soc. Civil Eng.*, vol. 70, no. 5, pp. 103–111, 2014.

[41] F. Seraj, B. J. van der Zwaag, A. Dilo, T. Luarasi, and P. Havinga, "RoADS: A road pavement monitoring system for anomaly detection using smart phones," in *Big Data Analytics in the Social and Ubiquitous Context*. Springer, 2015, pp. 128–146.

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

[43] A. Mohan and S. Poobal, "Crack detection using image processing: A critical review and analysis," *Alexandria Eng. J.*, vol. 57, no. 2, pp. 787–798, Jun. 2018.

[44] J. Masino, J. Thumm, M. Frey, and F. Gauterin, "Learning from the crowd: Road infrastructure monitoring system," *J. Traffic Transp. Eng.*, vol. 4, no. 5, pp. 451–463, Oct. 2017.

[45] J. Dixon-Warren, "Comparing the invensense and bosch accelerometers found in the iPhone 6," Chipworks, Sep. 2014. Accessed: Dec. 5, 2019. [Online]. Available: https://www.macrumors.com/2014/09/26/iphone-6-6-plus-two-accelerometers/

[46] V. Douangphachanh and H. Oneyama, "Formulation of a simple model to estimate road surface roughness condition from Android smartphone sensors," in *Proc. IEEE 9th Int. Conf. Intell. Sensors, Sensor Netw. Inf. Process. (ISSNIP)*, Apr. 2014, pp. 1–6.

[47] *Features for Expanded Ad Units*, Apple, Cupertino, CA, USA, Jun. 2015.

[48] V. Astarita *et al.*, "A mobile application for road surface quality control: UNIquALroad," *Procedia Soc. Behav. Sci.*, vol. 54, pp. 1135–1144, Oct. 2012.

[49] *Pavement Manual*, Texas Dept. Transp., Austin, TX, USA, 2018.

[50] A. Basavaraju, "Machine learning approaches to road surface anomaly assessment using smartphone sensors," M.S. thesis, Texas A&M Univ., College Station, TX, USA, 2018.

[51] E. S. Gadelmawla, M. M. Koura, T. M. A. Maksoud, I. M. Elewa, and H. H. Soliman, "Roughness parameters," *J. Mater. Process. Technol.*, vol. 123, no. 1, pp. 133–145, Apr. 2002.

[52] L. Sun, "Simulation of pavement roughness and IRI based on power spectral density," *Math. Comput. Simul.*, vol. 61, no. 2, pp. 77–88, Jan. 2003.

[53] D. M. Xu, A. M. O. Mohamed, R. N. Yong, and F. Caporuscio, "Development of a criterion for road surface roughness based on power spectral density function," *J. Terramech.*, vol. 29, nos. 4–5, pp. 477–486, Jul./Sep. 1992.

[54] 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. thesis, Univ. Bath, Bath, U.K., 2012.

[55] W. Staszewski and J. Giacomin, "Application of the wavelet based FRFs to the analysis of nonstationary vehicle data," in *Proc. Int. Soc. Opt. Eng.*, Feb. 1997. pp. 425–431.

[56] L. Wei, T. Z. Fwa, and Z. Zhe, "Wavelet analysis and interpretation of road roughness," *J. Transp. Eng.*, vol. 131, no. 2, pp. 120–130, Feb. 2005.

[57] *Statistics and Machine Learning Toolbox*, MathWorks, Natick, MA, USA, 2018.

[58] V. Cherkassky and F. M. Mülier, *Learning from Data: Concepts, Theory, and Methods*. Hoboken, NJ, USA: Wiley, 2007.

[59] S. Shalev-Shwartz and S. Ben-David, *Understanding Machine Learning: From Theory to Algorithms*. Cambridge, U.K.: Cambridge Univ. Press. 2014.

[60] *Multiclass Model for Support Vector Machines or Other Classifiers*, MathWorks, Natick, MA, USA, 2018.

[61] A. Gordon, L. Breiman, J. Friedman, R. Olshen, and C. Stone, "Classification and regression trees," *Biometrics*, vol. 40, no. 3, p. 874, Sep. 1984.

[62] F. Rosenblatt, "Principles of neurodynamics. Perceptrons and the theory of brain mechanisms," Cornell Aeronautical Lab Inc, Buffalo NY, USA, Tech. Rep., 1961.

[63] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," Inst for Cogn. Sci., Univ. California San Diego, Jolla, CA, USA, ICS Rep. 8506, 1985. ICS Report 8506

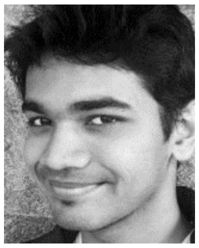
[64] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, Oct. 2011.

[65] *sklearn.Neural_Network.MLPClassifier—Scikit-Learn 0.19.1 Documentation*. [Online]. Available: http://scikit-learn.org

[66] G. Andrew and J. Gao, "Scalable training of L^1-regularized log-linear models," in *Proc. 24th Int. Conf. Mach. Learn.*, Jun. 2007, pp. 33–40.

[67] R. Malouf, "A comparison of algorithms for maximum entropy parameter estimation," in *Proc. 6th Conf. Natural Lang. Learn.*, Aug. 2002, pp. 1–7.

[68] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," Dec. 2014, *arXiv:1412.6980*. [Online]. Available: https://arxiv.org/abs/1412.6980



Akanksh Basavaraju received the B.Tech. degree in electronics and communication engineering from Amrita University, India, in 2015, and the M.S. degree in electrical engineering from Texas A&M University, College Station, in 2018. His research interests include biomedical signal processing, machine learning, and software development.

Jing Du received the M.B.A. and Civil Engineering degrees from Tianjin University, China, and the Ph.D. degree in construction management from Michigan State University, in 2012. He is currently an Associate Professor with the Department of Civil and Coastal Engineering, University of Florida. His research focuses on smart information technologies for construction projects. He has received about $2.3 million in external research funding from agencies, including the National Science Foundation (NSF), the National Institute of Standards and Technology (NIST), the U.S. Department of Transportation (US DOT), Fiatech, and the Zachry Group. He is an Assistant Specialty Editor of *ASCE Journal of Construction Engineering and Management* (JCEM).

Fujie Zhou received the Ph.D. degree in pavement engineering from Tongji University, China, in 1998. He was an Assistant Professor with Tongji University. He is currently a Research Engineer with the Texas A&M Transportation Institute (TTI), Texas A&M University System. He has published more than 50 journal articles and 25 conference papers. His research interests include connected and automated vehicles (CAV) safety and pavements, materials testing, performance (distress) models, and pavement designs. He has received more than $10 million in research funding from agencies, including the National Science Foundation (NSF), the National Cooperative Highway Research Program (NCHRP), the U.S. Department of Transportation (US DOT), the Texas Department of Transportation, the Ohio Department of Transportation, and GAF. He has received twice the Annual Best Paper Awards from the Association of Asphalt Paving Technologists in 2010 and 2014.

Jim Ji received the B.S. and M.S. degrees in electrical engineering from Tsinghua University, China, and the Ph.D. degree in electrical engineering from the University of Illinois at Urbana–Champaign (UIUC). He is currently an Associate Professor with the Department of Electrical and Computer Engineering, Texas A&M University, College Station. His research interests include biomedical imaging and image processing, machine learning for big data image processing, MRI and their applications in cancer, and neural applications. His recent work is focused on novel MRI methods for cancer interventions, MRI with large arrays and compressive sensing, and big data in MRI. He is the Program Co-Chair of the Annual International Conference of IEEE Engineering in Medicine and Biology Society in 2014. He received the Zhongwang Outstanding Graduate Student Prize from Tsinghua University, in 1997, the Sundaram Seshu Fellowship from UIUC in 2001, and the Faculty Early Career Development (CAREER) Award from the National Science Foundation (NSF) in 2008. His research support includes QNRF, NSF, and NIH. He also served as an Associate Editor for the IEEE TRANSACTIONS ON BIOMEDICAL ENGINEERING, IEEE JOURNAL OF BIOMEDICAL HEALTH INFORMATICS, *Quantitative Imaging in Medicine and Surgery*, and serves as an Editor for IEEE-EMBC Biomedical Imaging and Image Processing Theme.