ASHESI UNIVERSITY

BUILDING A POTHOLE DETECTION AND TRACKING SYSTEM

CAPSTONE PROJECT

B.Sc. Computer Engineering

Kalysa Abena Owusua Owusu

2019

ASHESI UNIVERSITY

BUILDING A POTHOLE DETECTION AND TRACKING SYSTEM

CAPSTONE PROJECT

Capstone Project submitted to the Department of Engineering, Ashesi University in partial fulfillment of the requirements for the award of Bachelor of Science degree in Computer Engineering.

Kalysa Abena Owusua Owusu

2019

Declaration

I hereby declare that this capstone is the result of my own original work and that no part of it has been presented for another degree in this university or elsewhere.

Candidate's Signature:

……

Candidate's Name:

……

Date:

……

I hereby declare that preparation and presentation of this capstone were supervised in accordance with the guidelines on supervision of capstone laid down by Ashesi University.

Supervisor's Signature:

……

Supervisor's Name:

……

Date:

……

Acknowledgement

And now, with the help of God, I shall become myself.

SØREN KIERKEGAARD

Abstract

Building and maintaining infrastructure is often a key challenge in developing countries, and Ghana is no exception. Increasing population and car ownership rates coupled with poor maintenance cultures result in a corresponding increase in the rate of damage of roads, causing deformities such as cracks and potholes. These road deformities not only negatively impact a country's road infrastructure and the cars which ply said roads, but also pose a threat to road users. In Ghana, only two mobile maintenance units are charged with monitoring the roads in all ten regions of the country. Thus, this project presents Pothole Tracker Ghana, a two-tiered application inspired by the idea of crowdsourcing. Consisting of a vision-based pothole classification system implemented on a Raspberry Pi and a map-based web application, this project aims to reduce the barriers to data collection on poor road infrastructure on the part of governments whilst allowing everyday road users to make informed decisions concerning their journeys. Three different algorithms are considered and compared for the classification task; logistic regression, support vector machines (SVM) and a hybrid algorithm incorporating a convolutional neural network (CNN) and SVM. The tuned SVM is chosen for the final system implementation.

Table of Contents

Chapter 1: Introduction

According to the Ghanaian Minister of Roads and Highways, 61% of the roads in Ghana are classified as poor - out of the approximately 72,000 kilometers of roads in the country, only 23% have been asphalted [1]. In addition, road accidents kill 6 people everyday in Ghana [2]. Although a large percentage of these accidents are caused by reckless driving, a significant number are also caused by poor road conditions, with a majority of such accidents taking place on major highways such as Accra-Tema motorway. As recently as November 11, 2018, a mother and her two-year old son were involved in a car accident, and calls have been made for motorway users to boycott toll payments [3].

Other lesser roads constitute the above mentioned 72,000 km, and the poor quality of the existing road infrastructure hinders the development of areas in which those roads are found. Considering the fact that certain road projects are abandoned and forgotten in the wake of governmental changes [4], as well as the fact that there is often a scramble to mobilize limited funds for road projects and a plethora of roads in deplorable condition to fix, this study assumes the need for classification of roads according to their conditions, such that decisions can be made concerning which roads are in urgent need of repair, and which can do without them. It also aims to facilitate the inclusion of Ghanaians to perform their civic duty by assisting in the development and improvement of infrastructure.

In this project, I present the Ghana Pothole Tracker, a system which uses a web application as a user interface for displaying road conditions in the city to drivers before they ply a route. This project proposes the use of classification based on pothole presence in order to improve record keeping concerning the state of roads in Ghana. To achieve this, a Raspberry Pi-based detector consisting of a camera and GPS module is installed in a car.

A classification algorithm - which serves as the decision-making engine of the system - is run on frames grabbed from a real-time video stream of the road, and upon detecting a pothole, the coordinates are transmitted to a database, from which data rich maps are generated. Consequently, a user is able to keep track of past routes and those taken by other users, whilst relevant authorities can access and save crowdsourced driving data, introducing an element of crowdsourcing for development.

Image processing is a subset of computer vision, which is a field of computer science concerned with "the automatic extraction, analysis and understanding of useful information from a single image or a sequence of images." [5] This is primarily done through the use of classification algorithms which are trained on an image dataset, and produce an output class given an unseen input image. In this project, the detection system makes use of a binary image classifier, which produces a class label given an input image. This approach can either be a supervised or unsupervised learning task, where supervised approaches are trained on labelled data, whilst unsupervised approaches train on data without labels. This project primarily considers supervised approaches, although a hybrid model is also considered in order to compare the efficacy of either approach.

Classification is chosen as the basis of this problem because of the binary nature of pothole presence, and the classification problem can be solved with various methods, ranging from the supervised; linear regression, logistic regression, and support vector machines (SVM) to unsupervised methods like neural networks. In the next chapter, relevant studies and technologies used to solve the problem of pothole detection will be considered, and an appropriate method shall be determined for this project in order to reduce the gaps identified and make improvements.

Chapter 2: Literature Review

2.1 Related Works

The problem of pothole detection and road classification is one that has been tackled by various parties all over the world through a variety of means, including vibration-based methods, mathematical derivations and vision-based methods. This study, however, to the best of my knowledge at this time, is the only one to implement both road condition classification and pothole detection using object classification.

In related literature concerning vibration-based methods, data collection is handled primarily using accelerometer readings of mobile devices installed in moving cars. For instance, Kulkarni et al [6] implement a Pothole Detection System using data obtained from accelerometer sensor readings of an Android mobile device, combined with GPS for plotting the location of the potholes on Google Maps. After data collection, a high pass filter is used to remove noise elements, and an Artificial Neural Network is used to identify potholes from the accelerometer data.

Much like Kulkarni et al, Vorgbe [7] acquired accelerometer data using an Android mobile development environment. In addition, the classification of roads by Vorgbe was based on a multi-class problem, making use of 3 classes (good, fair, and bad). Data was collected over 50km of road, with each example representing the data collected over 10 second intervals. In this study, two logistic regression algorithms were implemented to solve the classification problem. The binary model achieved a 92% true positive rate when distinguishing between good and bad roads, an 83% rate when distinguishing between

good and fair roads, but was unable to reliably distinguish between fair and bad roads. The multi-class algorithm fared worse, as it was only able to reliably distinguish good roads from fair and bad roads with a true positive rate of 87%.

Another take on the solution is that presented by Bhatt et al [8] in the form of the Intelligent Pothole Detection and Road Condition Assessment system, which uses an iOS mobile environment to detect potholes and assess road conditions in realtime. As a user driver, a support vector machine model uses phone sensor data to determine whether the road is good or bad, and whether it contains potholes. This leverages the experience of individuals to inform civic authorities about roads that need repair, and makes use of data-rich maps to illustrate road conditions across the city in an aesthetically appealing manner.

Punjabi et al, [9] utilizes a vision-based approach to implement an “Intelligent Pothole Detection System”, which detects a pothole at a distance, and alerts the driver to reduce their speed or change their route. According to Punjabi et al, the use of image processing results in efficiency and accuracy levels that are higher than conventional methods of pothole detection. The data is collected in the form of video, from which frames are extracted. Edge detection is used in order to aid in pothole analysis, after which which the presence/absence of potholes are validated and the established threshold is tested against for decision-making. The idea of instructing a user to slow down if there is a pothole on the road is the distinguishing feature of this study. However, a major limitation lies in the fact that high speeds cause a blurring in the video/images captured by the camera, which result in the image not being suitable for comparison.

Nienaber et. al [10] make use of a vision-based approach, making use of a Go Pro camera to create a representative dataset of pothole images under various driving conditions and deriving a model of said potholes using the collected images. An algorithmic approach is then used, consisting of image processing techniques such as contour detection and a Canny filter. Using this approach, a precision of 81.8% and a recall of 74.7% are achieved. In this approach, the authors purposely use a visual approach that does not require any machine learning algorithms, hence the model is not trained. Steps include the extraction of the road as the sole focus of the frame, due to the fact that extraneous features such as foliage do not contribute to the presence of potholes, and implementing contour/edge detection using the Canny edge detector, after which dilation is undergone in order to increase the area of the white pixels which result from the Canny edge detector. This idea of pre-processing using a portion of the image, as a means of improving accuracy is one to note, as well as the high values.

2.2 Relevance to Project

The studies presented above prove that whilst the methods used to collect, model and solve the problem of pothole detection/road quality classification vary, image-based methods may serve as effective solutions to said problem - and in conjunction with machine learning, an effective classifier can be built. Thus, the solution presented in this project serves to use a vision-based approach to compare the results of both supervised and unsupervised learning. The models chosen include those informed by previous studies, including Logistic Regression, Support Vector Machines and Neural Networks.

This project incorporates the idea of crowdsourcing road information through the use of GPS information, and goes a step further to create an intuitive user application to

display said data in an aesthetically and user-friendly manner. The gaps in existing works that this project intends to address include the blurring faced by Punjabi et al, and also introducing a routing element as compared to single pothole markers presented by Kulkarni et al. This is done in order to increase the usability of the application, and serves as an incentive for users to patronize the system, but unlike Bhatt et. al, this project shall consider the use of vision in order to determine how comparable the results will be, whilst taking away the necessity to monitor a user's phone. Additionally, this project intends to produce a definitive result as to whether an unsupervised or supervised method of learning would be better for this particular classification problem.

Chapter 3 : Design

This chapter first presents the objective of the design proposed, the scope of the project, and a description of the functional and non-functional requirements of the system. Additionally, it presents the design choices made considering said requirements, and in the case of the hardware component, uses a Pugh matrix to decide between various options. Finally, a high-level architecture of the system is presented.

3.1 Project Design Objective

This project intends to present a responsive system which detects and records potholes, as well as an intuitive user-facing interface for interacting with the collected data. This proposed system presents a modern solution to the inability of government institutions to efficiently track and monitor road conditions around the country. Currently, Mobile Maintenance Units (MMU's) exist in the northern and southern sectors of Ghana to carry out "routine and periodic" maintenance works on the roads in said sectors [11]. However, with the introduction of the system proposed by this project, the work of the Mobile Maintenance Units will be made easier, as the burden of scouting 10 regions will be carried by multiple system users.

Using hardware and software based approaches, this project intends to produce a system which uses a vision-based machine learning approach and incorporates GPS data to ensure tracking. The system requires a means of storing said GPS data in order to allow user interaction through a front-facing interface. The following section discusses the scope of the project, after which the functional and non-functional requirements of the system are determined. Finally, it considers decisions made concerning the system hardware and software, with respects to the functional and non-functional requirements.

3.2 Project Scope

The system implemented by this project is to consist of three parts; a user-facing interface, a server-side and a hardware-based system. The user interface is to display pothole and route information with the option of exporting said data for further analysis, whereas the server-side is responsible for keeping track of users, as well as the latitude and longitudes of the potholes they have encountered on their journeys.The hardware aspect of the system is to be installed in the car, and is responsible for monitoring the road and detecting potholes using the decision-making classification algorithm.

3.3 Functional Requirements

1. Data recording, classification and transfer
    - The user turns on the system in the car, facing the road
    - The system monitors a video stream of the road, grabbing a frame periodically and running the classification algorithm
2. Data storage
    - The transferred coordinates are received and inserted into a central database
3. Data visualisation
    - The user signs in to the application
    - The user enters an origin and destination
    - A route visualization is returned with the poor parts of the route highlighted in red, with pliable sections in green

3.4 Non-Functional Requirements

1. Data Integrity
    - Exported location data must be accurate and in an appropriate format
2. Responsiveness
    - The map must facilitate user interaction; a user should be able to pan and zoom
3. Reliability
    - Minimal to no delay when classifying potholes

3.5 Design Decisions

This section presents the design decisions made concerning the three major system components, as well as the high-level architecture of the proposed system, detailing the interactions between said components. The components in question are the user interface, the server side, and the hardware-based monitoring and classification system, and the relationship between them is detailed in a high-level architecture diagram.

3.5.1 User Interface

A major requirement of the system is usability, which raises the need for a user-facing application with an intuitive user interface (UI). The decision was made to build the UI with web technologies including HTML, CSS, and JavaScript. The choice of a web interface was made due to the fact that web development produces a more generalized experience - data can be accessed on a PC or mobile browser, as compared to a specialist mobile application. Further, it was determined that satisfying the mapping requirement would be simple due to the ability to access the Google Maps Javascript API using a generated API key.

3.5.2 Server Side & Database

Another requirement of the system was the need for storage of the detected GPS data in a robust, dynamic and persistent manner. MySQL was chosen to build the database, due to its compatibility with the chosen web technologies, as well as the data integrity ensured in its processes, which would ensure that the received GPS data would be stored and exported in a structured manner. Additionally, due to its widespread application, it has significant documentation. The scripting language chosen to manage the interaction between the frontend and the database was PHP.

The database consists of three tables; *location_data*, *users*, and *sessions,* and the relationship between them is presented in the figure below. The table of interest in this project is that of the location_data, which contains the longitude-latitude information from which the Google Maps API renders the markers.

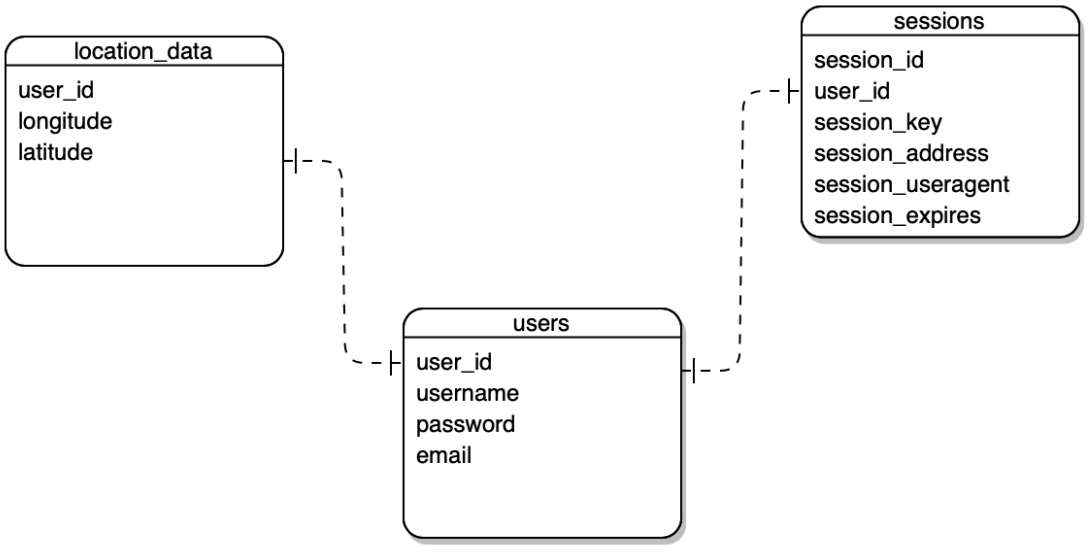

Figure 3.5.1 - Entity Relationship Diagram highlighting relationship between tables

3.5.3 System Hardware

Requirements for the hardware include a high quality camera module, portability, the ability to consistently transmit data, and the processing capacity to function over long distances. Considering the variety of hardware devices in existence which meet these criteria, a Pugh Matrix was used to compare and rank the top five options, which include an Arduino, Raspberry Pi, Go Pro camera, smartphone, and laptop. These options (apart from the Arduino and Pi) were chosen due to their use in previous studies. Baseline was chosen as a smartphone, due to the large-scale use of mobile environments (iOS and Android) in the field of pothole detection.

Table 3.1: Pugh Matrix

	Baseline	**Laptop**	**Arduino**	**Raspberry Pi**	**GoPro Camera**
Criteria					
Camera Module	0	-1	-1	0	+1
Portability	0	-1	0	0	0
Transmission	0	0	-1	0	-1
Processing Power	0	+1	-1	+1	-1
Overall Suitability	0	0	0	+1	-1
Net Score	0	-1	-3	2	-2
Rank	**2**	**3**	**5**	**1**	**4**

The Pugh Matrix above points to the Raspberry Pi as the best alternative in relation to the others. Additional advantages of the Pi include the large support for various hardware modules, including the GPS shield and Pi Camera, as well as prominent image processing and machine learning libraries such as openCV and scikit-learn (discussed further in Chapter 4). Moreover, the Pi is a low cost mini-computer, which means that computing power is not sacrificed in the stead of portability. A Raspberry Pi checks all the

necessary requirements for the system to be developed, as it is portable, supports a camera module for continuous monitoring of the road, and has an inbuilt Wifi and bluetooth module for communication with the server side. Support for a GPS module, as well as significant processing power means that coordinates can be transmitted to the MySQL database, which ties into the requirements of data integrity, persistence and accuracy.

3.6 System Architecture

This section presents the high-level architecture of the proposed system, which is a diagram describing the interactions between the three system components; the hardware-based Raspberry system, and the software-based MySQL Database and User Interface.

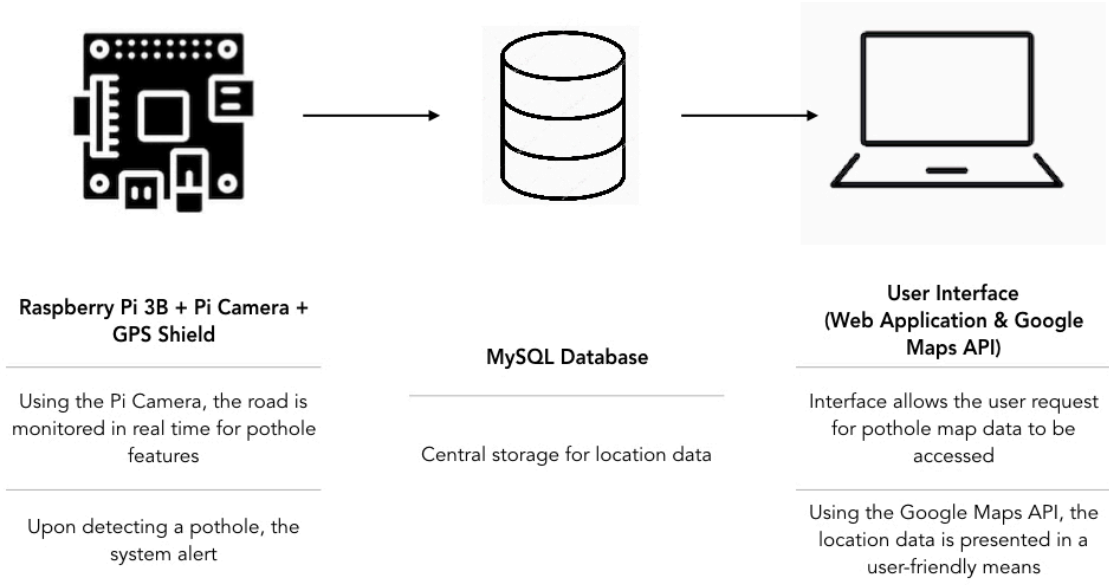

Figure 3.5.2 - High-Level Architecture

Chapter 4: Methodology

In this chapter, the methodology employed in developing the solution, as well as the approaches followed in data collection and system implementation are presented - in conjunction with a visual algorithm. Additionally, the technical specifications for the hardware and software used are detailed in the experimental setup. Further, the following sections highlight the different classification methods tested in this project, as well as the web application interface built for data visualization.

4.1 Classification System Implementation

The diagram below describes the flow of data within the pothole classification system, and is followed by a breakdown of the steps taken to implement the system, as well as the considerations made with regards to the system hardware, software and dataset.

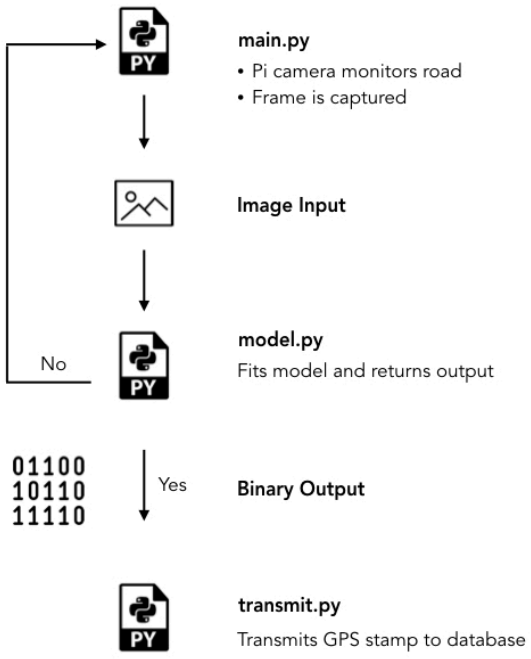

Fig 4.1: Visual flow diagram of the final system

4.1.1 Hardware

The Pugh matrix developed in Chapter 3 informed the decision to use the Raspberry Pi 3 as the hardware on which the system would be implemented, due to compatibility with a diverse array of modules and libraries.

For the purposes of this setup, the Pi Camera was used as the means through which input test data would be processed by the decision-making engine on the presence of a pothole. The Ublox Neo-6M module was the means of collecting latitude and longitude information as the car drives. Other peripherals included a keyboard, mouse, monitor, and 8GB SD card, which were obtained to allow the GUI of the Raspberry Pi to be used.

The following steps detail the Pi hardware setup of the system hardware.

1. Connecting the Pi camera to the built-in mount on the Pi 3.
2. Soldering header pins to the Neo-6M module; this was done to enable interaction through the GPIO pins on the Raspberry Pi.
3. Placing the SD card into the built-in slot, and connecting the power supply and peripherals; ensured the installation of the OS and the use of the GUI.

4.1.2 Software

The OS installed on the Pi was Raspbian Stretch. In order to execute the Python scripts the system would implement, external libraries were installed on the Pi via command line. The following table presents an overview of these libraries and their functionalities.

Table 4.1: Installed modules and their functionalities

Module Name	**Functionality**
NumPy	Allows images to be described as arrays
scikit-learn	Machine learning models, metrics and tuning support
Keras	Deep learning library
openCV	Real-time image processing
pynmea2	NMEA sentence parsing
picamera	Pi camera access
MySQLdb	Remote MySQL database access

4.1.3 Dataset

4.1.3.1 Collection & Labeling

The dataset used for this project was obtained from three main sources - extracted frames from video recorded by an iPhone 6 Plus, an iPhone 8 Plus, and Nienaber et al's GoPro recorded pothole dataset [10]. The iPhone data considered a journey in the Ashaley Botwe and Adenta suburbs of Accra, whereas Nienaber et al. considered roads in South African suburbs, and the Samsung data consisted of a variety of journeys around Ghana.

The idea during data collection was to introduce images with significant variation in the lighting and road conditions. This choice was made in order to account for intra-class variation, a phenomena describing the sometimes vast differences in the members of a particular class - in this case considering the variations in potholes. Additionally, viewpoint variation, illumination variation and background clutter were considered, as training the classifier with a variety of data would lead to a more robust recognizer which has been trained on enough varied data to generalize well in new environments [12].

The iPhone 6s image data was collected in a Suzuki Grand Vitara, whereas the iPhone 8 Plus data was collected in a Toyota Corolla, introducing different angles of inclination towards the road, as well as the inherently different fields of vision due to the different cameras used.

At the end of the collection stage, frame extraction took place using ffmpeg - a suite of libraries for handling multimedia files - with extracted frames from potholed stretches being named “pothole.%d.jpeg”, and placed in a folder called “pothole” whilst non-potholed frames were named “not_a_pothole.%d.png” and placed in a folder called “not_a_pothole”. This naming convention was chosen due to its explicit assignment of labels, and the fact that the format would be easy to deal with in further processing.

Cumulatively, the training dataset stood at 2,578 images, with 1,326 pothole images and 1,252 no pothole images.

4.1.3.2 Preprocessing & Feature Extraction

Images consist of a set pixels, which detail the color/intensity of the light in a particular part of the image on a gradient range [0, 255] [12]. This definition lends to the conceptual view of an image as a matrix of height and weight populated with pixel values, which represent either the grayscale channel or the color channel (usually the RGB color space). In this project, the color channel is the primary concern, as the assumption is made that certain features can be extracted as a result of image coloring, which will aid the model in distinguishing between potholes and non-potholes.

In the preprocessing and feature extraction stage, we consider the transformation of the input image into a NumPy matrix (of pixel intensities), into a form that a model is able

to understand. The assumption of the RGB color space means that each pixel will be represented by a list of three values, representing the intensity of red, blue and green contained in said pixel. These values also lie within the range [0, 255].

The feature extraction methodology chosen in this project is the method of Histogram of Oriented Gradients (HOG), as implemented by the OpenCV function cv2.calcHist. This function calculates an image histogram given the image, the color channels, an optional mask (for focusing on specific regions), the histogram bin size (color intensities), and the ranges of possible pixel values (usually [0, 255]). A histogram represents the distribution of pixel intensities in an image [13]. This particular method was chosen due to its wide scale use (and success) in object detection tasks in computer vision and image processing - the fact that it counts occurrences of gradient orientation in portions of an image [14] caused the assumption to be towards the applicability in the problem of detecting edges caused by potholes.

The vector returned after calling this histogram function is then flattened - the original size matches the number of bins specified in the function, whereas we require a (one-dimensional) vector to represent a single data item. This vector then serves as the main input to the chosen classifiers detailed in the following sections.

4.1.3.3 Metrics Considered

Metrics considered to evaluate the system developed in this project include precision, recall and F1 measure, metrics which are more effective than classification accuracy. This is because accuracy solely considers the number of correct predictions, as compared to precision, which considers false positives (proportion of incorrect positive

predictions), and recall, which considers false negatives (proportion of incorrect negative predictions) [15]. The F1 measure is simply an average of the two. These metrics are considered because they not only consider correctness, but holistically consider the correctness of the model's positive and negative predictions.

A well performing model is one that is able to best optimize these metrics, and a means through which this can be done is through tuning hyper-parameters - parameters that are not directly learnt by the model. Although the logistic regression implementation has no hyper-parameters, the Linear SVM model as implemented in scikit-learn allows for the tuning of hyper-parameters. This is discussed in detail in subsequent sections.

4.2 Logistic Regression

Logistic Regression is a supervised learning algorithm which derives a hypothesis function in order to determine the probability that a given input example, x_i belongs to a particular positive output class [16]. The value of this probability usually falls between 0 and 1, at which point a threshold is set to determine the value above which an input can be classified as positive (1). Usually, this threshold is 0.5, however for more specialized applications in which a higher degree of certainty is required - usually for security purposes (eg. Facial recognition), the threshold may be higher.

In this project, logistic regression is used due to its binary nature - it is able to discriminate between two classes; pothole and not a pothole. The equation governing logistic regression is $h(\theta) = g(\theta^T x)$, which considers input feature vector, x, the transpose of the weight vector θ^T - whose weights we seek to optimize, and a sigmoid

function $g(z)$, which is responsible for restricting the function output between probability bounds 0 and 1. The sigmoid function is defined as $g(z) = \frac{1}{1+e^{-z}}$.

In finding the best values for the input weights, an optimizing algorithm - usually gradient descent - is used to minimize the cost function, $J(\theta)$, which describes the difference between the predicted class of an input and the actual label. In logistic regression, this cost function is defined as

$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} [y^{(i)} log(h_\theta(x^{(i)})) + (1 - y^{(i)}) log(1 - h_\theta(x^{(i)}))] \quad [17]$$

The features extracted early on by the HOG are fed in as inputs to the system, along with their equivalent ground truth labels. A 75-25 train-test split is used in order to determine the effectiveness of the model on test data.

4.3 Support Vector Machines

Support Vector Machine (SVM) is also a supervised machine learning algorithm [17] which considers a training dataset x_i, and a corresponding set of labels, y_i, where $i = 1,...,N$ and $y_i = 1,...,K$ where N and K represent the number of training data points and classes respectively. SVMs work by building a hyperplane (separating line) which best divides the classes in a given dataset. The scikit-learn library implementation of the Linear SVM is based on the linear hypothesis function;

$$\bar{y}_i(x) = \mathbf{W}^T x_i + b \qquad [17]$$

The function $\bar{y}_i(x)$ takes an input example and outputs a prediction, $\bar{y}_i$, which can be compared against the ground truth label y_i in order to determine the mean squared error of the SVM. The term b represents the bias vector, of size $[K \times 1]$, which allows shifts and translations in the hypothesis function without impacting the adjustable weight matrix $\mathbf{W}$, which parameterizes the hypothesis function [12][17]. The best function parameters (weights and bias terms) are found when we are able to minimize the mean squared error function $J(x) = \sqrt{y_i^2 - \bar{y}_i^2}$, and ensure a high classification accuracy. The process of optimizing these terms involves finding the global minimum of the function $J(x)$.

In this method, after features are extracted from the training images, they are stored in a data array with corresponding labels (the built-in label encoder uses convention to assign 0 to non-potholes and 1 to potholes). Scikit-learn's train-test split functionality is then employed in order to test the model on 25% of the training data (which is held out) after training on the initial 75%. Since the ground truth labels of the 25% are known, it

then becomes simple to compare the predictions of the trained model with the ground truth to find out the accuracy.

4.3.1 Hyper-parameter tuning

In Support Vector Machines, the hyper-parameter term allows for the model to be tuned in order to avoid under-fitting - a phenomenon which occurs when the model is unable to fit the training data, resulting in poor results when tested on new data. Conversely, overfitting occurs when the classifier models both the training data and the underlying noise so effectively that it is unable to properly generalize and classify new, unseen data.

Generally, the ideal hyper-parameter values for a dataset is found via trial and error, however, scikit-learn provides two optimization methods for tuning; exhaustive grid search and randomized search. The grid search algorithm is a brute force optimizer which tests the model with all possible hyper-parameter combinations and returns the one with the highest cross-validation accuracy. The randomized search is not as thorough, and randomly samples parameters based on the distribution of possible parameter values - resulting in less time taken in the search.

In this project, the exhaustive grid search is chosen, as maximizing classification accuracy is more important than reducing the time/computation used to find the best c. The set of c values considered [0.01, 0.1, 1, 10, 100] are informed by literature [18]

4.4 Hybrid Classifier

Additionally, a hybrid classifier is considered which uses a convolutional neural network (CNN) for feature extraction instead of the HOG method used with the SVM. Convolutional neural networks are traditionally used in image classification, and have ben found to be good at identifying and extracting distinguishing features between examples. These features, extracted using through the layers in the neural network, serve as inputs to supervised classifiers such as SVMs, a methodology which has been found to produce better results than regular feature extraction algorithms [19].

Traditional neural networks consist of layers of interconnected neurons, from which outputs are multiplied by weights and passed as inputs to the next layer (in the form of the hypothesis for linear classifiers). The distinction here is the existence of an activation function, which introduces a non-linear element to the equation, and increases the ability of the network to model diverse regularities. CNNs go the extra step of passing filters over images, and pooling [19]. In this project, the CNN is used for feature extraction rather than learning.

4.4.1 Selecting a Neural Network

As stated earlier, a CNN is more suitable for image classification tasks than regular neural networks. The scikit-learn library, though not optimized for deep learning tasks, has a multi-layer perceptron function - a supervised learning algorithm based on neural network architecture. However, the implementation is not intended for large-scale tasks, nor has it been optimized for applications in image classification - regular neural networks produce too many weights when passed an input image. Thus, Keras, a deep learning library is chosen, since it contains a variety of model architectures to choose from.

The chosen architecture was InceptionV3, a relatively lightweight model developed by the Google Brain team for the ImageNet Large Visual Recognition Challenge. It is composed of 42 layers, and won 1st runner up in 2015. Initially, this seems like a too much for a classification task with relatively few images, however, the literature points to the fact that significant improvements over traditional SVM are possible.

4.5 Frontend Web Application Implementation

4.5.1 User Interface Considerations

Since one of the aims of this project is to present an intuitive interface to encourage participation of users on the platform, the user interface of the web application must be crafted in such a way as to be simple and provide effective insight.

Based on the functional requirements, the application must allow the user to enter the origin and destination points for their intended journey, and render on a map the recommended route, overlaid with pothole markers. A simple two-column interface is thus proposed, with the sidebar allowing for the inputs, and a larger column for the map render. Additionally, two buttons are to be added on the map controls; one for refreshing potholes and another for exporting the pothole data.

4.5.2 Development

As mentioned earlier, the web application was developed using HTML, CSS and JavaScript - key web technologies. HTML is the markup language used to build the structure of the web app using HTML elements, CSS introduces styling to those elements, and JavaScript allows the user to interact with these elements. Bootstrap, a front-end CSS

framework, is also used to define key web app elements, including text input boxes and buttons, after which custom CSS is used to introduce styles.

The map functionality is provided by the Google Maps API, which provides the autocomplete feature for the input boxes, used when entering journey origins and destinations. Additionally, the API has a directions functionality, which renders a colored line on the map when a valid route is entered, with the origin and destination makers labeled A and B respectively. After rendering this route the user can press a button to poll the database is for the latitude and longitudes of potholes in the area of said route.

A PHP script is run when this request is initiated, and renders an XML file containing the results of the SQL query sent to the *locations_data* table. All potholes satisfying said query are found, and using are overlaid on the map in the form of markers, as shown in Fig. 4.2 below. Additionally, the download functionality with confirmation is highlighted in Fig. 4.3 and 4.4.

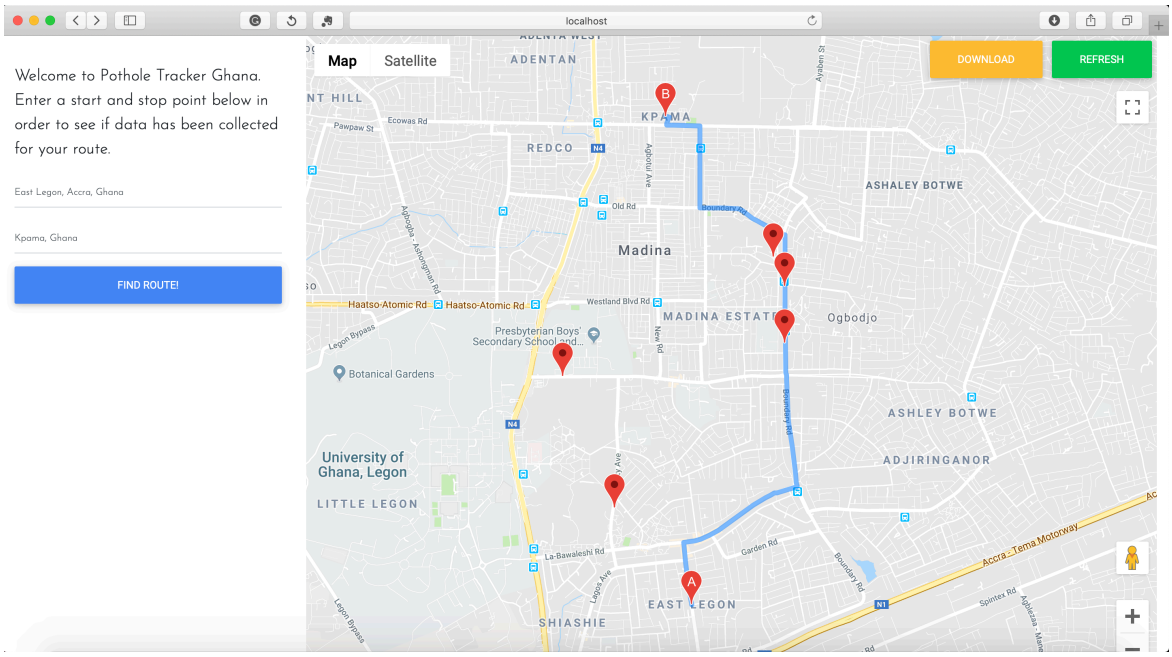

Fig. 4.2 Screenshot showing rendered route with pothole markers

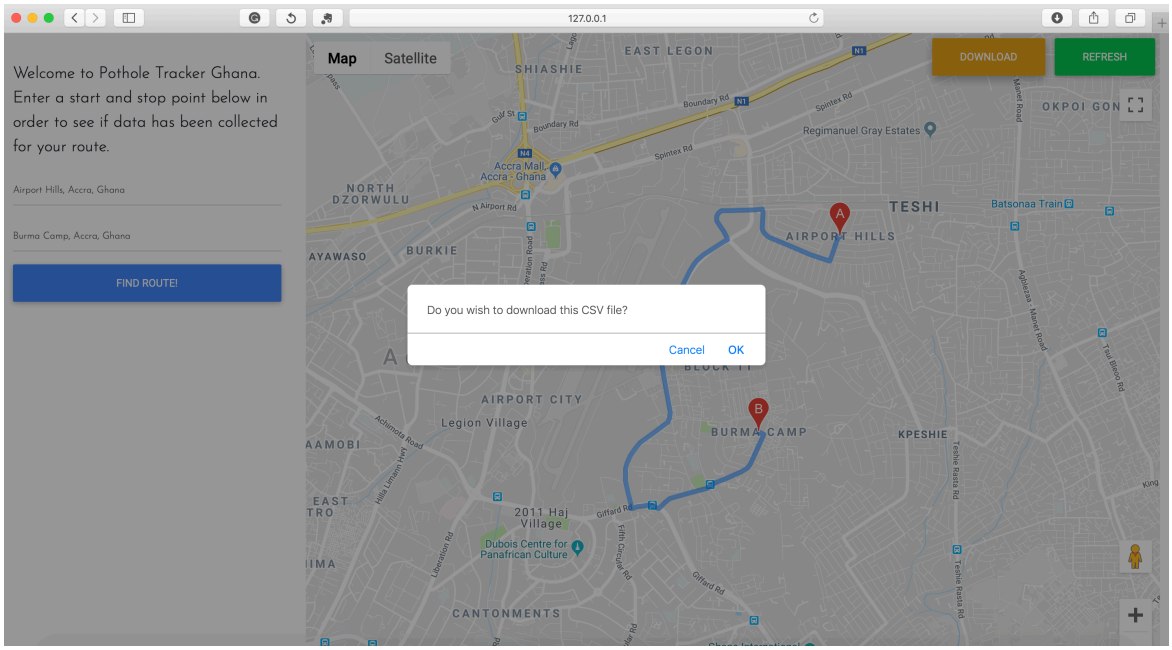

Fig. 4.3 Screenshot showing download confirmation

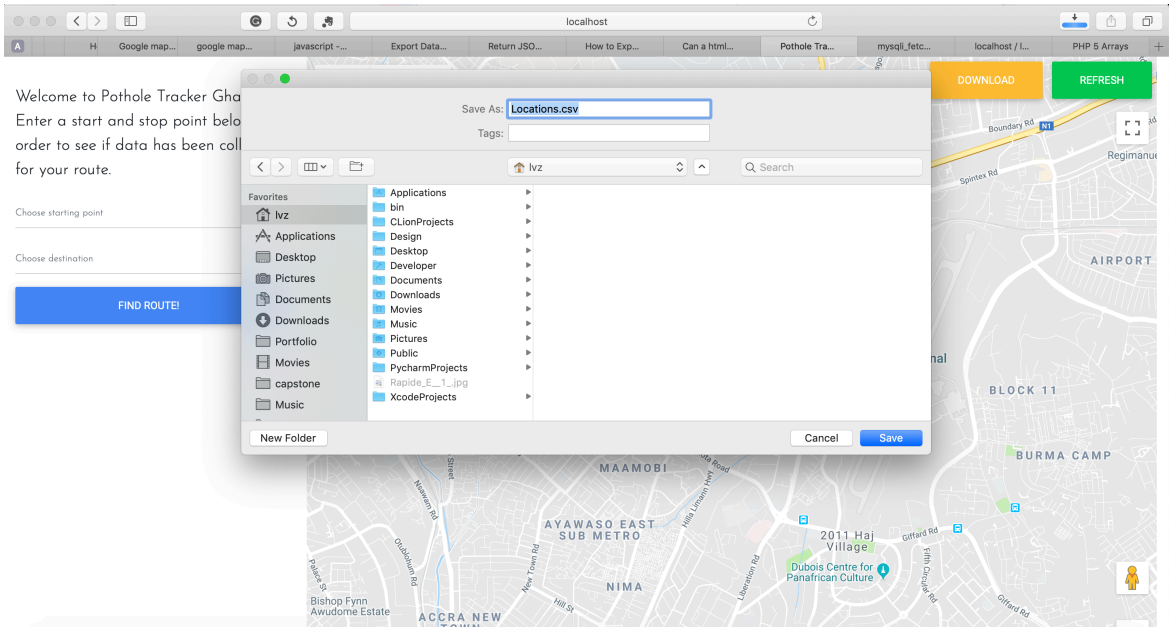

Fig. 4.4 Screenshot showing download feature

Chapter 5: Results

The training data was used to create a 75/25 train-test split on which to test the various models. This split was kept constant by utilizing the same random seed value in sklearn's train_test_split method for each model. The learning curves for each method, detailing the validation and training scores for a varying number of training examples was plotted, and confusion matrices and classification reports were generated.

Further, in order to validate the results received from the initial train-test split, the model was exposed to 10 new examples from a new dataset, consisting of 10 images; 5 potholes and 5 non-potholes. This additional testing was done to determine the ability of the model to generalize to unknown data points, since the images were taken of the potholes themselves and not from the dashboard of a car.

5.1 Testing the Logistic Regression Classification

Fig. 5.1 Logistic Regression learning curves

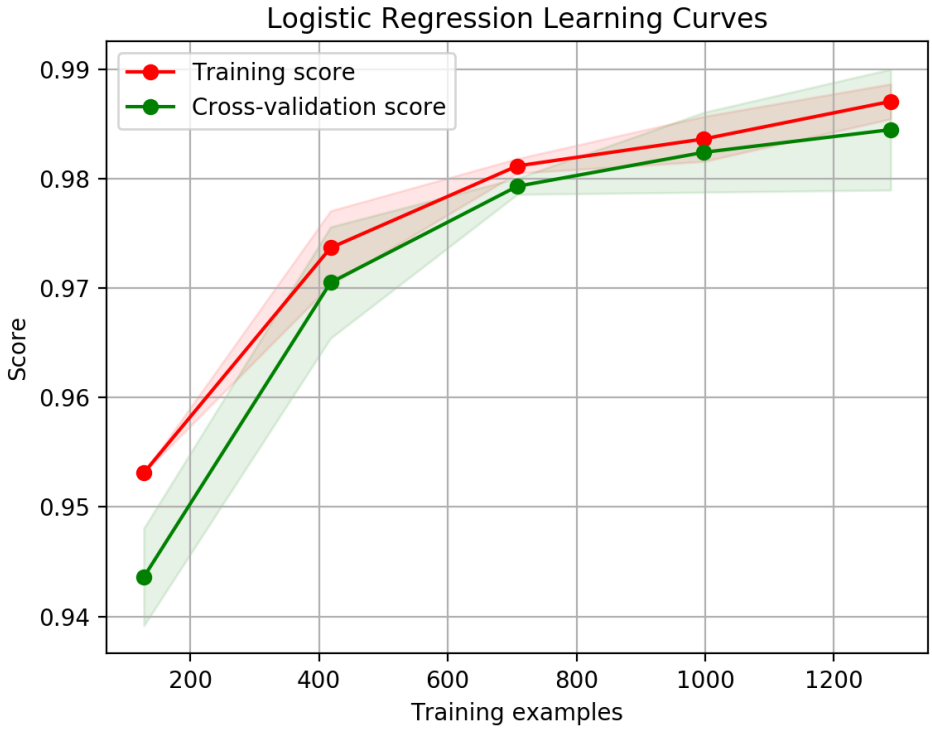

Table 5.1a Confusion matrix for the Logistic Regression classifier

	Predictions	
Actual	**not_a_pothole**	**pothole**
not_a_pothole	299	1
pothole	6	339

Table 5.1b Classification report for the Logistic Regression classifier

	Precision	**Recall**	**F1-score**	**Support**
not_a_pothole	0.98	1.00	0.99	300
pothole	1.00	0.98	0.99	345
Weighted avg	0.99	0.99	0.99	645

Table 5.1c Logistic Regression validation results

	Predictions	
Actual	**not_a_pothole**	**pothole**
not_a_pothole	0	5
pothole	0	5

5.2 Testing the SVM classifier (without hyper-parameter tuning)

Fig. 5.2 Untuned SVM learning curves

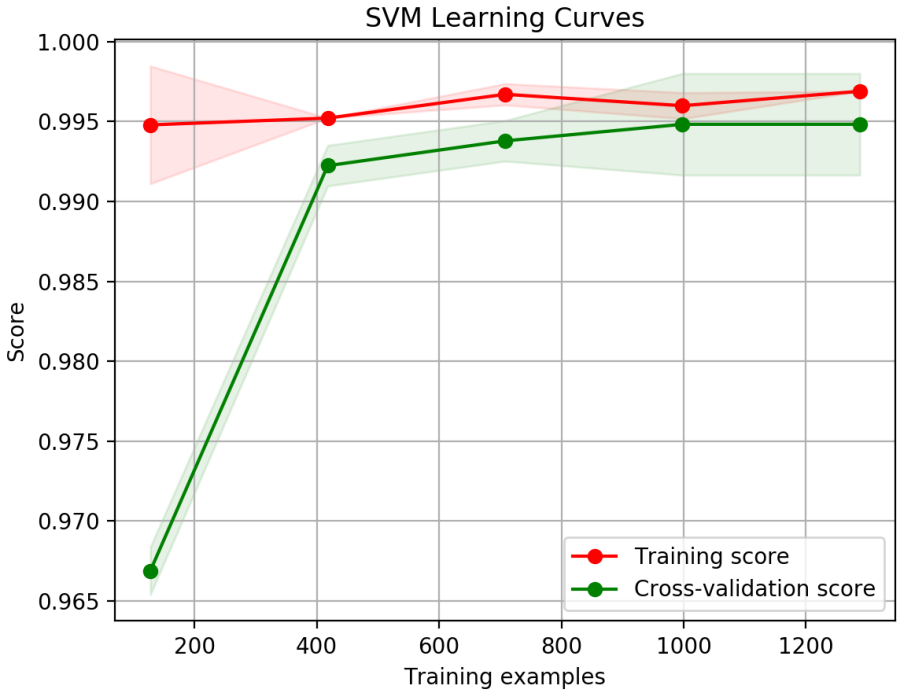

Table 5.2a Confusion matrix for the vanilla SVM classifier

	Predictions	
Actual	**not_a_pothole**	**pothole**
not_a_pothole	300	0
pothole	1	344

Table 5.2b Classification report for the vanilla SVM classifier

	Precision	**Recall**	**F1-score**	**Support**
not_a_pothole	1.00	1.00	1.00	300
pothole	1.00	1.00	1.00	345
Weighted avg	1.00	1.00	1.00	645

Table 5.2c SVM validation results

	Predictions	
Actual	**not_a_pothole**	**pothole**
not_a_pothole	0	5
pothole	0	5

5.2.1 Testing the SVM classifier with hyper-parameter tuning

The result of the grid search algorithm for hyper-parameter tuning resulted in a best value C = 100, which yielded a score of approximately 99.9% in classifying the various classes.

Fig. 5.3 Tuned SVM learning curves

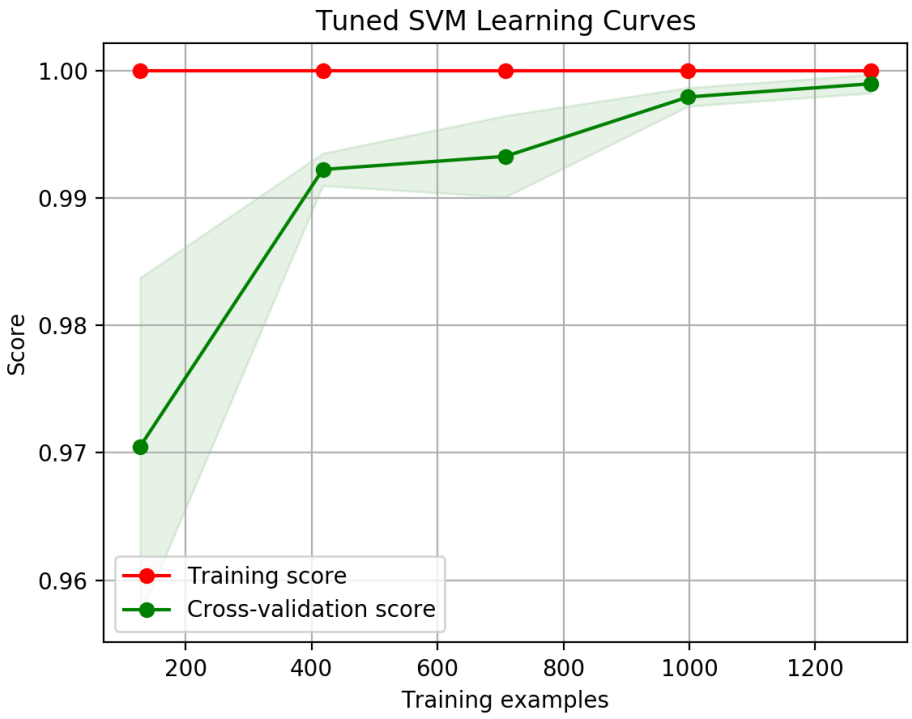

Table 5.2.1a: Confusion matrix for the SVM classifier with hyper-parameter tuning

	Predictions	
Actual	**not_a_pothole**	**pothole**
not_a_pothole	300	0
pothole	0	345

Table 5.2.1b: Classification report for the tuned SVM classifier

	Precision	**Recall**	**F1-score**	**Support**
not_a_pothole	1.00	1.00	1.00	300
pothole	1.00	1.00	1.00	345
Weighted avg	1.00	1.00	1.00	645

Table 5.4c SVM (tuned) validation results

	Predictions	
Actual	**not_a_pothole**	**pothole**
not_a_pothole	1	4
pothole	0	5

5.3 Testing the Hybrid Classifier

Fig. 5.3 Hybrid classifier learning curves

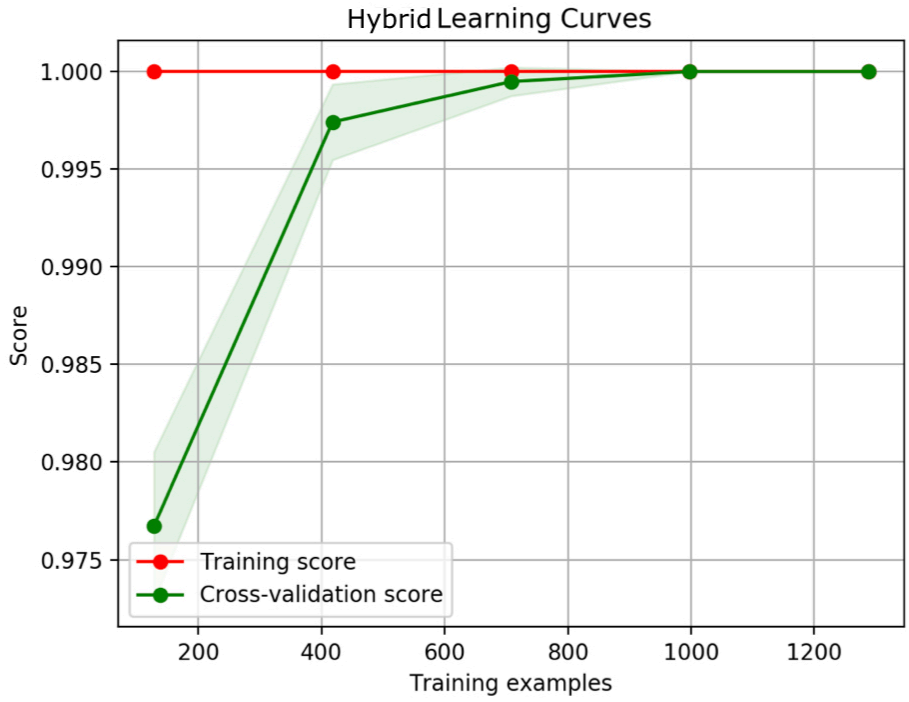

Table 5.3a: Confusion Matrix for the Hybrid classifier

	Predictions	
Actual	**not_a_pothole**	**pothole**
not_a_pothole	300	0
pothole	1	344

Table 5.3b: Classification Report for the Hybrid classifier

	Precision	**Recall**	**F1-score**	**Support**
not_a_pothole	1.00	1.00	1.00	300
pothole	1.00	1.00	1.00	345
Weighted avg	1.00	1.00	1.00	645

* Validation testing could not be carried out due to matrix dimension incompatibility

Chapter 6: Conclusion

6.1 Discussion

After testing, the SVM and hybrid classifiers are found to outperform the logistic regression classifier, which produces an average precision and recall rate of 99% - this is low compared to the 100% classification rates of other classifiers. With respect to the SVM, the untuned model almost achieves a 100% classification rate on the training data after 400 examples - similar to the hybrid model. This is not the case for the logistic regression learning curves, however, which grow steadily towards convergence.

Although the results of training propose that there is no room for improvement, the results of the validation testing show that the untuned SVM model makes no *not_a_pothole* predictions, even though there are actually 5 images with the true value *not_a_pothole*. This is a 50% false positive rate, as well as a 50% misclassification rate of roads that are not a pothole. Considering the nature of this application, where the average road is coated with gravel with a few potholes/deformities, such a high false positive rate would lead to more longitude-latitude values being sent to the web application as markers on the database. This then results in the intuition that a road network needs more work than it actually does, as well as increases (albeit incrementally) the overheads as a result of the constant communication of falsely identified potholes. Since this misclassification rate occurs for both the logistic regression and the untuned SVM, they are less than ideal for this application.

In the case of the tuned SVM, there's a 100% classification rate for both non potholes and potholes in the training data split. The gradual increase in the cross-validation score towards the training score highlights the fact that as the number of

examples increases, the model classification gets better. In validation testing, however, only the potholes are consistently classified. One out of five of the not potholes are also identified, giving the model a classification rate of 20% for negative examples. Although this is far from perfect, it is a better rate than that received from logistic regression and untuned SVM models.

In comparing the results of the validation test set with the original train-test split, it becomes evident that the models do not perform as well when exposed to situations that they have not been trained on - a characteristic of overfitting, a phenomenon in which a model is unable to generalize well on unseen data because it has too closely modeled the underlying pattern of the training data (including noise). This is justified by the fact that the learning curves plotted for each of the models either approach or converge towards 100%. Considering this independently of the validation data points to the fact that the models are effectively learning some features, however in the context of the results of the validation testing, it is not certain whether said features are potholes or otherwise.

In light of this, the decision of the classifier to be used in the system must take into consideration both the results of the initial training, as well as the validation testing. Additional considerations include the robustness needed of running the model on the Raspberry pi. With reference to these metrics, the use of the CNN to extract features for the hybrid classifier does not hold - the amount of time taken to extract image features is significantly high as compared to the other methods.

The remaining methods; logistic regression, as well as the tuned and untuned SVM each produced above-average results in the train-test regard, but appear to be overfitting in

the validation testing. In the end, the classification algorithm chosen for this project is the tuned SVM, which satisfies the requirements for simplicity and quick runtime, and distinguishes itself from the other methods due to the fact that it does better in the validation testing.

6.2 Limitations

One major limitation of this system is the memory constraints of the Raspberry Pi in this project application. Due to the fact that the system is expected to maintain constant uptime, tradeoffs have to be made between the computation power needed to use the chosen model, as well as the effectiveness of said model. Since this application is not as high-risk in terms of the impact of false positives/negatives, the received accuracies are acceptable - that is, if a false positive is received in a time step but is surrounded by true positives, it does not change the fact that a portion of the entire stretch of road needs to be re-done. However, it would be preferable if this tradeoff is not made.

Additionally, the fact that the validation test could not be carried out on the hybrid classifier means that there could potentially be a better means of classification. However, the fact that CNN feature extraction with InceptionV3 would have to be carried out on every grabbed frame is reason enough to remove this methodology for the comparison, as it takes away from the system requirement of responsiveness.

6.3 Future Work

The proposals for future works concern improvements to the technologies used, as well as the decisions made concerning the classification algorithm. With respect to technologies, leveraging the Android Auto SDK is proposed. This is a set of APIs created

by Google which allow a user application to access the various sensors and peripherals present in any compatible vehicle. Creating an application that is all encompassing would then allow for the full integration of the system code and application into the car, as compared to standalone hardware. This would not only help with uptime and robustness, but would also allow for embedded route visualizations.

Concerning the feature detection methodology employed, future works can take into consideration the fact that potholes are generally located in a particular position in an image when viewed from the dashboard - at the bottom of the image - and incorporating this knowledge would allow the use of a mask for the color intensity histogram for feature extraction. This could potentially increase the discrimination ability of the model. In line with this, images could potentially be cropped during the preprocessing process, as this would limit the external non-pothole features the model might be learning (e.g. foliage, sidewalks, sky etc.)

Another pre-processing step to explore would be exploring the angle from which data is collected as a design decision. The choice of said angle was made arbitrarily in this project, however a more structured methodology could be considered in order to explore potential effects on the classification results. Additionally, rather than increase the number of training examples, the number of unique training examples will be increased, as the system would incorporate a variety of images of potholes from different perspectives and under different conditions of illumination.

Finally, in future iterations, another major improvement to be implemented in this system would be the ability to detect road deformities dynamically through video. This

would reduce the processing time used to capture and pre-process frames, as well as make the system more cognizant of potholes in the distance, removing the need to capture frames over a particular interval. Including the ability to combine potholes in an area instead of counting them individually would also reduce the number of data points sent to the database, making the system more similar to human beings in terms of detection. Additionally, it would allow the routes to be plotted according to a key - much like Bhatt et al., increasing the user-friendliness of the web application.

References

[1] MyJoyOnline.com, “61% of Ghana’s roads classified as poor.” [Online]. Available: https:// www.myjoyonline.com/news/2017/December-31st/61-of-ghanas-classified-as-poor.php. [Accessed: 13-Nov-2018].

[2] G. Sarpong, "Safety Alert: 6 people die daily and 2,000 perish annually through road traffic crashes in Ghana - iWatch Africa", *iWatch Africa*, 2018. [Online]. Available: http://iwatchafrica.org/2018/09/06/safety-alert-6-people-die-daily-and-2000-perish-annually-through-road-traffic-crashes-in-ghana/. [Accessed: 13- Nov- 2018].

[3] GhanaWeb.com, “Potholes on Tema Motorway causes another accident.” [Online]. Available: https://www.ghanaweb.com/GhanaHomePage/NewsArchive/Potholes-on-Tema-Motorway-causes-another-accident-699898. [Accessed: 13-Nov-2018].

[4] Citifmonline, “Gov’t to award contracts for repair of 253 ‘bad’ roads,” 2017. [Online]. Available: http://citifmonline.com/2017/09/26/govt-to-award-contracts-for-repair-of-253-bad-roads-full-list/. [Accessed: 13-Nov-2018].

[5] The British Machine Vision Association and Society for Pattern Recognition, “What is computer vision?” [Online]. Available: http://www.bmva.org/visionoverview. [Accessed: 13-Nov-2018].

[6] A. Kulkarni, N. Mhalgi, S. Gurnani, and N. Giri, “Pothole Detection System using Machine Learning on Android,” vol. 4, no. 7, pp. 360–364, 2014.

[7] F. D. Vorgbe, “Classification of Road Surface Quality using Android Smartphone Devices,” Ashesi University, 2014.

[8] U. Bhatt, S. Mani, E. Xi, and J. Z. Kolter, “Intelligent Pothole Detection and Road Condition Assessment,” 2017.

[9] H. Punjabi, R. Nanwani, A. Vaswani, R. Jotwani, and A. Kunte, “Intelligent Pothole Detection System,” Int. J. Emerg. Technol. Adv. Eng., vol. 4, no. 7, 2014.

[10]S. Nienaber, M. Booysen, and R. Kroon, “Detecting Potholes using Simple Image Processing Techniques and Real-World Footage.” 2015.

[11] “Ghana Highway Authority.”

[12] Rosebrock, A. “Deep Learning for Computer Vision with Python,” 2017.

[13] Rosebrock, A. “Practical Python and OpenCV, 3rd Edition,” 2016.

[14] “Histogram of Oriented Gradients (HOG) Descriptor,” 2018. [Online]. Available: https://software.intel.com/en-us/ipp-dev-reference-histogram-of-oriented-gradients-hog-descriptor. [Accessed: 17-Mar-2019].

[15] Google Developers, “Machine Learning Crash Course - Classification: Precision and Recall,” Google Developers, 2019. [Online]. Available: https://developers.google.com/machine-learning/crash-course/classification/precision-and-recall. [Accessed: 19-Mar-2019].

[16] A. Rosebrock, “An intro to linear classification with Python,” 2016. [Online]. Available: https://www.pyimagesearch.com/2016/08/22/an-intro-to-linear-classification-with-python/. [Accessed: 04-Mar-2019].

[17] A. Ng, “Support Vector Machines,” 2011. [Online]. [Accessed 4-Mar-2019].

[18] X. Sun, L. Liu, W. Hanshi, S. Wei, and L. Jingli, “Image Classification via Support Vector Machine,” ICCSNT, 2015.

[19] S. Notley and M. Magdon-Ismail, “Examining the Use of Neural Networks for Feature Extraction: A Comparative Analysis using Deep Learning, Support Vector Machines, and K-Nearest Neighbor Classifiers,” 2018.

[20] A. Rosebrock, “linear_classifier.py,” 2016. (Version 1.0) [Source code]. Available: https://www.pyimagesearch.com/2016/08/22/an-intro-to-linear-classification-with-python/

[20] A. Rosebrock, “test_image.py” 2015. (Version 1.0) [Source code]. Available: https://www.pyimagesearch.com/2015/03/30/accessing-the-raspberry-pi-camera-with-opencv-and-python/

[21] Franky, “vgg16_pred_001.py” 2018. (Version 1.0) [Source code]. Available: https://medium.com/@franky07724_57962/using-keras-pre-trained-models-for-feature-extraction-in-image-clustering-a142c6cdf5b1