CSE3999 - Technical Answers for Real World Problems (TARP)

Project Report

Pothole Detection using Machine Learning

By

18BCE1077	Shreshth Vats
18BCE1100	Ishita Amit Saluja
18BCE1276	Anmol Choubey
18BCE1301	Siddhanth Harish Bist
18BCE1339	Shivika Manglick

B.Tech. Computer Science and Engineering

Submitted to

Prof S A Sajidha

School of Computer Science and Engineering

Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

June 2021

DECLARATION

I hereby declare that the report titled "**Pothole Detection using Machine Learning"** submitted by me to VIT Chennai is a record of bona-fide work undertaken by me under the supervision of **Prof S A Sajidha**, School of Computer Science and Engineering, Vellore Institute of Technology, Chennai.

Signature of the Candidate

Shreshth Vats

Reg. No. 18BCE1077

Ishita Amit Saluja

Reg. No. 18BCE1100

Anmol Choubey

Reg. No. 18BCE1276

Siddhant Harish Bist

Reg. No. 18BCE1301

Shivika Manglick

Reg. No. 18BCE1339

<u>CERTIFICATE</u>

Certified that this project report entitled "**Pothole Detection using Machine Learning"** is a bonafide work of **Shreshth Vats (18BCE1077), Ishita Amit Saluja (18BCE1100), Anmol Choubey (18BCE1276), Siddhant Harish Bist (18BCE1301), Shivika Manglick (18BCE1339)** and they carried out the Project work under my supervision and guidance for CSE3999 - Technical Answers for Real World Problems (TARP).

Prof S A Sajidha

SCOPE, VIT Chennai

ACKNOWLEDGEMENT

We wish to express our sincere thanks and deep sense of gratitude to our project guide, **Prof S A Sajidha**, School of Computer Science and Engineering for his/her consistent encouragement and valuable guidance offered to us throughout the course of the project work.

We are extremely grateful to **Dr. R. Jagadeesh Kannan**, Dean and **Dr. S. Geetha**, Associate Dean, School of Computer Science and Engineering (SCOPE), Vellore Institute of Technology, Chennai, for extending the facilities of the School towards our project and for their unstinting support.

We express our thanks to **Dr. Justus S,** Head of the Department, B.Tech. Computer Science and Engineering for his support throughout the course of this project.

We also take this opportunity to thank all the faculty of the School for their support and their wisdom imparted to us throughout the courses.

We thank our parents, family, and friends for bearing with us throughout the course of our project and for the opportunity they provided us in undergoing this course in such a prestigious institution.

Shreshth Vats

Reg. No. 18BCE1077

Ishita Amit Saluja

Reg. No. 18BCE1100

Anmol Choubey

Reg. No. 18BCE1276

Siddhant Harish Bist

Reg. No. 18BCE1301

Shivika Manglick

Reg. No. 18BCE1339

ABSTRACT

Roads connect everything, from villages to cities and have become a key part of the transportation infrastructure nowadays. The increase in the sum total of vehicles, the load of the traffic is getting larger by the day and road damage is quite obvious. Damaged roads may harm vehicles, are sometimes life risking to drivers and people nearby, as well as can also lead to a significant loss of property.

It is really important to find road conditions, which can be splitted into the four following stages:

1) **Manual detection**
2) **Semi-automated detection**
3) **Automated detection**
4) **Informatization detection**

We have tried to design a transfer learning with Convolutional Neural Networks (CNN) based solution to solve the image classification problems using Kaggle dataset, Pothole and Plain Road Images. The objective of our project is to build a binary classification model that classifies if a particular image has potholes or not.

CONTENTS

1. Introduction

1.1 Objective and goal of the project

Objective of the project is to reduce the rate of accidents which occur due to potholes and according to the data available on the internet 40% of accidents occur due to potholes. Therefore, the goal of our project is to predict the road quality, i.e, to detect the potholes which will make the most used transportation system more safe, efficient and comfortable using some machine learning and deep learning algorithms.

1.2 Problem Statement

Using deep learning and transfer learning techniques to solve the binary image classification problem of pothole detection by adopting an accurate model for savings in training time and computational efficiency.

As the number of cars on the road are increasing at a very high rate thus the traffic load is high and road damage is not avoidable. Damaged roads (e.g. potholes) are dangerous for drivers and pedestrians, even damage to the vehicle.

1.3 Motivation

Nowadays, roads connect cities, buildings, towns, villages and countries thus it is considered as the key transportation infrastructure in developed countries as well as developing countries. Due to the increase in traffic load which often results in pavement damage is unavoidable as it may be dangerous for drivers and pedestrians, even the vehicles get damaged. Thus, giving the information to drivers about the damaged road conditions, is directly related to increase in the safety level of drivers and pedestrians as well as comfort level in driving experience. The early notification of road surface irregularities is one the best ways to avoid accidents and motivate municipal authorities to upkeep and repair the anomalies.

1.4 Challenges

In this project, we have tried to implement a transfer learning and CNN based solution to solve image classification problems for pothole and plain road images. In this project, we will be using a solution based on pre-trained models. We will use models that are composed of two parts:

- Convolutional base
- Classifier

The challenges that we faced were:

- identifying the algorithm
- comparing the algorithm with already existing models
- optimizing the algorithm such that its better than the existing models
- integrating the algorithm in the project
- error identification and correction
- smooth running of the project
- enhancing the output speed

1.5 Existing System

The best two methods that we came across for Image Classification is CNN and Simple Image Processing. The testing accuracy of the Image Processing model was around 85% while that of CNN was more than 90%.

Hence we used CNN for our implementation.

In CNN we compared various models:

MODEL 1: Fully Connected Layer (97.2% testing accuracy)

In a neural network, fully linked layers are those where all of the inputs from one layer are linked to every activation unit of the following layer. The last few layers are fully connected, which assemble the data taken from previous layers to generate the final output.

MODEL 2: Global Average Pooling (94.3% testing accuracy)

Global Average Pooling is a pooling procedure used in CNN to substitute fully linked layers. The goal is to build one feature map for each matching category of the classification job. We take the average of each feature map and feed the resultant vector directly into the softmax layer, rather than constructing fully linked layers on top of the feature maps.

One advantage of global average pooling over fully connected layers is that it enforces correspondences between feature maps and categories, making it more natural to the convolution structure. As a result, the feature maps may be simply read as confidence map categories.

MODEL 3: Linear Support Vector Machine (95.1% testing accuracy)

The Support Vector Machine, or SVM, is a common Supervised Learning technique that may be used to solve both classification and regression issues. However, it is mostly utilised in Machine Learning for Classification difficulties. Linear SVM is used for linearly separable data, which implies that if a dataset can be categorised into two classes using a single straight line, it is linearly separable data, and a Linear SVM classifier is utilised.

MODEL 4: Random Forest Classifier (87.8% testing accuracy)

Random forests, also known as random decision forests, are an ensemble learning approach for classification, regression, and other problems that work by training a large number of decision trees. For classification tasks, the random forest's output is the class chosen by the majority of trees. The mean or average forecast of the individual trees is returned for regression tasks. Random decision forests address the problem of decision trees overfitting their training set.

MODEL 5: Bayesian Classifier (92.2% testing accuracy)

The job of a class is to forecast the values of characteristics for members of that class, according to a Bayesian classifier. Because the characteristics have similar values, the examples are divided into classes. A Bayesian classifier works on the principle that if an agent understands the class, it can anticipate the values of the other attributes. If it doesn't know the class, it can apply Bayes' rule to guess it based on the feature values.

Reason for choosing MODELS 1, 2 & 3

Out of the 5 models we studied, we selected the best three for this project with testing accuracy of 94% and above worked on these (fully connected layers, global average pooling, linear support vector machine) to make them better.

2. Literature Survey

[1] A Real-Time Pothole Detection Approach for Intelligent Transportation System

Wang, H.W., Chen, C.H., Cheng, D.Y., Lin, C.H. and Lo, C.C., 2015. A real-time pothole detection approach for intelligent transportation systems. *Mathematical Problems in Engineering*, *2015*.

This paper has proposed a pothole detection method which is based on mobile sensing, also it shares with the user pothole information. For this purpose, the mobile device should be equipped with a GPS to collect accelerometer data and location information. Then the pothole information is made available to the public to improve road safety.

[2] Pothole Detection Using Computer Vision and Learning

Dhiman, A. and Klette, R., 2019. Pothole detection using computer vision and learning. *IEEE Transactions on Intelligent Transportation Systems*, *21*(8), pp.3536-3550.

In this research, four different techniques are proposed and tested against each other. Each technique has its own benefits and can provide different pathways to a number of applications. The LM1 model can identify a pothole under challenging weather conditions with good precision and recall whereas the LM2 model is capable of real-time pothole identification. The SV2 approach can identify potholes and road manifolds with very high accuracy when used with stereo-vision cameras. The SV2 approach can can also be used to track potholes from one frame to another, and is relatively easy to implement.

[3] Pothole Detection System Using a Black-box Camera

Jo, Y. and Ryu, S., 2015. Pothole detection system using a black-box camera. *Sensors*, *15*(11), pp.29316-29331.

In this paper, we propose a novel pothole-detection system using a commercial black-box camera. The proposed system is mounted on the front windshield of a vehicle and can detect a pothole in real-time. A pothole-detection algorithm is installed on an embedded board in the black-box camera. This algorithm collects information regarding the size of potholes and their location, and this information is stored in the black box and then transmitted to a pothole-management server.

[4] An Automated Machine-Learning Approach for Road Pothole Detection Using Smartphone Sensor Data

Wu, C., Wang, Z., Hu, S., Lepine, J., Na, X., Ainalis, D. and Stettler, M., 2020. An automated machine-learning approach for road pothole detection using smartphone sensor data. *Sensors*, *20*(19), p.5564.

The objective of this study is to develop a method for detecting potholes using vibration sensors embedded in smartphone. They propose a useful crowd sourced automated road-monitoring system. The general workflow consists of four stages:

(1) data acquisition,

(2) data processing,

(3) feature extraction, and

(4) classification.

[5] Image-Based Pothole Detection System for ITS Service and Road Management System

Ryu, S.K., Kim, T. and Kim, Y.R., 2015. Image-based pothole detection system for ITS service and road management system. *Mathematical Problems in Engineering*, *2015*.

The proposed solution or method can be divided into three steps:

(1) segmentation,

(2) candidate region extraction, and

(3) decision.

[6] DeepBus: Machine Learning based real time pothole detection system for smart transportation using IoT

Bansal, K., Mittal, K., Ahuja, G., Singh, A. and Gill, S.S., 2020. DeepBus: Machine learning based real time pothole detection system for smart transportation using IoT. *Internet Technology Letters*, *3*(3), p.e156.

This paper proposes a machine learning based smart pothole detection called Deep-Bus for real time identification of surface irregularities on roads using IoT sensors. There is a design of an application for smartphones with IoT sensors (accelerometer, gyroscope, GPS) to identify if a vehicle is moving over a pothole or a speed bump in real time and update a common map for all the users with the precise location of road conditions.

[7] RIPD: Route Information and Pothole Detection

Garg, S., Mate, G.S., Das, R., Tiple, S. and Panicker, A., 2015. RIPD: Route Information and Pothole Detection. *International Journal of Advanced Research in Computer and Communication Engineering*, *4*(12), pp.194-197.

This paper proposed the use of Android smartphones. The built-in accelerometer in the phone is used to collect the required data. GPS is used to send the location coordinates of the potholes. At the server side, the information received is processed using aggregation algorithms and then the list of potholes is prepared. This will help the required authorities to take necessary steps to remove potholes.

[8] Pothole Detection System using Machine Learning on Android

Kulkarni, A., Mhalgi, N., Gurnani, S. and Giri, N., 2014. Pothole detection system using machine learning on Android. *International Journal of Emerging Technology and Advanced Engineering*, *4*(7), pp.360-364.

This system, called the Pothole Detection System, uses the Accelerometer Sensor of Android smartphones for detection of potholes and GPS for plotting the location of pothole on Google Maps. Using a simple machine-learning approach, it shows that we are able to identify the potholes from accelerometer data.

[9] Development of asphalt paved road pothole detection system using modified colour space approach

Isah, A.D., Aibinu, A.M., Olaniyi, O.M. and Campbell, O.O., 2018. Development of asphalt paved road pothole detection system using

modified colour space approach. ***Journal of Computer Science and Its Application*****,** ***25*****(2), pp.1-15.**

The paper discusses the reduction of the complexity of potholes detection systems using a model that reduces the stages involved in image processing technique for pothole detection compared with other methods from related literature which uses; 3D reconstruction method, vibration method, video images using artificial neural network models, machine vision, histogram shape-based thresholding and spectral clustering from histogram data obtain from gray scale images.

[10] Smart Sensing of Potholes and Obstacles in Automotive Applications using IoT

Ajitha, U., Sharma, S., Selvi, V.A. and Priyanka, A., 2018. Smart sensing of potholes and obstacles in automotive applications using the internet of things. ***International Journal of Pure and Applied Mathematics*****,** ***119*****(15), pp.1479-1486.**

In the proposed system pothole detection and obstacle detection is performed. When the vehicle moves over the uneven surface the accelerometer sensor fixed in it will detect the pothole based on the angle of inclination. At the time the pothole is detected the camera will be turned on and the image will be captured, the location is tracked using GPS to provide the longitude and latitude values of the pothole. This data will be initially stored in the raspberry pi and will be sent to the server.

3. Requirements Specification

3.1 Hardware Requirements

Intel i5 5th Generation 2.4 Ghz Processor

We tested our project on the 5th Generation Intel Core i5 processor which offers a smooth and powerful performance.

3.2 Software Requirements

Python 3.8.0

Python is an interpreted high-level general-purpose programming language. Python' design philosophy emphasizes code readability with its notable use of significant indentation.
We have used the Python language to execute our code on the Python 3.8.0 software, along with pip.

Jupyter Notebook

Jupyter is a free, open-source, interactive web tool known as computational notebook, which researchers can use to combine software code, computational output, explanatory text and multimedia resources in a single document.
We implemented our project using Jupyter Notebook.

Anaconda

Anaconda is a distribution of the Python language for scientific computing (data science, machine learning applications, large-scale data processing, predictive analytics, etc.) that aims to simplify package management and deployment. We have used Anaconda to manage our packages in our project.

4. System Design

4.1 Transfer Learning

- One of the most popular methods in computer vision is transfer learning as it helps in building accurate models with significant time saving.
- With the use of transfer learning, by learning the patterns of solutions of different problems, we can solve other problems.
- This way we don't have to start from scratch.
- A pre-trained model is trained on a dataset which solves some problems which are similar to the one we want.
- Because of the cost of computation which comes out of training models, it's better to import and use models from literature that are already published (e.g. VGG, Inception, MobileNet).

4.2 Convolutional Neural Networks

- Most of the pre-trained models that we use in transfer learning are based on large Convolutional Neural Networks.
- Two of the main factors which drive the popularity of CNN are high performance and ease of training.
- CNN typically has two parts:
    - ❖ Convolutional base - It is compass by a stack of pooling and convolutional layers with the primary goal of generating features from the image.
    - ❖ Classifier - It is usually composed of fully connected layers (layers whose neurons are having full connections to all the activation in the previous layer), and has the main goal of classifying the image based on the detected features.

4.3 CNN Architecture

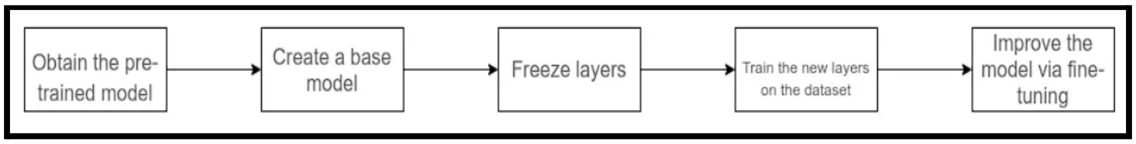
- CNN is a deep learning model that can automatically learn hierarchical feature representations.
- This means that features computed by the first layer are general and can be reused in different problem platforms, while features computed by the last layer are specific and depend on the chosen dataset.
- Hence, the convolutional base of CNN refers to general features, whereas the classifier part, and some of the higher layers of the convolutional base, refer to specialised features.

4.4 Transfer Learning Process

Transfer learning is a machine learning method where a model developed for a task is reused as the starting point for a model on a second task. We can also say that it is an approach in deep learning where pre-trained models are used as the starting point.

In our project we have used the pre-trained model generated by **VGG16** Convolution base. Therefore, the features extracted from the convolutional base will be the same for all the classifiers in our project. This way we can leverage previous learnings and avoid starting from scratch.

In our project, we have implemented the transfer learning process in the following way:


Obtain the pre-trained model
Create a base model
Freeze layers
Train the new layers on the dataset
Improve the model via fine-tuning

1. Obtain a pre-trained model based on the problem

   We used pre-trained weights from the VGG16 Convolution base.

2. Create a base model

   When creating the base model, we had to remove the final output layer. Later on, we added a final output layer that is compatible with our problem.

3. Freeze layers so they don’t change during training

   Freezing the layers from the pre-trained model is important. The reason is that we don’t want the weights in those layers to be restarted in further steps. If this happens, then we will lose all the learning that has already taken place. And thus, this will be no different from training the model from scratch.

4. Add new trainable layers and train them on the dataset

   In the next step we added new trainable layers that will turn old features into predictions on the new dataset. This is done as the pre-trained model is loaded without the final output layer.

   We then train the model with a new output layer.

5. Improve the model via fine-tuning

   After the previous step, we get a model that can make predictions on the dataset. We can improve its performance through fine-tuning. Fine-tuning is done by unfreezing the base model and training the entire model again on the whole dataset at a very low learning rate. The low learning rate will increase the performance of the model on the new dataset while preventing overfitting.

5. Implementation of System

Prepare Data

Choose an appropriate dataset. A different version of the dataset which is comparatively smaller can be used which runs the models faster and is a great tool for people whose computational powers are limited.

Extract Features

Perform feature extraction from convolutional base which will then help us to feed the classifiers we are training.

Using the pre-trained model

This is part of transfer learning. To use a pre-trained model for our own needs, we start by removing the original classifier, then we add a new classifier that fits our purposes, and finally we have to fine-tune our model according to one of three strategies:

- Obtain a pre-trained model based on the problem
- Create a base model
- Freeze layers so they don’t change during training
- Add new trainable layers and train them on the dataset
- Improve the model via fine-tuning

Description of Classifiers

Fully Connected Layers Classifier:

Fully Connected layers in neural networks are those layers where all the inputs from one layer are connected to every activation unit of the next layer. In most popular machine learning models, the last few layers are fully connected layers which compile the data extracted by previous layers to form the final output. A fully connected neural network consists of a series of fully connected layers. A fully connected layer is a function from $\mathbb{R}$ m to $\mathbb{R}$ n. Each output dimension depends on each input dimension.

Methodology

• For image classification problems, the standard approach is to use a stack of fully-connected layers followed by a softmax activated layer.

• The softmax layer outputs the probability distribution over each possible class label and then we just need to classify the image according to the most probable class.

• In comparison to the other layers, this category of layers has the highest number of parameters because every neuron is connected to every other neuron. The number of parameters is the product of the number of neurons in the current layer c and the number of neurons on the previous layer p plus the bias term. Thus, number of parameters here are: ((current layer neurons c * previous layer neurons p) +1*c).

• Softmax Function: The softmax function takes as input a vector z of K real numbers, and normalizes it into a probability distribution consisting of K probabilities proportional to the exponentials of the input numbers.

Working Algorithm

1. Perform feature extraction from convolutional base. These features will feed our fully-connected layer classifier for training.
2. This classifier adds a stack of fully-connected layers that is fed by the features extracted from the convolutional base.
3. Train the model by fixing the epochs and batch size.
4. Analyse the performance of the model using various performance metrics.

Global Average Pooling:

Global Average Pooling is a pooling operation designed to replace fully connected layers in classical CNNs. The idea is to generate one feature map for each corresponding category of the classification task in the last mlpconv layer. Instead of adding fully connected layers on top of the feature maps, we take the average of each feature map, and the resulting vector is fed directly into the softmax layer. Advantage over Fully Connected Layer is that the number of parameters is reduced.

Methodology

1. Input the images.
2. One feature map will be formed for each corresponding category of the classification task in the last mlpconv layer.
3. We take the average of each feature map, and the resulting vector is fed directly into the softmax layer.
4. The final layers consist simply of a Global Average Pooling layer and a final softmax output layer. As can be observed, in the architecture above, there are 64 averaging calculations corresponding to the 64, 7 x 7 channels at the output of the second convolutional layer. The GAP layer transforms the dimensions from (7, 7, 64) to (1, 1, 64) by performing the averaging across the 7 x 7 channel values.

Working Algorithm

1. Perform feature extraction from convolutional base. These features will feed our Global Average Pooling layer classifier for training.
2. This classifier adds a stack of Global Average Pooling layers and the output is fed as a sigmoid activated layer.
3. Train the model by fixing the epochs and batch size.
4. Analyse the performance of the model using various performance metrics.

Linear Support Vector Machine:

"Support Vector Machine" (SVM) is a supervised machine learning algorithm which can be used for both classification or regression challenges. However, it is mostly used in classification problems. In the SVM algorithm, we plot each data item as a point in n-dimensional space (where n is the number of features you have) with the value of each feature being the value of a particular coordinate. Then, we perform classification by finding the hyper-plane that differentiates the two classes very well. Support Vectors are simply the coordinates of individual observation. The SVM classifier is a frontier which best segregates the two classes (hyper-plane/ line).

Methodology

A simple linear SVM classifier works by making a straight line between two classes. That means all of the data points on one side of the line will represent a category and the data points on the other side of the line will be put into a different category. This means there can be an infinite number of lines to choose from.

What makes the linear SVM algorithm better than some of the other algorithms, like k-nearest neighbors, is that it chooses the best line to classify your data points. It chooses the line that separates the data and is the furthest away from the closest data points as possible.

Working Algorithm

1. Perform feature extraction from convolutional base. These features will feed our Linear SVM classifier for training.
2. This classifier adds a stack of Linear SVM and the output is fed as a sigmoid activated layer.
3. Train the model by fixing the epochs and batch size.
4. Analyse the performance of the model using various performance metrics.

Build Classifiers

Classification is the process of predicting the class of the given data. Classification modeling is the task of finding an approximate value using a mapping function (f) from input variables (X) to discrete output variables (y).

A classifier utilizes some training data to understand how given input variables relate to the class. The algorithm for implementation of classifiers in our models is given below:

1. *Define model:*
   a. *Import packages*
   b. *Set epochs*
   c. *Set parameters taken*
2. *Compile model*
3. *Train the model with set epochs and parameters*
4. *Plot the results using matplotlib library:*
   a. *Plot training and validation accuracy*
   b. *Plot training and validation loss*
5. *Define function to visualize predictions:*
   a. *Get picture*
   b. *Extract features*
   c. *Make prediction*
   d. *Show picture*
   e. *Write the prediction – 'pothole' or 'plain road'*
6. *Visualize predictions*
7. *Define performance metrics*

Train Model

Train the model by fixing the epochs and batch size.

Performance Analysis

Analyse the performance of the model using various performance metrics. The metric we used for calculating the testing accuracy of our classifiers is **Area under the ROC curve or AUC**.

The area under a receiver operating characteristic (ROC) curve, abbreviated as AUC, is a single scalar value that measures the overall performance of a binary classifier . The AUC value is within the range [0.5–1.0], where the minimum value represents the performance of a random classifier and the maximum value would correspond to a perfect classifier.

The AUC is a robust overall measure to evaluate the performance of score classifiers because its calculation relies on the complete ROC curve and thus involves all possible classification thresholds.

The AUC is typically calculated by adding successive trapezoid areas below the ROC curve. Figure 1 shows the ROC curves for two score classifiers A and B. In this example, classifier A has a larger AUC value than classifier B.

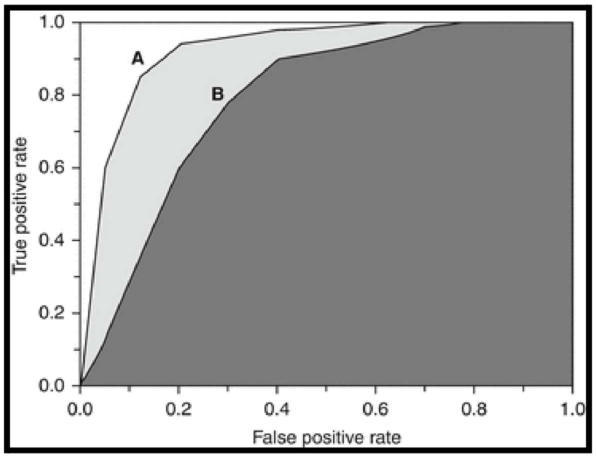


Figure 1 : ROC curves for two score classifiers A and B

Architecture model

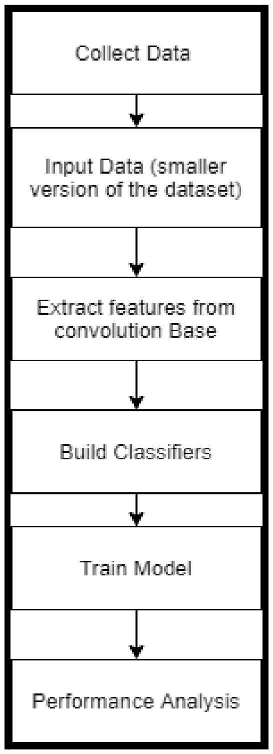
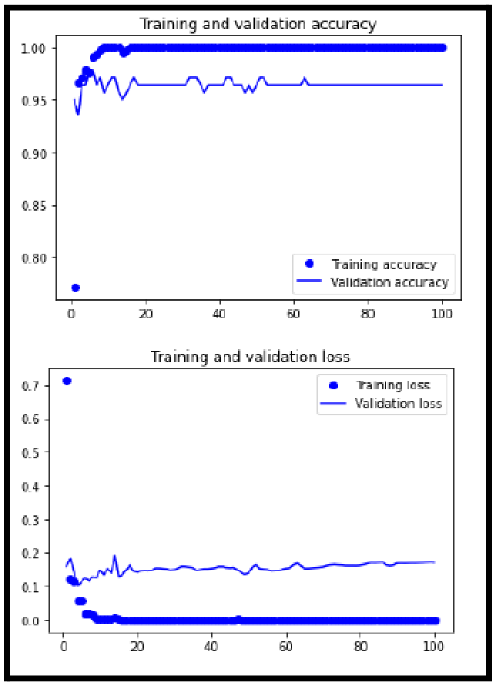
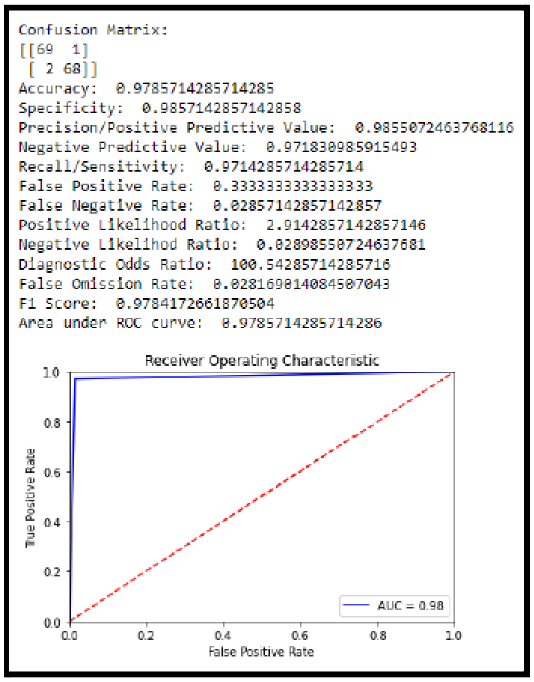

Collect Data
Input Data (smaller version of the dataset)
Extract features from convolution Base
Build Classifiers
Train Model
Performance Analysis

Figure 2 : Architecture model of the System

6. Results and Discussion

Fully-Connected Layers





```
Confusion Matrix:
[[69  1]
 [ 2 68]]
Accuracy:  0.9785714285714285
Specificity:  0.9857142857142858
Precision/Positive Predictive Value:  0.9855072463768116
Negative Predictive Value:  0.971830985915493
Recall/Sensitivity:  0.9714285714285714
False Positive Rate:  0.3333333333333333
False Negative Rate:  0.02857142857142857
Positive Likelihood Ratio:  2.9142857142857146
Negative Likelihod Ratio:  0.028985507246376681
Diagnostic Odds Ratio:  100.54285714285716
False Omission Rate:  0.028169014084507043
F1 Score:  0.9784172661870504
Area under ROC curve:  0.9785714285714286
```

Figure 3 : (i) Training and Validation Curves, (ii) Performance Metrics for Fully - Connected Layers

For Training accuracy/loss we used the ‘training data’ whereas for Validation accuracy/loss we used the ‘validation data’. Here, the validation set is only used to evaluate the model's performance.

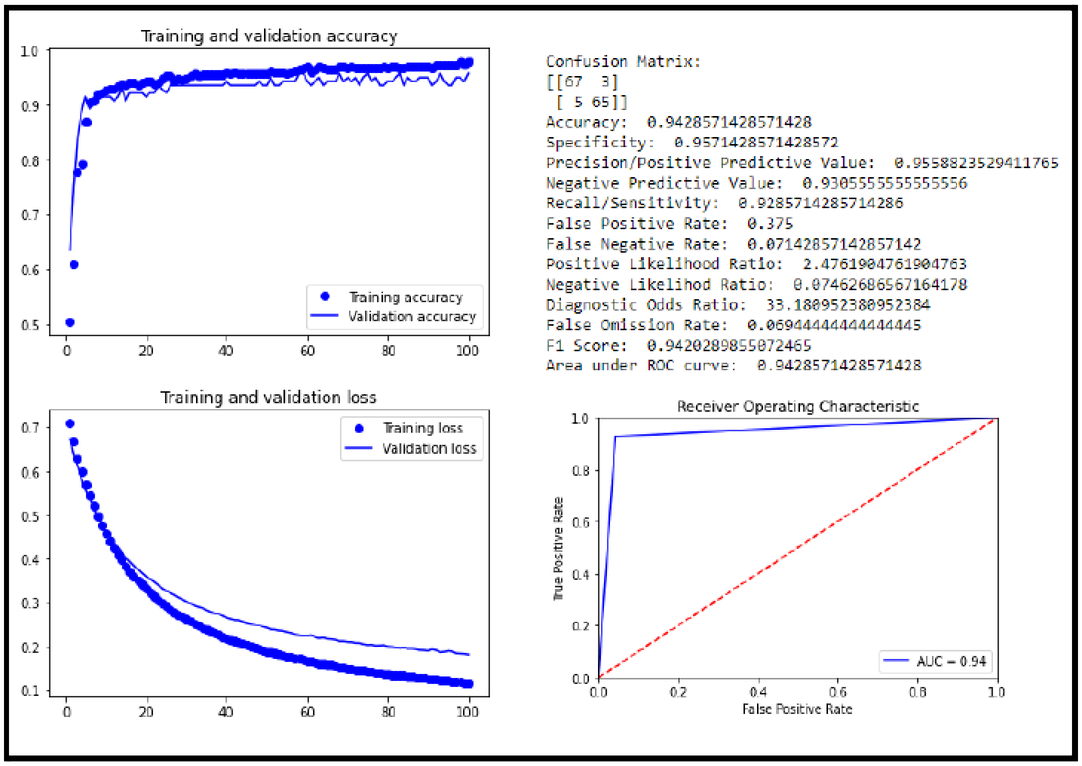
From the graph we can see that the Training accuracy is higher than the Validation accuracy whereas the Training loss is significantly lower as compared to Validation loss.

Also, we deduce that the testing accuracy of our classifier comes out to be 97.86%.

The plotted ROC curve also shows that our classifier is really accurate with area under curve close to 1, actually being 0.98.

Hence, with fully-connected layer classifiers we were able to successfully predict the images as pothole or plain with a high testing accuracy of **97.86%**.

Global Average Pooling



```
Confusion Matrix:
[[67  3]
 [ 5 65]]
Accuracy:  0.9428571428571428
Specificity:  0.9571428571428572
Precision/Positive Predictive Value:  0.9558823529411765
Negative Predictive Value:  0.9305555555555556
Recall/Sensitivity:  0.9285714285714286
False Positive Rate:  0.375
False Negative Rate:  0.07142857142857142
Positive Likelihood Ratio:  2.4761904761904763
Negative Likelihod Ratio:  0.07462686567164178
Diagnostic Odds Ratio:  33.180952380952384
False Omission Rate:  0.06944444444444445
F1 Score:  0.9420289855072465
Area under ROC curve:  0.9428571428571428
```

Figure 4 : (i) Training and Validation Curves, (ii) Performance Metrics for Global Average Pooling

For Training accuracy/loss we used the ‘training data’ whereas for Validation accuracy/loss we used the ‘validation data’. Here, the validation set is only used to evaluate the model's performance.

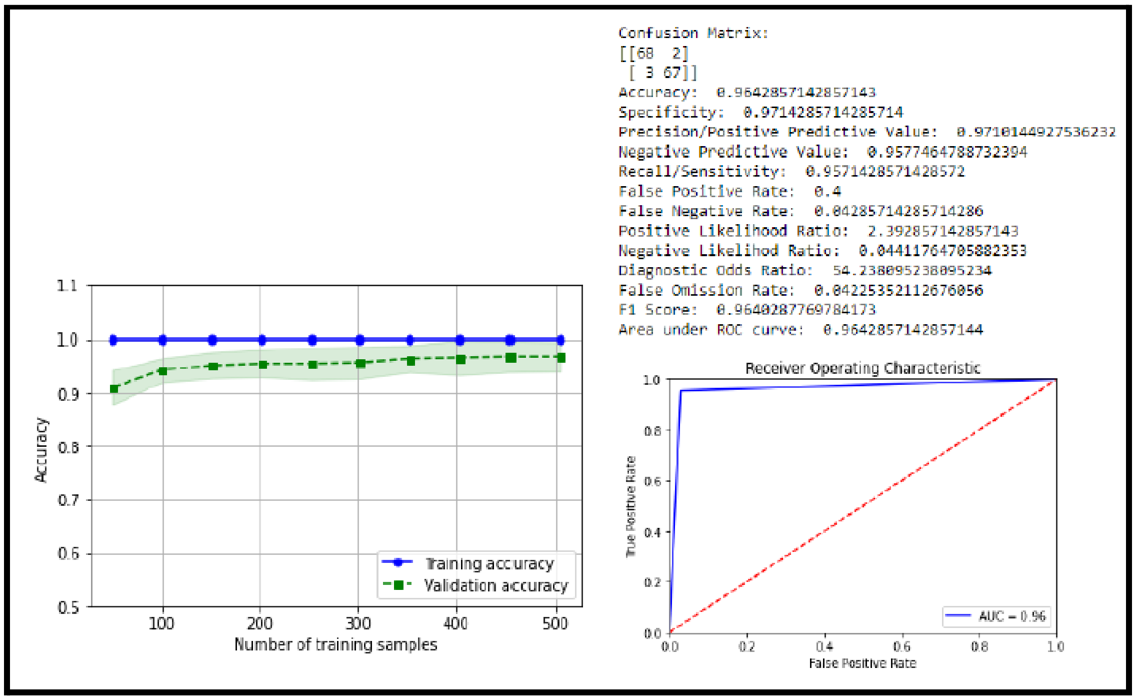
From the graph we can see that the Training accuracy is higher than the Validation accuracy whereas the Training loss is significantly lower as compared to Validation loss.

Also, we deduce that the testing accuracy of our classifier comes out to be 94.29%.

The plotted ROC curve also shows that our classifier is really accurate with area under curve close to 1, actually being 0.94.

Hence, with global average pooling classifiers we were able to successfully predict the images as pothole or plain with an testing accuracy of **94.29%**.

Linear Support Vector Machines



```
Confusion Matrix:
[[68  2]
 [ 3 67]]
Accuracy:  0.9642857142857143
Specificity:  0.9714285714285714
Precision/Positive Predictive Value:  0.9710144927536232
Negative Predictive Value:  0.9577464788732394
Recall/Sensitivity:  0.9571428571428572
False Positive Rate:  0.4
False Negative Rate:  0.04285714285714286
Positive Likelihood Ratio:  2.392857142857143
Negative Likelihod Ratio:  0.04411764705882353
Diagnostic Odds Ratio:  54.238095238095234
False Omission Rate:  0.04225352112676056
F1 Score:  0.9640287769784173
Area under ROC curve:  0.9642857142857144
```

Figure 5 : (i) Training and Validation Curves, (ii) Performance Metrics for Linear Support Vector Machines

For Training accuracy/loss we used the ‘training data’ whereas for Validation accuracy/loss we used the ‘validation data’. Here, the validation set is only used to evaluate the model's performance.

From the graph we can see that the Training accuracy is higher than the Validation accuracy.

Also, we deduce that the testing accuracy of our classifier comes out to be 96.43%.

The plotted ROC curve also shows that our classifier is really accurate with area under curve close to 1, actually being 0.96.

Hence, with linear support vector machine classifiers we were able to successfully predict the images as pothole or plain with a high testing accuracy of **96.43%**.

Learning Parameters:

	Fully Connected Layer Classifier	**Global Average Pooling Classifier**	**Linear Support Vector Machine**
Parameters	**6422784**	**513**	**C type parameter** `: [0.01, 0.1, 1, 10, 100]`

Table 1 : Parameters of different Classifiers

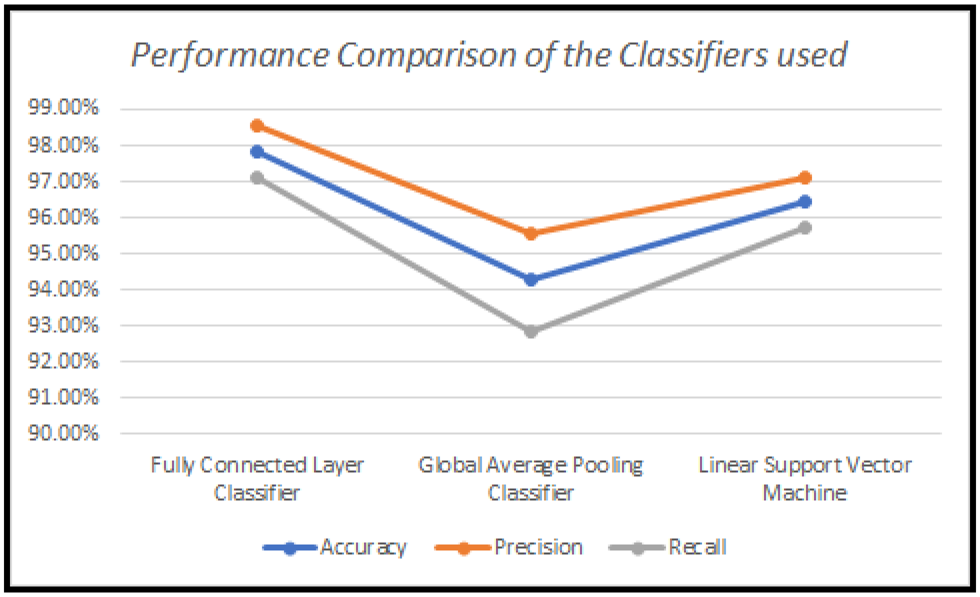
In the above table, we have mentioned the number of parameters used in the specified models.

Comparison of the Three Classifier:

	Fully Connected Layer Classifier	**Global Average Pooling Classifier**	**Linear Support Vector Machine**
Testing Accuracy	**97.86%**	**94.29%**	**96.43%**
Precision	**98.55%**	**95.58%**	**97.10%**
Recall	**97.14%**	**92.85%**	**95.71%**
Epochs	**100**	**100**	**10000**

Table 2 : Performance Comparison of the Classifiers used

The above table compares the different performance metrics of the different models we implemented. Also, the last row shows the number of epochs carried for each model.



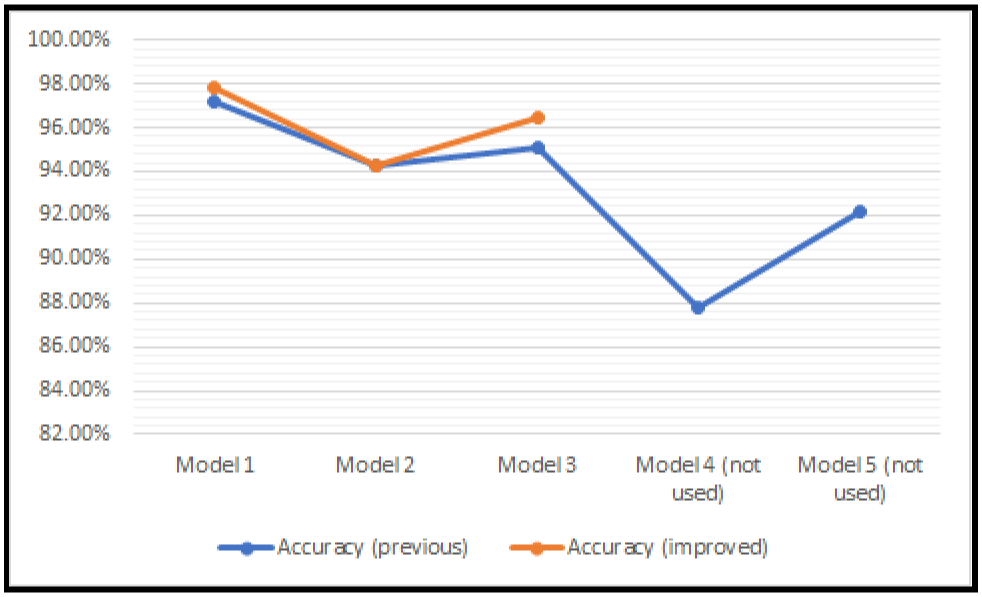
Graph 1 : Performance Comparison of the Classifiers used

Comparison of Models:

	Testing Accuracy (previous)	**Testing Accuracy (improved)**
Model 1 (Fully Connected Layers)	**97.2%**	**97.86%**
Model 2 (Global Average Pooling)	**94.3%**	**94.29%**
Model 3 (Linear SVM)	**95.1%**	**96.43%**

Table 3 : Testing Accuracy Comparison of the used models with existing models

In this table, we compare the new accuracy for each model that was implemented. We can see that there was a significant improvement in the testing accuracy of the models.



Graph 2 : Testing Accuracy Comparison of the used models with existing models

P.T.O.

Output from Trained Dataset:

- **Fully-Connected Layers**

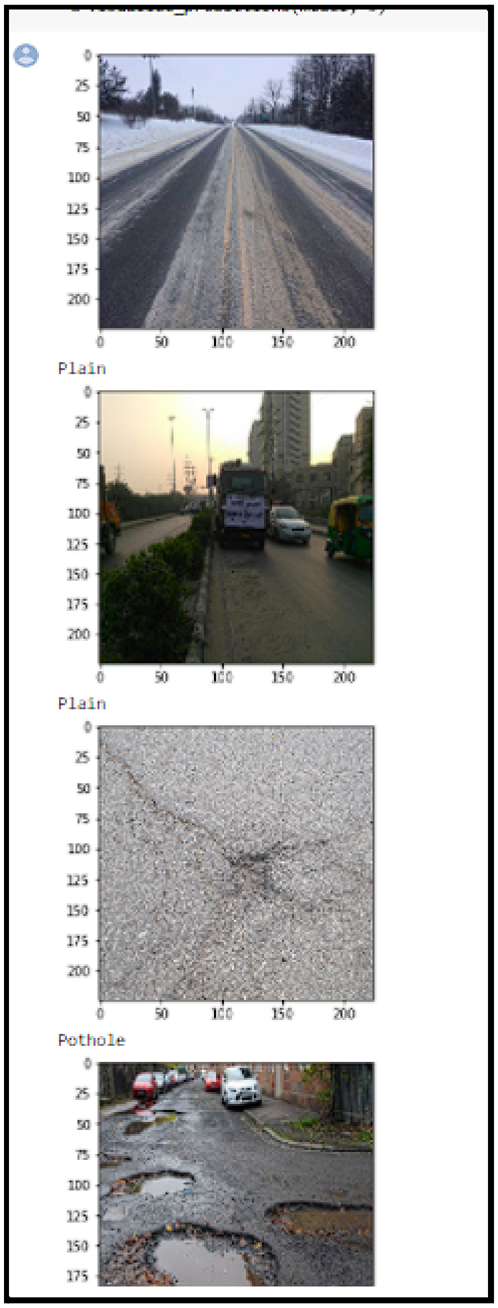


Figure 6 : Prediction outcome using Fully - Connected Layer

As predicted by the Fully Connected Layers classifier, the first two images were identified to be ‘plain’ and the third image is ‘pothole’.

- **Global Average Pooling**

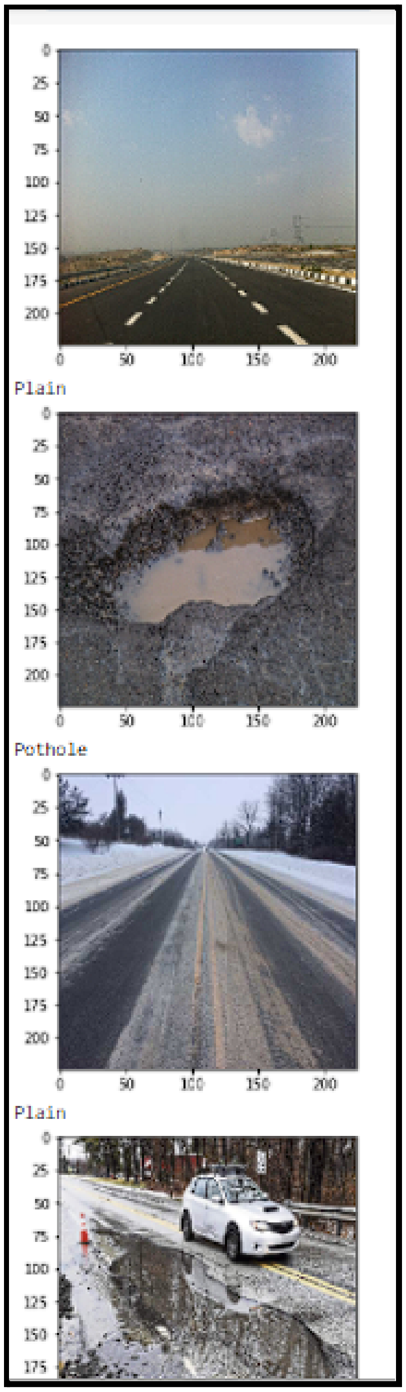
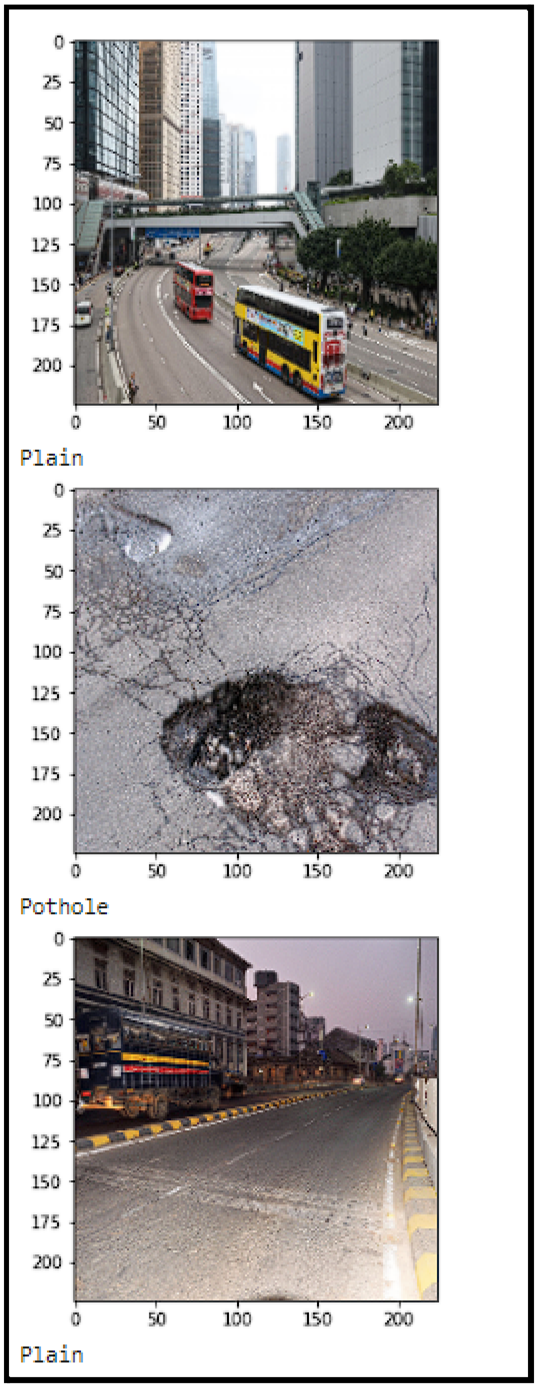


Figure 7 : Prediction outcome using Global Average Pooling

As predicted by the Global Average Pooling classifier, the first image is identified to be ‘plain’ and the next two images are ‘pothole’.

- **Linear Support Vector Machine**



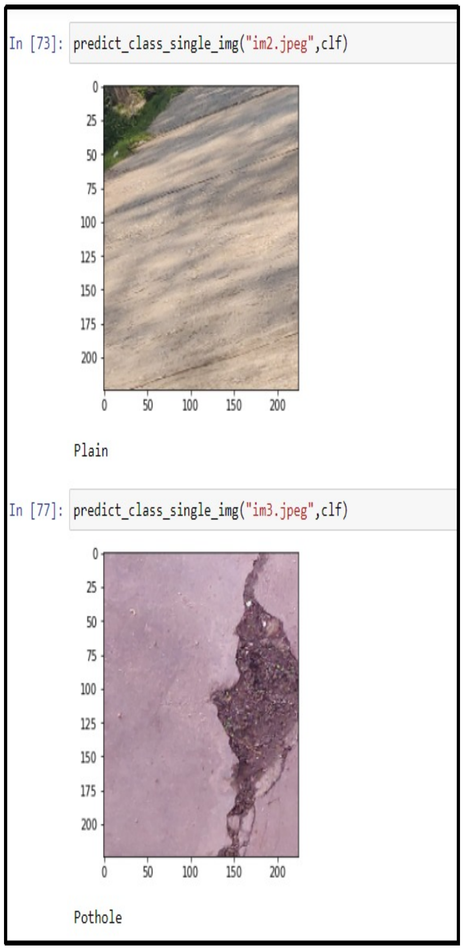
```
Plain
Pothole
Plain
```

Figure 8 : Prediction outcome using Linear SVM

As predicted by the Linear Support Vector Machine classifier, the first and last images were identified to be 'plain' and the second image is 'pothole'.

Output from User Input Images:

- Model used: Linear Support Vector Machine



```
In [73]: predict_class_single_img("im2.jpeg",clf)
```

```
Plain
```

```
In [77]: predict_class_single_img("im3.jpeg",clf)
```

```
Pothole
```

Figure 9 : Prediction outcome using User Input Images

Here we used our own images (clicked using our own camera) and tried to predict it using our own models and we found that the predictions made were correct. Thus, we can infer that the built models are correct.

7. Conclusion and Future Work

Conclusion

Thus, in the project we have:

- Reflected the idea of transfer learning, pre-trained models and neural networks
- Explained the basic fine-tuning approach to reuse a pre-trained model.
- Represented an organised approach to decide which fine-tuning approach is appropriate on the basis of similarity as well size of dataset.
- Provided three different classifiers that are used on top of the features extracted from the convolutional base.
- Presented insights about each of the three classifiers.
- Analysed each of the three classifiers.
- Drew ROC curve for each of the three classifiers.
- Reported the accuracy of the three classifiers and found out that Fully Connected Layer Classifier was best fit for our model with testing accuracy 97.86%.

Future Work

In the future, more practical results will be analyzed to provide the proposed method everywhere. Furthermore, due to a certain limit in the capacity of a battery of mobile phones, the issue about saving computation energy is found.

A new environment friendly approach namely green pothole detection approach is required to decrease the frequency of accelerometer data detection with higher accuracy for detection of potholes.

8. REFERENCES

[1] Wang, H.W., Chen, C.H., Cheng, D.Y., Lin, C.H. and Lo, C.C., 2015. A real-time pothole detection approach for intelligent transportation systems. *Mathematical Problems in Engineering*, *2015*.

[2] Dhiman, A. and Klette, R., 2019. Pothole detection using computer vision and learning. *IEEE Transactions on Intelligent Transportation Systems*, *21*(8), pp.3536-3550.

[3] Jo, Y. and Ryu, S., 2015. Pothole detection system using a black-box camera. *Sensors*, *15*(11), pp.29316-29331.

[4] Wu, C., Wang, Z., Hu, S., Lepine, J., Na, X., Ainalis, D. and Stettler, M., 2020. An automated machine-learning approach for road pothole detection using smartphone sensor data. *Sensors*, *20*(19), p.5564.

[5] Ryu, S.K., Kim, T. and Kim, Y.R., 2015. Image-based pothole detection system for ITS service and road management system. *Mathematical Problems in Engineering*, *2015*.

[6] Bansal, K., Mittal, K., Ahuja, G., Singh, A. and Gill, S.S., 2020. DeepBus: Machine learning based real time pothole detection system for smart transportation using IoT. *Internet Technology Letters*, *3*(3), p.e156.

[7] Garg, S., Mate, G.S., Das, R., Tiple, S. and Panicker, A., 2015. RIPD: Route Information and Pothole Detection. *International Journal of Advanced Research in Computer and Communication Engineering*, *4*(12), pp.194-197.

[8] Kulkarni, A., Mhalgi, N., Gurnani, S. and Giri, N., 2014. Pothole detection system using machine learning on Android. *International Journal of Emerging Technology and Advanced Engineering*, *4*(7), pp.360-364.

[9] Isah, A.D., Aibinu, A.M., Olaniyi, O.M. and Campbell, O.O., 2018. Development of asphalt paved road pothole detection system using modified

colour space approach. *Journal of Computer Science and Its Application*, *25*(2), pp.1-15.

[10] Ajitha, U., Sharma, S., Selvi, V.A. and Priyanka, A., 2018. Smart sensing of potholes and obstacles in automotive applications using the internet of things. *International Journal of Pure and Applied Mathematics*, *119*(15), pp.1479-1486.

APPENDIX

Colab Link:

https://colab.research.google.com/drive/1lpGAWEbX-I_J-nOGDjDZgtCF8oFChGAR?usp=sharing