A Comparative Study of Deep Learning Methods for Automating Road Condition Characterization

by

Zurana Mehrin Ruhi
20141049
Farahatul Aziz Sheetal
16101083
Farisha Hossain Prithu
16101259

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science

Department of Computer Science and Engineering
Brac University
April 2020

Declaration

It is hereby declared that

1. The thesis submitted is my/our own original work while completing degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

Student's Full Name & Signature:

Zurana Mehrin Ruhi
20141049

Farahatul Aziz Sheetal
16101083

Farisha Hossain Prithu
16101259

Approval

The thesis/project titled "A comparative study of Deep learning and Transfer learning methods for automating Road Condition Characterization in Bangladesh" submitted by

1. Zurana Mehrin Ruhi (20141049)
2. Farahatul Aziz Sheetal (16101083)
3. Farisha Hossain Prithu (16101259)

Of Spring, 2020 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science.

Examining Committee:

Supervisor:
(Member)

Hossain Arif
Assistant Professor
Department of Computer Science and Engineering
BRAC University

Program Coordinator:
(Member)

Md. Saiful Islam
Lecturer
Department of Computer Science and Engineering
BRAC University

Head of Department:
(Chair)

Mahbubul Alam Majumdar
Chairperson
Department of Computer Science and Engineering
Brac University

Abstract

Roads in Bangladesh provide infrastructural facilities to both agricultural as well as industrial sectors of the country. Distressed roads can cause fatal accidents as well as largely decelerate sector progress. This makes swift road inspection and repairs one of the most important aspects of our country's holistic growth. As much as it affects the general public, tackling this is as big a problem for the government as well. Currently, the problem for road repair is a multi-stage problem, which involves getting a complaint from a resident, physical road inspection by some official, identifying the type of damage and then comes the process of actually repairing it. Here, we intend to make this cumbersome process simpler, by automating the problem identification stage. We developed a method leveraging the Machine Learning and Deep Learning capabilities that can potentially detect a damaged road and identify the type of damage viz. pothole and crack. We self-captured data from the roads and streets, thus emulating the data we expect when this method is used in real-life by installing cameras on the city corporation's garbage trucks. We reviewed various models ranging from conventional machine learning to complex deep learning algorithms and ultimately shortlisted three models: CNN, CNN-XGboost, and ResNet. These three models were then optimized for our problem, and then extensive testing was performed to determine the one that outperforms the rest. ResNet-34 emerged as a clear winner, with an accuracy of 87.8 % on the test data. Here, we'll do an in-depth study of the efficacy of these models on our problem statement.

Keywords: CNN; Residual Network; Machine Learning; XGboost; Road Inspection; Potholes; Cracks

Acknowledgement

Firstly we would like to express our sincere gratitude to our advisor Hossain Arif sir and our co-advisor Md. Saiful Islam sir for their continuous support and guidance. his kind support and advice in our work. He helped us whenever we needed help. Finally to our parents without their throughout support it may not be possible. With their kind support and prayer, we are now on the verge of our graduation.

Table of Contents

List of Figures

List of Tables

Nomenclature

The next list describes several symbols & abbreviation that will be later used within the body of the document

ϵ	Epsilon
γ	Gamma
CNN	Convolutional Neural Network
ReLU	Rectified Linear Unit
SGD	Stochastic Gradient Descent

Chapter 1

Introduction

1.1 Overview

We wanted to create something that would contribute to the welfare of our country. The quality of the road infrastructure is crucial for people who drive. Potholes have been proven to cause catastrophes, especially during rainy seasons, so the drivers need to be cautious. Detecting potholes would highly contribute in order to minimize impact, make the ride smooth, and allow the vehicles to issue warnings to the drivers to slow down or avoid them. It would also contribute to updating the city corporation about potholes and cracks and help them to repair the damaged roads at the earliest using the data. Studies and research are still relatively few. Traditional approaches use sensors and expensive types of equipment with high computational costs and complexity of data along with manual inspection which highly demands on specialist's knowledge and experience, which leads to a labor-intensive, time-consuming process. For fast and reliable detection, automatic detection is expected to develop instead of a subjective and slower human inspection and sensor-based approach. There was no open-source data for our use case and specifically nothing that would fit the road structure of Bangladesh. So we collected and labeled the data on our own. Instead of making the system real-time, we thought of collecting video data by installing cameras on the city corporation's garbage trucks. This initiative would have a low installation cost and we will be able to collect more data as the trucks traverse around almost every corner of the city. In our study, we applied deep neural networks to extract information about the road and understand the environment surrounding the vehicle. The three models were made compatible with both image and video input.

1.2 Importance of Road Inspection

Z. kamal stated in his article [22] that the death toll due to road accidents in the last half of a single year was 2,297 while the number of injure people crossed five thousand. Senior Correspondent of BDNews24 [31] reported death of over seven thousand people due to road accidents across the country in 2018. Throughout the year over 5000 crashes were reported.In a country like Bangladesh monitoring road conditions is very important. Road maintenance and detection of road surface defects, such as cracks and potholes is a prolonged process, yet a very important part of development. The government needs accurate information to improve road

conditions and schedule maintenance at regular intervals. A driver finds it difficult to control the vehicles due to sudden potholes, bumps, and cracks. Thus, road image analysis is a very important aspect of the analysis of the road condition.

1.3 Motivation

Khandoker Maniruzzaman in his paper "Road Accidents in Bangladesh" [3] stated that the accident rate rose by 43% between 1982 and 200, however, the most alarming fact is that the accident fatalities increased by 400%. One of the contributing factors being road condition, as alarmed citizens it is our responsibility to help the government tackle this overwhelming problem. We should provide an outline to the government in order for them to take into account how bad the condition of the road in Bangladesh is right now and which roads need immediate repair.

1.4 Objective

Most of the roads in our country are not built maintaining any standard structure. The highways connecting the major divisions of the country are also poorly maintained and in dire need of repair. Most of the drivers often being inexperienced find it harder to drive in these damaged roads causing even fatal head-on collision.
We are conducting this study in order to address the underlying issue of these accidents. By automating the process of road inspection, our study will assist the government to effectively schedule routined maintenance of the major roads as well as identify the roads that need the firstmost attention. As no research has been done in our country regarding this field, we will collect data on our own to feed our machine learning models in order to accelerate the identification of damaged roads by classifying potholes, cracks and good roads.

1.5 Challenges faced

Firstly, there was no prior research done in this field, hence there was no readily available data or survey that would support our objective.
Secondly, the road infrastructure of our country does not follow the standard protocols.
Thirdly, the streets in the capital are so crowded that it is harder to find empty roads to collect images without any interference.

To tackle these challenges, we conducted a survey from all age group of people who travel on a regular basis to identify the majorly affected roads of Bangladesh.

1.6 Thesis outline

The aim of this paper was to construct a model capable of detecting potholes and cracks and determine is the road in a good or bad condition based on image processing. The aim of the authors was to formulate the best model for achieving the maximum accuracy and provide an accurate output. To begin with, in the first

chapter (Chapter 1), an overview of the system and its importance in Bangladesh's perspective are discussed. The Problem Statement was to develop and introduce a road inspection system in context to our country, thereby help avoid road accidents. Secondly, in (Chapter 2) related work in this field that identified cracks and potholes using various primitive methods are discussed along with their limitations. Outlining significant results achieved by researchers. As well as the lackings in the existing methods were overviewed.
Next, in (Chapter 3) background analysis of various supervised algorithms is discussed along with their implementation details. These algorithms are used in the research pipeline later on.
In (Chapter 4) the Research Methodology and workflows are proposed. Details about data collection, pre-processing, feature selection, project work-flow, and train and test data conducted in the research are elaborated.
Furthermore, in (Chapter 5) selected models are optimized, workflow overview and flow chart of the various models are provided along with the proposed model's overview (CNN, ResNet34, CNN-XGBoost) and training architecture details are elaborated.
In (Chapter 6) Experimental results and analysis are conducted. Comparison, confusion matrix, runtime evaluation, and final model evaluation are discussed. Visual comparisons of the models are provided. Accuracy metrics are tabulated.
Conclusions and further work were drawn in the last chapter (chapter 7)

Chapter 2

Related Work

Mednis et al. [6] proposed an automated system to detect potholes in real-time with little or no human interaction which will be based on Smartphones with Accelerometers and is implemented on the Android Operating System. They tried to avoid using expensive equipment by only using smartphones only. As they wanted little or no human interaction, they did not prefer collecting manual collections of pictures of roads with potholes. An automated survey approach carried out by smart-phones. Automated embedded sensing systems, including smartphones, have two general classes of sensors which are microphone and accelerometers to be used for pothole detection. The author focused on accelerometer data processing for pothole which is implemented on Android so that the system will be portable with less hardware complexity. The system running on a smart-phone should be able to detect while driving in different vehicles. Firstly using LynxNet collar devices, data from the accelerometer sensors were collected on an urban road. MansOS based software was used for the collection of raw acceleration data. Z THRESH which the simplest event detection algorithm was first tested on the data set. This algorithm identifies the types of potholes. After that, a bit advanced algorithm (Z-DIFF) was tested, which detected the fast changes in vertical acceleration data. Both the algorithm needs to know the Z-axis position. As the device is not using complex hardware resources standard deviation of vertical axis acceleration was used which was implemented in algorithm STDEV. For evaluating these algorithms the authors firstly made a ground rule for the test track using the Walking GPS approach and then run test drive sessions with 4 different smartphones (Samsung i5, Samsung Galaxy S, HTC Desire, HTC HD2). After using the event detection algorithms on collected data, statistical analysis was made in terms of previously marked ground truths which were performed using EGNOS. Moreover, tuning of the collected data with an appropriate threshold level for all algorithms was needed. Threshold values between 0.1g to 0.8g were used for tuning the Z-DIFF algorithm. For this value, 92% of all ground truth items were true positive 77% of all detected events were true hits. The test drive sessions detected irregularities 83-90% of potholes clusters. We can see 7% were not detected by any of the algorithms and 8% of the gapes escaped from the algorithms. Although they had an accuracy of 90% detection, the system is not enriched with self-calibration functionality. Moreover, using only smartphones limits the collection of data as it cannot handle huge amounts of data and it does not have such high computational power.
Madli et al.[18] The paper proposes a solution to track potholes and bumps of roads

within a low cost which gives timely alert to drivers about the presence of potholes and bumps. The author proposed a system that will identify potholes as well as bumps using ultrasonic sensors. It will not only calculate the depth and height of the potholes and bumps but also measure the geographical location coordinates of the irregularities of the roads using a GPS receiver. Capturing all the factors the system will alert the drivers so that they can take precautionary steps to avoid accidents. Peripheral Interface Control (PIC 16F887A) microcontroller is used in this system because it is cheap, available and it supports a high application. This microcontroller processes all the sensor inputs and alerts the drivers. Moreover, the HC-SR04 ultrasonic sensor is used which captures the distance using the reflection of sound waves. Again, the GPS Receiver uses a satellite navigation system to identify geographic position as well as time and weather conditions. In addition to that, GSM SIM 900 is used for mobile communication by which calls and text messages can be sent or received. Using all these components the systems divided it's architecture into 3 parts. Firstly, the microcontroller module is used to gather information about potholes and send the information to the server. Secondly, the server module acts like an intermediary layer microcontroller module and the mobile application processes and stores the information into a database by using an android device. Finally, a mobile application module is installed in the driver's android phone which gives timely alert to the drivers using the stored information in the database. They run an experimental setup in two phases, firstly they stored information about potholes in the server then generate alerts based on the information in the database. The microcontroller module was tested on a toy car with a threshold value of 5cm and worked as expected but the whole system is only based on sending text messages to drivers if there is any pothole or uneven bump. It did not employ any machine learning or image processing approach to solve the problem thus they were not able to show any percentage of accuracy level. Moreover, the information stored on a mobile server database will not be effective for a huge amount of data and if there is any network error the system will fail to alert the drivers in real-time.

Kawasaki et al. [23] The study approaches a system of shadow reduction to detect cracks with higher accuracy based on the percolation theory. Taking images from roads as input and then propose an algorithm for crack detection using a diagnostic analysis system so that cracks can be detected accurately and fast. To avoid manpower analysis in detecting and repairing cracks of road standardization and automation road diagnostic methods are required within low cost. In the study, at the first stage, they investigated the principles of cracks, potholes, and ruts. They found that repetitive construction, temperature and weather change, heavy load and vibration, increased traffic volume causes cracks where due to surface layer and underlying base layer weakness, sometimes tire pressure peel of the surface layer and creates potholes. On the other hand, passing through on the same point repeatedly causes ruts. So, to detect all these problems the defected road images are collected through stereo camera and areas are captured by a stereo machine and U-V disparity. The detected region is subjected to image processing. A pothole is detected by the inertial measurement unit (IMU) where a hole is identified by the data obtained from laser range finder (LRF) and then GPS mapping is done. The input images contain shadows of trees and other objects and due to camera conditions, a photo was taken time and weather also affect the image. All these reasons lead to the misdetection of cracks. For that, to detect cracks more accurately

firstly linear transformation is applied. Secondly, closing processing (morphological processing) is applied to eliminating depletion. Thirdly, a smoothing filter named Gaussian filter is used for removing noise by using Gaussian function. The next step is shadow removal where a Gaussian filter is applied. Here, a boundary is set between the shadow area and the non-shadow area. After that, the discriminant analysis is used to binarize images which calculate threshold automatically. After that, an anisotropic smoothing filter's equation is used to smooth the detected areas. Finally, the percolation theory is applied in 4 steps to image processing. The comparison of coincidence between the output images and ground truth images is used to find the crack detection rate. All the equations used to remove noise and shadow for detecting crack more accurately. If no noise is considered then the detection rate is 47.9% which is very low then the threshold value is changed and the problem is solved but the author said that still, noise appears which is again a problem.
Ajit et al.[7] Indian rural and suburban roads have faded lanes, irregular potholes, improper and invisible road signs, etc which causes many accidents. To unravel this acute problem, the study is undertaken with the objectives to create a survey of Indian roads, to suggest the tactic to detect lanes, potholes and road sign, and their classification and to suggest automated driver guidance mechanisms. During this case, Color Segmentation and Shape Modeling with Thin Spline Transformation (TPS) is used with the nearest neighbor classifier for road sign detection and Classification. Further, K-means Clustering-based algorithm is adopted for pothole detection. Road Image analysis is an extremely important aspect of automated driver networks. There are phases of algorithms to solve the issues. Firstly, ROI Segmentation with Image thresholding is employed to detect sign-color information. Then, Thinning and Edge Detection, Identifying the Region and Clustering, Thin Plate Spline (TPS) and Recognition, Path hole detection and lane detection thorough Hough transformation (HT) maps. Real-Time Road Images with real traffic conditions present many challenges for that image processing and analysis is required.
Jin at el. [5] proposes a histogram-based texture measure to extract the features of an image and identify potholes using a nonlinear support vector machine (SVM). Longitudinal cracks, transversal cracks, alligator cracks, and pothole are basically addressed as pavement defects. The pavement defect detection gives important information about the road network conditions. This defect detection system is a combination of acquisition and pre-processing of images, detection, and classification of defects and doing necessary measurements. The defect classification is based on image segmentation. Wavelet transform-based road crack extraction algorithm was proposed by the authors which extract linear features effectively. Image segmentation using the Otsu method, collected information of a selected area, contour tracking principle to calculate the perimeter of the defected region, helped to obtain the crack region. To discriminate against the defected targets, grey variance-based projection and correlation coefficients were used by the authors. The author of this paper mainly focuses on identifying potholes and cracks and distinguishing between them using partial differential equations (PDE) models. This method uses a nonlinear support vector machine to identify whether the area is a pothole or not. Image segmentation method based on partial differential equations is important because it uses the local boundary information of the image, image region statistics and characteristics to get closed one-pixel width edges to get satisfactory results that could

not be obtained from traditional boundary-based detection method.
Observing road images with cracks and potholes, it is found that they have different granular sections from other normal road images. So, the method targets to detect the difference in the granular section and then find out potholes. The authors used 80 pieces of sample images for the experiment where 50 images were for training and the remaining 30 images were for testing. The 50 sample images were transformed into grey-scale images. After that, for each image Eigenvalues were calculated and using image texture features including the average greyscale, contrast, 3-order moments, consistency and entropy as an eigenvector for the target region. After that, the values were normalized by SVM and then the model recognized 30 test images correctly. Despite all these, the model will not be effective for some complicated cases such as if the pothole is covered with mud or dust then the depression value becomes relatively flat with no grainy section. So, in case of the defect classes, the training model cannot correctly identify them and thus the number of training samples should increase for correct reorganization.

Chapter 3

Background Analysis

3.1 Neural Networks

The neural network forms the base of deep learning, a subfield of machine learning where the algorithms are inspired by human brain's structure. Neural networks take in data to train themselves to recognize the patterns in this data and predict the outputs for a new set of similar data. They decipher tangible information through a sort of machine recognition, marking or grouping crude info. The examples they perceive can be numerical, contained in vectors, into which all certifiable information, can be pictures, sound, content or time arrangement. Neural systems are comprised of layers of neurons.[14] The neuron is the core processing unit of the network. The first layer which receives the input is called the input layer and the output layer predicts the final output. The layers between this input and output layers are the hidden layers that perform the most the computations required by the network.

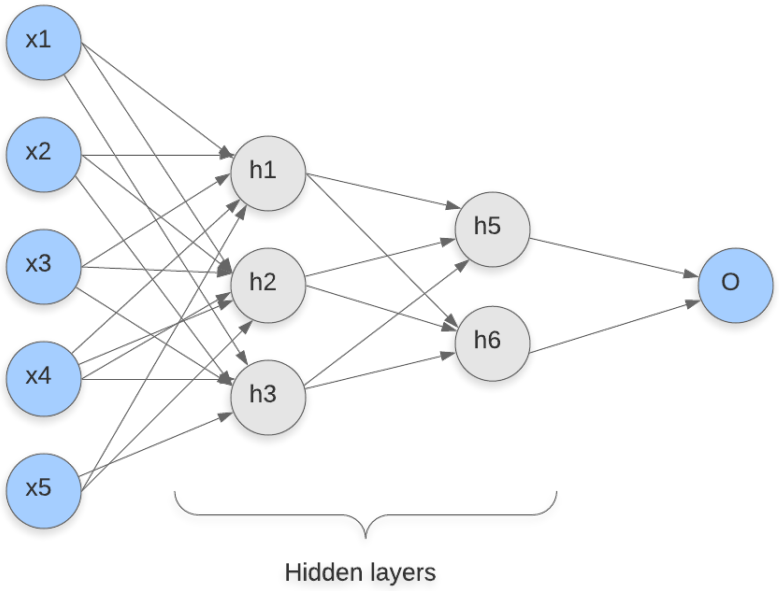

Figure 3.1: Two layer Neural Network Model

The neuron of one layer is connected to the next layer's neuron through channels having a weight value. A node combines input from the information with a hard and fast of coefficients, or weights, that either expand or hose down that input, thereby assigning importance to inputs in regards to the undertaking the algorithm is trying to learn; for example, identifying input with less error. These enter-weight products are summed after which the sum is passed through a node's activation function, to decide whether and to what quantity that signal should progress similarly through the network to affect the ultimate outcome. If the alerts pass via, the neuron has been activated. A node layer is a column of those neuron-like switches that switch on or off as the information is taken care of through the net. Each layer's output is simultaneously the following layer's enter, starting from a preliminary input layer receiving statistics. Pairing the model's adjustable weights with input features is how we assign significance to those capabilities in regards to how the neural community classifies and clusters enter.

3.2 Convolutionl Neural Network

The first convolutional neural system was proposed by Hubel and Wiesel[1] during the 1960s through investigations of neurons in monkey cortexes identified with neighborhood affectability and course determination. The concept of the convolutional neural network is the result of the well-known algorithms of artificial neural networks plus a set of operations that we will call convolution by combining these two kinds of ideas here we get the convolutional neural network or simply CNN. Neural networks are composed of artificial neurons that simulate biological neurons in a limited way. CNN is a Deep Learning calculation that can take in an info picture, allocate significance such as weight and bias to different viewpoints in the picture and have the option to separate one from the other. CNN's are primarily made out of a convolutional layer, a pooled layer, and a fully connected layer. The first layer of CNN is composed of one or more layers of convolution after this we can apply one or more steps of pooling after that we design what we call a fully connected layer. Firstly, the key convolution layer whose main work is to extricate highlights from input pictures or include maps. Each convolutional layer can have different convolution kernels, which is utilized to get numerous feature maps. The convolution layer's calculation is explained in the next page.

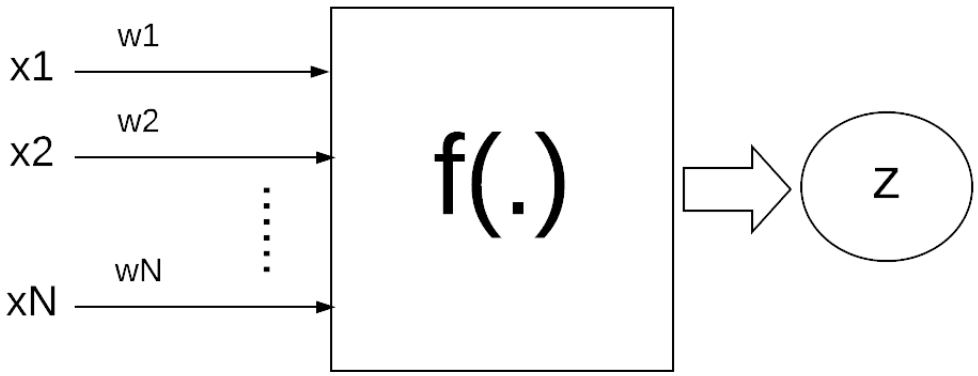

Figure 3.2: Convolutional layer calculation

Considering the figure 3.2, we have a set of elements that are represented by a set of inputs x_1, x_2, x_3 up to x_N which are connected to activation function f but the connection between input and the activation function is drawn by a set of weight which is represented by Ω_1, Ω_2 up to Ω_N. Moreover we have a bias which is represented by b and the output of the activation function is Z.

$$z = f(\sum_{i=1}^{N} x_i w_i + b) \tag{3.1}$$

Where, $X = (x_1, x_2, x_3.....,x_N)$ and $\Omega = (w_1, w_2,w_3.....,w_N)$

z is output of the function applied to the input weighted by all the weights that are inside Ω after adding the bias. So here we connect some input and have a single output.
After a convolution layer, once we get the feature maps, we generally include a pooling or a sub-inspecting layer in CNN layers. Like the convolutional Layer, the pooling layer is liable for diminishing the spatial size of the convolved feature map. This is to diminish the computational requirement to process the information through dimensionality decrease. Besides, it is required so as to make our model rotational and positional invariant, in this way keeping up the procedure of viably preparing the model. Pooling reduced the training time and complexity, while reducing the over-fitting. There are two types of pooling. One is max pooling and another is average pooling.

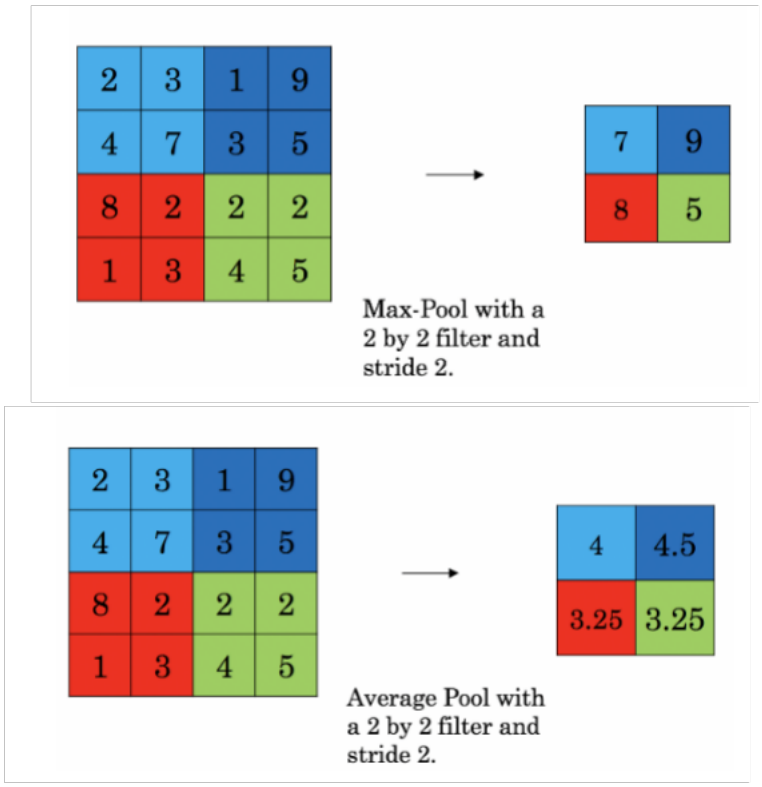

Figure 3.3: Maxpooling and Average pooling

Max Pooling restores the highest value from the part of the picture secured by the Kernel and it also acts as a noise suppressant and preserves the texture of the image well. Whereas, the average pooling restores the mean value from the part of the picture secured by the Kernel. Basically it simply performs dimensionality reduction as a noise suppressing mechanism.

After the convolution and pooling layer, the final output is flattened and feed it to a regular Neural Network for classification purposes. The capacity of the fully connected layer is to incorporate the numerous picture maps got after the picture is gone through various convolution layers and pooling layers to acquire the high-layer semantic highlights of the picture for consequent image classification. After converting the input image into a suitable form, the image is flattened into a column vector. After that, the flattened output is taken care of to a feed-forward neural system and backpropagation applied to each training iteration. After a number of epochs, the model can recognize ruling and low-level features in pictures and arrange them utilizing the Softmax Classification system.

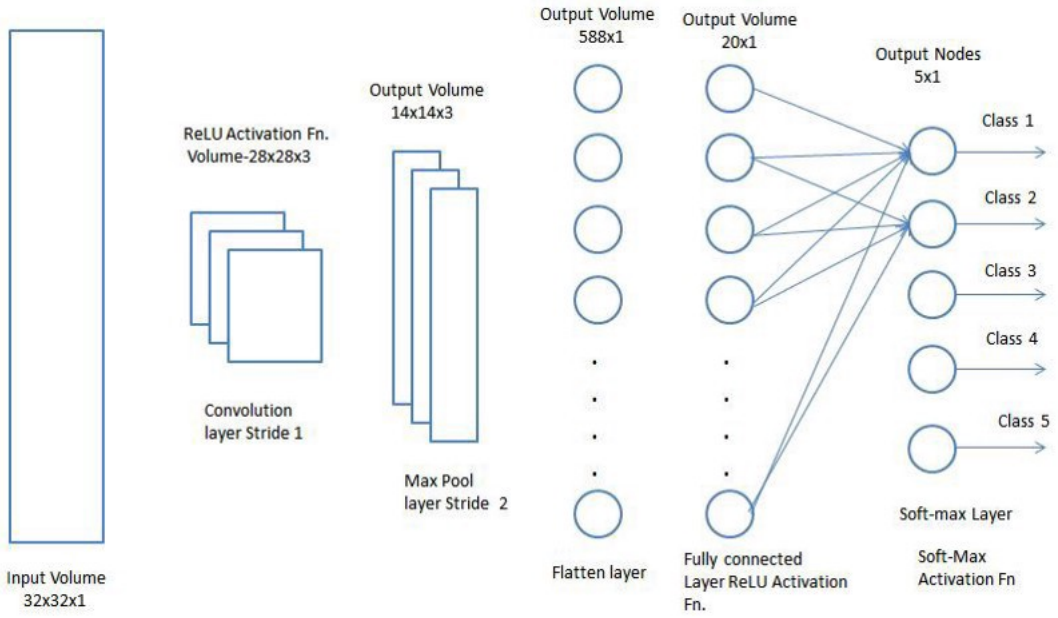

Figure 3.4: CNN model overview

3.3 Residual Network

ResNet, short for Residual Networks is a classic neural network used for an image classification task. For solving the problem in training very deep networks, residual neural networks were introduced.ResNets solve is the vanishing gradient problem for a deep network. Using ResNet[24], the gradients can flow directly through the skip connections backward from later layers to initial filters.

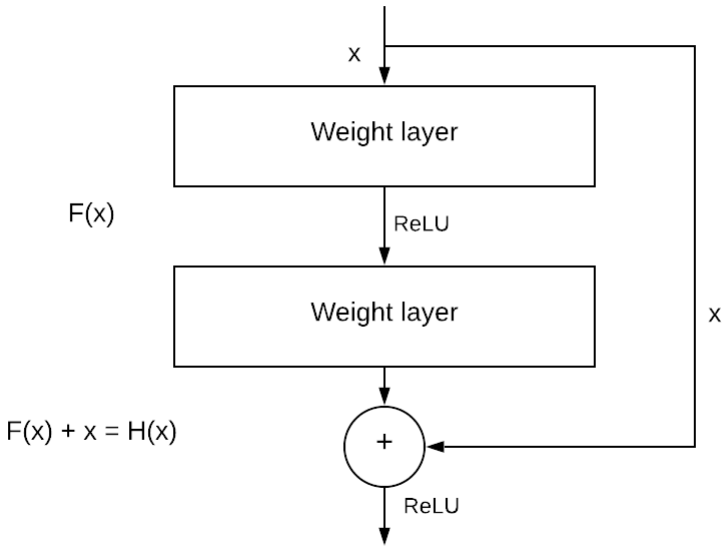

Figure 3.5: ResNet block

Residual functions formulate the layers having a reference to the input through identity or 'skip connection' such that theoretically if it required to push the layer

down to zero it could easily do it. This identity mapping having no parameters, adds the output from the previous layer to the next layer. Sometimes, the input function (x) and the output function (F(x)) will not have an identical dimension. The operation of convolution shrinks down the spatial resolution of an image. For example, a 3x3 convolution on a 32x32 image leads a 30x30 image. Moreover, the identity mapping is multiplied by a linear projection in order to expand the channels of shortcuts to match the residual. We can use these identity shortcuts (x) directly and for changing dimensions, x performs identity mapping with extra zero entries padded with the increased dimension and then to match the dimension projection shortcut is used.

3.4 Extreme Gradient Boosting

XGBoost stands for eXtreme Gradient Boosting. XGBoost is an enhanced dispersed gradient boosting library intended to be exceptionally proficient, adaptable and versatile. It actualizes machine learning calculations under the Gradient Boosting structure. It is an implementation of gradient boosted based on decision trees designed for better speed and performance [16]. The following figure below shows the evaluation of XgBoost from tree-based algorithms over the years.

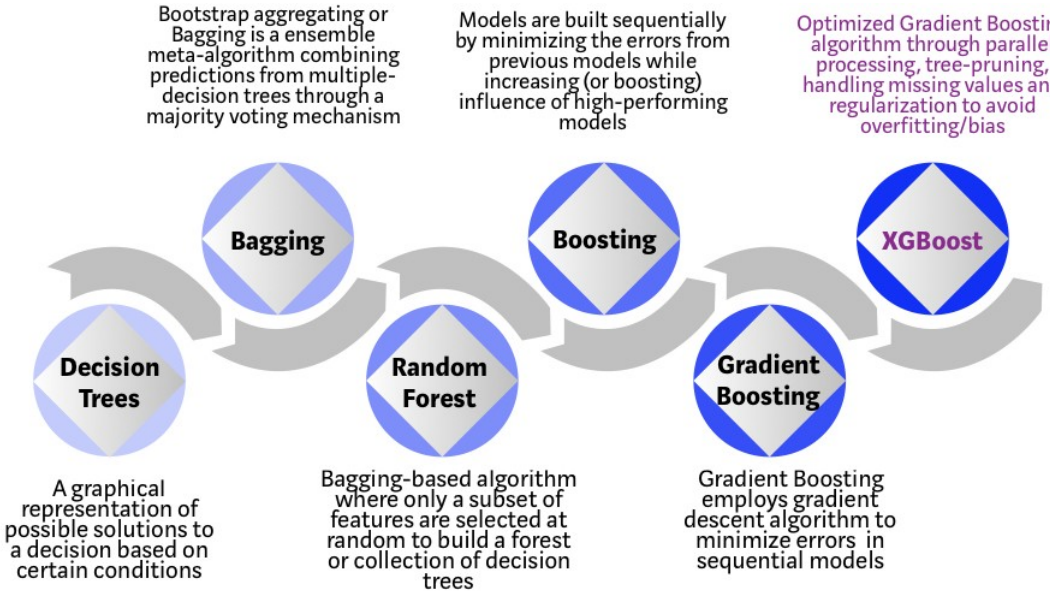

Figure 3.6: Evaluation of XGboost

The main two reasons for using XGBoost are to improve the execution speed and model Performance[10] especially engineered to exploit each bit of memory and resources of hardware. Artificial neural networks are known to perform better for images, videos but XgBoost is the king when it comes to small or medium structure data and it performs really well in classification, regression, ranking, user-defined prediction problems e.t.c. XgBoost algorithm was first developed as a research project at the University of Washington. It portable because it runs smoothly on Windows, Linux and OS X. It supports all major programming languages including C++, Python, Java, Scala, Julia e.t.c. Moreover, it supports cloud integration such as AWS, Azure, and Yarn Clusters. XG Boost targets to give a scalable, portable

and Distributed Gradient Boosting Library. There are some features such as clever penalization of trees, a proportional shrinking of leaf nodes, Newton Boosting, extra randomization parameter makes it different from other gradient boosting algorithms[2]. Tianqi Chen[19], the developer of XGBoost, believes this algorithm makes use of a greater regularized model formalization for you to manipulate over-fitting, leading to higher performance.

Chapter 4

Model Implementation and Optimization

4.1 Overview

Our primary focus was to build deep learning models while exploring the arena of convolutional and residual learning. So we focused on two neural network models: CNN and Resnet34. This section provides an overview of how we approached the designing and setup of the models' architecture to optimize and find the best match for our problem statement.

- The dataset is divided into training and test set with a split ratio of 80:20. This test set is not used anywhere in the training and kept out to perform the true evaluation for various models.
- The training set is further split into training and validation set, with an 80:20 ratio again. This brings our train, validation and test ratio as 64:16:20.
- Various models are developed and trained over this dataset. Model hyperparameters are tuned using the validation dataset to get the best results from that particular model.
- Finally all the models are evaluated and juxtaposed for analysis. Comparative results are obtained for all the models, and the best model is identified among those.

The following page includes a flowchart of our workflow:

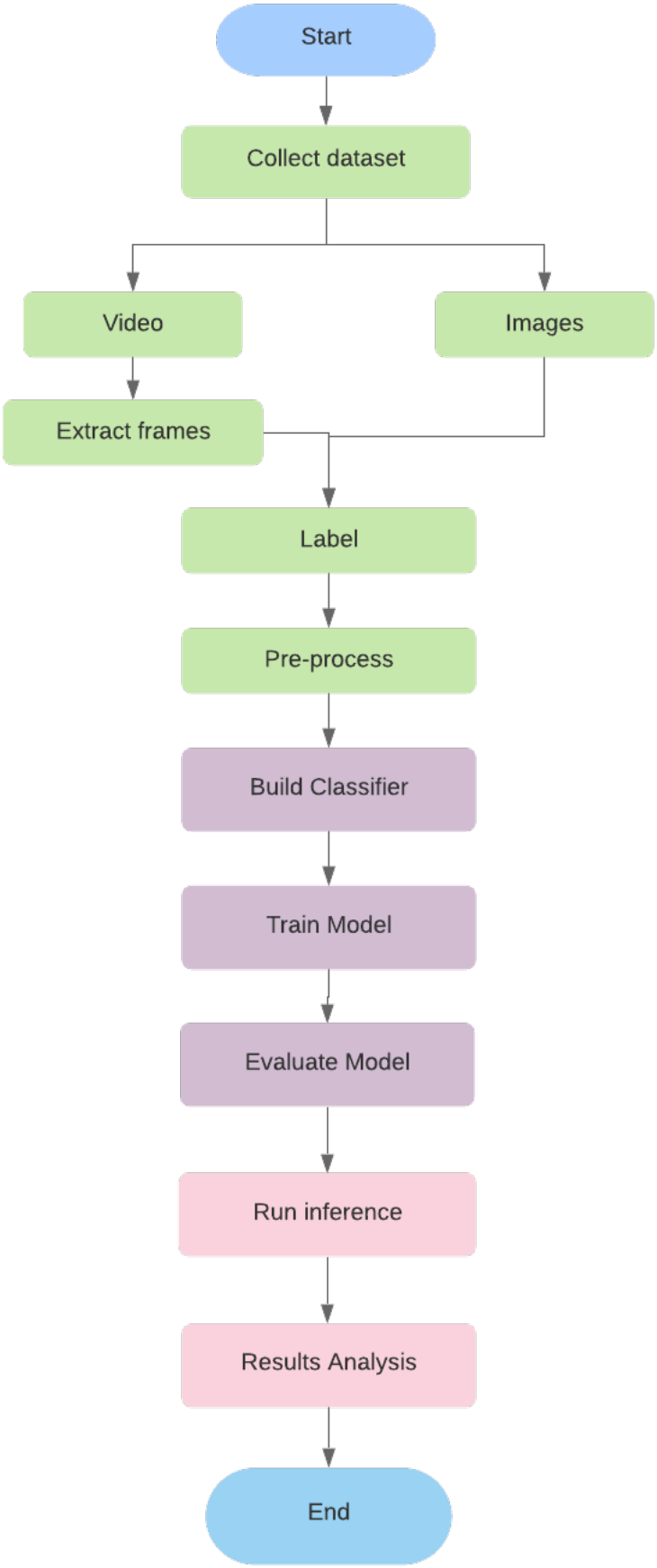

Figure 4.1: Work Flow

4.2 Dataset collection

There was no open-source data for our use case and specifically nothing that would fit the road structure of Bangladesh. We needed a dataset with images that contain all scenarios posing a challenge to our problem statement. The dataset must cover variations such as: size of potholes, width and length of cracks, brightness and so on while taking all corner cases into account. The manual method of taking pictures and videos seemed the best method to ensure variety and also economical feasibility for our study. The main components of the data collection process were:

- Going to various locations for taking pictures at different times, light and weather conditions
- Conducting surveys to find out which local roads are mostly damaged
- Using google street view to locate the most damaged parts in the highways

4.3 Pre-processing

First step of pre-processing was to extract frames from the videos that were recorded at 30fps. We extracted 2 frames per second to reduce redundancy in our training data by removing similar images, thus preventing our deep learning network model from overfitting. All images were then processed using PIL (Python image library) and resized to 128x128, thus minimizing the information loss from the image. Since we will be training our models in the RGB space, we also converted single-channel also known as grayscale images into 3 channeled (RGB) images by placing the same channel for all the three channels for the resultant RGB image.

Figure 4.2: Processed Images

In order to further diversify our data synthetically, we have also applied augmentation techniques which will be broadly discussed in (section 4.4.2). Then we labeled all images with relevant tags (Potholes, cracks and good roads) manually by making observations and judgments.

4.4 Design and training

CNNs and ResNets are well-known models that are previously used many times to classify images. Comparatively, integration of gradient boosting[25] with neural network models are new to the field and require more research. As our dataset contains a lot of variation, it provides a lot of space to learn. Hence, we have incorporated XGBoost model with our CNN feature extractor to expand the scope of our study. The following sections provide implementation and architectural details of these three models respectively.

4.4.1 CNN

CNN (Convolution Neural Network) model includes several layers like the input layer, convolution layers, pooling layers, non-linear activation layer and others. There are many standard architectures proposed that help solving complex classification problems. One of those is a VGG16 architecture that inspires our model.

Architecture

VGG architecture was first introduced by K. Simonyan and A. Zisserman in their paper "Very Deep Convolutional Networks for Large-Scale Image Recognition" [13]. The substitution of large kernel-size filters with consecutive 3x3 ones improved the performance of the model significantly. A traditional VGG16 architecture has 16 layers consisting of the above-mentioned layers. However, our developed architecture is a modified version of VGG16, such that it fits to our needs.
The architecture for our proposed model is given below in figure 4.3:

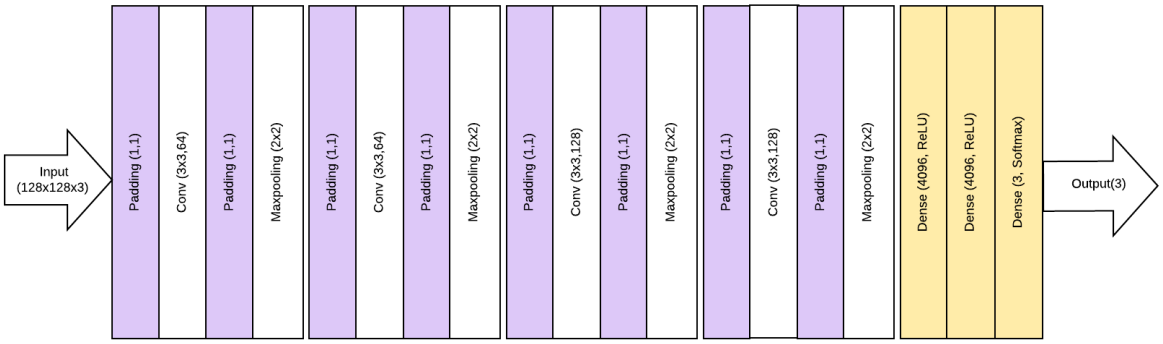

Figure 4.3: VGG16 inspired architecture

The input layer is fed 128x128 resized RGB images and 1x1 sized paddings are added. These two things are done to avoid overhead calculations while at the same time retain more information. Zero paddings add zero pixels (black frame) around the images that allow the kernel to cover more space. The network comprises of two

primary modules:

Feature Extraction: This module has 4 convolutional layer blocks with Rectified Linear Unit (ReLU) as the activation function. The convolutional layer performs the necessary calculations by determining the extraction of specific patterns amongst the images. We used a 3x3 kernel-size filter as it was the appropriate for our input data size. As for the number of filters, they are set in the power of 2. We tried out different combinations to find out the suitable numbers and ultimately set it as 64 and 128 accordingly. Given an input vector of dxd, padding size p, filter size k, and stride s, the output size of each convolution layer is determined as:

$$S = \frac{(d - k + 2p)}{s} + 1 \tag{4.1}$$

As we have set s=1, p = 1 and filter size 3, the equation for our model becomes:

$$S = d \tag{4.2}$$

The reason for using the kernel size of 3x3 is that it allows the convolutional layers to generate a feature map while using only minimal parameters. The values of the feature map are then calculated using the following formula where m and n denote the rows and column of the resultant matrix:

$$Conv[m, n] = (d * k)[m, n] = \sum_i \sum_j k[i, j]d[m - i, n - j] \tag{4.3}$$

This map is then passed through a non-linear activation function named ReLU that transforms all the negative inputs to zero. This function allows the neurons to be activated sparsely[28]:

$$ReLU(x) = max(x, 0) \tag{4.4}$$

The initial convolutional layers identify the simpler features, as we go into the deeper layers, more complex features are extracted. Maxpooling is applied to these feature maps to reduce overfitting and avoid slow convergence on validation data. The output size of this layer is given by S', Where pool size is denoted as P:

$$S' = \frac{(d - P) + 2p}{s} + 1 \tag{4.5}$$

In our case, Pool size is 2, stride s =2 and padding is 1. So the formula becomes:

$$S' = \frac{d}{2} + 1 \tag{4.6}$$

These pooling layers downsize the samples by reducing the parameters making it less complex. We have used maxpooling layers with window size 2 and default stride to make the model more efficient.

Classification module: The output of the final maxpooling layer is first flattened resulting in a one-dimensional array of length 10368, that data is then passed onto the fully connected layers. These Dense (fully connected) layers are stacked together with the last layer using Softmax activation function. The main goal of these layers is to use the extracted high-level features to classify the inputs into defined labels.

Here, within the final layer, there are three labels: Potholes, Cracks and Good roads. Softmax function is used in this layer specifically to get probabilistic values for each label to classify the input images into the above-mentioned classes. The function is defined as:

$$Softmax(x)_i = \frac{exp(x_i)}{\sum_j exp(x_j))} \tag{4.7}$$

where $exp(x_i)$ is the total weighted sum of inputs for each output unit j.
Below is a summary of our architecture:

Layer	Layer content	Tensor Size
0	input Image	3x128x128
1	zeropadding(1x1)	3x130x130
2	conv(3x3, 64 maps)	64x128x128
3	zeropadding(1x1)	64x130x130
4	maxpooling(2x2, stride 1)	64x65x65
5	zeropadding(1x1)	64x67x67
6	conv(3x3, 64 maps)	64x65x65
7	zeropadding(1x1)	64x67x67
8	maxpooling(2x2, stride 1)	64x33x33
9	zeropadding(1x1)	128x35x35
10	conv(3x3, 64 maps)	128x33x33
11	zeropadding(1x1)	128x35x35
12	maxpooling(2x2, stride 1)	64x17x17
13	zeropadding(1x1)	128x19x19
14	conv(3x3, 64 maps)	128x17x17
15	zeropadding(1x1)	128x19x19
16	maxpooling(2x2, stride 1)	128x9x9
17	flatten	10368x1x1
18	dense	4096x1x1
19	dense	4096x1x1
20	dense+softmax	3x1x1

Table 4.1: CNN architecture summary

Training

Preliminary training was performed on a smaller dataset with just 1315 images and later, more data was captured that increased the dataset to 2762 images. The validation accuracy improved by 22.84% in doing so. The comparison between two datasets and is given below:

	Old Dataset (1315)	New Dataset (2762)
Number of epochs	100	100
Training accuracy	91.6%	97.28%
Validation accuracy	63.89%	78.48%

Table 4.2: CNN training results

The training process comprises of two parts: Forward pass and backward pass. Initially, random weights are allotted to all the trainable parameters (weights and biases). During the forward pass, the training input data is fed to the network and predictions (label probabilities for each class) are computed. Using these predictions and the ground truths, we use categorical cross-entropy cost function to compute the model's loss. The loss function is defined as:

$$J(W) = -\frac{1}{N}\sum_{i=1}^{N}[y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)] \tag{4.8}$$

Then backward pass takes the computed loss into account and updates all the parameters with the right proportion so the model learns to make correct predictions i.e. the predicted output comes closer to the actual output. We start with a learning rate of 0.001, which is continuously optimized during the run using Adam optimizer[11]. Adam incorporates two other well-known optimizers RMSprop and SGD with momentum [9], making this an adaptive learning rate method. It incorporated momentum and moving averages($\hat{m}_t$,$\hat{v}_t$) while updating weights denoted with w_t using the following function:

$$w_t = w_t - 1\eta\frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} \tag{4.9}$$

Here, η is the learning rate which is initially set to 0.001.

Below is the list of hyperparameters that have been optimized to achieve the highest performance.

Parameter	Description	Value
K	Number of the kernels form convolution layer	64,128
f	Filter Size of convolution layer	(3,3)(3,3)
s	Stride size of convolution layer	Default
P	Pool size of convolution layer	(1,1)(1,1)
J(W)	Loss function	Categorical Cross-entropy
ReLU(x)	Activation function	ReLU
Softmax(x)	Activation function	Softmax
η	Learning Rate	0.001
b	Batch size	128

Table 4.3: CNN hyperparameters details

The following page contains our training accuracy curve:

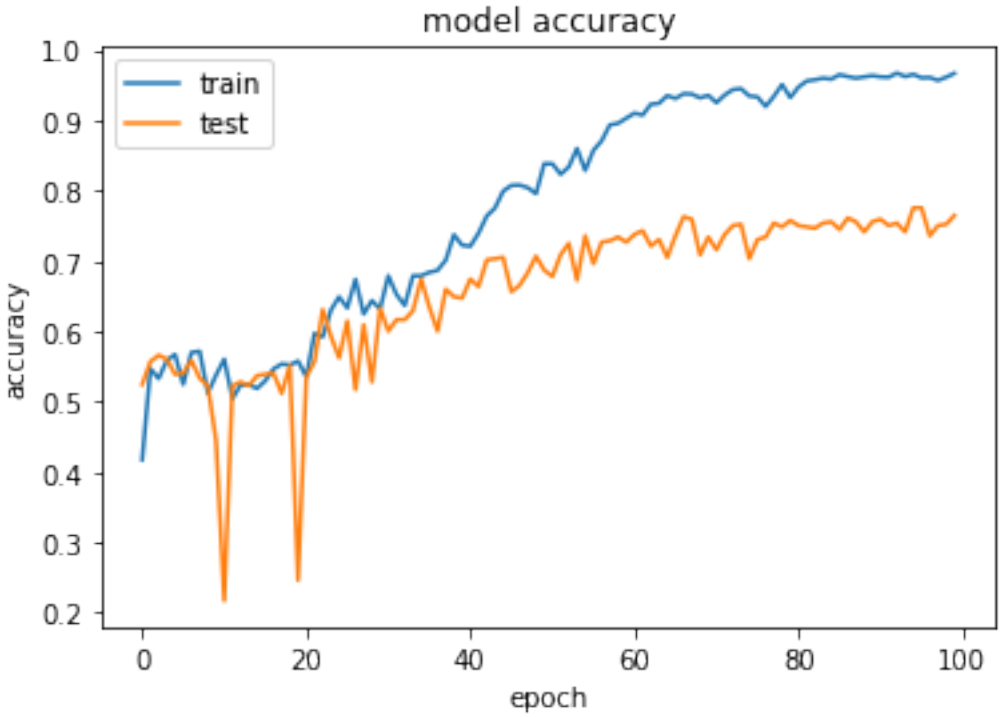

Figure 4.4: Epoch vs Accuracy curve for CNN training

4.4.2 ResNet34

ResNet34 [20] introduces a solution to the performance degradation problem of very deep networks due to accuracy saturation [10]. Very deep networks often face the problem of vanishing gradient, thus loosing substantial valuable information for deeper layer. ResNet successfully deals with this problem, by using skip connections to pass the information from past layers to the deeper layers. As per the building block discussed in (chapter 3.3) , we followed the below equation for our model:

$$Y = \mathcal{F}(x, \{W_i\}) + x \tag{4.10}$$

Where $\mathcal{F}(x, \{W_i\})$ is the residual mapping that we have learned. Generally, a residual function $\mathcal{F}(x)$ is defined as following, denoting the underlying mapping as $\mathcal{H}(x)$:

$$\mathcal{F}(x) = \mathcal{H}(x) - x \tag{4.11}$$

The addition operation between $\mathcal{F}$ and x, the inputs is performed by both shortcut and element-wise connection. Finally, a linear projection W_s has been included in the equation to ensure the dimensions are equal:

$$Y = \mathcal{F}(x, \{W_i\}) + W_s x \tag{4.12}$$

Data Augmentation

We added an augmentation module to the input pipeline in order to add more variations to the training data. Data Augmentation is a method that increases the number of images in a dataset by transforming the existing images [26] We choose a few transformation techniques to expand our dataset to reduce overfitting

furthermore. We have applied horizontal flipping, brightness and contrast changes, rotation and symmetrical warping.

Architecture

Our model uses the block given in figure 5.2 as its main component and starts with one convolutional layer followed by a maxpool layer. The output sizes of these layers were calculated using equations 5.1 and 5.4 respectively. These two steps were followed by 4 identical layers stacked together which perform convolution with a kernel-size of 3x3 and a restricted feature map dimension of 64, 128, 256 and 512 in order. The input is bypassed every two convolutional layers while the dimensions of that entire layer are kept constant.
A brief representation of our model is given below:

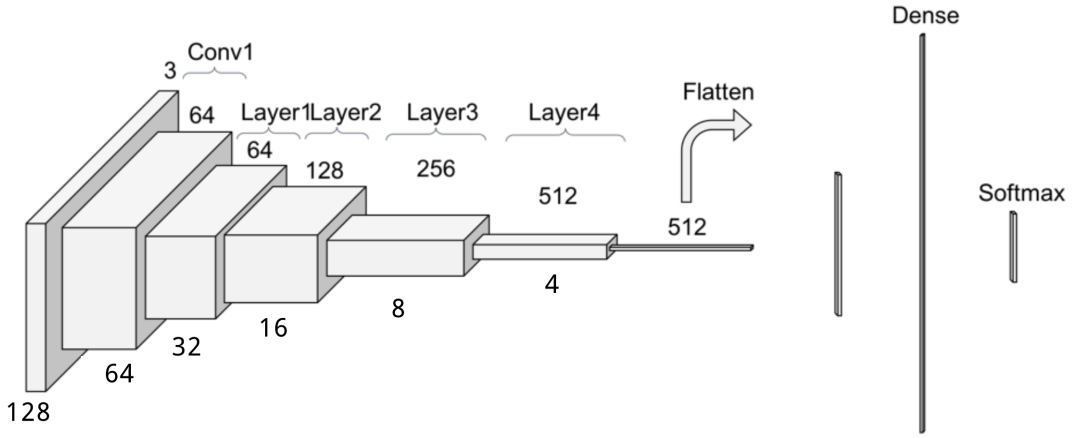

Figure 4.5: ResNet34 Architecture of our model

In every block, convolution is followed by batch normalization [17] and ReLU[28] activation, except the last convolution of the block. After these 4 layers are computed, two average pooling[29] layer operation is done where the kernel-size and stride are automatically adapted according to needs. Finally, a flattened feature map is passed through the linear layers using softmax function for the desired outcome.

The following page contains a summary of our architecture:

Layer type	Output Shape	Parameters
Conv2d	(64, 64, 64)	9,408
MaxPool2d	(64, 32, 32)	0
Conv2d	(64, 32, 32)	36,864
Conv2d	(64, 32, 32)	36,864
Conv2d	(64, 32, 32)	36,864
Conv2d	(64, 32, 32)	36,864
Conv2d	(64, 32, 32)	36,864
Conv2d	(64, 32, 32)	36,864
Conv2d	(128, 16, 16)	73,728
Conv2d	(128, 16, 16)	147,456
Conv2d	(128, 16, 16)	8,192
Conv2d	(128, 16, 16)	147,456
Conv2d	(128, 16, 16)	147,456
Conv2d	(128, 16, 16)	147,456
Conv2d	(128, 16, 16)	147,456
Conv2d	(128, 16, 16)	147,456
Conv2d	(128, 16, 16)	147,456
Conv2d	(256, 8, 8)	294,912
Conv2d	(256, 8, 8)	589,824
Conv2d	(256, 8, 8)	32,768
Conv2d	(256, 8, 8)	589,824
Conv2d	(256, 8, 8)	589,824
Conv2d	(256, 8, 8)	589,824
Conv2d	(256, 8, 8)	589,824
Conv2d	(256, 8, 8)	589,824
Conv2d	(256, 8, 8)	589,824
Conv2d	(256, 8, 8)	589,824
Conv2d	(256, 8, 8)	589,824
Conv2d	(256, 8, 8)	589,824
Conv2d	(256, 8, 8)	589,824
Conv2d	(512, 4, 4)	1,179,648
Conv2d	(512, 4, 4)	2,359,296
Conv2d	(512, 4, 4)	131,072
Conv2d	(512, 4, 4)	2,359,296
Conv2d	(512, 4, 4)	2,359,296
Conv2d	(512, 4, 4)	2,359,296
Conv2d	(512, 4, 4)	2,359,296
AdaptiveAvgPool2d	(512, 1, 1)	0
AdaptiveMaxPool2d	(512, 1, 1)	0
Flatten	(1024)	0
Dropout	(1024)	0
Linear	(512)	524,800
Dropout	(512)	0
Linear	(3)	1,539

Table 4.4: ResNet34 architecture details

Training

We used the transfer learning[12] approach here in which we transfer previously learned information to train our dataset. This is done by taking the same pre-trained neural network that has been trained on a larger dataset and the learned information is preserved. This not only spares us from designing a very deep network from scratch but also provides a performance boost without the need to accumulate vast data. A visual representation of transfer learning used by is given below:

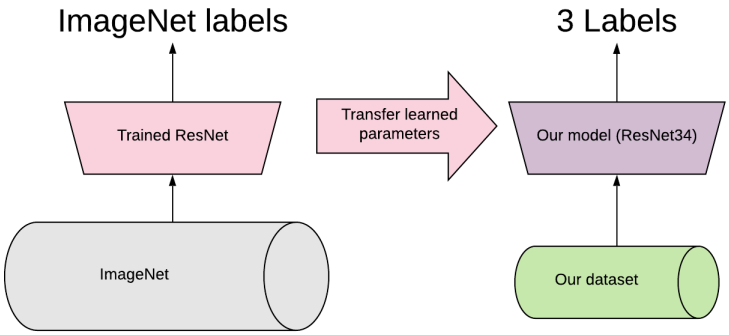

Figure 4.6: Transfer learning overview of our model

To achieve the transfer learning, we first loaded the parameters of a ResNet model trained over the ImageNet[4] dataset. Fully connected layer of this pre-trained model was removed and two fully connected layers had been added followed by adaptive pooling layers. To reduce overfitting, we also added two dropout [15] layers that randomly drops information units in order to prevent the network to memorize a certain pattern and adapt accordingly. The last layer was accompanied by softmax activation function in order to classify the inputs. Below is our training validation loss curve:

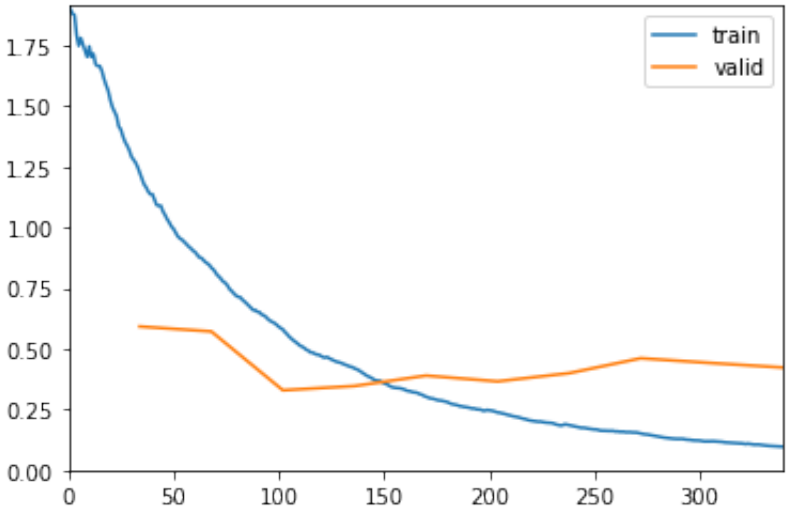

Figure 4.7: Epoch vs Loss curve for ResNet34 training

4.4.3 CNN-XGboost

The process of integrating a Convolutional neural Network with extreme gradient boosting starts with identifying the two main operations: feature extraction and multi-label classification [30]. XGBoost being a member of the gradient boosting family [8] accords the same principals that have been followed in our previous models. We have consolidated extreme gradient boosting as a classifier along with a convolutional neural network extracting the features from our dataset.

Architecture

Similar to the described network in the section (4.4.1), the two main operations needed to be done here are feature extraction and classification. The architecture and parameters of the feature extraction module of this model are similar to our previously built CNN model with XGBoost added at the end as a classifier to predict the classes. A simplified architecture is given below:

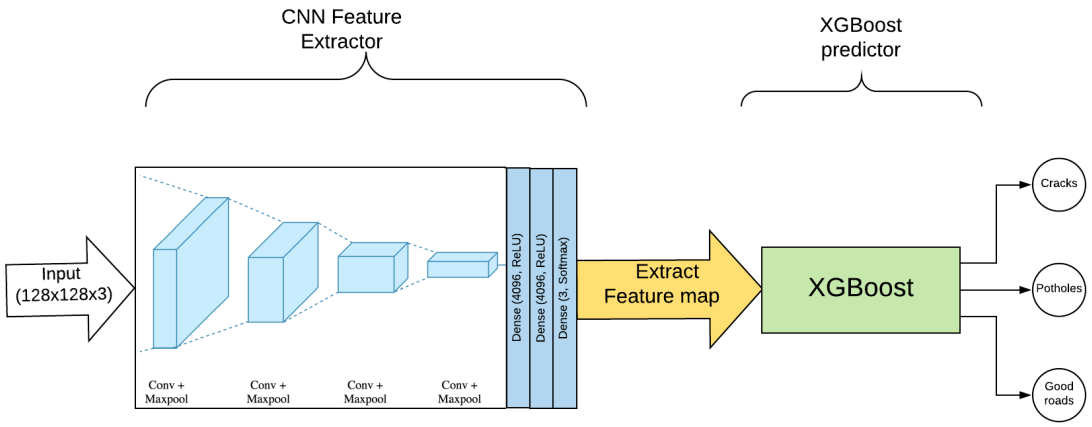

Figure 4.8: CNN and XGboost combined architecture

Training

After training our respective CNN model, we extracted a feature vector from the penultimate dense layer with 4096 features. Similar feature vector from across our training data acts as a new training dataset that has to be fed into the XGboost classifier. Denoting the inputs extracted from this feature map as x_f we derived the output as :

$$\hat{y}_i = \sum_j W_j X_f \tag{4.13}$$

Another representation of the output is:

$$\hat{y}_i = \sum_{t=1}^{t} f_t(X_f) \tag{4.14}$$

Where $f_t(X_f)$ represents the function performing necessary calculation related to output scores of leaf nodes.

As Extreme gradient boosting algorithm cannot be optimized [19] using conventional methods, we have used an objective function to compute the loss during training:

$$L_x = \sum_{i=1}^{n} l(y_i, \hat{y}_i + f_t(X_f)) + \Omega(f_t) \tag{4.15}$$

$$\Omega(f_t) = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2 \tag{4.16}$$

Here, γ is the loss reduction value that controls the split of a node and T is the number of leaves in the tree. The complexity of the tree has been limited by setting the depth of trees as 7. This decreases the chances of overfitting and supports the model to learn patterns related to a particular unit specifically. As the depth of the tree is set to 7 the value of T is automatically set to 7^2. We have set objective function as multiclass softmax in order to classify the inputs using the softmax objective. Softmax is used here by the objective function to get the class that has the maximum probability as an output.

The list of manually adjusted hyperparameters for optimizing the model is given below:

Parameter	Description	Value
η	Learning rate	0.1
L_x	Loss function	Objective(multi:softmax)
NBR	Number of trees	120
MD	Maximum depth of tree	7
T	Number of leaf nodes	128
Softmax(x)	Activation function	Softmax

Table 4.5: XGboost hyperparameters details

Chapter 5

Results and Analysis

5.1 Results

The three models that showed the potential for giving the best results for our classification problem are built and trained after further model optimization and hyper parameter tuning using the methodology described in (chapter 4). The models are evaluated using confusion matrices and compared using the accuracy, precision and recall scores.
We had 553 images in our test set that we fed to all three models and the results are discussed in this chapter. Below is the confusion matrix of CNN model that we have generated in order to visualize and further analyze the outcome.

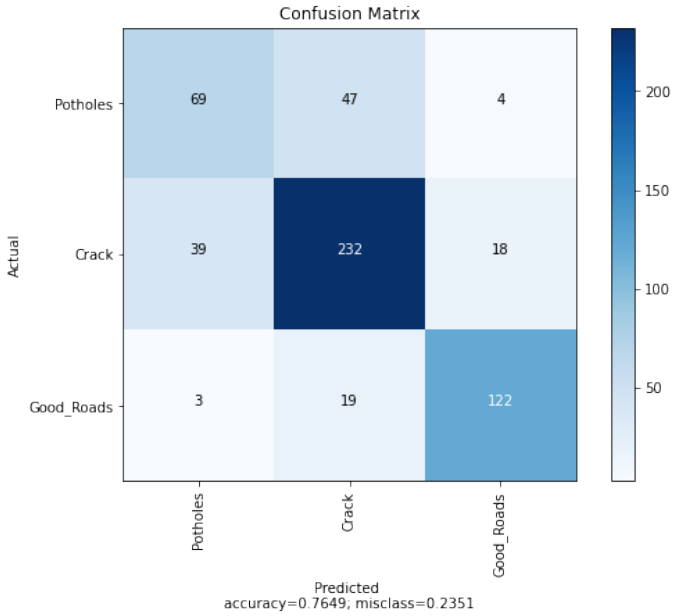

Figure 5.1: Confusion matrix of CNN model

As we have a multi-class problem, so we had to modify the binary approach of calculating Precision and Recall score using the below equations:

$$Precision = \frac{1}{3}\sum_{c=1}^{3}\frac{C_c}{T_c} \tag{5.1}$$

$$Recall = \frac{1}{3}\sum_{c=1}^{3}\frac{C_c}{A_c} \tag{5.2}$$

Where C_c denotes correctly predicted images of class "c", T_c denotes total predicted images of class "c" and A_c denotes number of actual images of class "c".
The confusion matrices for rest of the models' is shown below. Precision and Recall calculations were performed on this computed matrices in a similar fashion:

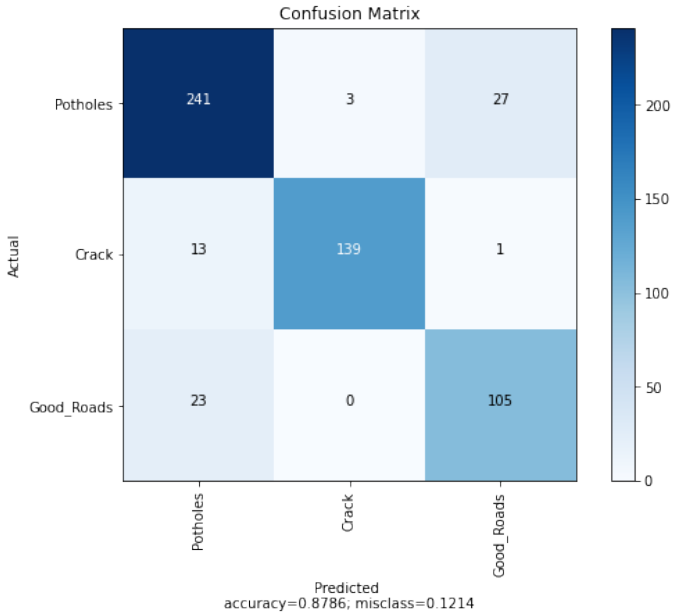

Figure 5.2: Confusion matrix of ResNet34 model

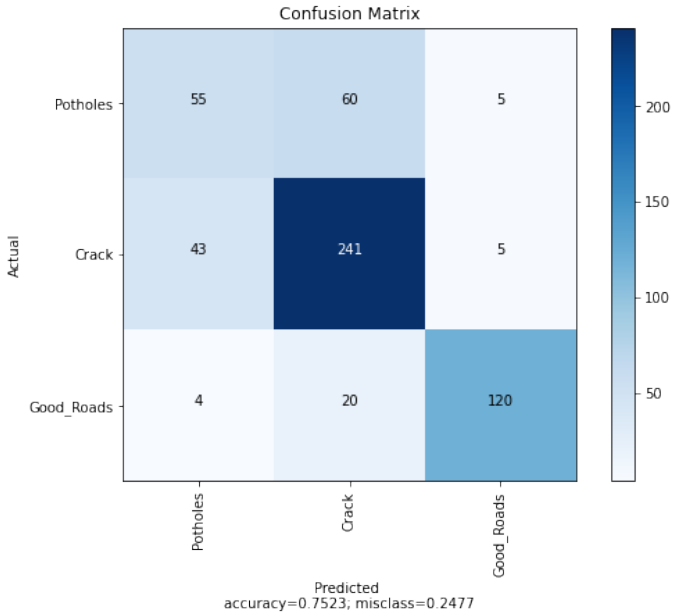

Figure 5.3: Confusion matrix of CNN+XGboost model

Training and evaluation for our model was performed on Google Colab service as they provide a K80 GPU for the GPU runitme. This accelerates the complete process by many-folds when compared to performing the same on normal CPU. Complete data was moved to our google drive, which was then linked with our colab notebook in order to access the data in our code. The training accuracy, precision and recall scores of all three models are graphically compared below:

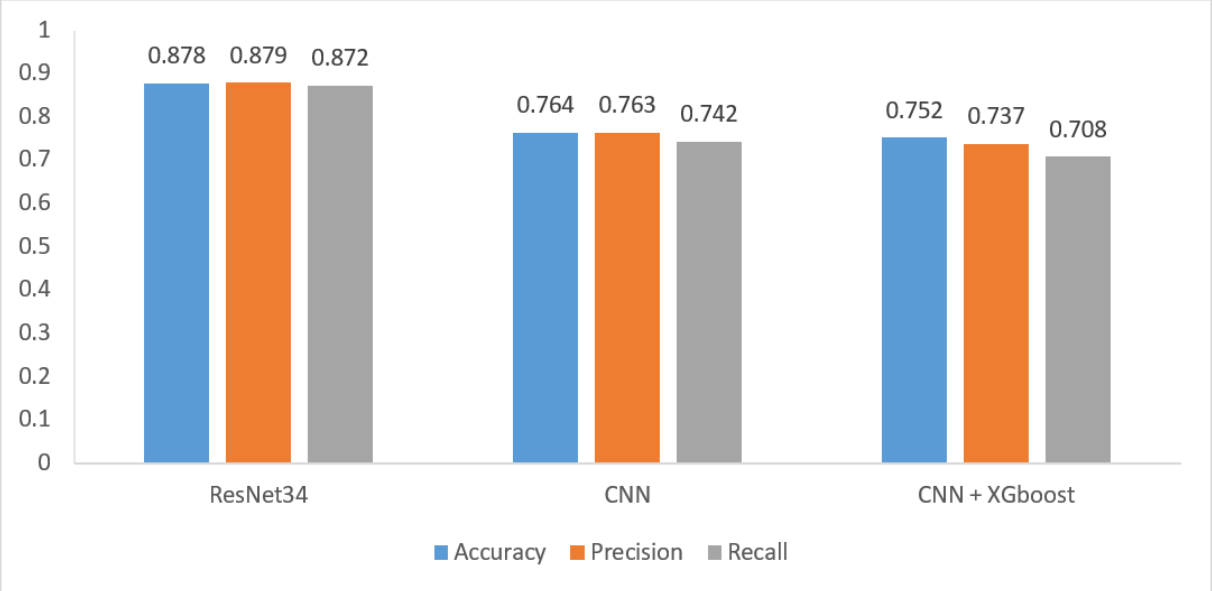

Figure 5.4: Accuracy, Precision and Recall of three models

The highest accuracy of 87.86% was achieved by ResNet34 model, while the other two models, lagged behind with an accuracy of 76.49% (CNN) and 75.23% (XG-boost).

5.2 Final model analysis

To analyze the ResNet34 model further, we looked into the training details. We can see, in the beginning, the error rate was 21.12% at epoch 0 that reduced down to 11.59% at the end, which is nearly halved than the initial rate. Validation loss has also been reduced by 28.55% . The details are given in the following page:

Epoch	Training loss	Validation loss	Error rate	Accuracy	Time
0	1.249568	0.592242	0.211957	0.788043	04:35
1	0.846661	0.571916	0.199275	0.800725	00:13
2	0.585616	0.329725	0.134058	0.865942	00:11
3	0.425057	0.347088	0.139493	0.860507	00:11
4	0.306629	0.389954	0.148551	0.851449	00:11
5	0.240832	0.366185	0.117754	0.882246	00:11
6	0.187858	0.399993	0.125000	0.875000	00:11
7	0.153490	0.461374	0.125000	0.875000	00:11
8	0.118940	0.441579	0.115942	0.884058	00:11
9	0.097417	0.423106	0.115942	0.884058	00:11

Table 5.1: ResNet34 training details

For comparison, in our thesis, we use the algorithm proposed by Penghui Wang [27] and Amila Akagic [21], further referred to as model 1 and model 2 respectively. These two models are chosen, as they are the most recent works that target a similar problem statement and give better results when compared with other previous models. As per the accuracy, precision and recall given in figure 5.4, it is pretty evident that our model can detect potholes and cracks quite robustly even in varying conditions of climate, light and road type. On the other hand, both [27] and [21] try to deal only with the problem of pothole detection exclusively. So, to draw a more uniform comparison between these models with the proposed model, we just use the numbers for class potholes. As computed from the accuracy matrix shown in figure 5.2, the accuracy for our proposed model for class Potholes is 88.92%, while for model 1 and 2 it is claimed to be 86.7% and 82%. These results can be better visualized in the bar charts shown below.

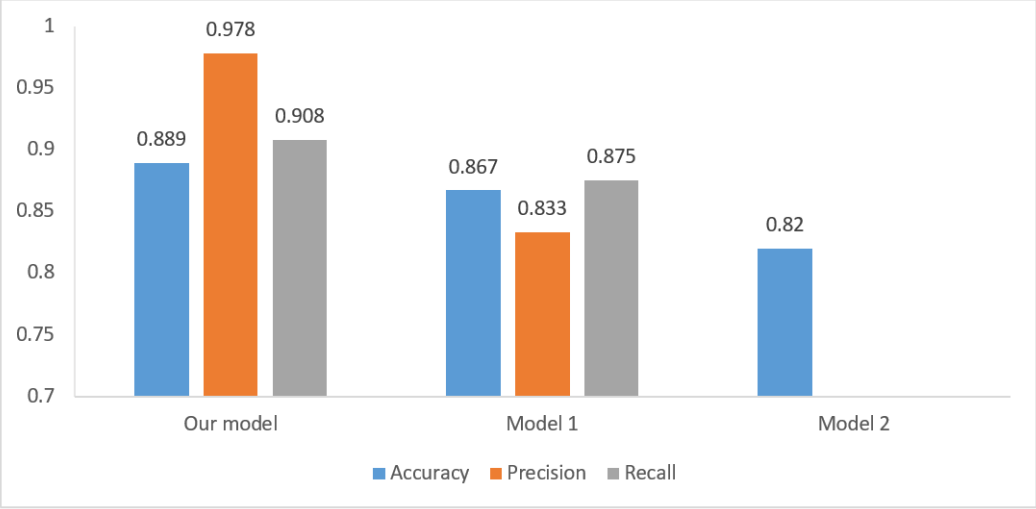

Figure 5.5: Comparison with existing method

Our model is able to differentiate between both pothole and cracks, along with good roads with an overall accuracy of 87.8%. Though the CNN model provided a 97.28% training accuracy, it failed to achieve a higher validation accuracy, thus clearly showing that it faces a bad variance problem. The XGboost model, which

used CNN as a feature extractor module also performed poorly on validation data with a precision score of 16.28% lower than the ResNet34 model. Thus, we conclude that the CNN feature extractor network, when trained from scratch, was getting overfitted due to limited training data available with us. That is where the prowess of ResNet34 model, trained using transfer learning approach, makes it the most viable and robust one for our problem statement.

Below are some visuals of our results.

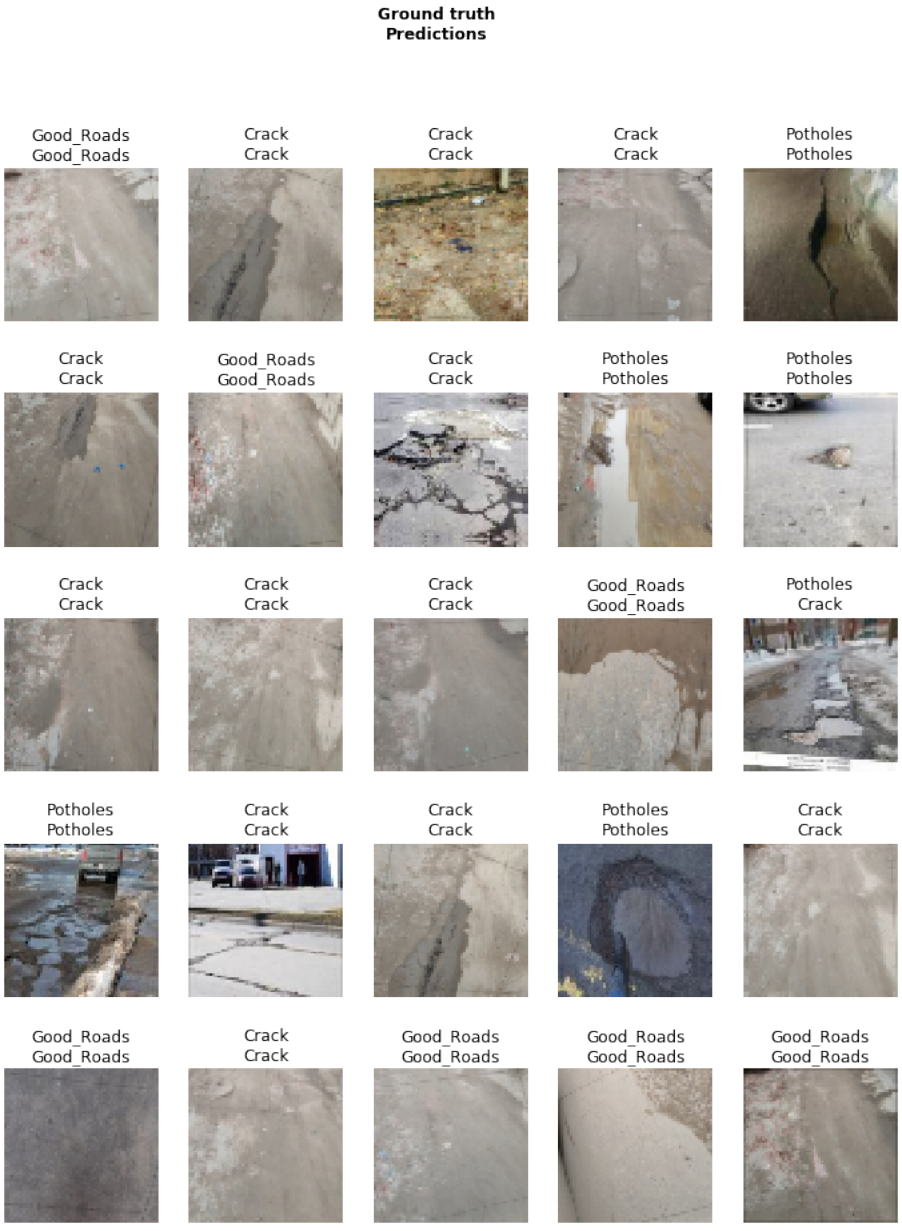

Figure 5.6: ResNet34 Ground truth vs Predictions

Figure 5.7: Predictions vs Actual vs Loss vs Probability

Chapter 6

Conclusion and Future work

This paper proposes an architecture based on the Residual Neural Network algorithms to detect the condition of roads. The model differentiates between good and bad roads and further detects potholes and cracks in order to speed up the process of road reconstruction and maintenance. After extensive shortlisting, we evaluated three different algorithms: Convolution neural network, Residual Network (34 layer) and Extreme Gradient Boosting algorithm to solve this problem. Moreover, the algorithms were assessed on our own collected dataset. All these models' performance were comparatively close, but the Residual Network (ResNet34) model gives the highest test accuracy around 87.8% for the given dataset. Although, currently we are only focusing on classifying potholes and cracks of roads, there are other factors which affect the roads such as type of surface, surface conditions, road spillage, shoulder drop-off and so on. In the future, we will work towards including these features as well, in our study. The features that would be focused are width and dimensions of the roads, surface conditions and shoulder drop-off as per our survey analysis. Additionally, we intent to detect roads covered with mud in the rainy seasons where detecting these features will be difficult, as mud can cause slippage of vehicles. Most importantly, our future plan is to execute and test our project in real world situation. As our topic is a broad and nearly unexplored one, we see huge scope of research to improve our project. We wish to further work on this project for the safety of our people as well as for the country.

Bibliography

[1] D. H. Hubel and T. N. Wiesel, "Receptive fields and functional architecture of monkey striate cortex.", *The Journal of physiology*, vol. 195 1, pp. 215–43, 1968.

[2] R. E. Schapire, "The boosting approach to machine learning: An overview", 2003.

[3] K. Maniruzzaman and R. Mitra, "Road accidents in bangladesh", *IATSS Research*, vol. 29, Dec. 2005. DOI: 10.1016/S0386-1112(14)60136-9.

[4] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A Large-Scale Hierarchical Image Database", in *CVPR09*, 2009.

[5] J. Lin and Y. Liu, "Potholes detection based on svm in the pavement distress image", *2010 Ninth International Symposium on Distributed Computing and Applications to Business, Engineering and Science*, pp. 544–547, 2010.

[6] A. Mednis, G. Strazdins, R. Zviedris, G. Kanonirs, and L. Selavo, "Real time pothole detection using android smartphones with accelerometers", Jun. 2011, pp. 1–6. DOI: 10.1109/DCOSS.2011.5982206.

[7] A. Danti, J. Kulkarni, and P. Hiremath, "An image processing approach to detect lanes, pot holes and recognize road signs in indian roads", *International Journal of Modeling and Optimization*, vol. 2, pp. 658–662, Jan. 2012. DOI: 10.7763/IJMO.2012.V2.204.

[8] A. Natekin and A. Knoll, "Gradient boosting machines, a tutorial", *Frontiers in neurorobotics*, vol. 7, p. 21, Dec. 2013. DOI: 10.3389/fnbot.2013.00021.

[9] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, "On the importance of initialization and momentum in deep learning", *30th International Conference on Machine Learning, ICML 2013*, pp. 1139–1147, Jan. 2013.

[10] K. He and J. Sun, "Convolutional neural networks at constrained time cost", Dec. 2014.

[11] D. Kingma and J. Ba, "Adam: A method for stochastic optimization", *International Conference on Learning Representations*, Dec. 2014.

[12] M. Oquab, L. Bottou, I. Laptev, and J. Sivic, "Learning and transferring mid-level image representations using convolutional neural networks", *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Jun. 2014. DOI: 10.1109/CVPR.2014.222.

[13] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition", *arXiv 1409.1556*, Sep. 2014.

[14] Sonali, B. Maind, and P. Wankar, "Research paper on basic of artificial neural network", 2014.

[15] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting", *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, Jun. 2014.

[16] T. Chen and C. Guestrin, "Xgboost : Reliable large-scale tree boosting system", 2015.

[17] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift", Feb. 2015.

[18] R. Madli, S. Hebbar, P. Pattar, and P. GV, "Automatic detection and notification of potholes and humps on roads to aid drivers", *IEEE Sensors Journal*, vol. 15, pp. 1–1, Aug. 2015. DOI: 10.1109/JSEN.2015.2417579.

[19] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system", Aug. 2016, pp. 785–794. DOI: 10.1145/2939672.2939785.

[20] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition", Jun. 2016, pp. 770–778. DOI: 10.1109/CVPR.2016.90.

[21] A. Akagic, E. Buza, and S. Omanovic, "Pothole detection: An efficient vision based method using rgb color space image segmentation", May 2017. DOI: 10.23919/MIPRO.2017.7973589.

[22] Z. Kamal. (Jul. 2017). At a glance: Bangladesh road accident fatalities on the rise, [Online]. Available: https://www.thedailystar.net/world/south-asia/bangladesh/road-accident-in-bangladesh-2017-statistical-data-essay-at-a-glance-1427245.

[23] Y. Kawasaki, K. Matsushima, and Z. Zhong, "Image-based pavement crack detection by percolation theory", Oct. 2017, pp. 1–6. DOI: 10.1109/ICITEED.2017.8250508.

[24] S. Li, J. Jiao, Y. Han, and T. Weissman, "Demystifying resnet", *ArXiv*, vol. abs/1611.01186, 2017.

[25] X. Ren, H. Guo, S. Li, S. Wang, and J. Li, "A novel image classification method with cnn-xgboost model", 2017, pp. 378–390. DOI: 10.1007/978-3-319-64185-0_28.

[26] C. Vasconcelos and B. Vasconcelos, "Increasing deep learning melanoma classification by classical and expert knowledge based image transforms", Feb. 2017.

[27] P. Wang, Y. Hu, Y. Dai, and M. Tian, "Asphalt pavement pothole detection and segmentation based on wavelet energy field", 2017.

[28] A. F. Agarap, "Deep learning using rectified linear units (relu)", Mar. 2018.

[29] B. Mcfee, J. Salamon, and J. Bello, "Adaptive pooling operators for weakly labeled sound event detection", *IEEE Transactions on Audio Speech and Language Processing*, vol. 26, Aug. 2018. DOI: 10.1109/TASLP.2018.2858559.

[30] A. Azhar, M. Khodra, and A. Sutiono, "Multi-label aspect categorization with convolutional neural networks and extreme gradient boosting", Jul. 2019, pp. 35–40. DOI: 10.1109/ICEEI47359.2019.8988898.

[31] *Road accidents caused 7,221 deaths in 2018: Bangladesh passenger welfare organisation*, Accessed: 2020-02-25. [Online]. Available: https://bdnews24.com/bangladesh/2019/01/25/road-accidents-caused-7221-deaths-in-2018-bangladesh-passenger-welfare-organisation.