Damaged Road Detection using Image Processing and Deep Learning

by

Shimran Mahbub Swadesh
21341029
Rifat Ahmed
18101710
MD. Imran Hossain
17201093
MD. Raihan Rahman
18101169

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
January 2022

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:

Shimran Mahbub	Rifat Ahmed
Shimran Mahbub Swadesh	Rifat Ahmed
21341029	18101710
MD IMRAN HOSSAIN	MD. RAIHAN RAHMAN
MD. Imran Hossain	MD.Raihan Rahman
17201093	18101169

Approval

The thesis/project titled "Damaged Road Detection Using Image Processing and Deep Learning" submitted by

1. Shimran Mahbub Swadesh (21341029)
2. Rifat Ahmed (18101710)
3. Md. Imran Hossain (17201093)
4. MD. Raihan Rahman (18101169)

Of Fall, 2021 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science on January 18, 2022.

Examining Committee:

Supervisor:
(Member)

[illegible]

Ahnaf Rodoshi
Lecturer
Department of Computer Science and Engineering
Brac University

Co-Supervisor:
(Member)

[illegible]

Nafis Mostafa
Contractual Lecturer
Department of Computer Science and Engineering
Brac University

Program Coordinator:
(Member)

Md. Golam Rabiul Alam, PhD
Associate Professor
Department of Computer Science and Engineering
Brac University

Head of Department:
(Chair)

Sadia Hamid Kazi, PhD
Chairperson and Associate Professor
Department of Computer Science and Engineering
Brac University

Abstract

Computer Science has evolved enormously in the last few decades. It has now far exceeded the Human and Computer interfaces. Its recent sights are scaling, measuring, object detection, etc. Image processing and deep learning have gone through many groundworks in the last few years. Our research paper, based on YOLO V4, LeNet-5, Retina Net, and Faster-RCNN algorithms, proves that these algorithms can detect damaged roads and analyze whether we can enhance any new ways to improve the damaged road detection in real-time. However, in a real-world scenario, it is essential to comprehend the various damages in taking the appropriate action. Thus automotive industries are looking forward to innovations that can increase the efficiency in damage categorization.
Along with worldwide industrialization, road damage detection systems have become significantly important both in terms of maintenance and establishment. As the Artificial Intelligence sector is making a lot of progress, faulty road detection through Image Processing and Machine Learning has proved to be a flourishing technique. We can detect damaged roads within specific provisional categories with combinations of such technological stems. In our solution, we propose a futuristic deep learning method for object recognition with the help of four different algorithms. More specifically, our approach uses a convolutional neural network to train our model with a large dataset solely made for the project and categorize the results into a set of damages along with its comparative analysis.

Index Terms—Deep Learning, Road Crack Detection, Categorization, Object Recognition, Transfer Learning and RELU.

Acknowledgement

“It is not possible to prepare a project report without the assistance encouragement of other people. This one is certainly no exception.”

First and foremost we are grateful to our Almighty and thankful to our family and friends for their constant source of inspirations and support Moreover we would like to express our sincere gratitude to our advisor Ahnaf Rodoshi for the continuous support of our research. Besides our advisor, we would like to thank our co-supervisor Nafis Mostafa. We are ineffably indebted to Nafis Mostafa for conscientious guidance and encouragement to accomplish this research. Their guidance helped us in all the time of our research and writing of this thesis. We could not have imagined having a better advisor and co-advisor for giving us this opportunity to research this exceptional topic.

All the exertions here are the result of combined work from the group members. We have tried to cover all the topics provided by our supervisors Ahnaf Rodoshi and our co-supervisor, Nafis Mostafa.

Table of Contents

List of Figures

Chapter 1

Introduction

In this 21st century, infrastructure is crucial for speedier financial development and alleviating poverty. It should be emphasized that an excellent foundation is significant not just for quicker financial development but also to guarantee a comprehensive outcome. According to the World Economic Forum Global Competitiveness Report, out of 142 countries, Bangladesh stood 134th(2013). This rating is done based on the quality of the overall infrastructure within the country. According to the Global Competitiveness Index for 2017-2018, Bangladesh has the second-worst roads in Asia on a scale of one to seven. Bangladesh received just a 3.1 for road quality, a subcategory of infrastructure. Whereas Nepal, which has a poor score of 2.8 and is prone to earthquakes, floods, and landslides, is the only country below Bangladesh. Given the condition of infrastructure management today, more feasible and efficient road maintenance approaches are necessary. Pavement distress, often known as damaged roads, is a significant criterion for pavement condition that may be evaluated manually, moderately automated, as well as completely autonomously. Manual and moderately automated surveys are the two most common methods for collecting data on road conditions. However, due to the length of road networks, extensive human contact is required, which requires time. Moreover, road damage detection findings vary from person to person because they rely exclusively on the assessors' experience. Thus, there is a raising need for autonomous road condition evaluation methods that can detect as well as locate road damage quickly and precisely.

Currently, autonomous identification of road damage is mainly accomplished using three methods: laser, radar, and vision. Deep learning-based technology has been quickly developing. Deep neural network-based computer vision research and product developments are fastest gaining tractions. Academics have conducted a fresh round of research and practice on a strategy for detecting road damage using image processing technologies and deep neural networks.

1.1 Research Problem:

For emergent nations, one of the most important criteria is having smooth roads that maximize sustainability. Road Damage Detection is important to maintain the infrastructure quality and condition. However, it is challenging for a few experts to accurately carry out the on-field inspection over a vast chunk of land.

In the recent past, many researchers came up with new ideas on Artificial intelligence to detect exceptional road damage but lacked precision and accuracy. The unavailability of image datasets required for model training is a significant drawback. Image quality is another concern, as most algorithms do not function properly with lower quality images, and thus wrong predictions are made. A few researchers were able to detect damages but failed to categorize the damaged road and mention how fast the condition needed attention. Many CNN-based algorithms have been presented as the growth of deep convolutional networks has made significant advances. The R-CNN technique detects objects in two steps: suggestion of object regions and classification. Algorithms such as CNN and R-CNN were able to detect damages. However, they took a considerable amount of time to complete and therefore worked for greater optimal approaches introduced Faster RCNN. R-CNN learns RPN faster by extracting all anchors from a single image in a 256-simple-batch. Because all samples from one picture may still be connected, it takes a long time for the network to reach convergence.

1.2 Research Objectives:

This research provides an intelligent solution for detecting cracking roads from images and videos by using image processing. This paper has the following objectives:

- Constructing a new benchmark over the use of algorithms in image processing and deep learning can be established from this research. Unfortunately, there are only a few papers concerning image processing and deep learning, so our foremost objective is to deliver arithmetical refinements for public usage.
- A brief comparison between our firsthand result and any existing result on the internet, which will be helpful for other academic purposes in the future.
- The CNN model would be trained with different images around the world, and thus we would be able to provide a perfect road damage detection system that would have high accuracy and precision all over the world.
- Perform a comparative analysis between the algorithms used and provide accurate reasoning for which algorithms work best.
- Provide a new dataset of more than 3000 images which can be classified into several sorts of damages. There are not many datasets available on the internet; thus, providing a free dataset source for public research is one of our primary objectives.
- Categorize the risk based on the damage. Risk categorization will make it easy to identify which roads are highly damaged and provide priority to repair as this system can be categorized into low, medium, and high risk.
- Evaluate the research outcome based on the diverse datasets available on the internet and newly formed datasets from any remote area. The datasets we have found on the internet are not from our region, so our newly formed dataset and other datasets will be different. Thus we will be able to assess the difference in roads.

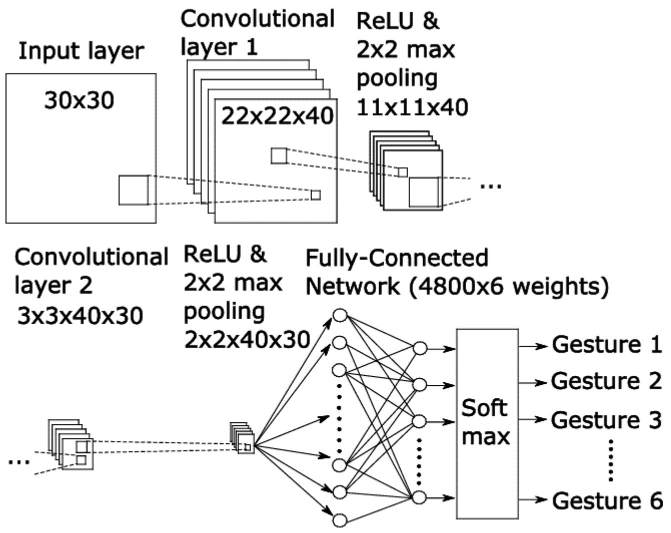
Chapter 2

Background

2.1 Convolutional Neural Networks (CNN)

The convolutional neural network, also known as CNN, is a deep learning technique that dominates the classes of processing such as image base processing technic for seeking cracks on the road's surface. CNN can abstract an array of numbers using the feature extraction process of linear operation, which activate by activation functions. Inputs are implemented kernel with tensor numbers generated from the array that has previously been introduced. Based on the cross fold of elements, generated tensor inputs and the output, we can create a featured map. These kernels have their own characteristics that can separate different features.

2.2 CNN Architecture



Input layer
30x30
Convolutional layer 1
22x22x40
ReLU & 2x2 max pooling 11x11x40
...
Convolutional layer 2 3x3x40x30
ReLU & 2x2 max pooling 2x2x40x30
Fully-Connected Network (4800x6 weights)
...
Soft max
Gesture 1
Gesture 2
Gesture 3
Gesture 6

Figure 2.1: CNN Architecture

Traditional CNN is complex functionality of convolutions layers, ReLU(rectified unit layers), max-pooling, fully connected layers for neurons. There are other layers named propagation layers which can transform the input data to output data. As mentioned, there are many components that are the basic element of CNN. Convolution layers are one of the major components that harness features using linear and non-linear functions. Values of linear function then pass through via convolution layer. The convolution neural network uses the ReLU function; ReLU is far better in performance. It has only two edges for functional representations; Y=0; Y=X;

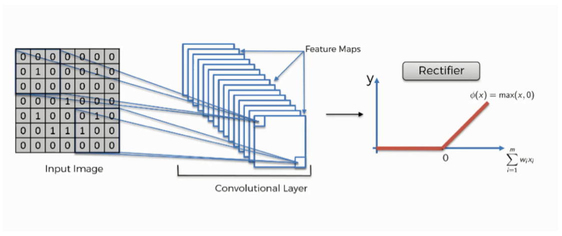


Figure 2.2: Max Pooling

It is one of the features that downsamples the operations for decreasing the in-plane dimensionality as it can manipulate the variances of small vector distorted vectors and reduce the subsequent learnable parameters. The learnable parameters are likely to be not present in the first pooling layers. There are some hyperparameters like stride, filtration, padding etc. they are ones that came along the pooling layers. In earlier layers, the strides are 2X2 in the filter size. These are strides comes from extracting feature maps. Also, extraction features are reduced in this phase in 2 folds.

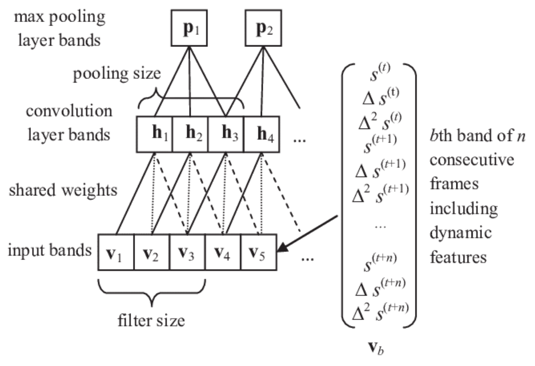

max pooling layer bands
$\mathbf{p}_1$
$\mathbf{p}_2$
pooling size
convolution layer bands
$\mathbf{h}_1$ $\mathbf{h}_2$ $\mathbf{h}_3$ $\mathbf{h}_4$ ...
shared weights
input bands
$\mathbf{v}_1$ $\mathbf{v}_2$ $\mathbf{v}_3$ $\mathbf{v}_4$ $\mathbf{v}_5$ ...
filter size
$s^{(t)}$ $\Delta s^{(t)}$ $\Delta^2 s^{(t)}$ $s^{(t+1)}$ $\Delta s^{(t+1)}$ $\Delta^2 s^{(t+1)}$ ... $s^{(t+n)}$ $\Delta s^{(t+n)}$ $\Delta^2 s^{(t+n)}$
$\mathbf{v}_b$
bth band of n consecutive frames including dynamic features

Figure 2.3: Pooling Layers

The pooling layers have also become inputs for some of the layers, such as Dense layers that connect the multiple layers into output layers. These layers then assign some weights and transform the layers into a one-dimensional vector array. The vector is an integer-valued array. Then it goes through the final output via fully connected layers. Such the Classified value that we have got from the previous layers like convolution or downsamples are being sent as the output of the function. The function has consisted of the same number of classes as the fully connected layers that have been implemented in the system. Action functions such as ReLU are not the same fully connected layers we have previously encountered. Rather the system chose the right activation function. Softmax is one of the functions implemented to normalise real output values into binary values such as 0 and 1. It can also be called vectoring the value.

Chapter 3

Literature Review

This section aims to critically review previous relevant work in Road Damage Detection in the context of Deep Learning and Image processing. We analyze the different techniques used for the main results achieved, and we show how CNN has its specific challenges due to the lack of Precision and accuracy.

Road accidents are one of the most frequently occurring causes of death; statistics show that around 3500 people die every day (World Health Organization). Many of the accidents are also related to damaged road tracks. To reduce such an unprecedented event, detecting damaged roads and a well-planned recovery system is one of the keys. The neural network study area is a flourishing subject of modern technology, specialized in finding solutions in application fields that are challenging to describe with standard statistical techniques Adeli(2001).

Image Processing and its advancement in road damage detection and differentiation have been investigated thoroughly. In the early days, the researchers utilized target level implementation to discover shear areas. The mean and standard deviation distinguish blocks with damage from blocks without a crack in CrackIT. Cord and his team use AdaBoost to describe crack images. The effectiveness of these approaches depends on the features that are obtained. But unfortunately, it is difficult to find features that work for all the pavements.

Because neural networks are the result of a training process, their behaviour differs considerably from that of traditional computer science techniques. In contrast, traditional techniques predetermine the system's behaviour. OverFeat in the studies outperformed a similar methodology by Huval et al., 2015 like an instance of a deep edge learning method that excelled standard techniques. However, as per our knowledge, there is no example of a method that uses deep learning to detect road damage identification properly.

Applying cutting-edge deep learning technology, a group of specialists created photograph-based algorithms for assessing the street condition. In previous work, Chun et al. (2015) developed an improved road surface crack identification system that used image processing techniques as well as a machine-learning approach.
In 2016, Deep neural networks were utilized in the work of Zhang et al. and Maeda et al. They compared the findings of SSD Inception V2 to the findings of SSD Mo-

bileNet. Linear cracks and white-line blur had good Recall and Precision, however construction joint pothole demonstrated low Recall owing to a lack of training data. MobileNet beats Inception in six domains where the value of Recall is evaluated, with the exception of a pothole and cross-walk blur. In general, SSD MobileNet outperforms HDD MobileNet. Furthermore, they did error analysis to better understand the detection findings. False positives and false negatives were utilized to classify the errors. In the year 2017, Zhang et al. made it feasible to assess road surface deterioration rather precisely. Zhang et al., 2017 developed a crack net that anticipated output values for all regions. CrackNet is defined in this article as a low-cost architecture based on Convolutional Neural Networks for automatic pixel-perfect pavement fracture identification on 3D asphalt surfaces (CNN). As input data, the CrackNet employs feature maps created by the feature extractor using suggested line filters in various orientations, widths, and lengths. For each pixel, CrackNet generates a list of anticipated class grades. Convolutional and fully linked layers make up CrackNet's hidden layers. CrackNet was trained on 1,800 3D pavement images before being demonstrated to be effective at detecting cracks in a variety of circumstances using a second set of 200 3D pavement images. Using 200 testing 3D pictures, the Precision we got from the CrackNet was able to achieve high Precision (90.13%), Recall (87.63%), and F-measure (88.86%) all at the same time. Nevertheless, current faulty road diagnostic systems can only identify the presence of deterioration. Although there exist some studies that define damage by kind, Zalama et al. divide it into two sectors: (i) parallelly and (ii) perpendicularly. Akarsu and their team, on the other hand, classified damage into three categories: vertical, horizontal, and zigzag.

R. Fan et al. developed an identification procedure based on deep learning and responsive image classification for detecting road fractures; this neural network is utilized to determine the cracks. They randomly chose 15000 positive and 15000 negative photos from the dataset to train the neural network. The remaining images are used to evaluate the suggested technique's effectiveness. The beginning learning rate, max number of epochs, and evaluation frequency are equal to zero, sixteen, and sixty. With a momentum of 0.9, the optimizer is backpropagation descent with momentum. To examine the accuracy of our proposed image, they calculated Recall and Precision, which reflects the number of actual positive, false negative, and real negative testing photos consecutively. Fan, Z. offers a supervised technique, which is deep learning-based, to deal with varying pavement conditions. CNN is especially utilized to grasp the structure of the fractures from raw pictures with no pre-processing. Small patches from damage pictures are used as sources to build a large training dataset, a CNN is taught, and cracking detection is treated as a categorized technique.

Taking this into account, we use edge deep learning-based object recognition algorithms to identify street surface deterioration. Initially, we test each algorithm's accuracy results and processing speed. We specifically study if we can recognize and categorize different types of road damage using cutting-edge object detection algorithms. Moreover, we would use four different algorithms, such as YOLO, Faster RCNN, RetinaNet, and LeNet-5, which works instantly and analyze the algorithms for the best result. We also look forward to finding out which algorithms perform better in different circumstances.

Furthermore, as we can assess the risk from even a single frame, we have the utmost advantage of using it for any complaint, which will help us minimize the accidents and a safe route for the people as early as possible. Also, the anchor points can help us to identify the highly damaged area. Despite the usefulness, there is a possibility of miscalculation as many systematic problems or bugs may appear on the software. Malware is one of the reasons why we have to evaluate physically, as it may delude workflow sometimes. Apart from that, the proposed algorithms have the advantage in accuracy, rapidity, and other sectors. Though we can undoubtedly assume, the system has the edge in benefits as faults can be fixed throughout the implementation process, and updates may resolve the problem of bugs fixing or any other unruly.

Chapter 4

Methodology

This section will focus on the description and detailed architecture of the Algorithms used for our proposed road damage detection. Secondly, we will discuss the dataset we used and the division of the images into training and test categories. Finally, we would also enlighten the readers on our proposed workflow and the pre-processing mechanisms, including their brief explanations.

4.1 Algorithms

4.1.1 YOLO

Recently, an objection detection system has been built with two sections, a backbone and a head. The Backbone is pre-trained, and the head is required to predict classes and bounding boxes of an image. YoloV4 is an essential improvement of YoloV3, and it has a very unique distinctive that its previous predecessors did not have at the time. It has significantly changed the mAp (mean per average) value along with its FPS rate. We can easily train datasets with such a neural network.

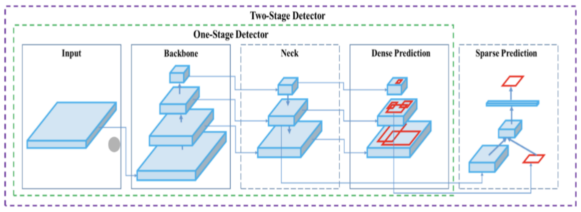


Figure 4.1: Two-Stage Detector

In the YOLO v3 portion, we will talk about types of object detection models are; one of them is one stage model that can recognize objects without any pre-processing phase. But two, stage models have been more popular due to their classification capability. But when it comes to speed and real-time performance, then one stage model has the upper hand.

For Crack detection, we have to use a two-stage model. The Two-stage model consists of image classification and image segmentation. The Two-stage model is in advance due to its convolution neural network. The output of the image is given

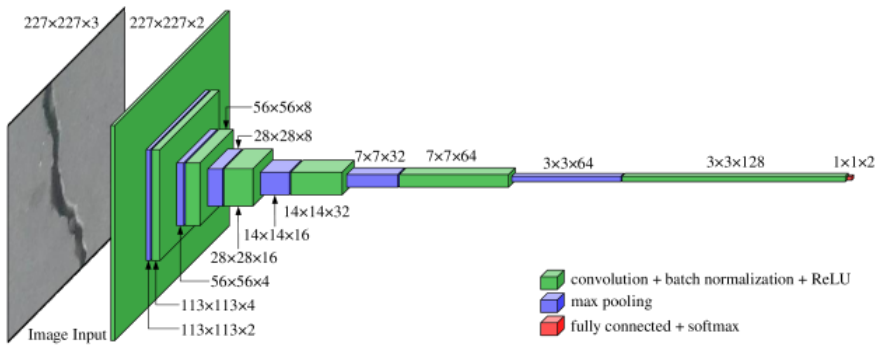


Figure 4.2: YOLO CNN Architecture

based on the presence of cracks. First, the system tries to separate the image based on the cracks as a negative or positive image. Then the deep neural convolution network extracts the negative image as cracks of the real picture.

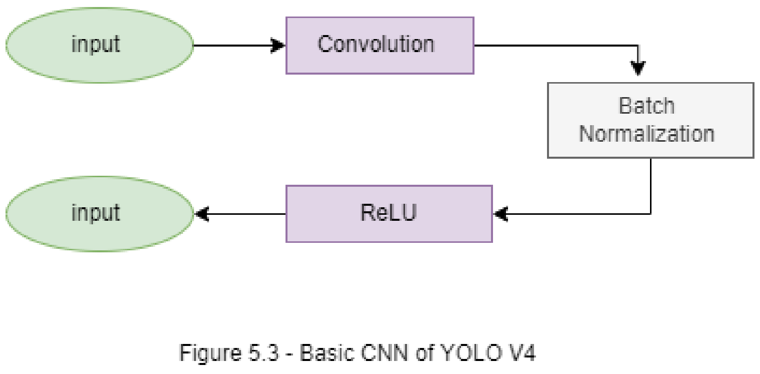


Figure 4.3: Green Zone of CNN briefly explained.

Then, the max-pooling calculates the maximum value for patches of the featured map. But the current patches are too high, so the network downsamples representations. Meanwhile, the softmax function translates the vectors into probable distributions.

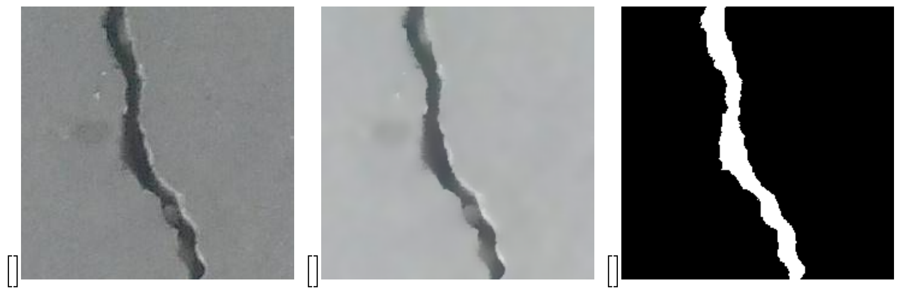


Figure 4.4: Bilateral filtering and image segmentation;(a) original picture; (b) filtered positive; (c) segmented picture.

After the classification, the system will only take a positive image as the input. Before starting the next stage, the image has to go through the Bilateral filter process because it smooths the image and performs the edge preservation.

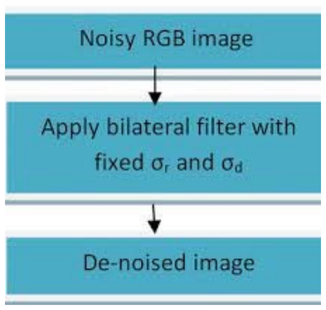


Figure 4.5: Edge Preservation

To further reduce the image more; we have to downsample the image to reduce the image size. The pixel of the image after downsampling is normalized.

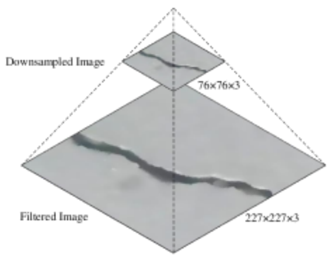


Figure 4.6: Normalization of Down Sampling

Then we can assume that our image has two portions. One of them is the normal surface of the road, and another one is the cracked surface. The cracked surface is assumed to be the darker part. So, using dense prediction, YOLO v4 can detect the cracks of the image.

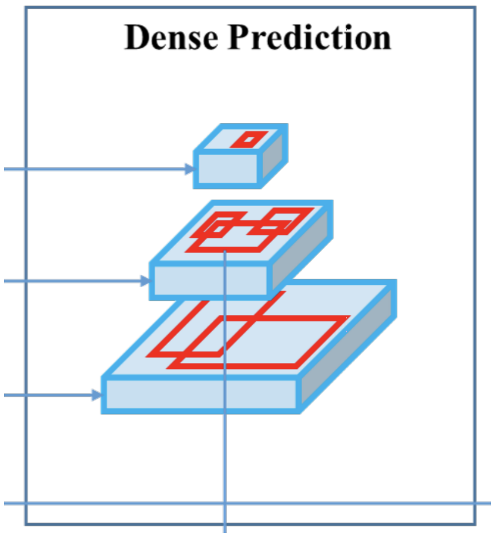


Figure 4.7: Dense Prediction Model

4.1.2 LeNet-5

LeNet-5 is one of the earliest architectural models suitable for resolving problems associated with Images. It has all the features of CNN, but the only difference is its layer distribution. It consists of 7 layers in total; 3 convolutional layers, 2 subsampling layers and 2 fully connected layers.

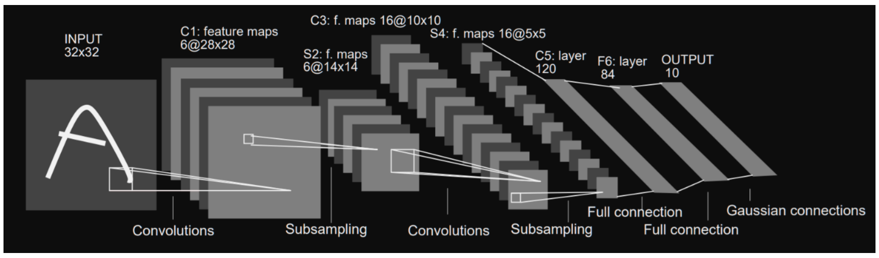


Figure 4.8: LeNet-5 Architecture

The Subsampling layer distinguishes between CNN and LeNet-5's performance. LeNet5's subsampling acts as a means of local averaging and sub-sampling the data size. Due to its precession in average and sub-sampling, it is mostly used in Bank sectors for recognizing handwriting. It can also be used for object detection such as cracks, vehicles etc. In Figure-4.9, The convolution is connected to the subsampling layer because the layers need to convolute the image as it fits, then it averages the image size along with its filtration. Via this combination of layers, it becomes easy for the dataset to reform as the requirement is imposed on it. We have two such layers among them. Also, LeNet-5 uses Tanh as an activation function. It has a total of 5 learnable parameters in total. The images given are grayscale. It has thousands of trainable parameters as well.

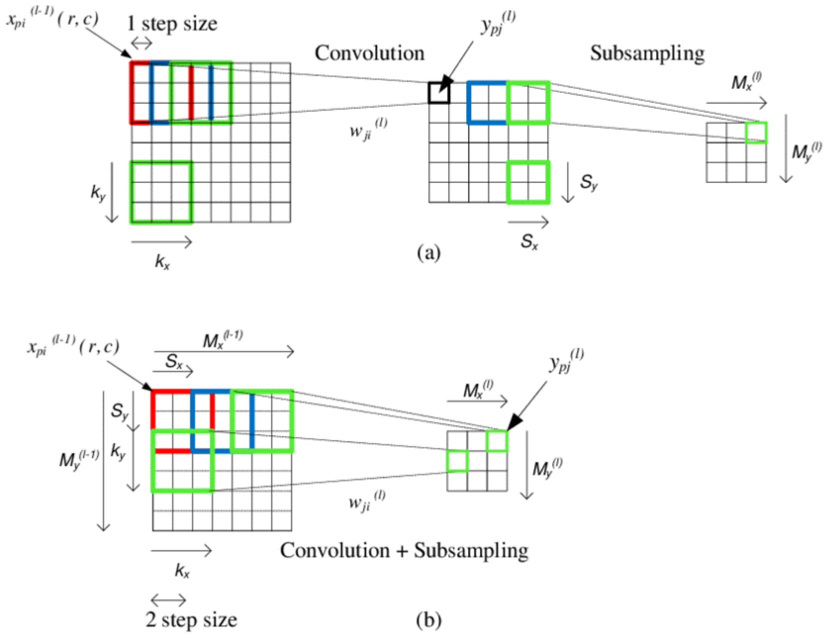


Figure 4.9: Process of Sub Sampling

After the pre-processing of the dataset, the fully connected layers multiply the given data by assigning weights to it and adding a bias vector to it. So, ultimately, it fits the data according to the instructions given to the model.

4.1.3 Retina Net

RetinaNet is structured on ResNet architecture with Feature Pyramid Net (FNP) built on top, which was modelled to increase the accuracy of one-stage detectors. This Neural Network could bring good outcomes on crack detection in roads as well as on pavement surfaces. To achieve accuracy, it splits FPN into two sub-networks executing simultaneously, with similar input layers to reduce execution time and make it faster. Moreover, incorporating class imbalance during training helps RetinaNet reshape loss function.

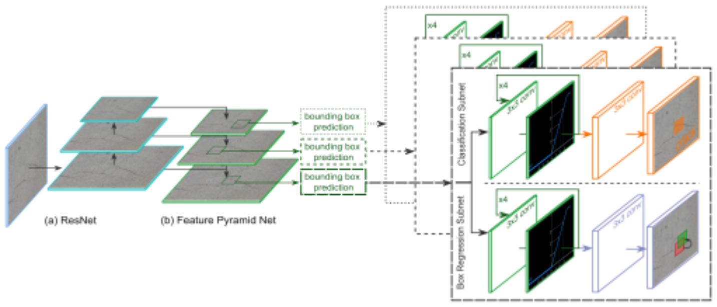


Figure 4.10: RetinaNet Architecture

The FPN architecture assembles a single input image into several scaled images where the detector separately evaluates each level. Based on the level of FPN, three types of ratios are used (2:1, 1:1, 1:2) for each multiple scaled prediction where the anchor area is implemented in different sizes.

The Intersection establishes the weighting over the Union (IoU) ratio between the projected output and the ground-truth bounding box in the training phase. Above 0.5 matchings are considered positive, indicating that the projected anchor is allocated to a single ground-truth box. Overlapping between multiple predictions and ground-truth bounding boxes must be ignored.

The scale of the ground-truth boxes is not dependent on the scale of FNP; even the anchor boxes are also used to rescale the prediction to the original image size so that it may be compared to the ground-truth bounding boxes.

The classification subnet estimates the likelihood of a fracture in the FPN output and classifies it properly. Each of the FPN levels is served by the subnet. At each level, the subnet employs a Fully Convolutional Network (FPN).

The box registration subnet is almost identical to the classification subnet in terms of design, but it is run four times for each of the anticipated boxes in order to calculate the offset scale and positioning. These outputs show the difference between the predicted box and the ground truth, including two for bounding box scaling and two for bounding box placement. The two sub-networks are then fed into a targeted loss function that concentrates on outliers and misclassified cases.

4.1.4 Faster RCNN

A number of boxes are suggested by the RCNN algorithm in order to test if any of them contain any objects rather than a huge number of regions to work on. To retrieve these boxes from an image, RCNN uses a specific search. These boxes are referred to as regions.
Firstly, we scrutinize what it's called and learn how it distinguishes and pinpoints regions. Fundamentally, an object comprises four parts: variable scales (size), colours, textures, and boundaries. These patterns in the image are identified using selective search, which makes suggestions based on those patterns. As an input file, a picture is used. This is followed by the formation of primary sub-segments so that we can have many areas from this picture. A bigger area is created by combining related regions based on colour consistency, texture resemblance, similar size, and shape suitability. Then the areas are used to determine the final positions of the objects (Region of Interest).
Unfortunately, because of the several procedures and methods, R-CNN becomes quite sluggish. Faster R-CNN eliminates the issue of selective search by replacing R-CNN with Region Proposal Network (RPN). The first thing that we do to get feature maps is to use ConvNet from the training images and then send them through an RPN to generate object suggestions. Hence, the bounding boxes of these maps are projected and categorized.

Faster R-CNN is a truncated variant of Fast R-CNN. The main distinction is that Fast RCNN generates Regions of Interest by selective search, whereas Faster RCNN generates Regions of Interest using "Region Proposal Network," aka RPN. RPN takes image representation patterns as input and generates a sequence of image recommendations, each with a bounding box score.

Faster-RCNN is a fully end-to-end CNN object detection model. After recognizing items in an image, the following stages are followed by a Faster R-CNN algorithm:

1. Give an input picture to the ConvNet, which provides feature maps for the picture.
2. (RPN) extracts the feature maps to get the output image.
3. Use the ROI pooling layer to reduce the size of all proposals to the same size.
4. Lastly, the recommendations are then sent to a convolutional layer to identify and forecast the image's anchor boxes.

Actual Functions of RPN:
Faster RCNN begins by taking CNN's feature maps and passing them along to the Region Proposal Network (RPN). Over these feature maps, RPN creates k Anchor boxes of varying sizes and shapes by sliding a window over them.

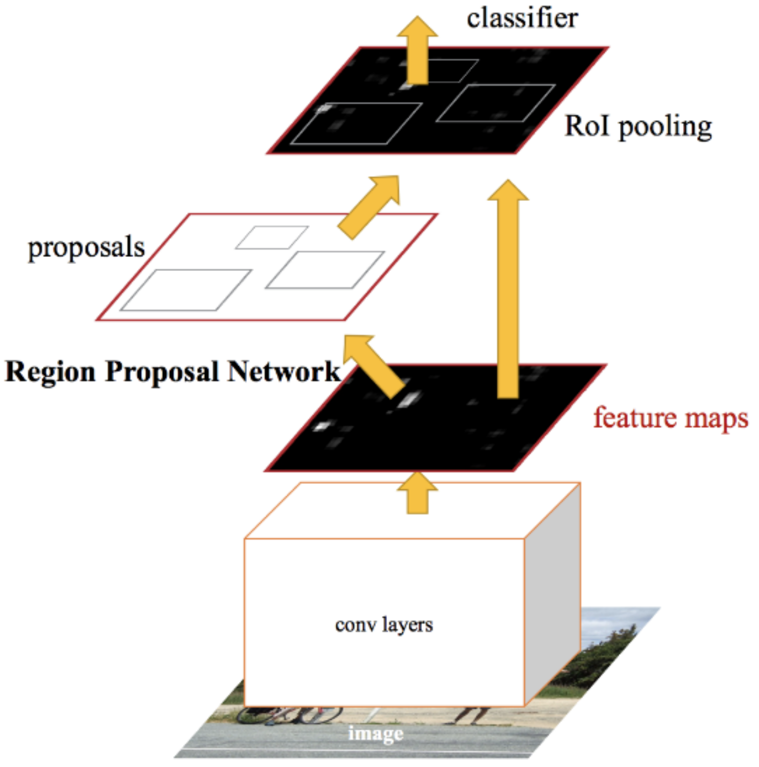


Figure 4.11: Region Proposal Network

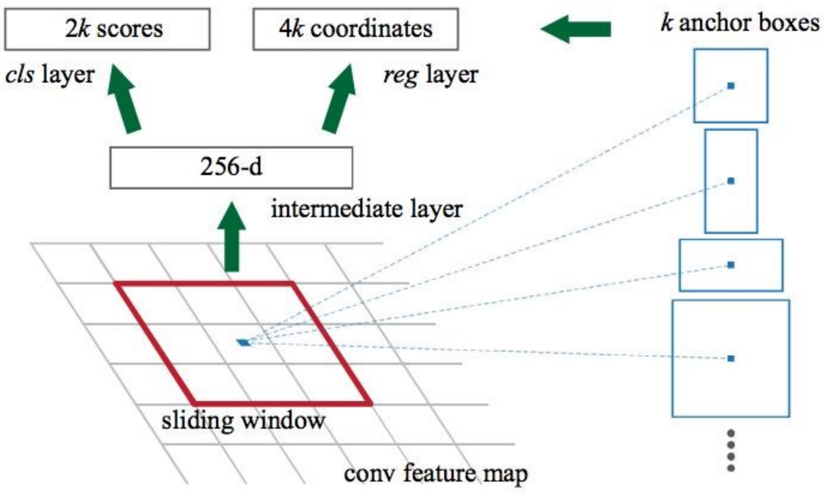


Figure 4.12: Sliding Window

Anchor boxes are fixed-size boundary boxes that are randomly dispersed across the picture and come in a variety of shapes and sizes. RPN predicts two outcomes for each anchor:

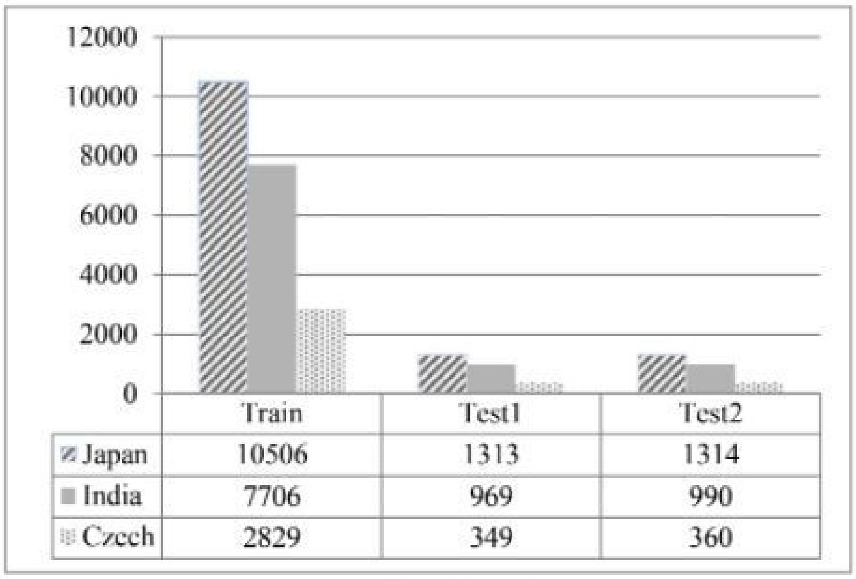
- The first factor is the likelihood that an anchor is an item (it does not take into account the class to which the object belongs.)
- The bounding box regressor is used to modify the anchors to better suit the object.

Boundary boxes of various forms are now transferred to the RoI layer. There will be proposals with no classes assigned to them following the RPN phase. We may crop each proposal such that each proposal contains an item. The RoI pooling layer does this. For each anchor, it generates fixed-size feature maps.

4.2 Dataset

The maintenance of roads is critical to a country's socio-economic growth. This involves a frequent examination of the road's condition, which is often carried out separately by multiple governmental authorities. Some organizations conduct road condition readings with pavement survey trucks outfitted with a variety of sensors to measure the quality and degradation of the pavement. In these vehicles, optical machine vision-based cameras and 3-D sensors are frequently used to image street conditions with maximum clarity and sharpness. There are low-cost technologies that can be used to extensively scanning road surfaces. However, local governments with limited resources can't afford to install such technology on dedicated automobiles. A version of the device that supports assessment for road conditions was recently created by the University of Tokyo (Maeda et al., 2018). The GRDD challenge is intended to advance modern technology in identifying street damage. The challenge consists of two parts: i) an extensive accessible collection of pavement pictures and annotations. (ii) An online competition and workshop. Road photos from three different countries around the world were used to create the GRDDC data. There were three sections of the data: Train, Test1, and Test2. Road images annotated in XML files in PASCAL VOC format have been used to create the train set. The remaining two sets were published without annotations to assess the authors' solutions.

There are different interpretations for the allocation of photographs in three distinct data sets provided in Figure 16. Labelling for various forms of road damage accompanies the training data. And in the next Figure illustrates the quantity of each damage category in the dataset.



	Train	Test1	Test2
▨ Japan	10506	1313	1314
■ India	7706	969	990
░ Czech	2829	349	360

Figure 4.13: Image Distribution

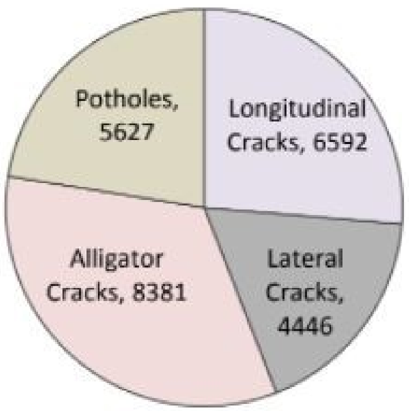


Figure 4.14: Different Damage Types

India, and in particular, Japan and the Czech Republic, are areas of the world that the road damage detection challenge focuses on. There are four distinct types of damage. These cracks are - longitudinal, transverse and alligator on top that potholes are also included. The quality of road security depends on where the data originated from. Furthermore, each form of the fault has its quirks, and there is a significant difference in the degree and size of the fault.

4.3 Proposed Workflow

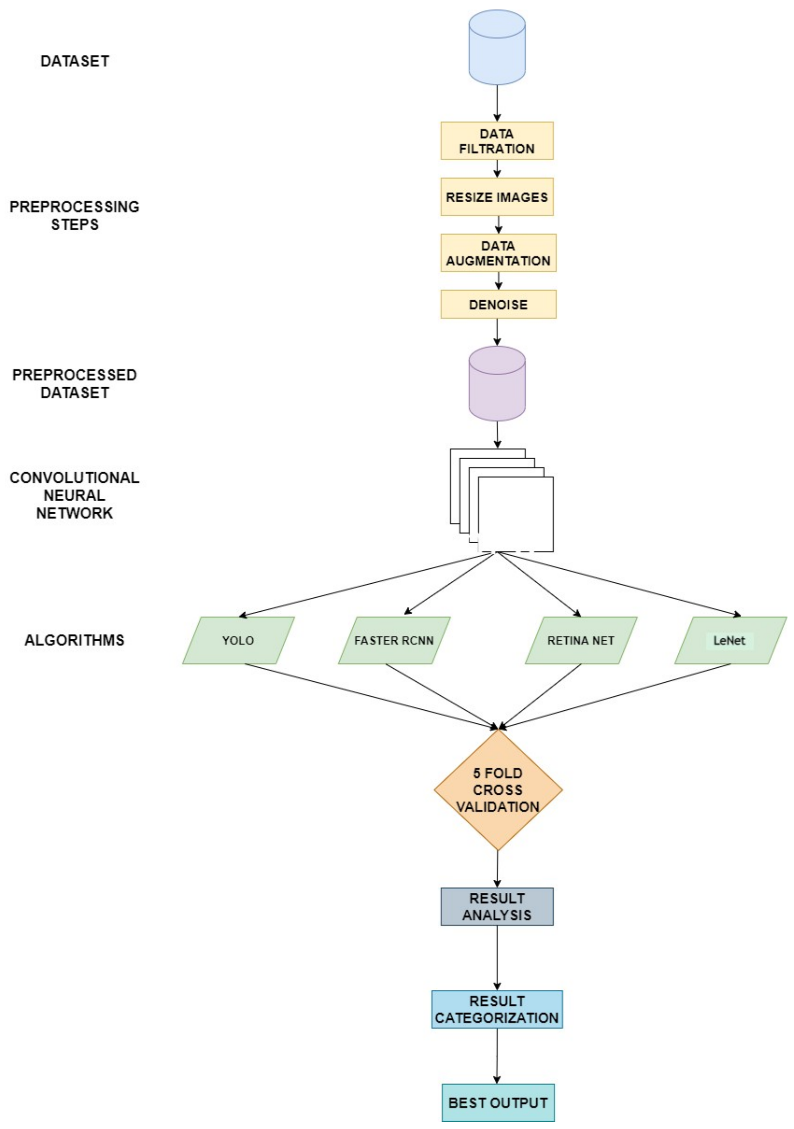


Figure 4.15: Proposed Workflow

Classic Workflow diagram of any image processing method has two objectives; classification and object detection. In our research, our primary goal here is to detect cracks of a damaged road in a real-time process. Though we are working with multiple algorithms, we will use the same dataset for each algorithm as all of the algorithms will work parallelly.
Before sending the data to the CNN phase, the dataset has to go through multiple processes. First, the dataset has to go through a data filtration process where images from the dataset; will be augmented, resized, image de-noised etc. We have to remember that De-noise is an essential part of image processing as a dataset consists of many images, so that data De-noising can occur very frequently. There are two types of de-noising; internal and interference. Internal noise occurs due to the camera's internal errors such as electricity, heat, and sensor illumination levels. Interference occurs due to the high magnetic or radio transmission. However, interference de-noise is a sporadic case. The term resized is used for referring to

mandatory requirements for image quality for data processing. Augmentation is manipulating the existing data to create some more data for the model training process. Then, in the next phase, the filtrated dataset will enter the pre-processing data state. After that, the dataset will enter the CNN (convolution neural network) phase as all the algorithms we are working with; are based on CNN convolution. CNN is used for object recognition or detection. So, in this phase, the CNN will detect the state of the cracks as this is the learning phase, so each image will be assigned weights and biases. After the CNN training phase, each algorithm will apply its activation functions and methods to detect cracks. Then all of the datasets will have to go through Fold in Cross-Validation. Cross-Validation in Fold is when we take a particular dataset and then create multiple datasets randomly. After that, the 'K' number of datasets takes a single dataset for each training phase. The process continues for K times for each anomaly. This is a process that statistically generalizes the result as an independent dataset. Then it will show us the fastest output from the algorithm.

4.4 Data Pre-Processing

4.4.1 Hardware Setup:

PC: Intel(R) Core(TM) i5-9300H CPU @ 2.40GHz 2.40 GHz RAM: 24 Gb GPU: NVIDIA GTX GeForce 1650

4.4.2 Library Preparation:

Numpy(array or vector manipulation), Pandas (For data pre-processing), Seaborn (data visualisation), Matplotlib (data visualisation), OpenCV(object detection), Tensorflow (for neural networks and hyperparameter tuning).

Coming up next are the pre-processing methods that we would perform for the preparation of the data:

DATA FILTRATION: The quality of an induced model is also determined by the training data, which can be assessed by the number of deleterious cases present, for example. Induced models with poor quality training data have poor quality. Searching the training set space for an optimal subset that minimizes the empirical error: argument tP(T) E(g(t,), V), where t is a subset of T and P(T) is the power set of T The induced model is unaffected by the removed instances.

DATA AUGMENTATION: In data analysis, data augmentation refers to procedures that add slightly modified versions of current data or freshly produced data from available data to increase the volume of data accessible. It acts like establishers and aids in the reduction of overfitting while training a machine learning model. Here are some examples of image data augmentation techniques: Position augmentation techniques include rotation and translation. To improve the color of an image, brightness and hue might also be utilized.

RESIZE: In computer vision, resizing images is a crucial step. On tiny photos, our machine learning models learn faster. Furthermore, many deep learning model designs demand that our pictures have the same size, although our raw gathered images may vary in size. Image data is commonly resized in two ways to meet the network's input size. The height and width of a picture are multiplied by a scaling factor when it is rescaled. If the scaling factor is not quite the same both in the two perpendicular directions, rescaling changes the geometric extents of the pixels as well as the image resolution. By cropping we can extract a sub-region of an image while preserving the spatial extent of each pixel.

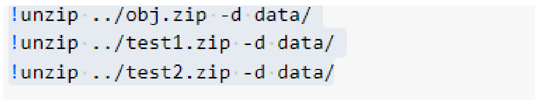
DENOISE: Image de-noising is a basic task in field of computer vision, with the initial objective of estimating the actual picture by dampening noise from a noise-contaminated copy of either the picture. The simplest algorithm for de-noising time-series data is taking a summary statistic using a rolling window. A rolling window collects observations into groups of n size. The groups are shifted one observation at a time, creating a "window" that passes over the dataset.

Chapter 5

Model Implementation

5.1 YOLO-v4

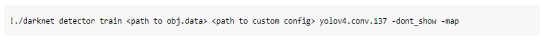
Yolo v4 can be implemented in two ways; pre-trained version or Customized datasets. The variant in the pre-trained set is very limited than the customized dataset with pertained limited classes among them. In the pre-trained, there are 4 classes in total. So, adjusting a large amount of data is pretty tough. Here, the implementation is based on the customized dataset. So, the first step is to gather sufficient datasets and label them accordingly to train them efficiently. But we have collected previously used datasets to compare the result with those previously mentioned in some papers. Also, as we work with 4 algorithms, we can use the same dataset with another algorithm for better comparison. The dataset is also annotated. Then we zipped three files as train and test1, test2. We unzip them in Colab using the shell command.



```
!unzip ../obj.zip -d data/
!unzip ../test1.zip -d data/
!unzip ../test2.zip -d data/
```

Figure 5.1: Unzipping Dataset

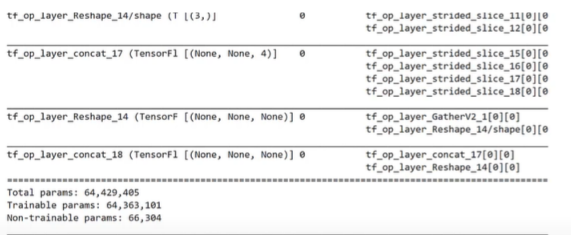
After that, we configured the file by converting it into a .cfg file and .txt file of the test and train dataset. In this phase, we added the batch of 64. Then we used a pre-trained convolutional dataset that can be downloaded from GitHub. Then we started training the dataset using the shell command in the Colab.



```
!./darknet detector train <path to obj.data> <path to custom config> yolov4.conv.137 -dont_show -map
```

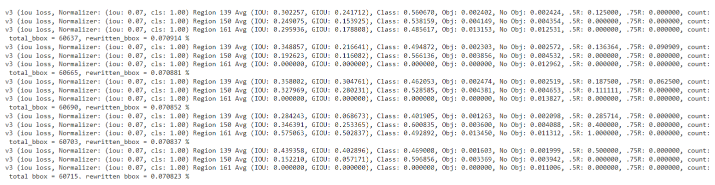
Figure 5.2: Training Shell Command

We implement the trained model using Darknet into the TensorFlow model. We also need DeepSort with Yolo-V4 to implement the trained model as well.
After executing the command shell for the training phase, prompt windows will show us the normalization, IOU, average loss Etc. We can also see the graph after the



```
tf_op_layer_Reshape_14/shape (T [(3,)]              0            tf_op_layer_strided_slice_11[0][0
                                                                 tf_op_layer_strided_slice_12[0][0
__________________________________________________________________________________________________
tf_op_layer_concat_17 (TensorFl [(None, None, 4)]    0            tf_op_layer_strided_slice_15[0][0
                                                                 tf_op_layer_strided_slice_16[0][0
                                                                 tf_op_layer_strided_slice_17[0][0
                                                                 tf_op_layer_strided_slice_18[0][0
__________________________________________________________________________________________________
tf_op_layer_Reshape_14 (TensorF [(None, None, None)] 0            tf_op_layer_GatherV2_1[0][0]
                                                                 tf_op_layer_Reshape_14/shape[0][0
__________________________________________________________________________________________________
tf_op_layer_concat_18 (TensorFl [(None, None, None)] 0            tf_op_layer_concat_17[0][0]
                                                                 tf_op_layer_Reshape_14[0][0]
==================================================================================================
Total params: 64,429,405
Trainable params: 64,363,101
Non-trainable params: 66,304
__________________________________________________________________________________________________
```

Figure 5.3: Trainable and non-trainable parameters



```
v3 (iou loss, Normalizer: (iou: 0.07, cls: 1.00) Region 139 Avg (IOU: 0.302257, GIOU: 0.241712), Class: 0.560670, Obj: 0.002402, No Obj: 0.002424, .5R: 0.125000, .75R: 0.000000, count:
v3 (iou loss, Normalizer: (iou: 0.07, cls: 1.00) Region 150 Avg (IOU: 0.249075, GIOU: 0.153925), Class: 0.538159, Obj: 0.004149, No Obj: 0.004354, .5R: 0.000000, .75R: 0.000000, count:
v3 (iou loss, Normalizer: (iou: 0.07, cls: 1.00) Region 161 Avg (IOU: 0.295936, GIOU: 0.178808), Class: 0.485617, Obj: 0.013153, No Obj: 0.012531, .5R: 0.000000, .75R: 0.000000, count:
 total_bbox = 60637, rewritten_bbox = 0.070914 %
v3 (iou loss, Normalizer: (iou: 0.07, cls: 1.00) Region 139 Avg (IOU: 0.348857, GIOU: 0.216641), Class: 0.494872, Obj: 0.002303, No Obj: 0.002572, .5R: 0.136364, .75R: 0.090909, count:
v3 (iou loss, Normalizer: (iou: 0.07, cls: 1.00) Region 150 Avg (IOU: 0.192623, GIOU: 0.116082), Class: 0.566136, Obj: 0.003856, No Obj: 0.004532, .5R: 0.000000, .75R: 0.000000, count:
v3 (iou loss, Normalizer: (iou: 0.07, cls: 1.00) Region 161 Avg (IOU: 0.000000, GIOU: 0.000000), Class: 0.000000, Obj: 0.000000, No Obj: 0.012962, .5R: 0.000000, .75R: 0.000000, count:
 total_bbox = 60665, rewritten_bbox = 0.070881 %
v3 (iou loss, Normalizer: (iou: 0.07, cls: 1.00) Region 139 Avg (IOU: 0.358002, GIOU: 0.304761), Class: 0.462053, Obj: 0.002474, No Obj: 0.002519, .5R: 0.187500, .75R: 0.062500, count:
v3 (iou loss, Normalizer: (iou: 0.07, cls: 1.00) Region 150 Avg (IOU: 0.327969, GIOU: 0.280231), Class: 0.528585, Obj: 0.004381, No Obj: 0.004653, .5R: 0.111111, .75R: 0.000000, count:
v3 (iou loss, Normalizer: (iou: 0.07, cls: 1.00) Region 161 Avg (IOU: 0.000000, GIOU: 0.000000), Class: 0.000000, Obj: 0.000000, No Obj: 0.013827, .5R: 0.000000, .75R: 0.000000, count:
 total_bbox = 60690, rewritten_bbox = 0.070852 %
v3 (iou loss, Normalizer: (iou: 0.07, cls: 1.00) Region 139 Avg (IOU: 0.284243, GIOU: 0.068673), Class: 0.401905, Obj: 0.001263, No Obj: 0.002098, .5R: 0.285714, .75R: 0.000000, count:
v3 (iou loss, Normalizer: (iou: 0.07, cls: 1.00) Region 150 Avg (IOU: 0.346391, GIOU: 0.253365), Class: 0.600835, Obj: 0.003600, No Obj: 0.004088, .5R: 0.400000, .75R: 0.000000, count:
v3 (iou loss, Normalizer: (iou: 0.07, cls: 1.00) Region 161 Avg (IOU: 0.575063, GIOU: 0.502837), Class: 0.492892, Obj: 0.013450, No Obj: 0.011312, .5R: 1.000000, .75R: 0.000000, count:
 total_bbox = 60703, rewritten_bbox = 0.070837 %
v3 (iou loss, Normalizer: (iou: 0.07, cls: 1.00) Region 139 Avg (IOU: 0.439358, GIOU: 0.402896), Class: 0.469008, Obj: 0.001603, No Obj: 0.001999, .5R: 0.500000, .75R: 0.000000, count:
v3 (iou loss, Normalizer: (iou: 0.07, cls: 1.00) Region 150 Avg (IOU: 0.152210, GIOU: 0.057171), Class: 0.596856, Obj: 0.003369, No Obj: 0.003942, .5R: 0.000000, .75R: 0.000000, count:
v3 (iou loss, Normalizer: (iou: 0.07, cls: 1.00) Region 161 Avg (IOU: 0.000000, GIOU: 0.000000), Class: 0.000000, Obj: 0.000000, No Obj: 0.011006, .5R: 0.000000, .75R: 0.000000, count:
 total bbox = 60715. rewritten bbox = 0.070823 %
```

Figure 5.4: Training Progress

command gets executed. We can see mAP using another shell command to run our detector. We used a box for each crack to indicate it after running the command shell properly.

5.1.1 5-fold Cross Validation

It is pretty tough to validate a very large dataset. So, rather than validate a very large dataset, we can cross-validate. In Cross validate, we randomly take a large portion of the dataset as a training model and a fair test dataset. For example, we have taken 80% train set and 20% test set in our dataset. Then, fork-fold cross-validation, we have distributed the dataset into 5-folds. Each time we validated, we took 1-fold for validation and the rest of the folds for training. All through this sort of low folds is biased, but as our dataset is very large and the capacity of the GPU is not suitable for this dataset, we have to use 5-fold cross-validation. If we had set a higher value, we could have achieved the LOOCV approach to achieve n-fold validation. In our dataset, we have used python code for running the cross-validation. We can also run cross-validation language such as R. There are some ways to ensure higher accuracy and efficiency in cross-validation though we have not shown such a process in our research. Stratified k-fold validation is one of those processes that rearrange the data. But we have graphically measured some of the cross-validation results using the matplotlib library.

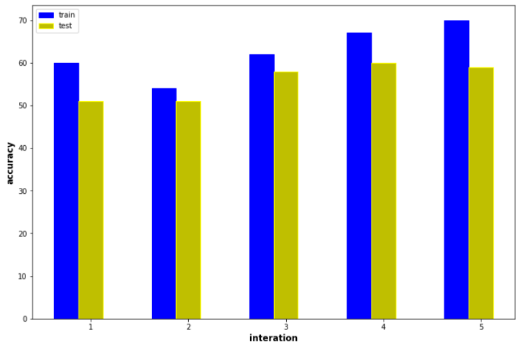


Figure 5.5: Single Fold Validation

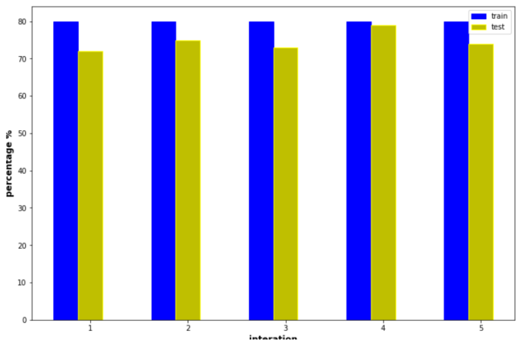
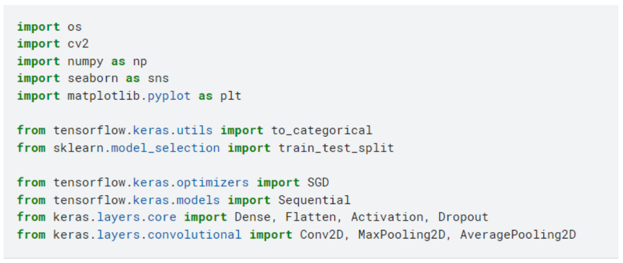


Figure 5.6: Single Fold Accuracy

5.2 LeNet-5

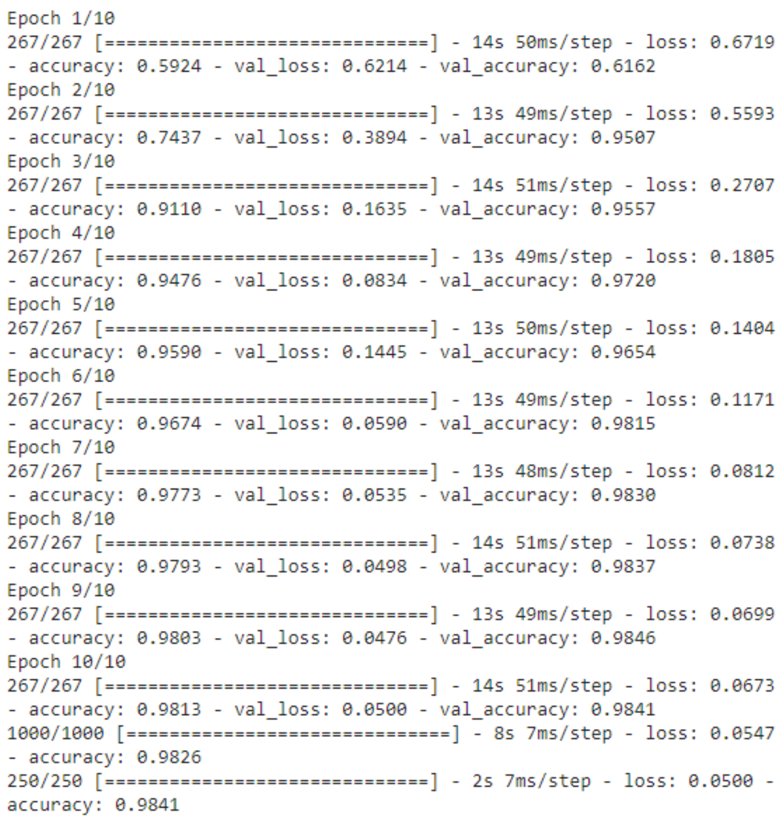
leNet-5 is one of the earliest models that can successfully detect road cracks. Before running properly, this implementation requires certain libraries, such as NumPy, os, TensorFlow, seaborn, etc. As for another implantation (Keras TensorFlow), we can import it. As it is an earlier model, it cannot classify the model, so we have accessed



```
import os
import cv2
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

from tensorflow.keras.utils import to_categorical
from sklearn.model_selection import train_test_split

from tensorflow.keras.optimizers import SGD
from tensorflow.keras.models import Sequential
from keras.layers.core import Dense, Flatten, Activation, Dropout
from keras.layers.convolutional import Conv2D, MaxPooling2D, AveragePooling2D
```

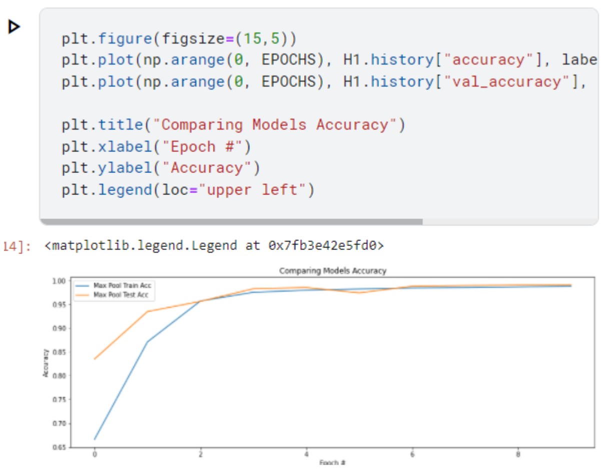
Figure 5.7: Library preparation

a dataset that we connected in the Kaggle and run it, which contained cracked and uncracked images as a negative and positive folder. After that, we label the dataset and resize them with 64. We have 40000 images in the datasets, labelled as 0 and 1. We randomly take images to select the image required for training. We run 7 layers of LeNet-5 on the datasets. It flattens, Dense, Dropout, run activation etc., to train itself. Then we take 10 epochs for the algorithm. After that, we run the epochs on the epochs properly. We will see the accuracy, lose, Val loss, Val accuracy. Result: We used matplotlib to plot a graph of pool train with test pool inaccuracy and loss. It can also be average or max. So, we have to plot 4 different graphs to compare them. This plot makes us understand the basic comparison among the given dataset as it is an earlier model, so it cannot classify the images like another algorithm. It cannot account for video crack detection like Yolo-V4 or other modern algorithms.



```
Epoch 1/10
267/267 [==============================] - 14s 50ms/step - loss: 0.6719
- accuracy: 0.5924 - val_loss: 0.6214 - val_accuracy: 0.6162
Epoch 2/10
267/267 [==============================] - 13s 49ms/step - loss: 0.5593
- accuracy: 0.7437 - val_loss: 0.3894 - val_accuracy: 0.9507
Epoch 3/10
267/267 [==============================] - 14s 51ms/step - loss: 0.2707
- accuracy: 0.9110 - val_loss: 0.1635 - val_accuracy: 0.9557
Epoch 4/10
267/267 [==============================] - 13s 49ms/step - loss: 0.1805
- accuracy: 0.9476 - val_loss: 0.0834 - val_accuracy: 0.9720
Epoch 5/10
267/267 [==============================] - 13s 50ms/step - loss: 0.1404
- accuracy: 0.9590 - val_loss: 0.1445 - val_accuracy: 0.9654
Epoch 6/10
267/267 [==============================] - 13s 49ms/step - loss: 0.1171
- accuracy: 0.9674 - val_loss: 0.0590 - val_accuracy: 0.9815
Epoch 7/10
267/267 [==============================] - 13s 48ms/step - loss: 0.0812
- accuracy: 0.9773 - val_loss: 0.0535 - val_accuracy: 0.9830
Epoch 8/10
267/267 [==============================] - 14s 51ms/step - loss: 0.0738
- accuracy: 0.9793 - val_loss: 0.0498 - val_accuracy: 0.9837
Epoch 9/10
267/267 [==============================] - 13s 49ms/step - loss: 0.0699
- accuracy: 0.9803 - val_loss: 0.0476 - val_accuracy: 0.9846
Epoch 10/10
267/267 [==============================] - 14s 51ms/step - loss: 0.0673
- accuracy: 0.9813 - val_loss: 0.0500 - val_accuracy: 0.9841
1000/1000 [==============================] - 8s 7ms/step - loss: 0.0547
- accuracy: 0.9826
250/250 [==============================] - 2s 7ms/step - loss: 0.0500 -
accuracy: 0.9841
```

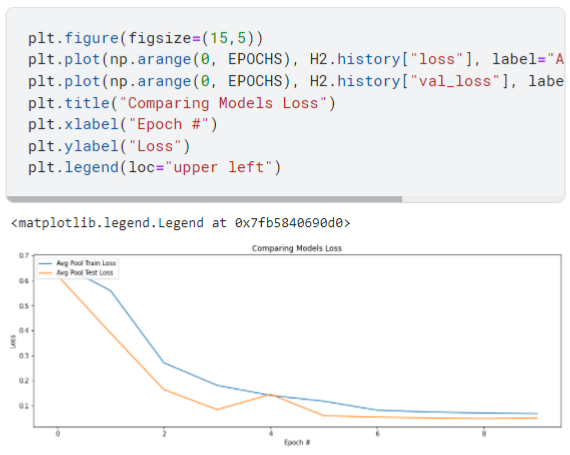
Figure 5.8: Training Progress



```
plt.figure(figsize=(15,5))
plt.plot(np.arange(0, EPOCHS), H1.history["accuracy"], labe
plt.plot(np.arange(0, EPOCHS), H1.history["val_accuracy"],

plt.title("Comparing Models Accuracy")
plt.xlabel("Epoch #")
plt.ylabel("Accuracy")
plt.legend(loc="upper left")
```

```
14]:  <matplotlib.legend.Legend at 0x7fb3e42e5fd0>
```

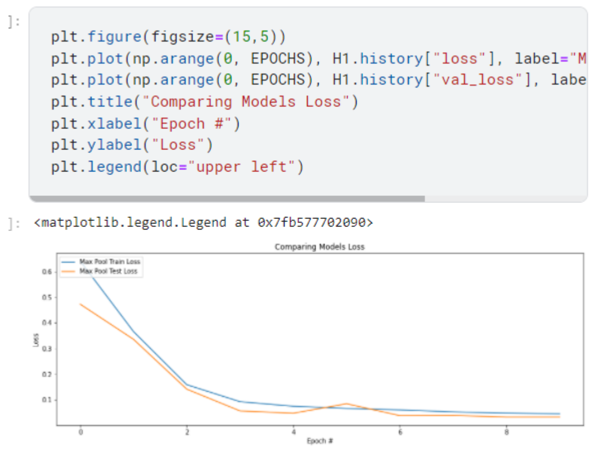
Figure 5.9: : Max pool vs max train accuracy



```
plt.figure(figsize=(15,5))
plt.plot(np.arange(0, EPOCHS), H2.history["loss"], label="A
plt.plot(np.arange(0, EPOCHS), H2.history["val_loss"], labe
plt.title("Comparing Models Loss")
plt.xlabel("Epoch #")
plt.ylabel("Loss")
plt.legend(loc="upper left")
```

```
<matplotlib.legend.Legend at 0x7fb5840690d0>
```

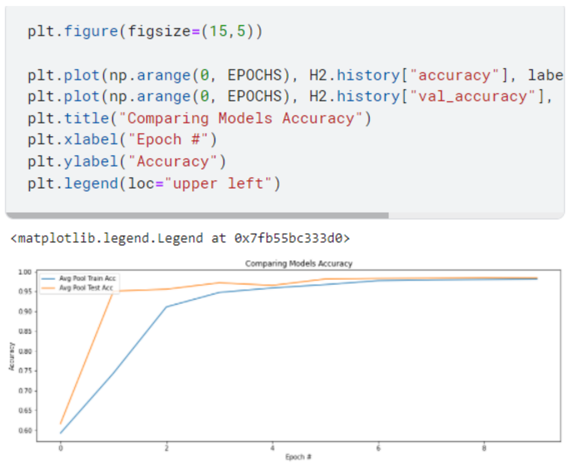
Figure 5.10: Avg pool vs Avg train loss



```
plt.figure(figsize=(15,5))
plt.plot(np.arange(0, EPOCHS), H1.history["loss"], label="M
plt.plot(np.arange(0, EPOCHS), H1.history["val_loss"], labe
plt.title("Comparing Models Loss")
plt.xlabel("Epoch #")
plt.ylabel("Loss")
plt.legend(loc="upper left")
```

```
<matplotlib.legend.Legend at 0x7fb577702090>
```

Figure 5.11: Max pool vs max train loss



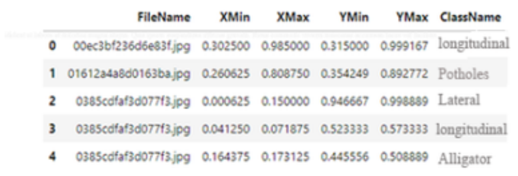
```
plt.figure(figsize=(15,5))

plt.plot(np.arange(0, EPOCHS), H2.history["accuracy"], labe
plt.plot(np.arange(0, EPOCHS), H2.history["val_accuracy"],
plt.title("Comparing Models Accuracy")
plt.xlabel("Epoch #")
plt.ylabel("Accuracy")
plt.legend(loc="upper left")
```

```
<matplotlib.legend.Legend at 0x7fb55bc333d0>
```

Figure 5.12: Avg pool vs Avg train Accuracy

5.3 Faster-RCNN

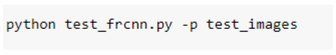
This chapter will explain the implementation process of the proposed system of detecting cracking roads from images. We constructed a more powerful R-CNN neural network. We used google COLAB to train the algorithm as it provides free GPU resources. We have used an already pre-trained version of the dataset, which was annotated previously. In google COLAB, we had given k80 GPU processor to run the algorithm. Before Running the algorithm, we need some libraries to be implemented, for example, pandas, matplotlib, TensorFlow, Keras – 2.0.3, NumPy, OpenCV-python, sklearn, h5py. We can directly install the libraries using the command shell. But before that, we need to down the requirement txt file. Then for the object information, we set the bounding boxes as Ymin, Xmin, Ymax, Xmax, FileName, ClassName and their labels. Next, we will display the bounding box and label. We then resized the input image to (H=416, W=416). To convert the input image to our required output image, the picture will go through the VGG16. We can see the result using the shell command as follows: Then we will use the CVS file



	FileName	XMin	XMax	YMin	YMax	ClassName
0	00ec3bf236d6e83f.jpg	0.302500	0.985000	0.315000	0.999167	longitudinal
1	01612a4a8d0163ba.jpg	0.260625	0.808750	0.354249	0.892772	Potholes
2	0385cdfaf3d077f3.jpg	0.000625	0.150000	0.946667	0.998889	Lateral
3	0385cdfaf3d077f3.jpg	0.041250	0.071875	0.523333	0.573333	longitudinal
4	0385cdfaf3d077f3.jpg	0.164375	0.173125	0.445556	0.508889	Alligator

Figure 5.13: Damage Classification

copied in the repository along with the train and test model to train the dataset. Then we convert the .csv file to a txt file. We minimize the number of epochs to train the data faster. For the data we imported, we ran one of the files as ”mfrncc.hdf5” to assign weight. We trained the model with 100 epochs.



```
python test_frcnn.py -p test_images
```

Figure 5.14: Detection Command

5.4 RetinaNet

RetinaNet outperforms previous single-stage detectors, resulting in models suitable for embedded applications. We followed the procedures outlined below to train the above-mentioned deep learning models. The filtered dataset was randomly combined and divided into two disjoint groups, with 70% of the images used for training and 30% for validation. We employed regularization techniques like dropout and sample augmentation on the larger dataset provided to reduce overfitting in our models.
For model description, training, and validation, Keras and Tensorflow were utilized. Google's collaborative platform, powered by a Tesla K80 GPU, was used for all testing and training. The following settings were used to train the tested architectures: 24 batch sizes and 300 steps in each epoch. As indicated in we employed Adam to optimize the model parameter search by varying the learning rate. Our RetinaNet-based approach outperformed the dataset in [25], particularly for underrepresented classes in the dataset, as the results show.

As previously indicated, we ran all of our experiments with Google Colab, a tool that allowed us to train our models with cloud computing resources. Furthermore, we could host our databases on Google Cloud, reducing computational strain. The picked models were also remotely evaluated using both training and validation data. In addition, we used the Cartucho/mAP evaluation tool to evaluate the proposed technique's capabilities in various scenarios.

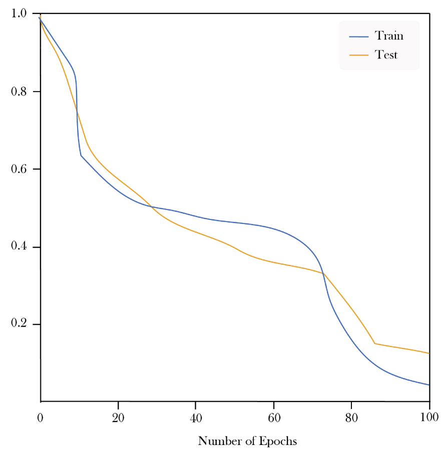
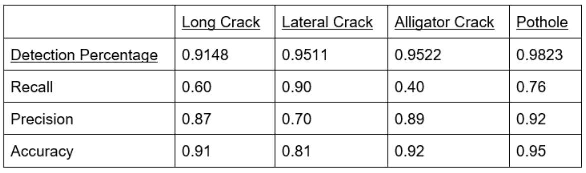


Figure 5.15: Loss vs Epoch curve



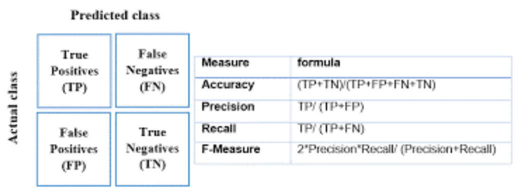
	Long Crack	Lateral Crack	Alligator Crack	Pothole
Detection Percentage	0.9148	0.9511	0.9522	0.9823
Recall	0.60	0.90	0.40	0.76
Precision	0.87	0.70	0.89	0.92
Accuracy	0.91	0.81	0.92	0.95

Figure 5.16: Accuracy

Chapter 6

Discussion

We have worked with four different types of models that have been known throughout time for their development of objection detection algorithms. We have already mentioned the implementation in chapter 5. Also, we have discussed further modification for optimization and hyper tuning etc. Here, we would further discuss the algorithm's performance and point out the basic Comparison among them. For, Comparison we will use confusion matrices to evaluate the basic performance. We can get the accuracy, precision, recall, and F-1 score from the confusion matrices. The equation to get the value is given below: There are certain approaches to get



Measure	formula
Accuracy	(TP+TN)/(TP+FP+FN+TN)
Precision	TP/ (TP+FP)
Recall	TP/ (TP+FN)
F-Measure	2*Precision*Recall/ (Precision+Recall)

Figure 6.1: Predicted Class Actual Curve

precision and recall value from equations. We used the equation for four models. The equations are given below: From the equations below, Cc denotes accurately

$$Precision = \frac{1}{4}\sum_{c=1}^{4}\frac{C_c}{T_c}$$

Figure 6.2: Precision Equation

$$Recall = \frac{1}{4}\sum_{c=1}^{4}\frac{C_c}{A_c}$$

Figure 6.3: Recall Equation

predicted images of class "c." Tc denotes the total or the overall predicted images of class "c." Ac denotes the actual images of class "c".

$$F_1\text{–score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} = \frac{2\text{TP}}{2\text{TP} + \text{FP} + \text{FN}}$$

Figure 6.4: F1 Equation

We can also find the F-1 score from the equation given below:
We have used two datasets. The RDD-2020 dataset for our YOLO-V4, Faster RCNN and Retina-Net evaluation. Another new pre-trained dataset that we get from the Kaggle.We have a total of four types of classification for our research. There is a certain number of images for each of the classified types. So, with the confusion matrix, we have to enlighten that among the total number of images, how many are correctly classified in the category that we have labelled them as. For example, In Yolo-v4, the first of its confusion matrix row of alligator cracks are correctly predicted 94.6% of the total 8381 images as alligator cracks which are approximately 7600 images. Alligator labelled cracks are predicted as 2.1% longitudinal, 1.7% potholes and 1.6% as lateral cracks. Alligator cracks have perfectly correct 94.6% correctly, so according to the indicator, it is deep blue while others are marginally low, so those are almost white. In the later rows, the predictions are less blue for each category of the same cracks, which are lighter than the first row. This is how we use the confusion metrics for evaluating the performance of a classification model. The Confusion matrices for all algorithms are provided below:-

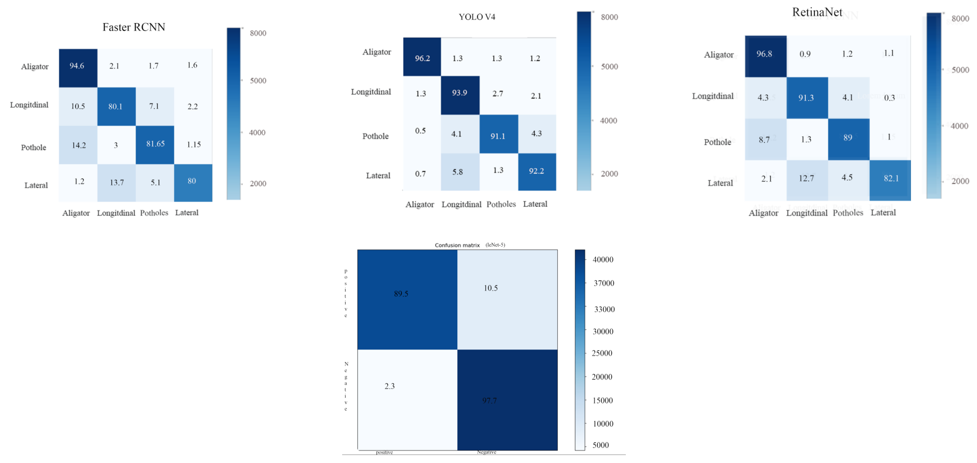


Figure 6.5: Confusion Matrix Of all four Algorithms

Comparison

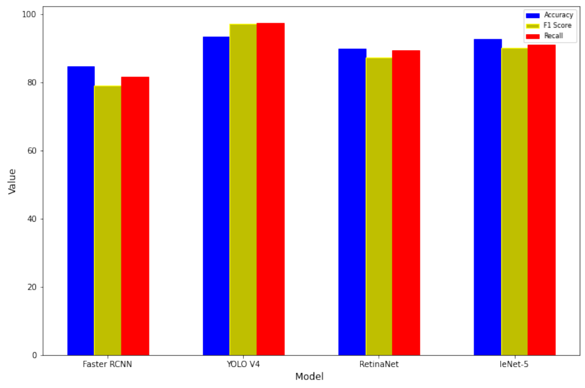


Figure 6.6: Accuracy, F-1 and Recall Comparison

Here, we have compared the value that we achieved through our research on the models. We have compared the Accuracy, Efficiency, Recall and precision to understand the difference in their performance. We have considered four models, and from the graph, we can clearly say that Yolo-V4 outperforms all other 3 models in terms of accuracy, precision and F1 Score. It achieved an accuracy of 93.4%, along with an F1 score of 97%. Closest to it is the Faster-RCNN, which is another widely used algorithm. Retina-Net and LeNet-5 are the other models, but their performance is not equivalent. But they are still being used for many purposes. For example, LeNet-5 is widely known for its use in signature pattern recognition, handwriting reorganization etc.
The outcomes from our algorithms are as follows:

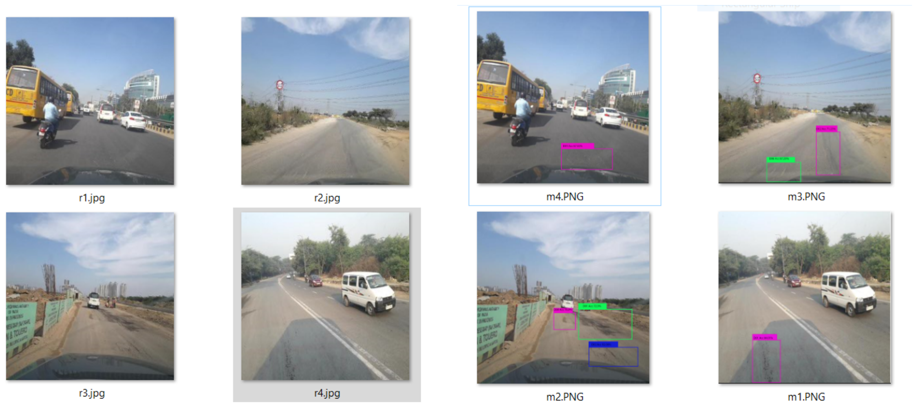


Figure 6.7: Raw and Detected images.

Chapter 7

Conclusion

Our paper has proposed and implemented different latest state-of-the-art convolutional neural network (CNN) architectures to train our dataset. Our training execution was neat and orderly. Our major goal here is to highlight the finest technique that can assist us in determining the best algorithm and their methods of detecting damaged roads in real-time. Different algorithms use various techniques to identify cracks. Hence we used four algorithms to detect the damaged road, and then the result would be categorized into three classes as discussed in the paper. However, some methods are superior to others, and our study may provide us with the chance to produce a breakthrough in the field of real-time damaged road identification.

7.1 Limitations

While working on this project, we encountered a number of issues and drawbacks. First and foremost, our initial idea was to make a local dataset in Bangladesh by performing fieldwork to include local images and road access conditions in the country. However, due to the global epidemic and health risks, we were unable to collect the necessary data. Next, while the accuracy percentages obtained from our training models are significant, there is still room for improvement because our another goal was to achieve values greater than 90%. Finally, the margin of loss is another area where the values were less than satisfactory, and we can reduce the loss here by analyzing some of the insights from the models and attempting to change it further for better accuracy with available regularization techniques. Because the approach we are focusing on is mainly focusing on our country's infrastructure, the increased accuracy and lower data loss are crucial to have a better understanding of damages in roads of our country.

7.2 Future Works

In terms of our research, the models we trained and described, successfully attained a high level of accuracy. As a result, our recent research can serve as a foundation for future research and development on road damage detection in Bangladesh. Our prime objective is to first perfect and enhance the models in order to ensure higher accuracy and reliability. Another aspect that needs to be improved is the percentage of loss of data we experienced all through data processing, which we will work to reduce and thus make it more effective. We also intend to work with regional data to understand and predict statistical data in our country, which we have been unable to do due to the current pandemic. Furthermore, we intend to apply Damage Detection to other research areas being conducted in the object detection fields to make information more accessible and open for people.

Bibliography

[1] Araya, L., Espada, N., Tosini, M., Leiva, L. (2018). Simple detection and classification of road lanes based on image processing. Int. J. Inf. Technol. Comput. Sci.(IJITCS), 10(8), 38–45.

[2] Chun, P.j., HASHIMOTO, K., KATAOKA, N., KURAMOTO, N., OHGA, M. (2015). Asphalt pavement crack detection using image processing and naive bayes based machine learning approach. Journal of Japan Society of Civil Engineers, Ser. E1 (Pavement Engineering), 70(3).

[3] Maeda, H., Sekimoto, Y., Seto, T. (2016). Lightweight road manager: smartphone-based automatic determination of road damage status by deep neural network. In Proceedings of the 5th ACM SIGSPATIAL International Workshop on Mobile Geographic Information Systems (pp. 37–45).

[4] Zhang, A., Wang, K. C., Li, B., Yang, E., Dai, X., Peng, Y., Fei, Y., Liu, Y., Li, J. Q., and Chen, C. (2017). Automated pixel-level pavement crack detection on 3d asphalt surfaces using a deep-learning network. Computer-Aided Civil and Infrastructure Engineering, 32(10):805–819.

[5] Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R., and LeCun, Y. (2013). Over feat: Integrated recognition, localization and detection using convolutional networks. arXiv preprint arXiv:1312.6229.

[6] Huval, B., Wang, T., Tandon, S., Kiske, J., Song,W., Pazhayampallil, J., Andriluka, M., Rajpurkar, P.,Migimatsu, T., Cheng-Yue, R., et al. (2015). An empirical evaluation of deep learning on highway driving. arXiv preprint arXiv:1504.01716.

[7] Maeda, H; Sekimoto, Y; Seto, T; Kashiyama, T. et al. (2018) Road Damage Detection Using Deep Neural Networks with Images Captured Through a Smartphone, Hiroshi Omata University of Tokyo, 4-6-1 Komaba, Tokyo, Japan.

[8] R. Fan et al., "Road Crack Detection Using Deep Convolutional Neural Network and Adaptive Thresholding," 2019 IEEE Intelligent Vehicles Symposium (IV), 2019, pp. 474-479, doi: 10.1109/IVS.2019.8814000.

[9] S. Mathavan, K. Kamal, and M. Rahman, "A review of threedimensional imaging technologies for pavement distress detection and measurements," IEEE Transactions on Intelligent Transportation Systems, vol. 16, no. 5, pp. 2353–2362, 2015.

[10] Tzogka, C., Refanidis, I. (2021). Addressing Computer Vision Challenges Using an Active Learning Framework. In International Conference on Engineering Applications of Neural Networks (pp. 259–270).

[11] Mauri, A., Khemmar, R., Decoux, B., Haddad, M., Boutteau, R. (2021). Real-Time 3D Multi-Object Detection and Localization Based on Deep Learning for Road and Railway Smart Mobility. Journal of Imaging, 7(8).

[12] Hossain, S., Lee, D.j. (2019). Deep learning-based real-time multiple-object detection and tracking from aerial imagery via a flying robot with GPU-based embedded devices. Sensors, 19(15), 3371.

[13] Bochkovskiy, A., Wang, C.Y., Liao, H.Y. (2020). Yolov4: Optimal speed and accuracy of object detection. arXiv preprint arXiv:2004.10934.

[14] He, K., Zhang, X., Ren, S., Sun, J. (2015). Spatial pyramid pooling in deep convolutional networks for visual recognition. IEEE transactions on pattern analysis and machine intelligence, 37(9), 1904–1916.

[15] Liu, S., Qi, L., Qin, H., Shi, J., Jia, J. (2018). Path aggregation network for instance segmentation. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 8759–8768).

[16] Chen, K., Chen, Y., Zhou, H., Mao, X., Li, Y., He, Y., Xue, H., Zhang, W., Yu, N. (2020). Self-supervised adversarial training. In ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (pp. 2218–2222).