

DEEP LEARNING BASED POTHOLE DETECTION SYSTEM

Submitted in partial fulfillment of the requirements for the award of
Bachelor of Engineering degree in Computer Science and Engineering

By

RAJEEV NAYAN (Reg.No - 39110828)
RAHUL ROONWAL (Reg.No – 39110825)



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF COMPUTING

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)

Accredited with Grade “A” by NAAC | 12B Status by UGC | Approved by AICTE
JEPPIAAR NAGAR, RAJIV GANDHI SALAI,
CHENNAI - 600119

APRIL - 2023



SATHYABAMA

**INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

Accredited with Grade "A" by NAAC | 12B Status by UGC | Approved by AICTE

www.sathyabama.ac.in

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **Rajeev Nayan (Reg.No - 39110828)** and **Rahul Roonwal (Reg.No - 39110825)** who carried out the Project Phase-2 entitled **"DEEP LEARNING BASED POTHOLE DETECTION SYSTEM"** under my supervision from January 2023 to April 2023.

Internal Guide

Ms. DEEPA. D M.E.

Head of the Department

Dr. L. LAKSHMANAN, M.E., Ph.D.



Submitted for Viva voce Examination held on 20.04.2023

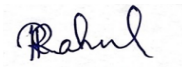
Internal Examiner

External Examiner

DECLARATION

I, **Rahul Roonwal** (Reg.No- 39110825), hereby declare that the Project Phase-2 Report entitled "**DEEP LEARNING BASED POTHOLE DETECTION SYSTEM**" done by me under the guidance of **Ms. Deepa D., M.E.** is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in **Computer Science and Engineering**.

DATE: 20.04.2023

A handwritten signature in blue ink, appearing to read 'Rahul', is placed over a light gray rectangular background.

PLACE: Chennai

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management of SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T.Sasikala M.E., Ph. D, Dean**, School of Computing, **Dr. L. Lakshmanan M.E., Ph.D.**, Head of the Department of Computer Science and Engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Ms. Deepa D** ,for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my phase-2 project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

ABSTRACT

India has a vast network of roads which connects the various cities and villages, due to which its maintenance becomes a challenging task and also many accidents takes place because of the damaged roads and also the accidents are increasing every year because of the increase in the number of potholes. As the inspection of these roads for damages is done manually, a significant amount of time is consumed, cost increases as it is a labor intensive process and also the possibility of human error increases. Also, as the road network is very large, it is not feasible for the Authority travel to all the places for inspection. In order to solve this problem and to increase the efficiency we can use various Image Processing techniques (such as CNN, YOLO V5) in order to find the damages and then comparing the same to find the technique with maximum accuracy. This data can further be sent to the government for repairing the roads and this helps in reducing accidents and further ensuring the safety of the citizens especially during rainy season. By using these techniques, the problem of damaged roads can be solved and also a lot of resources can be saved by repairing the roads in the right time and also by reducing the labor cost.

Chapter No	TITLE		Page No.
	ABSTRACT		v
	LIST OF FIGURES		viii
	LIST OF TABLES		ix
1	INTRODUCTION		1
2	LITERATURE SURVEY		7
	2.1 Inferences from Literature Survey		
	2.2 Open problems in Existing System		9
3	REQUIREMENTS ANALYSIS		10
	3.1	Feasibility Studies/Risk Analysis of the Project	10
	3.2	Software Requirements Specification Document	12
	3.3	System Use case	13
4	DESCRIPTION OF PROPOSED SYSTEM		15
	4.1	Selected Methodology or process model	15
	4.2	Architecture / Overall Design of Proposed System	17
	4.3	Description of Software for Implementation and Testing plan of the Proposed Model/System	19
	4.4	Project Management Plan	19
	4.5	Transition/ Software to Operations Plan	19
5	IMPLEMENTATION DETAILS		22
	5.1	Development and Deployment Setup	22
	5.2	Algorithms	25
	5.3	Testing	26
6	RESULTS AND DISCUSSION		29
7	CONCLUSION		30
	7.1	Conclusion	30
	7.2	Future work	30
	7.3	Research Issues	30

			32
	7.4	Implementation Issues	
	REFERENCES		34
	APPENDIX		36
	A. SOURCE CODE		36
	B. SCREENSHOTS		47
	C. RESEARCH PAPER		52

TABLE OF CONTENTS

FIGURE NO	FIGURE NAME	Page No.
1.1	Machine Learning	2
1.2	Deep Learning	3
1.3	Image Recognition	4
1.4	Object Detection Using CNN	5
1.5	Yolo v5	6
4.1	Methodology Flow Diagram	16
4.2	System Architecture Diagram	17
4.3	YOLO v5 Architecture	18
5.1	Sample Dataset Image	22
5.2	Sample Dataset Image – Plain and Pothole Image	23
6.1	Example Output from YOLOv5 Model	29

LIST OF FIGURES

TABLE NO	TABLE NAME	Page No.
6.1	YOLOv5 Results	9

LIST OF TABLES

CHAPTER 1

INTRODUCTION

In recent years, due to advancement in transportation, poorly maintained highways and roads results in traffic and also leads to accidents. This can be reduced by properly maintaining the roads and filling the potholes regularly which also reduces the maintenance cost. As road inspection is done manually, it becomes a time consuming job and it also requires human labour and it is also subjected to errors. Due to the damaged roads, traffic jams occur which is indirectly responsible for economic losses. Also, if these damaged roads are not repaired timely the conditions of the roads may become worse and the cost of repairing these roads also increases immensely. But it is difficult to monitor the road conditions by visiting all the roads and checking for the damages manually as it is a labor-intensive process and it also involves high inspection cost. In order to solve these problems modern techniques can be used. In the proposed system, first the images of the roads are collected after which the images are then being processed using various machine learning algorithms such as YOLO V5, CNN, etc. The dataset used in this model for the purpose training, testing and for validation is collected from various websites such as Roboflow , Kaggle ,etc .Then the dataset collected is divided into three smaller datasets in which one is used for training purpose, the other dataset is used for the purpose of testing and the third set is used after the model is trained and tested, for the purpose of validation. After the process of training, testing and validation, the algorithm with the most accuracy is used to develop an application in which the video of the road is processed and is converted into images after which the images are processed using the algorithm with the best accuracy and the potholes are found. These reports can then be sent to the concerned department to look into the matter and repairing the damaged roads thus reducing the traffic and accidents.

1.1 MACHINE LEARNING

Machine learning (ML) is the study of computer algorithms that can improve automatically through experience and by the use of data. It is seen as a part of artificial intelligence. Machine learning algorithms build a model based on sample data, known as training data, in order to make predictions or decisions without being explicitly programmed to do so. Machine learning algorithms are used in a wide variety of applications, such as in medicine, email filtering, speech recognition, and computer vision, where it is difficult or unfeasible to develop conventional algorithms to perform the needed tasks.

A subset of machine learning is closely related to computational statistics, which focuses on making predictions using computers; but not all machine learning is statistical learning. The study of mathematical optimization delivers methods, theory and application domains to the field of machine learning. Data mining is a related field of study, focusing on exploratory data analysis through unsupervised learning. Some implementations of machine learning use data and neural networks in a way

that mimics the working of a biological brain. In its application across business problems, machine learning is also referred to as predictive analytics.

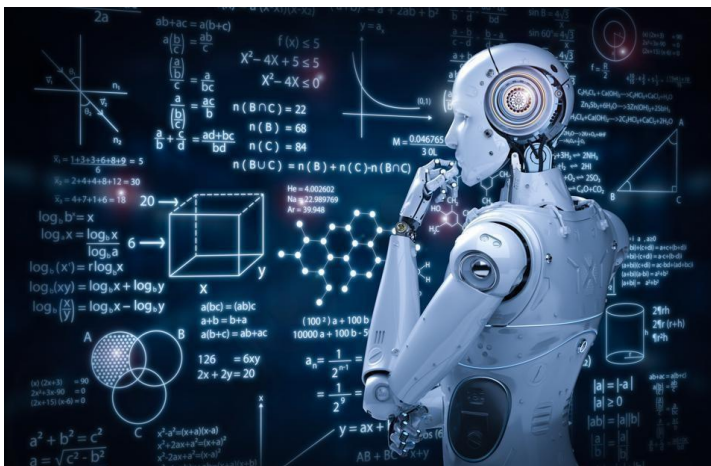
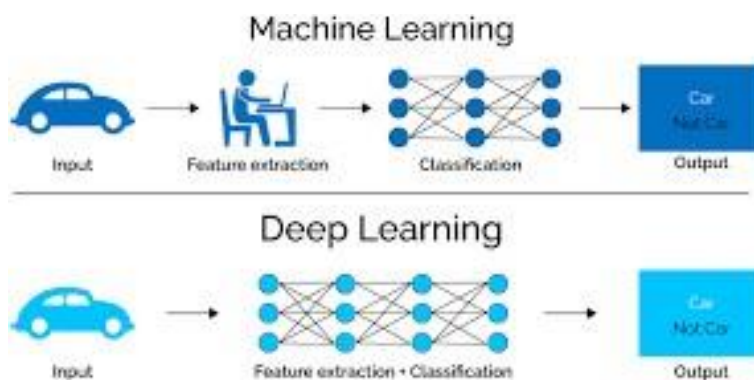


Fig 1.1: MACHINE LEARNING

1.2 DEEP LEARNING

Deep learning (also known as deep structured learning) is part of a broader family of machine learning methods based on artificial neural networks with representation learning. Learning can be supervised, semi-supervised or unsupervised.

Deep-learning architectures such as deep neural networks, deep belief networks, deep reinforcement learning, recurrent neural networks, convolutional neural networks and Transformers have been applied to fields including computer vision, speech recognition, natural language processing, machine translation, bioinformatics, drug design, medical image analysis, climate science, material inspection and board game programs, where they have produced results comparable to and in some cases surpassing human expert performance



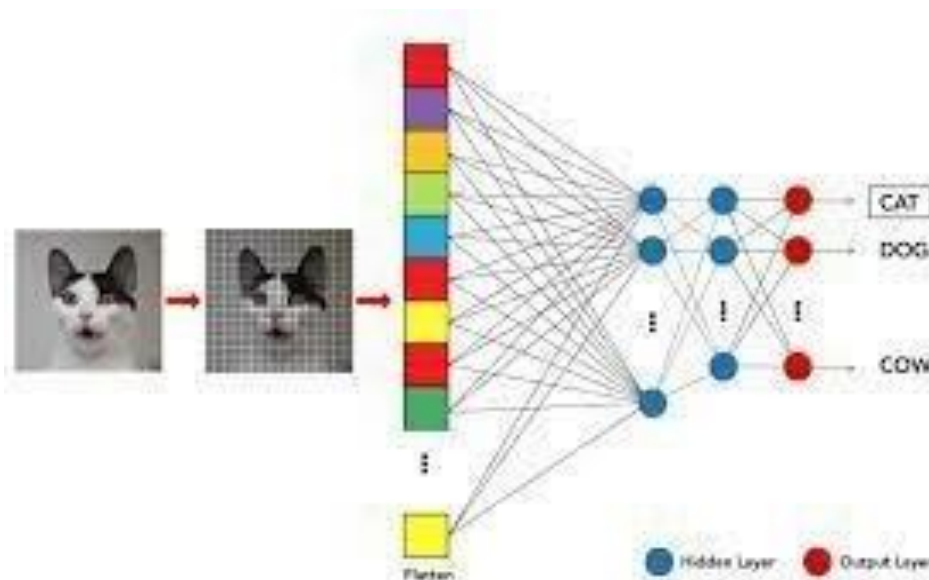
Artificial neural networks (ANNs) were inspired by information processing and distributed communication nodes in biological systems. ANNs have various differences from biological brains.

Specifically, artificial neural networks tend to be static and symbolic, while the biological brain of most living organisms is dynamic (plastic) and analogue.

Fig 1.2: DEEP LEARNING

1.3 IMAGE RECOGNITION

A common evaluation set for image classification is the MNIST database data set.



MNIST is composed of handwritten digits and includes 60,000 training examples and 10,000 test examples. As with TIMIT, its small size lets users test multiple configurations. A comprehensive list

of results on this set is available.

Deep learning-based image recognition has become "superhuman", producing more accurate results than human contestants. This first occurred in 2011 in recognition of traffic signs, and in 2014, with recognition of human faces.

Deep learning-trained vehicles now interpret 360° camera views. Another example is Facial Dysmorphology Novel Analysis (FDNA) used to analyze cases of human malformation connected to a large database of genetic syndromes.

Fig 1.3: IMAGE RECOGNITION

1.4 CONVOLUTIONAL NEURAL NETWORK (CNN)

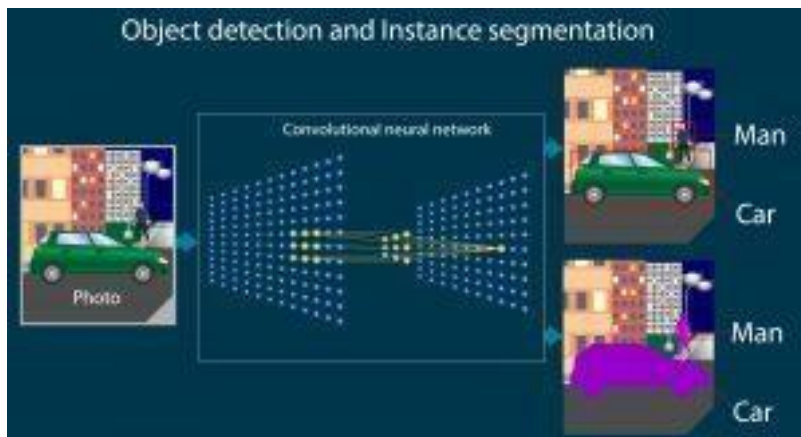
A Convolutional Neural Network or CNN is a type of artificial neural network, which is widely used for image/object recognition and classification. Deep Learning thus recognizes objects in an image by using a CNN. CNNs are playing a major role in diverse tasks/functions like image processing problems, computer vision tasks like localization and segmentation, video analysis, to recognize obstacles in self-driving cars, as well as speech recognition in natural language processing. As CNNs are playing a significant role in these fast-growing and emerging areas, they are very popular in Deep Learning.

CNNs have fundamentally changed our approach towards image recognition as they can detect patterns and make sense of them. They are considered the most effective architecture for image classification, retrieval and detection tasks as the accuracy of their results is very high.

They have broad applications in real-world tests, where they produce high-quality results and can do a good job of localizing and identifying where in an image a person/car/bird, etc., are. This aspect has made them the go-to method for predictions involving any image as an input.

Fig 1.4: OBJECT DETECTION USING CNN

1.5 YOLO v5 ALGORITHM



YOLOv5 is a family of compound-scaled object detection models trained on the COCO dataset, and includes simple functionality for Test Time Augmentation (TTA),

model ensembling, hyperparameter evolution, and export to ONNX, CoreML and TFLite.

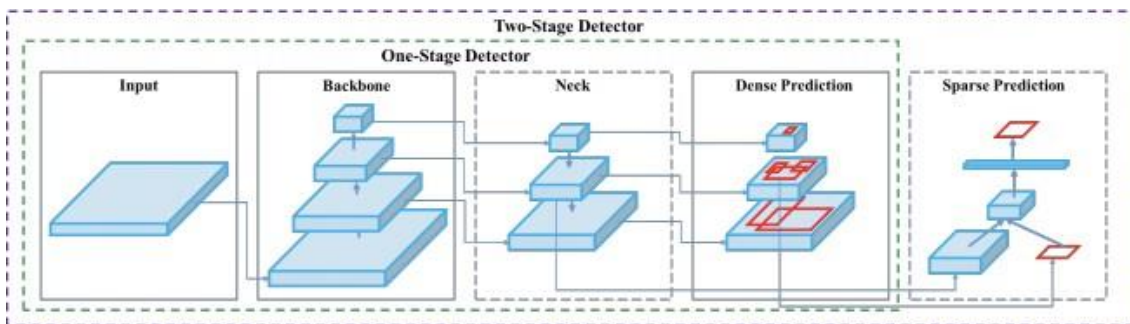
The YOLO family of models consists of three main architectural blocks i) Backbone, ii) Neck and iii) Head.

1. YOLOv5 Backbone: It employs CSPDarknet as the backbone for feature extraction from images consisting of cross-stage partial networks.
2. YOLOv5 Neck: It uses PANet to generate a feature pyramids network to perform aggregation on the features and pass it to Head for prediction.
3. YOLOv5 Head: Layers that generate predictions from the anchor boxes for object detection.

Apart from this YOLOv5 uses the below choices for training –

1. Activation and Optimization: YOLOv5 uses leaky ReLU and sigmoid activation, and SGD and ADAM as optimizer options.
2. Loss Function: It uses Binary cross-entropy with logits loss.

Fig 1.5: Yolo v5



CHAPTER 2

LITERATURE SURVEY

This problem statement has been extensively studied over the past years by researchers in a bid to create a solution, and all their solutions vary from the algorithms used to the approaches taken for predicting the damaged roads.

The work of Mr. Anup Kumar Pandey [1] introduced a system for the detection of potholes in which the accelerometer present in the phone was used to collect the data which was further processed by training the ID-CNN model.

Deepak Kumar Dewangan [6] introduced a system for the inspection of roads and for finding the damaged roads in which a smartphone and a raspberry pi camera is used and the whole setup is placed on the vehicle and the pi camera records the road condition and then the recorded video is converted into images which is then processed and the pothole is detected using the various CNN algorithms.

Lei Chen [15] presented an image recognition system in which various techniques are used to process the given images and finding the agricultural disease using image processing. In this system, CNN model is used for detecting the agricultural disease.

Nachuan Ma [11] proposed a system for road imaging and pothole detection using various cameras , laser scanners and various other devices for road imaging which is placed on the vehicle and the images are then processed using various machine learning algorithms for pothole detection.

Ratnajit Mukherjee [13] proposed an AI based road maintenance inspection in which the images are processed for predicting the damaged roads and damaged road signs using RGPNet algorithm and then visualizing the damages on the map.

Dharneeshkar J [7] proposed a system for pothole detection which uses various image processing techniques such as YOLO v3, YOLO v2, etc and the results obtained are then compared using precision and recall for best accuracy.

Kang BH and Choi SI [2] developed a model for the detection of potholes using a 2D Lidar sensor and a Camera in which first the images are captured using a 2D Lidar sensor and the potholes are detected using various algorithms and after this step, the Camera setup is incorporated to increase its accuracy.

Kavitha R and Nivetha S [10] proposed a system for the detection of potholes which uses a Raspberry pi for recording images and then using various object detection techniques for the detection of potholes.

K. Vigneshwar, B. Hema Kumar [9] developed a system for pothole detection using various machine learning techniques such as clustering based on image segmentation and Gaussian-Filtering for the detection of potholes.

2.1 INFERENCES FROM LITERATURE SURVEY

Based on the studies from the literature survey, it is noted that the pothole detection problem is solved using various algorithms and different approaches are followed but with less accuracy and only some algorithms are used in a particular approach. Also, it is noted that the latest algorithms are not implemented for pothole detection.

2.2 OPEN PROBLEMS IN EXISTING SYSTEM

The main problem in the existing system is that the accuracy in the existing system is less as well as only few algorithms are used for prediction and only images can be used in the existing system. So, in our system we are trying to use videos and various algorithms such as Yolo V5, CNN, etc for predicting the potholes and the algorithm with the maximum accuracy is used.

CHAPTER 3

REQUIREMENT ANALYSIS

3.1 FEASIBILITY STUDIES/RISK ANALYSIS OF THE PROJECT

To conduct a feasibility study and risk analysis for a pothole detection system using YOLOv5 and CNN, we need to consider the following:

3.1.1 FEASIBILITY STUDY:

3.1.1.1 TECHNICAL FEASIBILITY:

Evaluate the technical feasibility of implementing a pothole detection system using YOLOv5 and CNN. This would involve understanding the technical aspects of YOLOv5 and CNN and assessing whether they are suitable for this project.

3.1.1.2 ECONOMIC FEASIBILITY:

Evaluate the economic feasibility of implementing the system. This would involve assessing the costs associated with developing the system and the potential revenue generated from its implementation.

3.1.1.3 OPERATIONAL FEASIBILITY:

Evaluate the operational feasibility of implementing the system. This would involve understanding how the system would be integrated into the existing road infrastructure, such as how data would be collected and how it would be transmitted to maintenance teams.

3.1.2 RISK ANALYSIS:

3.1.2.1 TECHNICAL RISKS:

Identify the technical risks associated with implementing the system. This would involve identifying potential technical issues that could arise during the development process, such as issues with data quality or accuracy.

3.1.2.2 OPERATIONAL RISKS:

Identify the operational risks associated with implementing the system. This would involve understanding how the system would be used in the real world and identifying potential issues that could arise during its operation, such as false positives or false negatives.

3.1.2.3 ECONOMIC RISKS:

Identify the economic risks associated with implementing the system. This would involve understanding the financial risks associated with the project, such as cost overruns or potential revenue shortfalls.

3.1.2.4 LEGAL RISKS:

Identify the legal risks associated with implementing the system. This would involve understanding the legal framework surrounding the use of such a system, such as data privacy laws or liability issues.

Based on the above methods, the feasibility of the project is being studied and some risks are also analyzed such as if the speed of the vehicle is too fast the system may not be able to detect the potholes from the video or the images captured and also as the accuracy is not 100% there is a chance that some potholes can be missed and left undetected and if the image is not of good resolution then it would be difficult to detect the potholes. Also, the model is not trained on small potholes. So, it is recommended that when images are captured the speed of the vehicle should be less so that clear images are captured.

Now, as the feasibility study and risk analysis of the project is completed, we can use the findings to make informed decisions about whether to proceed with the project, how to design the system, and how to manage potential risks.

3.2 SOFTWARE REQUIREMENTS SPECIFICATION DOCUMENT

The various software used for developing a model for pothole detection are:

3.2.1 *Python:*

Python is a high-level, general-purpose programming language. Its design philosophy emphasizes code readability with the use of significant indentation. Python is dynamically-typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly procedural), object-oriented and functional programming. It is often described as a "batteries included" language due to its comprehensive standard library.

3.2.2 *Jupyter Notebook:*

Project Jupyter is a project with goals to develop open-source software, open standards, and services for interactive computing across multiple programming languages. It was spun off from IPython in 2014 by Fernando Pérez and Brian Granger. Project Jupyter's name is a reference to the three core programming languages supported by Jupyter, which are Julia, Python and R.

3.2.3 Google Colab:

It is a product from Google Research. Colab allows anybody to write and execute arbitrary python code through the browser, and is especially well suited to machine learning, data analysis and education. More technically, Colab is a hosted Jupyter notebook service that requires no setup to use, while providing access free of charge to computing resources including GPUs.

3.2.4 VS Code:

Visual Studio Code, also commonly referred to as VS Code,[9] is a source-code editor made by Microsoft with the Electron Framework, for Windows, Linux and macOS.[10] Features include support for debugging, syntax highlighting, intelligent code completion, snippets, code refactoring, and embedded Git. Users can change the theme, keyboard shortcuts, preferences, and install extensions that add additional functionality.

3.3 SYSTEM USE CASE:

A pothole detection system model developed using YOLOv5 and CNN can have the following use case:

3.3.1 INPUT DATASET COLLECTION:

The system would collect a dataset of images that contain potholes. This dataset would be used to train the system to recognize potholes in images.

3.3.2 PREPROCESSING:

The input dataset would be preprocessed to ensure that it is suitable for training the model. This would involve resizing the images, normalizing the pixel values, and labeling the potholes in the images.

3.3.3 TRAINING:

The system would use the preprocessed dataset to train a YOLOv5 and CNN model to detect potholes in images. This process involves feeding the labeled images to the model and adjusting the model parameters until it can accurately detect potholes in new images.

3.3.4 TESTING:

Once the model has been trained, it would be tested using a separate dataset of images that contain potholes. This dataset would be used to evaluate the accuracy of the model and to identify areas where the model may need improvement.

3.3.5 DEPLOYMENT:

Finally, the pothole detection system would be deployed to the real world, where it would be used to detect potholes in images captured by cameras or other sensors. The system could be integrated with other systems such as navigation apps, which could use the information to provide alternate routes to avoid potholes or alert drivers to the presence of potholes on their current route.

CHAPTER 4

DESCRIPTION OF PROPOSED SYSTEM

Looking at the disadvantages of all the above methodologies used in previous systems, the most common point that pops up is in most of the algorithms proposed and used have less accuracy and the approaches followed are also less feasible which makes a way for us to propose a system with the maximum accuracy. For this various algorithms are first implemented such as YOLO V5, CNN, etc and the one with maximum accuracy is used. In the available system only few algorithms are used for a particular approach due to which the maximum accuracy can't be attained. In our system first the video of the road is fed into the system which is then converted into images after which the images are processed in order to classify the images with damaged roads.

4.1 SELECTED METHODOLOGY OR PROCESS MODEL

The process model followed for the purpose of detection of pothole is that, as the dataset for the road images is not available in India, we should first collect the image dataset and then divide the images for training and testing purpose and then apply the various machine learning techniques such as pre-processing and then apply various algorithms for detecting the potholes and then finding the best algorithm with the highest accuracy.

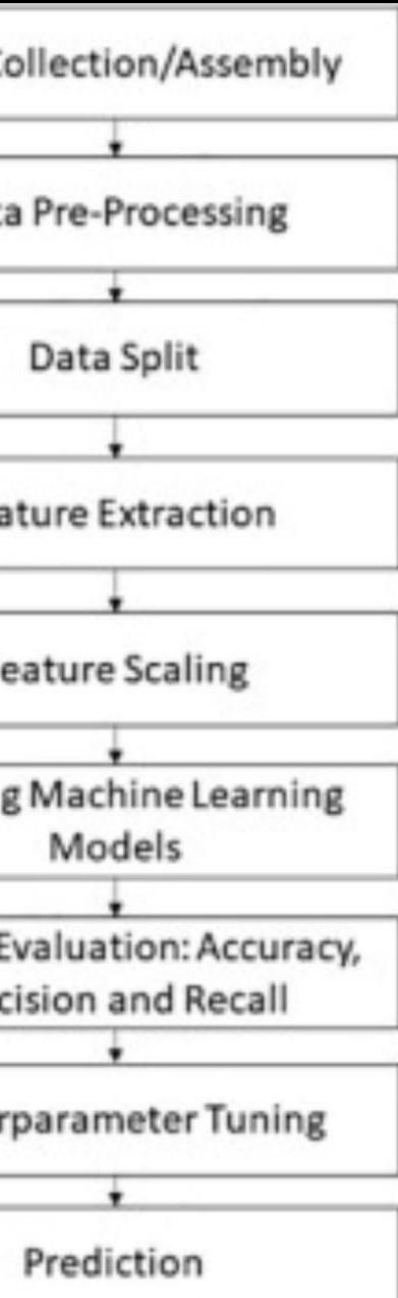


Fig 4.1: Methodology Flow Diagram

4.2 ARCHITECTURE / OVERALL DESIGN OF PROPOSED SYSTEM

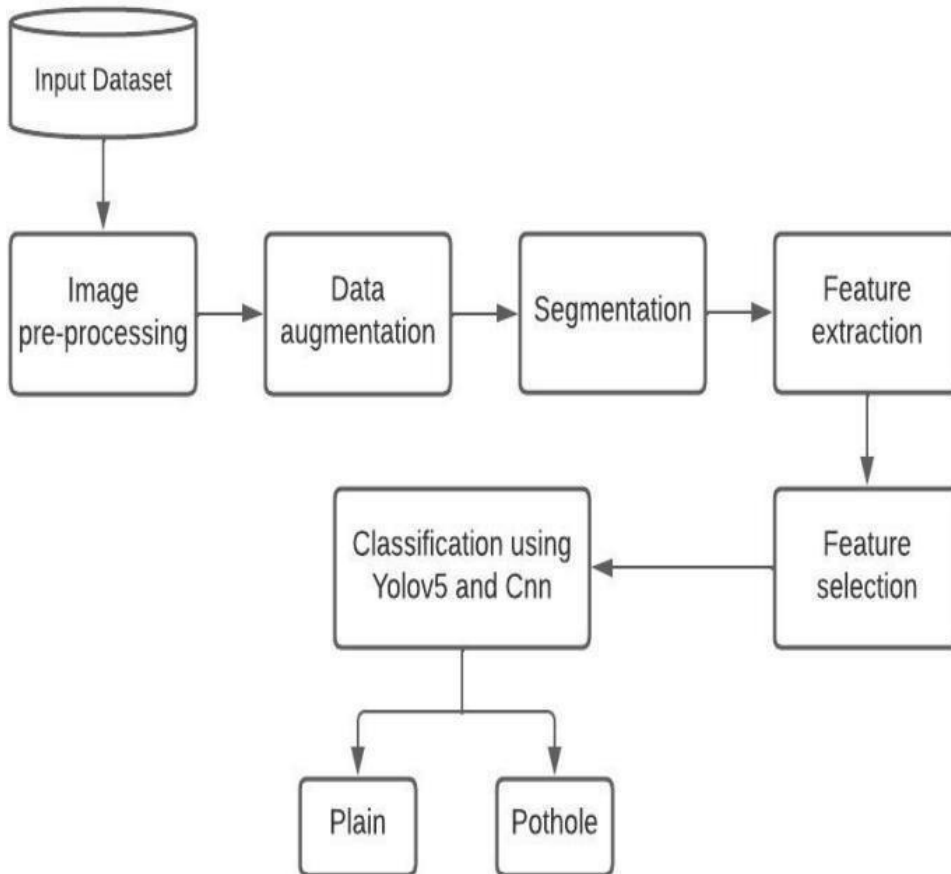


Fig 4.2: System Architecture Diagram

The architecture of our model consists of 4 convolution layers, followed by an activation – “Relu” and max pooling layer with size of [2,2] respectively. At the last two dense layers are added with “Relu” and “Softmax” as activation function respectively.

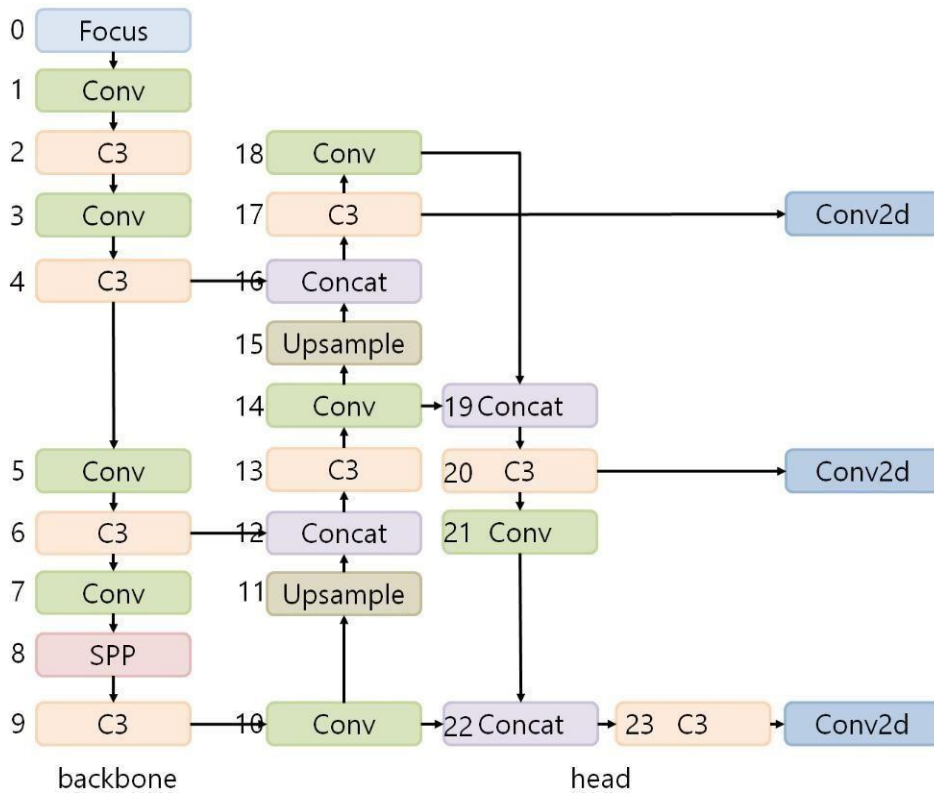


Fig 4.3: YOLO v5 Architecture

The block diagram of the proposed system has been shown in the above figures. In the above architecture, data is first collected followed by data processing, after which various machine learning techniques are applied which results in predicting the potholes.

4.3 DESCRIPTION OF SOFTWARE FOR IMPLEMENTATION AND TESTING PLAN OF THE PROPOSED MODEL/SYSTEM

In order to test the system developed, various testing techniques are used and also after training the model, a separate dataset is used to test for the accuracy of the model developed.

4.4 PROJECT MANAGEMENT PLAN

In the initial phase, we collect various images of damaged roads for both plain and images with potholes from different databases.

In the next phase, we pre-process the collected images and divide the dataset into two parts training and testing the model.

Now, when the model is trained using different algorithms we evaluate the model for the best accuracy.

The model with best and precise accuracy is used for the implementation of the project.

4.5 TRANSITION/ SOFTWARE TO OPERATIONS PLAN:

A transition plan outlines the steps needed to move a project from development to operations, ensuring that the system is properly deployed and maintained. The outline of a Transition/Software to Operations plan for a pothole detection system using YOLOv5 and CNN is given below:

4.5.1 PROJECT CLOSEOUT:

- Finalize all documentation and training materials.

- Obtain approval from stakeholders to move forward with the deployment.

4.5.2 INFRASTRUCTURE:

- Define the hardware and software requirements for the deployment environment.
- Ensure that the infrastructure is in place and configured properly.

4.5.3 DEPLOYMENT:

- Deploy the system to the designated environment.
- Configure the system to work in the target environment.

4.5.4 TESTING:

- Conduct testing to ensure that the system is working as expected.
- Validate the results of the system against manual inspections of potholes.

4.5.5 TRAINING:

- Train the operations team on the proper use and maintenance of the system.
- Create training materials and documentation to support the training.

4.5.6 MAINTENANCE:

- Establish a maintenance plan to ensure the system is operating correctly and efficiently.
- Define roles and responsibilities for the maintenance team.
- Establish a schedule for maintenance activities, including software upgrades and system backups.

4.5.7 MONITORING:

- Establish a monitoring plan to ensure the system is performing as expected.
- Define metrics and thresholds for monitoring system performance.
- Create alerts to notify the operations team of potential issues.

4.5.8 SUPPORT:

- Establish a support plan to provide assistance to users in the event of system issues or questions.
- Define the process for reporting issues and escalating them if necessary.

4.5.9 CONTINUOUS IMPROVEMENT:

- Establish a process for monitoring and evaluating the system's performance over time.
- Collect feedback from users and incorporate it into future system updates and improvements.

By following this plan, we can ensure that the pothole detection system is deployed effectively and maintained properly, minimizing disruptions and maximizing the system's effectiveness in reducing road hazards.

CHAPTER 5

IMPLEMENTATION DETAILS

5.1 DEVELOPMENT AND DEPLOYMENT SETUP

5.1.1 DATASET USED

The data set is a collection of records in a particular table, each column corresponds to a particular variable in the data set, and each row corresponds to a particular record in the data set. The Record in the dataset can be of any type such as file, document, text or numerical values, etc. Since, Our Project is to detect the pothole in the road images, therefore in our case the data type in the dataset is image file of pothole and plain images. For each of the algorithm that is YOLOv5 and CNN, the dataset which has been used in training of the model is not same for both the algorithm.

For YOLOv5 the dataset has been imported directly from the Roboflow Website. Roboflow is a website which has a large collection of different kind of datasets. It also provides the data in different format options for various machine learning models. Therefore, YOLOv5 format option has been chosen and the dataset has been imported using a code generated by a roboflow. The dataset consists of road


images,  for each image there exist a text file which gives the annotation based on the width, height, depth and size of potholes

image that form a bounding box around the potholes in the images. The dataset consists a total of 665 images which is divided in training, validation and testing in ratio 7, 2 and 1 respectively.

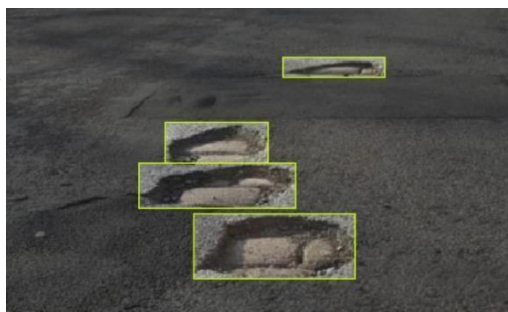


Fig. 5.1. Sample Dataset Image

For the other algorithm, which is CNN the dataset has been downloaded from



Kaggle website. Kaggle is a website which provides large collection of datasets of various kinds as well as a workspace. The dataset consists of two categories of images plain and

pothole images respectively. For each category the number of images used are 364 for plain images and 357 for Pothole images. The image file in the dataset does not contain any type of pre- defined annotations or bounding boxes as it was given for Yolov5.

Fig. 5.2. Sample Dataset Image – Plain and Pothole Image

5.1.2 DEVELOPMENT AND DEPLOYMENT STEPS:

In order to develop and deploy the pothole detection system, we need to follow certain steps.

5.1.2.1 COLLECT DATA

Collect images of potholes using a camera or collecting the dataset from various databases. This data will be used to train the model.

5.1.2.2 LABEL DATA:

Use a labeling tool to label the images as potholes or non-potholes and also for creating bounding boxes around the pothole in the images. This will create a dataset that your model can learn from.

5.1.2.3 TRAIN MODEL:

Use a library like TensorFlow or PyTorch to train a model on the labeled dataset. You can use a pre-trained model like YOLO as a starting point.

5.1.2.4 TEST MODEL:

Test the trained model on a separate test dataset to evaluate its accuracy and make any necessary adjustments.

5.1.2.5 DEPLOY MODEL:

Deploy the trained model to a server or edge device that can perform real-time pothole detection.

5.2 ALGORITHMS

5.2.1 Convolution Neural Network (CNN):

In deep learning, convolution neural network is a part of artificial neural network. Convolution neural network is widely used in image recognition and processing of the images due to its pattern recognition capabilities. Convolution neural network consist of three different and main layers.

5.2.1.1 Convolution layer :

This is the first layer used for extracting various features from the input image provided. Feature extraction from input images is done through a filter or Kernel method.

$$\frac{W - F}{S} + 1$$

5.2.1.2 Pooling layer:

It aims to reduce the computational costs by shrinking the size of the convolved feature map.

5.2.1.3 The Fully Connected (FC) layer:

It consists of the weights and biases along with the neurons and is used to connect the neurons between two different layers. This layer is placed prior to the output layer.

5.2.2 You Only Look Once (Yolov5):

There are several deep learning algorithms which fails to solve the problem of speed for real time object detection. The yolo algorithm gives much better performance than other algorithms along with the higher frame per second for fast detection speed and exact precision. The yolo algorithm predicts classes and bounding boxes for an entire image in one run instead of selecting the interesting part in each run.

There are four variant models available in YOLOv5 namely s, m, l and x. Different detection accuracy and performance are offered by each of them.

5.3 TESTING

Testing is an important process in software development to ensure that the system meets the requirements and performs as expected. The Pothole Detection System is a software application that uses Yolov5 and Convolutional Neural Network (CNN) to detect potholes on roads. In this section, we will discuss the testing process for the Pothole Detection System.

5.3.1 TYPES OF TESTING:

The various types of testing that can be used for testing the pothole detection system are:

5.3.1.1 UNIT TESTING:

In this type of testing, individual components of the system are tested to ensure that they work as expected. For example, the CNN algorithm and Yolov5 model can be tested individually to ensure they are performing accurately and efficiently.

5.3.1.2 INTEGRATION TESTING:

This type of testing is performed to ensure that the individual components of the system work together as expected. For example, testing the integration between the input images and the CNN algorithm to detect potholes.

5.3.1.3 SYSTEM TESTING:

In this type of testing, the entire system is tested to ensure that it meets the functional and non-functional requirements. For example, testing the system's ability to detect potholes with a minimum accuracy of 90% and a response time of less than 1 second.

5.3.2 TESTING PROCESS:

In order to perform effective testing, a particular testing process needs to be followed:

5.3.2.1 TEST PLAN:

The testing process starts with a test plan that outlines the testing objectives, scope, and methods.

5.3.2.2 TEST CASES:

Test cases are developed based on the requirements and the test plan to ensure that each requirement is tested thoroughly.

5.3.2.3 TEST EXECUTION:

The test cases are executed by running the system and inputting various test scenarios. The test results are recorded and analyzed to identify any issues or defects in the system.

5.3.2.4 DEFECT REPORTING:

Any defects or issues identified during the testing process are reported to the development team for resolution.

5.3.2.5 REGRESSION TESTING:

After the defects are fixed, regression testing is performed to ensure that the system still meets the requirements and functions as expected.

5.3.2.6 USER ACCEPTANCE TESTING:

Once the system has passed all the tests and meets the requirements, user acceptance testing is performed to ensure that the system meets the needs of the end-users.

For developing an effective model, testing is considered as an essential part of software development and is critical to ensure that the system meets the functional and non-functional requirements. The Pothole Detection System can be tested using unit testing, integration testing, and system testing to ensure that it performs as expected. The testing process includes developing a test plan, test cases, test execution, defect reporting, regression testing, and user acceptance testing. A thorough testing process can help ensure that the developed model for the detection of pothole is accurate, efficient, and user-friendly.

CHAPTER 6

RESULTS AND DISCUSSION

For YOLOv5 , the training has been done on different epochs and batch sizes,

Batch size	Epoch	Accuracy(mean average precision)	Time taken(in hrs)
2	100	0.669	0.673
20	100	0.677	0.238
20	180	0.715	0.421
32	280	0.734	0.608

then the epoch value which results in better accuracy was taken into consideration. A total 465 images data has been used in the training.

TABLE 6.1 YOLOv5 Results

For



CNN, the model is trained on different epochs and on different architectures, and the batch size taken was kept default i.e., 32 .And, the dataset which is used consist total of 960 images of data, out of which 486 are of

plain images and 474 are of pothole images.

Fig. 6.1. Example Output from YOLOv5 Model

While forwarding the data to the model for training, the dataset was split into two parts train and test in 80:20 ratio. Therefore 768 image data was passed to the model for training and 192 image data was kept for testing.

The accuracy for CNN comes out to be 93.34%.

CHAPTER 7

CONCLUSION

7.1 CONCLUSION

The identification of various deep learning techniques for detecting the potholes was done and then the accuracy obtained from various models is compared to obtain the best model with the highest accuracy. The CNN algorithm was preferred over YOLO v5 algorithm as CNN algorithm gave 93.34% accuracy were as YOLO v5 algorithm gave 73.34% accuracy. Further the accuracy of the YOLO v5 algorithm can be increased by training the model using a large volume of training dataset. The best model obtained can be used in future for various other applications.

7.2 FUTURE WORK

My future work would be implementing this model in real world environment in which an camera placed in an vehicle can be used to collect live data and then performing live processing of the data and then using the results to design an application in which the location of the damaged roads can be mapped and the same images can be shown on the map.

7.3 RESEARCH ISSUES:

The various research issues faced while developing the pothole detection system using YOLOv5 and CNN are:

7.3.1 DATASET SIZE AND QUALITY:

The accuracy of the pothole detection system largely depends on the size and quality of the dataset used to train the model. Building a large dataset with high-quality images can be a time-consuming and resource-intensive task.

7.3.2 DATASET BIAS:

A pothole detection model trained on a biased dataset may not perform well when deployed in the real world. For example, if the dataset used to train the model contains images of roads from a specific region or climate, the model may not perform well in other regions or climates.

7.3.3 GENERALIZATION:

It is important to ensure that the pothole detection system can accurately detect potholes in a wide range of conditions, such as different lighting, weather, and road types. This requires building a model that can generalize well to new, unseen data.

7.3.4 MODEL OPTIMIZATION:

Training a pothole detection model involves adjusting many parameters, such as the number of layers, the learning rate, and the batch size. Finding the optimal set of parameters can be a challenging and time-consuming task.

7.3.5 DEPLOYMENT:

Deploying a pothole detection system in the real world requires overcoming several challenges, such as integrating with existing infrastructure, dealing with privacy concerns related to image and video data, and ensuring that the system is reliable and accurate.

By addressing these research issues, it can help improve the accuracy and effectiveness of pothole detection system model, making it a more useful tool for improving road safety and maintenance.

7.4 IMPLEMENTATION ISSUES:

Implementing a pothole detection system can pose several implementation issues, such as:

7.4.1 HARDWARE REQUIREMENTS:

The training of a YOLOv5 and CNN model requires a significant amount of computational resources, including powerful GPUs and large amounts of memory. Therefore, implementing the system using a large dataset may require a significant investment in hardware.

7.4.2 DATA PREPARATION:

Preparing the dataset for training can be a time-consuming and challenging process. The images need to be labeled correctly to train the model accurately, and the dataset must be balanced to avoid bias.

7.4.3 MODEL SELECTION AND CONFIGURATION:

There are several different versions of YOLO and selecting the appropriate version and configuration can be a challenging task. Additionally, hyperparameters such as learning rate and batch size must be optimized to achieve high accuracy.

7.4.4 TRAINING TIME:

Training a YOLOv5 and CNN model can take several hours or even days, depending on the size of the dataset and the complexity of the model.

7.4.5 PERFORMANCE ISSUES:

Once the system is deployed, it may face performance issues such as slow detection times or high false positive rates. These issues may require further optimization of the model or adjustments to the implementation.

7.4.6 INTEGRATION WITH EXISTING INFRASTRUCTURE:

Integrating the pothole detection system with existing infrastructure, such as cameras and databases, can be a complex task that requires careful planning and execution.

7.4.7 PRIVACY AND ETHICAL CONCERNS:

The system will collect and process large amounts of image and video data, raising privacy and ethical concerns that must be addressed to ensure the system complies with relevant laws and regulations.

By addressing these implementation issues, it can help in successful development, deployment, and operation of a pothole detection system using YOLOv5 and CNN.

REFERENCES:-

- [1] Anup Kumar Pandey, Rahat Iqbal, Tomasz Maniak, Charalampos Karyotis, Stephen Akuma, Vasil Paladea, Convolution neural networks for pothole detection of critical road infrastructure.
- [2] Byeong-ho Kang and Su-il Choi, Pothole Detection System using 2D LiDAR and Camera.
- [3] Chitale, Pranjali A., Kaustubh Y. Kekre, Hrishikesh R. Shenai, Ruhina Karani, and Jay P. Gala. "Pothole detection and dimension estimation system using deep learning (yolo) and image processing." In 2020 35th International Conference on Image and Vision Computing New Zealand (IVCNZ), pp. 1-6. IEEE, 2020.
- [4] Deepa, D., and A. Sivasangari. "An effective detection and classification of road damages using hybrid deep learning framework." Multimedia Tools and Applications (2022): 1-34.
- [5] Deepa, D., R. Vignesh, A. Sivasangari, Suja Cherukullapurath Mana, B. Keerthi Samhitha, and Jithina Jose. "Visualizing road damage by monitoring system in cloud." International Journal of Electrical Engineering and Technology 11, no. 4 (2020): 191-203.
- [6] Deepak Kumar Dewangan, Satya Prakash Sahu, PotNet: Pothole detection for autonomous vehicle system using convolutional neural network.
- [7] Dharneeshkar J, Soban Dhakshana V, Aniruthan S A, Karthika R; Latha Parameswaran, Deep Learning based Detection of potholes in Indian roads using YOLO.

- [8] Fang, Yiming, Xianxin Guo, Kun Chen, Zhu Zhou, and Qing Ye. "Accurate and Automated Detection of Surface Knots on Sawn Timbers Using YOLO-V5 Model." *BioResources* 16, no. 3 (2021).
- [9] K. Vigneshwar, B. Hema Kumar, Detection and Counting of Pothole using Image Processing Techniques.
- [10] Kavitha R and Nivetha S, Pothole and Object Detection for an Autonomous Vehicle Using YOLO.
- [11] Nachuan Ma, Jiahe Fan, Wenshuo Wang, Jin Wu, Yu Jiang, Lihua Xie and Rui Fan, Computer Vision for Road Imaging and Pothole Detection: A State-of-the-Art Review of Systems and Algorithms.
- [12] Pathak, Ajeet Ram, Manjusha Pandey, and Siddharth Rautaray. "Application of deep learning for object detection." *Procedia computer science* 132 (2018): 1706-1717.
- [13] Ratnajit Mukherjee, Haris Iqbal, Shabbir Marzban, Ahmed Badar, Terence Brouns, Shruthi Gowda, Elahe Arani and Bahram Zonooz, AI Driven Road Maintenance Inspection.
- [14] Song, Hyunwoo, Kihoon Baek, and Yungcheol Byun. "Pothole detection using machine learning." *Advanced Science and Technology* (2018): 151-155.
- [15] Yuan Yuan, Lei Chen, Huarui Wu, Lin Li, Advanced agricultural disease image recognition technologies.

APPENDIX

A. SOURCE CODE:

A.1 USING YOLOv5:

```
## cloning the yolov5 github repository
!git clone "https://github.com/ultralytics/yolov5.git"

## installing the required libraries
!pip install -qr yolov5/requirements.txt
print("The requirements for the yolov5 has been installed Successfully.")

%cd yolov5

from IPython.display import Image, clear_output
!pip install utils

import matplotlib.pyplot as plt
import matplotlib.image as mpimg

## Download the custom Dataset that you are interested in using Roboflow api
    bold text
%cd /content
!curl -L "https://public.roboflow.com/ds/Ocn6NAj6XI?key=zesorLzE9A" >
    roboflow.zip; unzip roboflow.zip; rm roboflow.zip

## Create the Custom Model Configuration file
%cat data.yaml

import yaml

with open("data.yaml","r") as stream:
    num_classes = str(yaml.safe_load(stream)["nc"])
```

```

%cat /content/yolov5/models/yolov5s.yaml
from IPython.core.magic import register_line_cell_magic

@register_line_cell_magic
def writetemplate(line,cell):
    with open(line,'w') as f:
        f.write(cell.format(**globals()))

%%writetemplate /content/yolov5/models/custom_yolov5s.yaml

# Parameters
nc: 1 # number of classes
depth_multiple: 0.33 # model depth multiple
width_multiple: 0.50 # layer channel multiple
anchors:
  - [10,13, 16,30, 33,23] # P3/8
  - [30,61, 62,45, 59,119] # P4/16
  - [116,90, 156,198, 373,326] # P5/32

# YOLOv5 v6.0 backbone
backbone:
  # [from, number, module, args]
  [[-1, 1, Conv, [64, 6, 2, 2]], # 0-P1/2
  [-1, 1, Conv, [128, 3, 2]], # 1-P2/4
  [-1, 3, C3, [128]],
  [-1, 1, Conv, [256, 3, 2]], # 3-P3/8
  [-1, 6, C3, [256]],
  [-1, 1, Conv, [512, 3, 2]], # 5-P4/16
  [-1, 9, C3, [512]],
  [-1, 1, Conv, [1024, 3, 2]], # 7-P5/32
  [-1, 3, C3, [1024]],
  [-1, 1, SPPF, [1024, 5]], # 9
  ]

```

```
# YOLOv5 v6.0 head
```

```
head:
```

```
[-1, 1, Conv, [512, 1, 1]],  
[-1, 1, nn.Upsample, [None, 2, 'nearest']],  
[-1, 6, 1, Concat, [1]], # cat backbone P4  
[-1, 3, C3, [512, False]], # 13
```

```
[-1, 1, Conv, [256, 1, 1]],  
[-1, 1, nn.Upsample, [None, 2, 'nearest']],  
[-1, 4, 1, Concat, [1]], # cat backbone P3  
[-1, 3, C3, [256, False]], # 17 (P3/8-small)
```

```
[-1, 1, Conv, [256, 3, 2]],  
[-1, 14, 1, Concat, [1]], # cat head P4  
[-1, 3, C3, [512, False]], # 20 (P4/16-medium)
```

```
[-1, 1, Conv, [512, 3, 2]],  
[-1, 10, 1, Concat, [1]], # cat head P5  
[-1, 3, C3, [1024, False]], # 23 (P5/32-large)
```

```
[[17, 20, 23], 1, Detect, [nc, anchors]], # Detect(P3, P4, P5)  
]
```

```
## Training the model using the train dataset
```

```
%%time
```

```
%cd /content/yolov5
```

```
!python train.py --img 665 --batch 20 --epochs 300 --data './data.yaml'  
  --cfg ./models/custom_yolov5s.yaml --weights " " --name yolov5s_results  
  --cache
```

```
## Plotting the result and confusion matrix of trained model
```

```
from IPython import display
```

```
display.Image(f"runs/train/yolov5s_results/results.png")
## Detecting the test Images and plotting the images after it got tested
!python detect.py --weights runs/train/yolov5s_results/weights/best.pt --img 416
--conf 0.4 --source ../test/images
```

```
import glob
from IPython.display import Image,display
for imagename in glob.glob("/content/yolov5/runs/detect/exp3/*"):
    display(Image(filename=imagename))
    print("\n")
## detecting new images for potholes
!python detect.py --weights ../yolov523/best.pt --img 416 --conf 0.6 --source
../pthole.jpg
```

```
import glob
from IPython.display import Image,display

for imagename in glob.glob("runs/detect/exp/*"):
    display(Image(filename=imagename))
    print("\n")
```

A.2 USING CNN:

```
## Importing the required libraries

import numpy as np
import cv2
import os
import random
import matplotlib.pyplot as plt
import pickle

import matplotlib.pyplot as plt
import seaborn as sns

# setting the path of training dataset
DIRECTORY=r"My Dataset/train"
CATEGORIES=["Plain","Pothole"]

## Counting and labelling the datasets
for category in CATEGORIES:
    folder = os.path.join(DIRECTORY,category)
    count=0
    for img in os.listdir(folder):
        count+=1
    print(f"The {category} images are {count}")

print("-----")

labels = [i for i in range(len(CATEGORIES))]
labeldict={}
for i in range (len(CATEGORIES)):
    labeldict[CATEGORIES[i]]=labels[i]
```

```
print(labeldict)
```

```
## renaming image files in a folder
```

```
data=[]
```

```
for category in CATEGORIES:
```

```
    folder = os.path.join(DIRECTORY,category)
```

```
    count=1
```

```
    label = CATEGORIES.index(category)
```

```
    for img in os.listdir(folder):
```

```
        img_path = os.path.join(folder,img)
```

```
        count+=1
```

```
        dst=f"{category}{str(count)}.jpg"
```

```
        os.rename(img_path,dst)
```

```
## Increasing the data by performing Data Augumentation
```

```
from keras.preprocessing.image import
```

```
    ImageDataGenerator,array_to_img,img_to_array,load_img
```

```
datagen = ImageDataGenerator(
```

```
    rotation_range=5,
```

```
    shear_range=0.03,
```

```
    zoom_range=0.04,
```

```
    fill_mode='nearest'
```

```
)
```

```
# img = load_img('My Dataset/train/Pothole/Pothole2.jpg')
```

```
# img = load_img('My Dataset/train/Pothole/Pothole4.jpg')
```

```
# img = load_img('My Dataset/train/Plain/Plain345.jpg')
```

```
img = load_img('My Dataset/train/Plain/Plain239.jpg')
```

```
x = img_to_array(img)
```

```
x = x.reshape((1,)+x.shape)
```

```
i=0
for batch in datagen.flow(x,batch_size=1,save_to_dir='My
    Dataset/train/Plain',save_prefix='Plaindatagen',save_format='JPEG'):
    i+=1
    if i>60:
Break
```

```
## after applying imagedata generator, the count of data has increased
for category in CATEGORIES:
    folder = os.path.join(DIRECTORY,category)
    count=0
    for img in os.listdir(folder):
        count+=1
    print(f"The {category} images are {count}")
```

```
## Preparing the dataset
data=[]
for category in CATEGORIES:
    folder = os.path.join(DIRECTORY,category)
    # print(folder)
    label = CATEGORIES.index(category)
    for img in os.listdir(folder):
        img_path = os.path.join(folder,img)
        img_arr = cv2.imread(img_path)
        # plt.imshow(img_arr)
        try:
            img_arr = cv2.resize(img_arr,(120,120))
            data.append([img_arr,label])
        except:
            print(img)
```

```
## Total count of data
print(len(data))
```

```
## Plotting the count of data
list = []
countPlain=0
countPothole=0
for i in data:
    if(i[1] == 0):
        list.append("Plain")
        countPlain+=1
    else:
        list.append("Pothole")
        countPothole+=1
sns.set_style('darkgrid')
sns.countplot(list)
print(f"Total number of data: \nPlain: {countPlain}\nPothole: {countPothole}")
plt.show()
```

```
## shuffling the dataset, so that the model dont get biased
random.shuffle(data)
```

```
## Splitting the features and target, in X and Y
X=[]
Y=[]
```

```
for feature,labels in data:
    X.append(feature)
    Y.append(labels)
```

```
## printing the length of the X and Y data
print(f"length of X : {len(X)}\nlength of Y : {len(Y)}")
```

```
## Converting the X and Y values to numpy array
X=np.array(X)
Y=np.array(Y)
```



```
## reshaping X and Y
```

```
X=X/255
```

```
X=np.reshape(X,(X.shape[0],120,120,3))
```

```
print(X[0])
```

```
from keras.utils import np_utils
```

```
Y=np_utils.to_categorical(Y)
```

```
print(Y[0])
```

```
## Creating Cnn-Architecture and Modelling the data
```

```
from keras.models import Sequential
```

```
from keras.layers import
```

```
    Conv2D,MaxPooling2D,Flatten,Dense,Dropout,Activation
```

```
# creating cnn model
```

```
modell=Sequential()
```

```
modell.add(Conv2D(32,(3,3),input_shape=X.shape[1:]))
```

```
modell.add(Activation('relu'))
```

```
modell.add(MaxPooling2D(pool_size=(2,2)))
```

```
modell.add(Dropout(0.5))
```

```
modell.add(Conv2D(32,(3,3),input_shape=X.shape[1:]))
```

```
modell.add(Activation('relu'))
```

```
modell.add(MaxPooling2D(pool_size=(2,2)))
```

```
modell.add(Dropout(0.5))
```

```
modell.add(Conv2D(32,(3,3),input_shape=X.shape[1:]))
```

```
modell.add(Activation('relu'))
```

```
modell.add(MaxPooling2D(pool_size=(2,2)))
```

```
modell.add(Dropout(0.5))
```

```

modell.add(Conv2D(64,(3,3)))
modell.add(Activation('relu'))
modell.add(MaxPooling2D(pool_size=(2,2)))
modell.add(Dropout(0.5))

modell.add(Conv2D(64,(3,3)))
modell.add(Activation('relu'))
modell.add(MaxPooling2D(pool_size=(2,2)))
modell.add(Dropout(0.5))

modell.add(Flatten())

modell.add(Dense(128,activation='relu'))
modell.add(Dropout(0.5))
modell.add(Dense(2,activation='softmax'))

modell.compile(loss='categorical_crossentropy',optimizer='adam',metrics=['ac
    curacy'])
modell.summary()

# splitting the data into train and test using train test split
from sklearn.model_selection import train_test_split
train_data,test_data,train_target,test_target=train_test_split(X,Y,test_size=0.2
    )

print(f'train data : {len(train_data)}\ntest data : {len(test_data)}')

# fitting the training data into model
history=modell.fit(train_data,train_target,epochs=40,validation_split=0.1,verbo
    se=1)

```

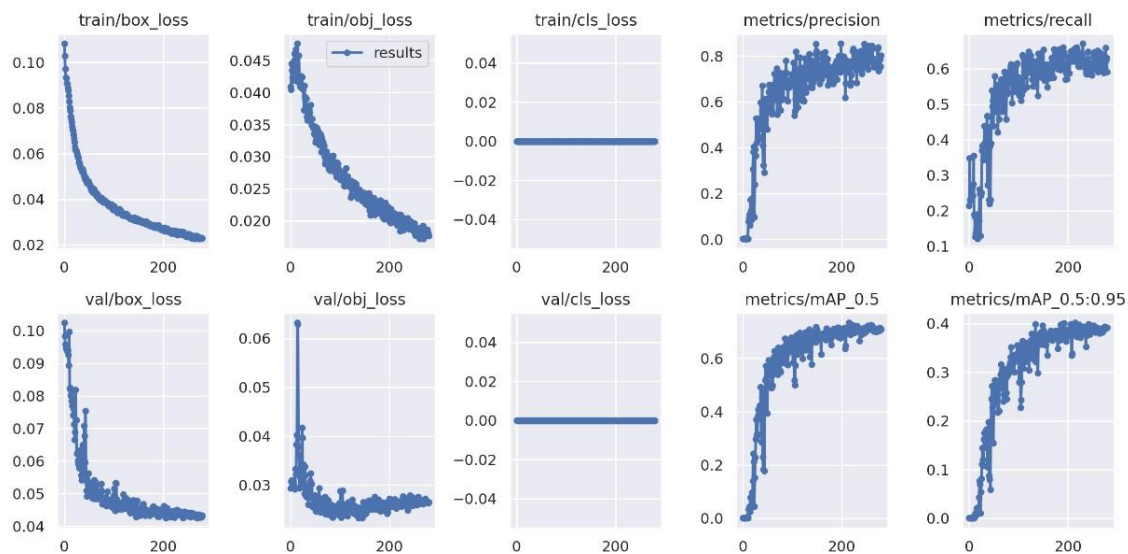
```
# plotting the line of training loss
plt.plot(history.history['loss'],'red',label='training loss')
plt.plot(history.history['val_loss'],'blue',label='validation loss')
plt.xlabel("epochs->",size=15)
plt.ylabel("loss->",size=15)
plt.legend()
plt.show()
```

```
# plotting the line of training accuracy
plt.plot(history4.history['accuracy'],'red',label='training accuracy')
plt.plot(history4.history['val_accuracy'],'blue',label='validation accuracy')
plt.xlabel("epochs->",size=15)
plt.ylabel("accuracy->",size=15)
plt.legend()
plt.show()
```

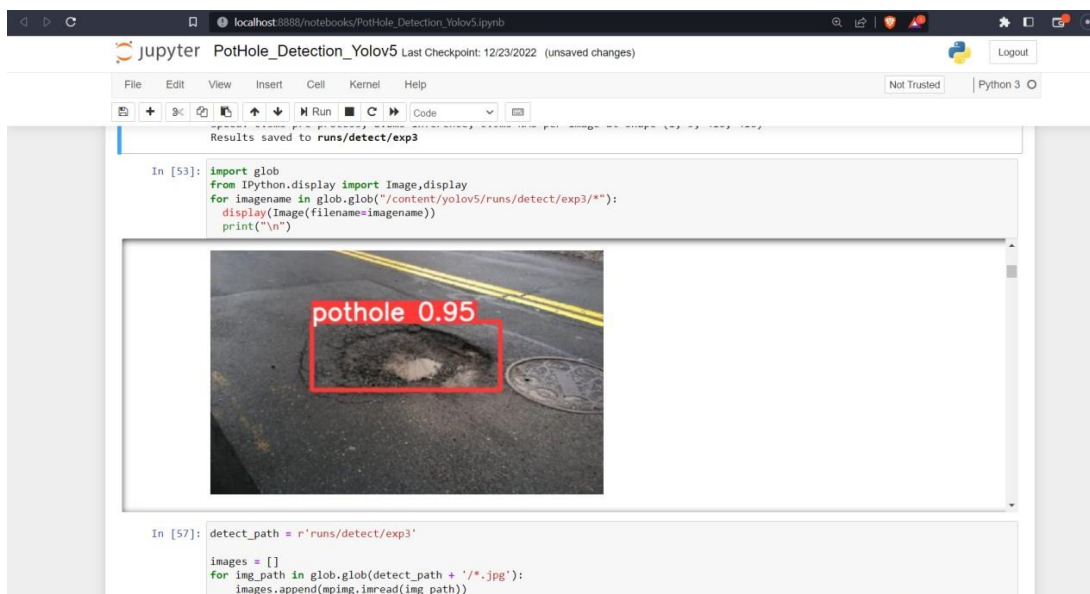
```
[25]: from IPython import display
# import matplotlib.image as mpimg
display.Image(f"runs/train/yolov5s_results5/results.png")
```

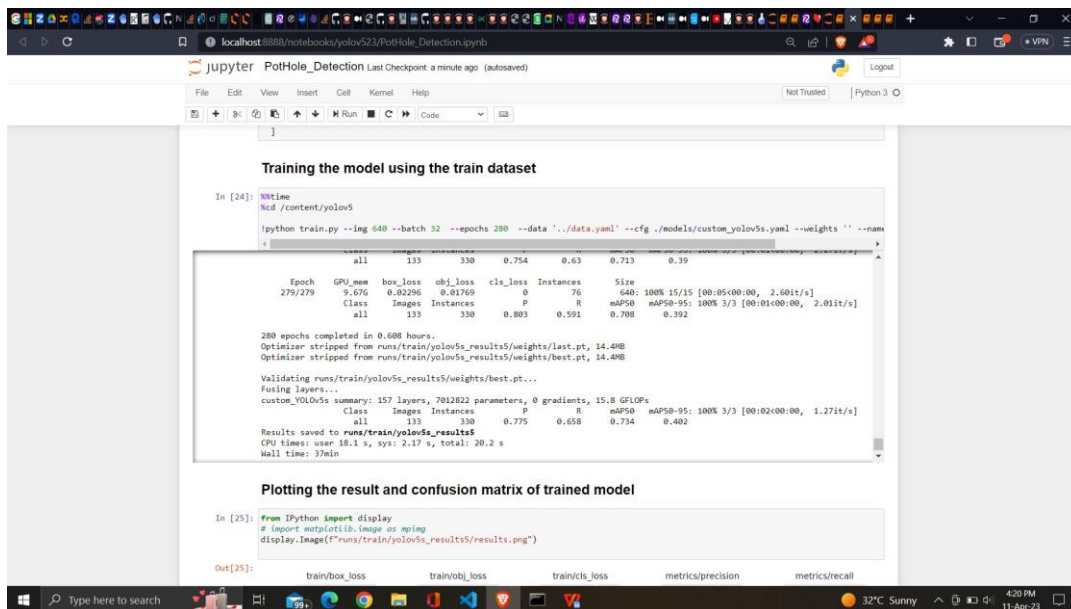
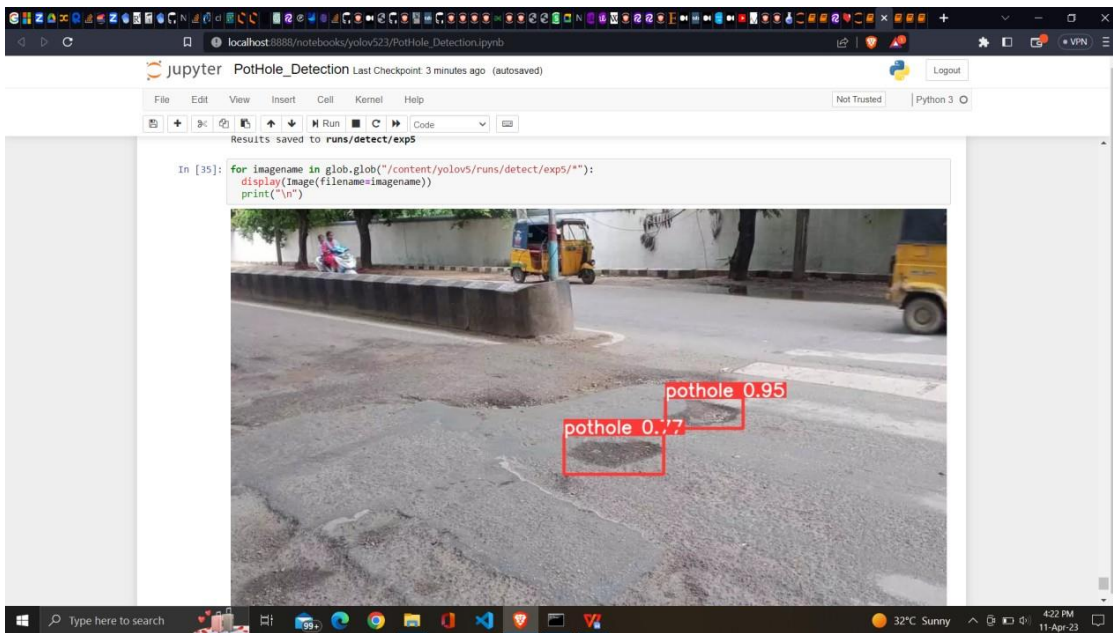
A.

[25]:

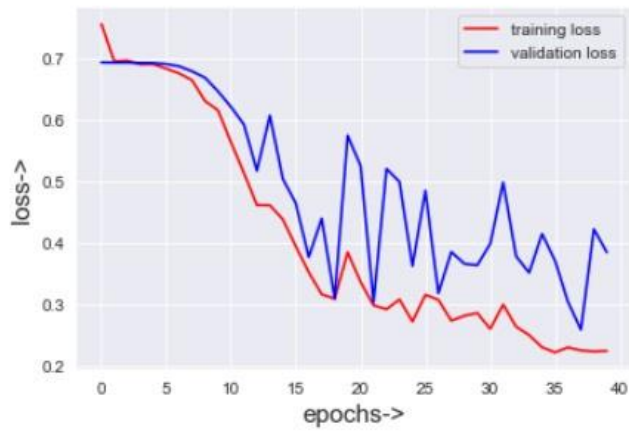


SCREENSHOTS:





```
In [88]: # plotting the line of training loss
plt.plot(history.history['loss'],'red',label='training loss')
plt.plot(history.history['val_loss'],'blue',label='validation loss')
plt.xlabel("epochs->",size=15)
plt.ylabel("loss->",size=15)
plt.legend()
plt.show()
```



```
In [43]: # plotting the line of training accuracy
plt.plot(history4.history['accuracy'],'red',label='training accuracy')
plt.plot(history4.history['val_accuracy'],'blue',label='validation accuracy')
plt.xlabel("epochs->",size=15)
plt.ylabel("accuracy->",size=15)
plt.legend()
plt.show()
```

