# ROAD DAMAGE DETECTION
# USING YOLO V7 OBJECT DETECTION TECHNIQUE

*A project submitted by*

**KONA SIVA DURGA PRASAD**

**(Regd.No: 421206421022)**

*In the partial fulfillment of the requirements for the award of the degree of*

**MASTER OF SCIENCE**

in

**COMPUTER SCIENCE**

Under the guidance of esteemed

**Dr. KONDAPALLI VENKATA RAMANA**

Associate Professor


**DEPARTMENT OF INFORMATION TECHNOLOGY AND COMPUTER APPLICATIONS**

**ANDHRA UNIVERSITY**

**VISAKHAPATNAM – 530003**

**(2021-2023)**

# DEPARTMENT OF INFORMATION TECHNOLOGY AND COMPUTER APPLICATIONS

# ANDHRA UNIVERSITY

# VISAKHAPATNAM - 530003

## CERTIFICATE

This is to certify that the project report entitled **"ROAD DAMAGE DETECTION USING YOLO V7 OBJECT DETECTION TECHNIQUE."**, is the bonafide work carried out by with **KONA SIVA DURGA PRASAD** Regd.No:421206421022 ,a student of M.Sc(Computer Science) in **AU COLLEGE OF ENGINEERING(A) , ANDHRA UNIVERSITY, VISAKHAPATNAM,** during the year 2021-2023, in partial fulfillment of the requirements for the award of degree of **MASTER OF SCIENCE**

**Dr. KONDAPALLI VENKATA RAMANA**
**Project Guide**
**DEPT OF CS & SE**
**ANDHRA UNIVERSITY**

**Prof. KUNJAM NAGESWARA RAO**
**Head of the Department**
**DEPT OF IT & CA**
**ANDHRA UNIVERSITY**

# ACKNOWLEDGEMENT

I have the great pleasure to acknowledge my sincere gratitude to my Project Supervisor,**DR.KONDAPALLI VENKATA RAMANA**, Department of Computer Science and Systems Engineering, Andhr University, for his valuable help and resourceful guidance during every stage of this project work. His useful suggestions and encouragement helped us a lot in carrying out this project work as well as in bringing this project to this form. I express my sincere gratitude for having accorded us permission to take up this project work and for helping us graciously throughout the execution of this work.

I express my sincere thanks to **PROF. KUNJAM NAGESWARA RAO**, Head of the Department of Information Technology and Computer Applications, Andhra University College of Engineering for his keen interest and providing necessary facilities for this project study.

I am sincerely thankful to **PROF. D. LALITHA BHASKARI**, Chairman Board of Studies For her valuable support.

I express my sincere gratitude to **PROF. G. SASIBHUSANA RAO, PRINCIPAL**, Andhra University College of Engineering for his keen interest and for providing necessary facilities for this project study.

I am also thankful to all academic teaching staff members and non-teaching staff members in the Department of Information Technology & Computer Applications, for their feedback in the reviews and kind help throughout my project and study.

**KONA SIVA DURGA PRASAD**

**(421206421022)**

# DECLARATION

I Declare that the project report entitled, "**ROAD DAMAGE DETECTION USING YOLOv7 OBJECT DETECTION TECHNIQUE**" has been done by me in partial fulfillment of requirement for the award of degree of "Master of Science" during the academic year 2021-2023 under the guidance of "**DR.KONDAPALLI VENKATA RAMANA**" ,department of Computer Science & Systems Engineering , AU College of Engineering (A), Andhra University, Visakhapatnam. I here by declare that this project work has not been submitted to any other Universities/Institutions for the award of any degree

PLACE : VISAKHAPATNAM KONA SIVA DURGA PRASAD

DATE : (421206421022)

# ABSTRACT

Road damage detection is a critical task in infrastructure maintenance and ensuring road safety. Traditional manual inspection methods are time-consuming and prone to human error. To overcome these limitations, machine learning techniques have emerged as a powerful tool for automated road damage detection. In this study, we explore the application of machine learning algorithms in identifying and classifying different types of road defects, such as patholes, cracks, and pavement distress.

The proposed approach involves training a machine learning model using a large dataset of labeled road images. Various deep learning architectures, such as convolutional neural networks (CNNs), are employed to learn complex patterns and features associated with different types of road damage. The trained model is then used to automatically analyze new road images and detect instances of damage.

Experimental results demonstrate the effectiveness of the proposed method in accurately identifying road defects. The model achieves high classification accuracy, outperforming traditional manual inspection methods. The automated detection process significantly reduces the time and cost associated with manual inspections while improving detection efficiency.

Furthermore, the scalability of machine learning algorithms allows for the analysis of large volumes of road data. This capability facilitates comprehensive monitoring of road conditions over time, enabling authorities to gain valuable insights into the deterioration patterns and prioritize maintenance efforts accordingly.

The application of machine learning in road damage detection has the potential to revolutionize infrastructure maintenance practices. By automating the identification and classification of road defects, this technology enhances road safety, minimizes traffic disruptions, and optimizes resource allocation. The findings of this study contribute to the advancement of intelligent transportation systems and provide a foundation for further research in the field of road damage detection using machine learning.

# TABLE OF CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

# LIST OF ABBREVIATIONS

| [Abbreviation Short Form] | [Abbreviation Full Form] |
|---|---|
| RPN | Region Proposal Network |
| YOLO | You Only Look Once |
| MMS | Mobile Measurement System |
| GPS | Global Positioning System |
| CPU | Central Processing Unit |
| TPU | Tensor Processing Unit |
| CNN | Convolutional Neural Network |
| R-FCN | Region-based Fully Convolutional Networks |

# CHAPTER 1

# INTRODUCTION

Maintenance departments need to regularly assess the quality of the roads to properly maintain them. Currently, this is done by yearly inspections or in response to reports from the public. The inspection is sometimes done by workers walking along the streets and recording the conditions on paper which later is put into a database. In other cases, the agency makes use of special vehicles that measure the road surface. On the other hand, research on damage detection of road surfaces using image processing techniques has been actively conducted, achieving considerably high detection accuracies. However, in a real-world scenario, when the road managers from a governing body need to repair such damage, they need to clearly understand the type of damage to take effective action, to achieve this we need an accurate information about the type of damage on the road. Since roads have a great influence on people's lives, maintenance and management of roads should be done periodically and exhaustively. However, due to lack of financial resources, many local governments are not able to conduct enough inspections. In fact, some municipalities automate damage determination by using high performance sensors, but because of their high cost, many municipalities are unable to introduce them. Therefore, there is a need for a method that makes it easy to judge the damage of the road surface at low cost. In this project we are going to automate the process of detection of damage on the road using Machine Learning.

Road damage detection using machine learning is a cutting-edge approach that harnesses the power of artificial intelligence to automatically identify and classify various types of road defects. With the increasing need to ensure safe and efficient transportation infrastructure, this technology has emerged as a game-changer in the field of road maintenance and inspection.

Traditional methods of road damage detection often rely on manual inspections, which can be time-consuming, expensive, and subject to human error. By contrast, machine learning algorithms can be trained to recognize patterns and features in road

imagery, enabling them to identify and categorize different types of damage accurately and efficiently.

The process begins with the collection of road images or videos using various sources such as cameras mounted on vehicles, drones, or even satellite imagery. These images are then fed into the machine learning model, which has been trained on a vast dataset of labeled road damage examples. The model learns to recognize common road defects such as potholes, cracks, pavement distress, and other anomalies.

## 1.1 Problem statement:

**1. Manual Inspection Limitations:** The current method of visually inspecting roads for damages is time-consuming, expensive, and prone to human errors. Automating the process will significantly improve efficiency and reduce costs.

**2. Safety Risks:** Road damages pose significant safety risks for drivers and pedestrians. Detecting and repairing these damages promptly is crucial to ensure the safety of road users. An automated system can expedite the detection process and enable timely repairs.

**3. Maintenance Planning:** Identifying and documenting road damages accurately is essential for effective maintenance planning. The system should provide detailed information about the type, severity, and location of damages to assist in prioritizing repair efforts and optimizing resource allocation.

**4. Data Volume and Variability:** Road networks are extensive, and the amount of data collected for analysis can be massive. The system needs to handle large volumes of data efficiently while accounting for variations in lighting conditions, weather effects, and road surface materials.

**5. Classification Accuracy:** Accurately classifying different types of road damages is crucial for appropriate repair strategies. The system should be capable of distinguishing between various damage categories, such as cracks, potholes, and erosion, with a high degree of precision.

**6. Real-Time Detection:** To enable timely repairs, the system should operate in near real-time, detecting damages as soon as they occur or shortly after. This requirement is

particularly crucial for rapidly deteriorating damages like potholes, which can worsen quickly and lead to accidents.

**7. Scalability:** The solution should be scalable to accommodate varying road network sizes and expansions. It should be adaptable to different geographical regions, considering regional differences in road conditions, climate, and materials.

## 1.2 INTRODUCTION TO MACHINE LEARNING

Machine learning is a dynamic field of study that lies at the intersection of computer science, statistics, and artificial intelligence. It empowers computers to learn and improve from experience without being explicitly programmed. By utilizing algorithms and statistical models, machine learning enables systems to analyze vast amounts of data, recognize patterns, and make intelligent decisions or predictions.

At its core, machine learning aims to develop computational models that can automatically uncover meaningful insights from data and generalize from that knowledge to make accurate predictions or take appropriate actions in new, unseen situations. These models learn from labeled examples or historical data, identifying intricate relationships and dependencies that might not be easily discernible to humans.

The potential applications of machine learning are vast and diverse, spanning numerous fields such as healthcare, finance, marketing, robotics, and more. It has revolutionized industries by enabling advancements like personalized medicine, autonomous vehicles, natural language processing, fraud detection, and recommender systems.

There are various types of machine learning techniques, including supervised learning, unsupervised learning, semi-supervised learning, reinforcement learning, and deep learning. Each approach addresses different problem domains and leverages different algorithms to tackle challenges unique to each category.

Machine learning has the power to unlock valuable insights from complex data, automate decision-making processes, and enhance human productivity in countless ways.

As the field continues to evolve, it holds the promise of transforming industries, driving innovation, and shaping the future of technology and society as a whole.

**"A computer program is said to learn from experience E with some class of tasks T and performance measure P if its performance at tasks in T, as measured by P, improves with experience E." -Tom M. Mitchell**

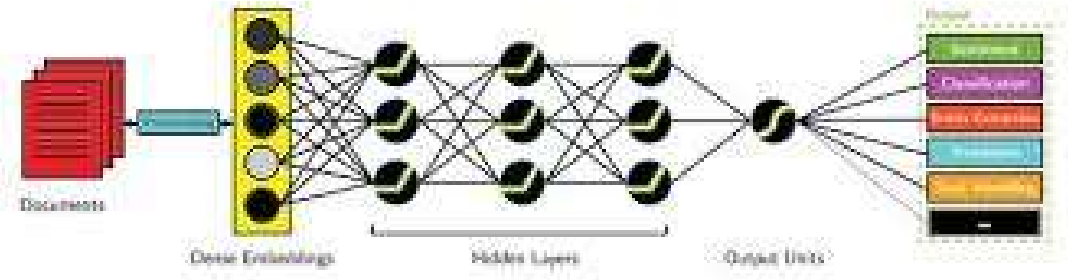


Fig 1.1. Machine Learning Model

## 1.3 Why is Machine Learning Essential?

Machine learning is essential for several reasons:

1. Handling Big Data: In today's digital era, vast amounts of data are generated every second. Machine learning techniques enable us to efficiently process, analyze, and extract valuable insights from this enormous volume of data. Traditional methods and manual analysis would be impractical or impossible due to the sheer scale and complexity of the information.
2. Automation and Efficiency: Machine learning automates repetitive tasks and decision-making processes, reducing the need for manual intervention and saving time and effort. By leveraging algorithms that learn from data, machines can perform complex computations and tasks much faster and with higher accuracy than humans, leading to improved efficiency and productivity across various domains.
3. Predictive Capabilities: Machine learning enables predictive modeling, allowing us to make accurate predictions or forecasts based on historical data. This capability has far-reaching implications in fields such as finance, weather

forecasting, sales forecasting, healthcare, and more. By leveraging patterns and trends in data, machine learning algorithms can help us anticipate future outcomes and make informed decisions.

4. Personalization and Recommendation Systems: Machine learning powers personalized experiences and recommendation systems that we encounter daily. Platforms like Netflix, Amazon, and Spotify leverage machine learning algorithms to analyze user behavior, preferences, and historical data to offer personalized recommendations. This not only enhances user satisfaction but also drives business revenue.
5. Fraud Detection and Cybersecurity: Machine learning plays a crucial role in detecting anomalies, identifying patterns of fraudulent behavior, and preventing cyber-attacks. By analyzing vast amounts of data and learning from past incidents, machine learning algorithms can detect suspicious activities, classify potential threats, and enhance overall cybersecurity measures.
6. Medical Diagnostics and Healthcare: Machine learning has the potential to revolutionize healthcare by aiding in disease diagnosis, personalized treatment plans, and drug discovery. By analyzing patient data, medical images, genomic information, and clinical records, machine learning algorithms can assist doctors in making accurate diagnoses, predicting disease progression, and suggesting optimal treatment strategies.
7. Autonomous Systems: Machine learning is at the heart of developing autonomous systems such as self-driving cars, drones, and robotics. These systems learn from real-time sensor data to navigate, make decisions, and adapt to changing environments. Machine learning algorithms enable these autonomous systems to perceive and interpret their surroundings, making them safer and more reliable.
8. Natural Language Processing and Translation: Machine learning techniques, particularly deep learning, have made significant strides in natural language processing, speech recognition, and machine translation. These advancements power virtual assistants like Siri and Alexa, language translation services, and chatbots, enabling more seamless human-computer interactions and bridging language barriers.

## Types of Machine Learning:

There are several types of machine learning techniques, each addressing different problem domains and learning from data in distinct ways. Here are some of the common types of machine learning:

- **Supervised Learning:** In supervised learning, the algorithm learns from labeled examples, where the input data is paired with corresponding desired output or target values. The algorithm aims to generalize from these labeled examples to make accurate predictions or classifications on unseen data. Examples of supervised learning algorithms include linear regression, logistic regression, decision trees, support vector machines (SVM), and neural networks.
- **Unsupervised Learning:** Unsupervised learning involves learning from unlabeled data, where the algorithm aims to discover patterns, relationships, or structures within the data. Unlike supervised learning, there are no specific target values or labels associated with the input data. Common unsupervised learning techniques include clustering algorithms (such as k-means clustering and hierarchical clustering) and dimensionality reduction techniques (like principal component analysis).
- **Semi-Supervised Learning:** Semi-supervised learning is a combination of supervised and unsupervised learning. It utilizes a small amount of labeled data along with a larger amount of unlabeled data to improve learning accuracy. The labeled data provides some initial guidance, while the unlabeled data helps to uncover underlying patterns or structures in the data. This approach is useful when obtaining labeled data is expensive or time-consuming.
- **Reinforcement Learning:** Reinforcement learning involves an agent learning how to interact with an environment to maximize rewards or minimize penalties. The agent learns through trial and error, taking actions in the environment and receiving feedback in the form of rewards or punishments. Reinforcement learning algorithms, such as Q-learning and deep reinforcement learning, are commonly used in robotics, game playing, and autonomous systems.

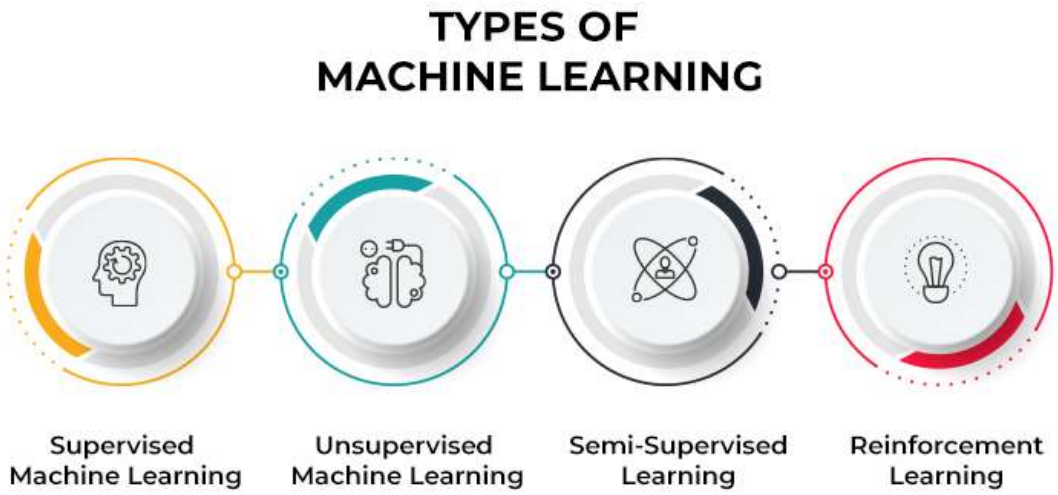



Fig 1.2: Types of machine learning

## 1.4 Examples of Machine Learning

Machine Learning technology has widely changed the lifestyle of a human beings as we are highly dependent on this technology. It is the subset of Artificial Intelligence, and we all are using this either knowingly or unknowingly. For example, we use Google Assistant that employs ML concepts, we take help from online customer support, which is also an example of machine learning, and many more.

Machine Learning uses statistical techniques to make a computer more intelligent, which helps to fetch entire business data and utilize it automatically as per requirement. There are so many examples of Machine Learning in real-world, which are as follows:

## Object detection:

Object detection using machine learning involves training algorithms to identify and locate objects within images or videos. It has various applications in fields such as autonomous driving, surveillance systems, robotics, and image analysis. Here's a high-level overview of how object detection using machine learning works:

- Data Collection: A large dataset of images or videos is collected, where each image or video frame contains labeled bounding boxes around the objects of

interest. The dataset should represent a wide range of object instances, viewpoints, lighting conditions, and backgrounds.

- Data Preparation: The collected dataset is then preprocessed to normalize the images, resize them, and extract relevant features. Common feature extraction methods include Histogram of Oriented Gradients (HOG), Scale-Invariant Feature Transform (SIFT), or Convolutional Neural Networks (CNNs).
- Training: Various machine learning algorithms can be used for object detection, such as Support Vector Machines (SVMs), Decision Trees, or Convolutional Neural Networks (CNNs). The algorithm is trained using the preprocessed dataset, where it learns to classify and localize objects based on the extracted features.
- Object Localization: During training, the algorithm learns to predict bounding boxes around the objects in the images. This is typically done by assigning coordinates to the corners of the bounding boxes or by using anchor boxes to define potential object locations and sizes.
- Inference: Once the algorithm is trained, it can be used to detect objects in new, unseen images or videos. The algorithm scans the input image or video frame at different scales and positions, applying the trained model to classify and localize objects based on the learned patterns.
- Post-processing: After object detection, post-processing techniques are applied to refine the results. This may involve removing duplicate detections, filtering out low-confidence predictions, or employing techniques like non-maximum suppression to select the most accurate bounding boxes.
- Evaluation and Iteration: The performance of the object detection algorithm is evaluated using metrics such as precision, recall, and mean average precision (mAP). If the performance is not satisfactory, the process can be iterated by collecting more data, adjusting parameters, or changing the model architecture until the desired accuracy is achieved.

It's worth mentioning that there are several popular object detection frameworks available, such as Faster R-CNN, YOLO (You Only Look Once), and SSD (Single Shot MultiBox Detector), which provide pre-implemented models and training pipelines to simplify the object detection process.

## Google Translation:

Google Translate is an online translation service developed by Google that utilizes machine learning techniques to provide translations between different languages. Here's an overview of how Google Translate uses machine learning:

- Training Data: Google Translate is trained on a vast amount of multilingual data. This includes parallel corpora, which are collections of texts in different languages that are translated versions of each other. These datasets can include books, websites, official documents, and other sources.
- Neural Machine Translation (NMT): Google Translate leverages Neural Machine Translation, a type of machine learning model that uses artificial neural networks to learn how to translate text. NMT models are trained to map the input text in one language to the output text in another language.
- Training Process: During training, the neural network learns to translate by analyzing the parallel corpora. It learns to encode the source language input into a representation that captures its meaning and context, and then decodes this representation into the target language output.
- Optimization: The neural network's parameters are optimized through a process called backpropagation, where the model adjusts its internal weights based on the comparison between its predicted translations and the correct translations from the training data. This optimization process minimizes the difference between predicted and actual translations.

## Video Surveillance:

Video surveillance using machine learning is a powerful application that leverages artificial intelligence to enhance the capabilities of surveillance systems. Traditional video surveillance methods require manual monitoring, which can be time-consuming and prone to human error. By incorporating machine learning algorithms, video surveillance systems can automatically analyze video footage, detect anomalies, and provide valuable insights for security and safety purposes.

Video surveillance using machine learning offers several advantages. It can automate the detection of suspicious activities, such as unauthorized access, vandalism, or unusual behavior, thereby enhancing security and enabling rapid response. The technology can also identify specific objects or individuals, such as vehicles or persons of interest, aiding in investigations and forensic analysis.

Furthermore, machine learning models can learn to recognize normal patterns of activity in a specific environment. This allows them to identify deviations from the norm, such as crowd congestion, traffic accidents, or abnormal movements, and trigger alerts or notifications to alert security personnel or relevant authorities.

By integrating machine learning with video surveillance, large amounts of video data can be processed and analyzed more efficiently. The technology can handle real-time monitoring and provide valuable insights by extracting meaningful information from the video streams. These insights can aid in decision-making, resource allocation, and the development of proactive security strategies.

Some popular uses of video surveillance are:

- Facility protections
- Operation monitoring
- Parking lots
- Traffic monitoring
- Shopping patterns

## Statistical Arbitrage:

Statistical arbitrage is an investment strategy that seeks to exploit pricing inefficiencies in financial markets by using statistical models and quantitative analysis. Machine learning techniques can be applied to enhance and automate the process of identifying and executing statistical arbitrage opportunities. Here's how machine learning can be utilized in statistical arbitrage:

- Data Collection: Machine learning models in statistical arbitrage require vast amounts of historical financial data, including price data, trading volumes, financial

statements, and macroeconomic indicators. These datasets are collected and organized for analysis.

- Model Development: Machine learning algorithms, such as regression models, decision trees, random forests, support vector machines (SVMs), or neural networks, are trained on the historical data using the engineered features. The models learn patterns and relationships between the features and target variables, which could be signals for trading decisions.
- Strategy Generation: Based on the trained machine learning models, statistical arbitrage strategies are generated. These strategies typically involve identifying pairs or groups of financial instruments that exhibit historically correlated price movements. Deviations from this historical relationship are used as signals for potential trading opportunities.
- Risk Management: Machine learning models in statistical arbitrage need robust risk management techniques to control exposure and mitigate potential losses. Risk factors such as position sizing, stop-loss mechanisms, and portfolio diversification are considered to manage the downside risk associated with the strategy.
- Execution: Once the strategy is developed and validated, it can be executed in real-time or near real-time. The execution can be manual or automated using algorithmic trading systems. Trade execution should consider factors like market liquidity, transaction costs, and market impact to ensure practical feasibility.
- Monitoring and Adaptation: Machine learning models need continuous monitoring and adaptation to changing market conditions. The models may require periodic retraining or adjustment of parameters to maintain their effectiveness. Ongoing performance monitoring and model refinement are essential to stay competitive in the dynamic financial markets.

# CHAPTER 2

# LITERATURE REVIEW

Road surface inspection is primarily based on visual observations by humans and quantitative analysis using expensive machines. Among these, the visual inspection approach not only requires experienced road managers, but also is time consuming and expensive. Furthermore, visual inspection tends to be inconsistent and unsustainable, which increases the risk associated with aging road infrastructure. Considering these issues, municipalities lacking the required resources do not conduct infrastructure inspections appropriately and frequently, increasing the risk posed by deteriorating structures. In contrast, quantitative determination based on large-scale inspection, such as using a mobile measurement system (MMS) (KOKUSAI KOGYO CO., 2016) or laser-scanning method (Yu and Salari, 2011) is also widely conducted. An MMS obtains highly accurate geospatial information using a moving vehicle; this system comprises a global positioning system (GPS) unit, an internal measurement unit, digital measurable images, a digital camera, a laser scanner, and an omnidirectional video recorder. Though quantitative inspection is highly accurate, it is considerably expensive to conduct such comprehensive inspections especially for small municipalities that lack the required financial resources. Therefore, considering the abovementioned issues, several attempts have been made to develop a method for analyzing road properties by using a combination of recordings by in-vehicle cameras and image processing technology to more efficiently inspect a road surface. For traffic capacity analysis, focusing on roads in good condition will simplify the problem, no matter how the roads are destroyed. The whole road line is provided by road network vector of appropriate scale. Subtract the good road and the damaged parts are clearly shown. This method can realize quantitative analysis on road seismic damage when lack of pre-earthquake images. In this paper it is applied to road damage extraction in Wenchuan earthquake, using high resolution remote sensing images shot just several days after the event.

A mobile robot moves along the route for investigation and obtain shape information of road surface using 2D laser scanner. From this road surface information, road damage section will be automatically detected. By showing the detection result

instead of site investigation by human workers, it expects to reduce the burden of human workers. Road surface have gradual curves and some road damage is small and less than 2cm. Hence, our method uses random sampling to detect irregularity as road damage. This paper explains the measurement of road surface using mobile robot equipped with 2D laser scanner and the road damage detection method.

1. "Road Damage Detection and Classification Using Deep Convolutional Neural Networks" (2017) by M. S. Rahman et al.:

   This study proposed a deep learning approach for road damage detection using Convolutional Neural Networks (CNNs). The authors collected a large dataset of road images and trained the CNN model to detect and classify various types of road defects. The results showed high accuracy and demonstrated the effectiveness of CNNs in road damage detection.

2. "Road Damage Detection Using Fully Convolutional Networks and Transfer Learning" (2018) by V. K. Kukar and M. Vranić:

   The authors presented a road damage detection system based on Fully Convolutional Networks (FCNs) and transfer learning. They used a pre-trained VGG-16 model and fine-tuned it on a road damage dataset. The study showed that transfer learning improved the model's performance and achieved accurate road damage detection.

3. "DeepRoadMapper: Extracting Road Topology from Aerial Images" (2019) by L. Lu et al.:

   This research focused on extracting road topology and detecting road damage using aerial images. The authors proposed a deep learning-based framework called DeepRoadMapper that utilized convolutional and recurrent neural networks. The study achieved accurate road damage detection and topology extraction, demonstrating the potential of aerial imagery for road inspection.

4. "Pavement Distress Detection Using Deep Convolutional Neural Networks" (2019) by T. Hasan et al.:

   The study employed deep convolutional neural networks for pavement distress detection, including road cracks and potholes. The authors utilized the transfer learning technique and fine-tuned pre-trained models to achieve high accuracy in detecting pavement distress. The research highlighted the effectiveness of deep learning for automated road damage detection.

5. "DeepCrack: A Deep Hierarchical Feature Learning Architecture for Crack Detection in Concrete" (2018) by Y. Li et al.:

   This study focused on crack detection in concrete structures, which includes road surfaces. The authors proposed a deep learning architecture called DeepCrack, which utilized both low-level and high-level features for accurate crack detection. The results demonstrated the effectiveness of the approach in detecting road cracks.

6. "Unsupervised Road Damage Detection Using Convolutional Neural Network with Synthetic Data" (2019) by T. Sonobe et al.:

   The research aimed to overcome the limitations of labeled road damage datasets by proposing an unsupervised approach. The authors utilized synthetic data generation techniques to train a convolutional neural network for road damage detection without relying on labeled data. The study showed promising results in detecting road damage without the need for extensive labeling efforts.

7. "Road Damage Detection and Classification using Multi-Scale Feature Learning with Convolutional Neural Networks" (2019) by M. Xie et al.:

   This study proposed a multi-scale feature learning approach for road damage detection and classification. The authors used a convolutional neural network architecture to extract multi-scale features from road images. The research demonstrated improved performance in detecting and classifying different types of road defects.

These studies highlight the growing interest in utilizing machine learning, particularly deep learning techniques, for road damage detection. They showcase various approaches and architectures that leverage convolutional neural networks, transfer learning, synthetic data, and aerial imagery to achieve accurate and automated road damage detection.

## 2.1 Existing System:

Maintenance departments need to regularly assess the quality of the roads to properly maintain them. Currently, this is done by yearly inspections or in response to reports from the public. The inspection is sometimes done by workers walking along the streets and recording the conditions on paper which later is put into a database. In other cases, the agency makes use of special vehicles that measure the road surface.

There are several existing systems for road damage detection using machine learning. Here are a few examples:

**1. RoadAI:**

RoadAI is a system developed by the University of Tokyo that uses deep learning algorithms for road damage detection. It utilizes a convolutional neural network (CNN) trained on a large dataset of road images to identify and classify different types of road defects. The system can analyze images captured by cameras mounted on vehicles or drones, enabling real-time detection and mapping of road damage.

**2. Road Damage Detection and Classification (RDDC):**

RDDC is a system developed by researchers at the University of Bristol. It combines image processing techniques with machine learning algorithms to detect and classify road damage. The system extracts feature from road images and uses support vector machines (SVMs) to classify the detected damage into different categorie such as cracks, potholes, or surface deterioration.

**3. Road Crack Detection System (RCDS):**

RCDS is a system developed by researchers at the University of California, Berkeley. It uses a combination of deep learning and image processing techniques to detect and

classify road cracks. The system employs a deep convolutional neural network (CNN) to automatically learn crack features from road images and classify them into different severity levels.

**4. Road Damage Detection and Evaluation System (RDDES):**

RDDES is a system developed by researchers at the University of California, Davis. It uses machine learning algorithms to detect and evaluate road damage. The system utilizes a combination of image processing techniques and support vector machines (SVMs) to identify and classify different types of road defects, including cracks, potholes, and surface distress.

**5. Road Crack Detection and Analysis System (RCDAS):**

RCDAS is a system developed by researchers at the University of Twente in the Netherlands. It uses machine learning algorithms to detect and analyze road cracks. The system combines image processing techniques with random forest classifiers to identify cracks in road images and estimate their severity levels.

These existing systems demonstrate the effectiveness of machine learning in automating the road damage detection process. By leveraging deep learning algorithms, image processing techniques, and classification models, these systems can accurately identify and classify different types of road defects, enabling timely repairs and maintenance of road infrastructure.

## 2.2 Proposed Method:

### 2.2.1 YOLO:

You Only Look Once (YOLO) proposes using an end-to-end neural network that makes predictions of bounding boxes and class probabilities all at once. It differs from the approach taken by previous object detection algorithms, which repurposed classifiers to perform detection.

Following a fundamentally different approach to object detection, YOLO achieved state-of-the-art results, beating other real-time object detection algorithms by a large margin.

While algorithms like Faster RCNN work by detecting possible regions of interest using the Region Proposal Network and then performing recognition on those regions separately, YOLO performs all of its predictions with the help of a single fully connected layer.

Methods that use Region Proposal Networks perform multiple iterations for the same image, while YOLO gets away with a single iteration.

Several new versions of the same model have been proposed since the initial release of YOLO in 2015, each building on and improving its predecessor. Here is a timeline showcasing YOLO's development in recent years.

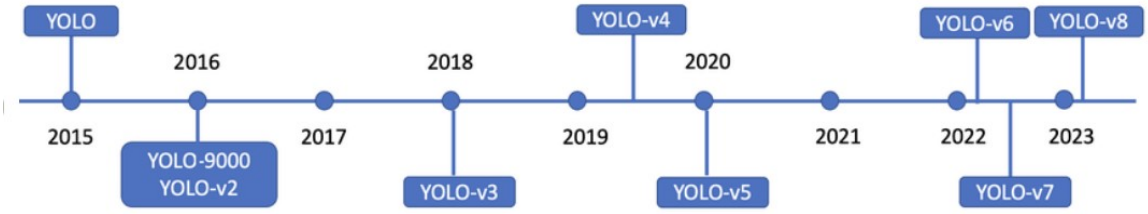



Fig:Time line of yolo variant

## How does YOLO work? YOLO Architecture

The YOLO algorithm takes an image as input and then uses a simple deep convolutional neural network to detect objects in the image. The architecture of the CNN model that forms the backbone of YOLO is shown below

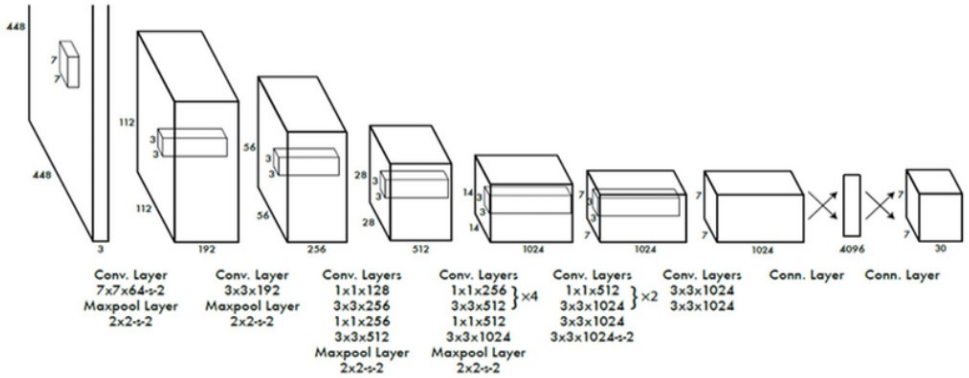



Fig: Yolo Architecture

The first 20 convolution layers of the model are pre-trained using ImageNet by plugging in a temporary average pooling and fully connected layer. Then, this pre-trained model is converted to perform detection since previous research showcased that adding convolution and connected layers to a pre-trained network improves performance. YOLO's final fully connected layer predicts both class probabilities and bounding box coordinates.

YOLO divides an input image into an S × S grid. If the center of an object falls into a grid cell, that grid cell is responsible for detecting that object. Each grid cell predicts B bounding boxes and confidence scores for those boxes. These confidence scores reflect how confident the model is that the box contains an object and how accurate it thinks the predicted box is.

YOLO predicts multiple bounding boxes per grid cell. At training time, we only want one bounding box predictor to be responsible for each object. YOLO assigns one predictor to be "responsible" for predicting an object based on which prediction has the highest current IOU with the ground truth. This leads to specialization between the bounding box predictors. Each predictor gets better at forecasting certain sizes, aspect ratios, or classes of objects, improving the overall recall score.

One key technique used in the YOLO models is **non-maximum suppression (NMS)**. NMS is a post-processing step that is used to improve the accuracy and efficiency of object detection. In object detection, it is common for multiple bounding boxes to be generated for a single object in an image. These bounding boxes may overlap or be located at different positions, but they all represent the same object. NMS is used to identify and remove redundant or incorrect bounding boxes and to output a single bounding box for each object in the image.

## What's new with YOLO v7?

YOLO v7, the latest version of YOLO, has several improvements over the previous versions. One of the main improvements is the use of anchor boxes.

Anchor boxes are a set of predefined boxes with different aspect ratios that are used to detect objects of different shapes. YOLO v7 uses nine anchor boxes, which allows it to

detect a wider range of object shapes and sizes compared to previous versions, thus helping to reduce the number of false positives.

A key improvement in YOLO v7 is the use of a new loss function called “focal loss.” Previous versions of YOLO used a standard cross-entropy loss function, which is known to be less effective at detecting small objects. Focal loss battles this issue by down-weighting the loss for well-classified examples and focusing on the hard examples—the objects that are hard to detect.

YOLO v7 also has a higher resolution than the previous versions. It processes images at a resolution of 608 by 608 pixels, which is higher than the 416 by 416 resolution used in YOLO v3. This higher resolution allows YOLO v7 to detect smaller objects and to have a higher accuracy overall.

One of the main advantages of YOLO v7 is its speed. It can process images at a rate of 155 frames per second, much faster than other state-of-the-art object detection algorithms. Even the original baseline YOLO model was capable of processing at a maximum rate of 45 frames per second. This makes it suitable for sensitive real-time applications such as surveillance and self-driving cars, where higher processing speeds are crucial.

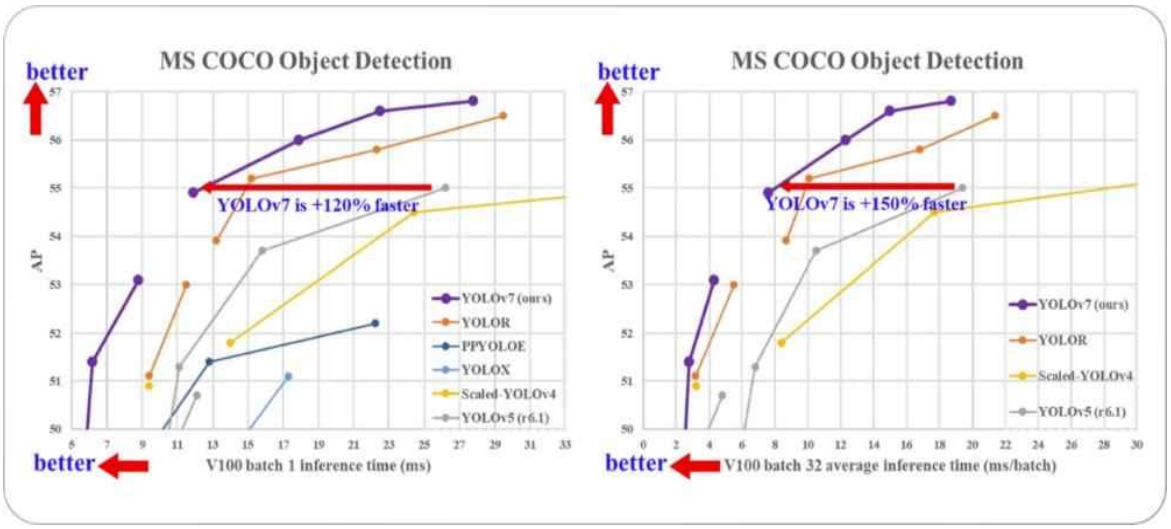



Fig: perfomance of Yolo

Regarding accuracy, YOLO v7 performs well compared to other object detection algorithms. It achieves an average precision of 37.2% at an IoU (intersection over union)

threshold of 0.5 on the popular COCO dataset, which is comparable to other state-of-the-art object detection algorithms. The quantitative comparison of the performance is shown below.

However, it should be noted that YOLO v7 is less accurate than two-stage detectors such as Faster R-CNN and Mask R-CNN, which tend to achieve higher average precision on the COCO dataset but also require longer inference times

## Limitations of YOLO v7

YOLO v7 is a powerful and effective object detection algorithm, but it does have a few limitations.

1. YOLO v7, like many object detection algorithms, struggles to detect small objects. It might fail to accurately detect objects in crowded scenes or when objects are far away from the camera.
2. YOLO v7 is also not perfect at detecting objects at different scales. This can make it difficult to detect objects that are either very large or very small compared to the other objects in the scene.
3. YOLO v7 can be sensitive to changes in lighting or other environmental conditions, so it may be inconvenient to use in real-world applications where lighting conditions may vary.
4. YOLO v7 can be computationally intensive, which can make it difficult to run in real-time on resource-constrained devices like smartphones or other edge devices.

## Benefits Of YOLO:

- Fast. Good for real-time processing.
- Predictions (object locations and classes) are made from one single network. Can be trained end-to-end to improve accuracy.
- YOLO is more generalized. It outperforms other methods when generalizing from natural images to other domains like artwork.

# CHAPTER 3

# METHODOLOGY

## 3.1 Image Dataset of Road Surface Damage:

Though an image dataset of the road surface exists, called the road damage detector dataset (Geiger et al., 2013), it is primarily used for applications related to automatic driving. In each study, the researchers independently propose unique methods using acquired road images. Therefore, a comparison between the methods presented in these studies is difficult. Furthermore, according to Mohan et al. (Mohan and Poobal, 2017), there are few studies that construct damage detection models using real data, and 20 of these studies use road images taken directly from above the road. In fact, it is difficult to reproduce the road images taken directly from above the roads, because doing so involves installing a camera outside the car body, which, in many countries, is a violation of the law; in addition, it is costly to maintain a dedicated car solely for road images. Therefore, we have taken a dataset of road damage images using the road images captured using a smartphone on the dashboard of a general passenger car; in addition, we made this dataset publicly available. Moreover, we show that road surface damage can be detected with considerably high accuracy even with images acquired by employing such a simple method.

To assemble a dataset for YOLO v7, which is a popular object detection algorithm, follow these steps:

**1. Define the Object Classes:**

Determine the specific object classes you want to detect using YOLO v7. For example, if you're interested in detecting cars, pedestrians, and bicycles, those would be your object classes.

**2. Data Collection:**

Gather a diverse set of images or videos that contain the desired object classes. You can acquire these images through various means, such as capturing them yourself, using publicly available datasets, or scraping images from the internet. Ensure that the

dataset covers a wide range of lighting conditions, backgrounds, scales, and viewpoints to make the model robust.

**3. Annotation:**

Annotate the objects of interest in the images or videos. For each object instance, you need to provide the bounding box coordinates (top-left corner coordinates and width and height) along with the corresponding object class label. Annotation tools like LabelImg, RectLabel, or VIA can assist in this process.

**4. Organize the Dataset:**

Create a folder structure for your dataset. Divide it into separate directories for images and annotations. Each image should have a corresponding annotation file in a compatible format, such as YOLO format (.txt) or Pascal VOC format (.xml).

**5. Data Augmentation (optional):**

To improve model generalization and increase dataset size, consider applying data augmentation techniques such as flipping, rotation, scaling, or adding noise to your images. This step can help the model handle variations in the real-world scenarios.

**6. Splitting the Dataset:**

Divide your dataset into training, validation, and testing subsets. Typically, a common split is 80% for training, 10% for validation, and 10% for testing. Ensure that images from the same scene or location are not present in multiple subsets to avoid biased evaluation results.

**7. Convert Annotations to YOLO Format:**

If your annotations are not already in YOLO format, convert them accordingly. For YOLO v7, the format typically includes the object class index followed by the normalized bounding box coordinates (center coordinates, width, and height) relative to the image dimensions.

**8. Generating Labels File:**

Create a labels file that maps the object class names to their corresponding class indices. This file is necessary for training YOLO v7.

**9. Train, Validate, and Test:**

With your assembled dataset, split into subsets, and properly formatted annotations, you can proceed to train YOLO v7 using the training subset, validate its performance on the validation subset, and evaluate it on the testing subset.

Remember to follow ethical guidelines, ensure the legality of image usage, and respect privacy regulations when collecting or using datasets for object detection purposes.

### 3.1.1 Object Detection System

In general, for object detection, methods that apply an image classifier to an object detection task have become mainstream; these methods entail varying the size and position of the object in the test image, and then using 3 the classifier to identify the object. In the past few years, an approach involving the extraction of multiple candidate regions of objects using region proposals as typified by R-CNN, then making a classification decision with candidate regions using classifiers has also been reported. However, the R-CNN approach can be time consuming because it requires more crops, leading to significant duplicate computation from overlapping crops. As described above, image processing methods have historically developed at a considerable pace. In our study, we primarily focus on four recent object detection systems: The Faster R-CNN (Ren et al., 2015), the You Look Only Once (YOLO) (Redmon et al., 2016; Redmon and Farhadi, 2016) system, the Region-based Fully Convolutional Networks (R-FCN) system (Dai et al., 2016), and the Single Shot Multibox Detector (SSD) system (Liu et al., 2016).

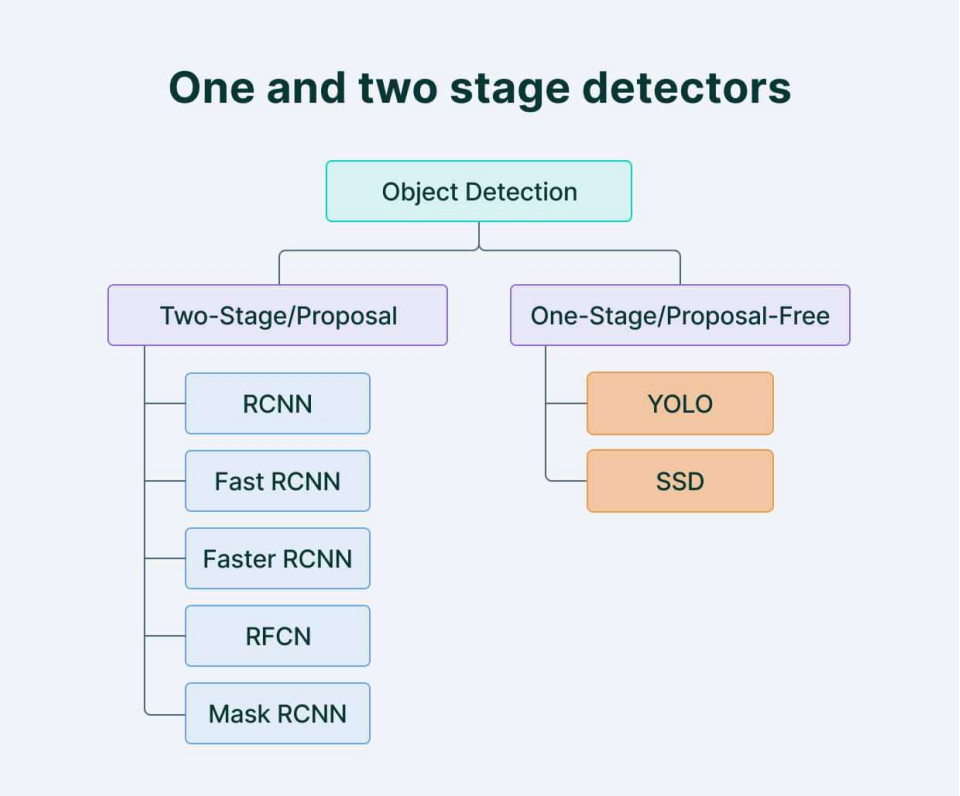



Fig: Object Detection types

## 3.1.2 Project Methodology:

We used a dataset of 9,053 road damage images which contains the bounding box of each class for the eight types of road damage. Furthermore, we showed that the type of damage from among the eight types can be identified with high accuracy. To build a model using YOLO according to our requirement.

**Steps given below were followed to make our project**

Step 1: Installation of pandas,opendatasets and NumPy

Step 2: Selecting the Desired Algorithm i.e., Tiny YOLO

Step 3: Collecting data and annotations

Step 4: Modifying Model

Step 5: Running and deploying the model

## 3.1.3 Flow Chart of System:

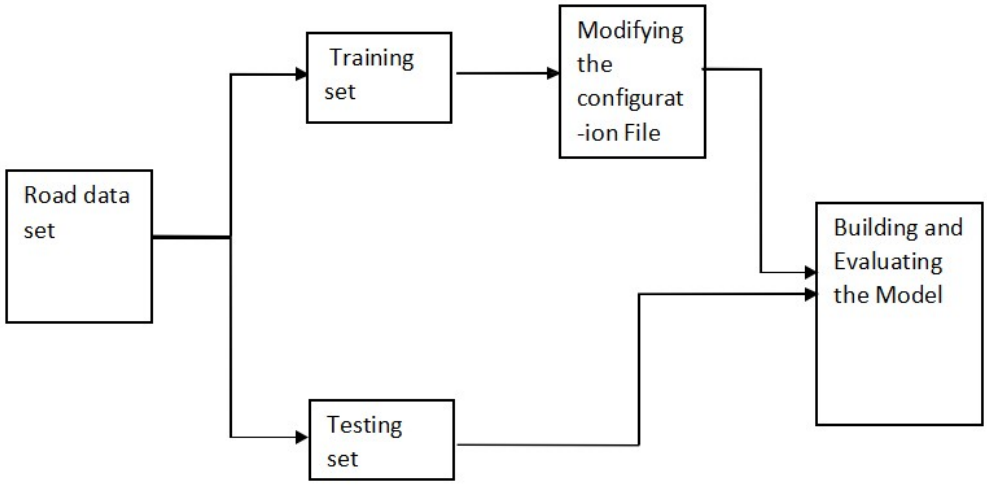



Fig 3.1. Flow chart of the model.

## 3.1.4 Architecture of the Model:

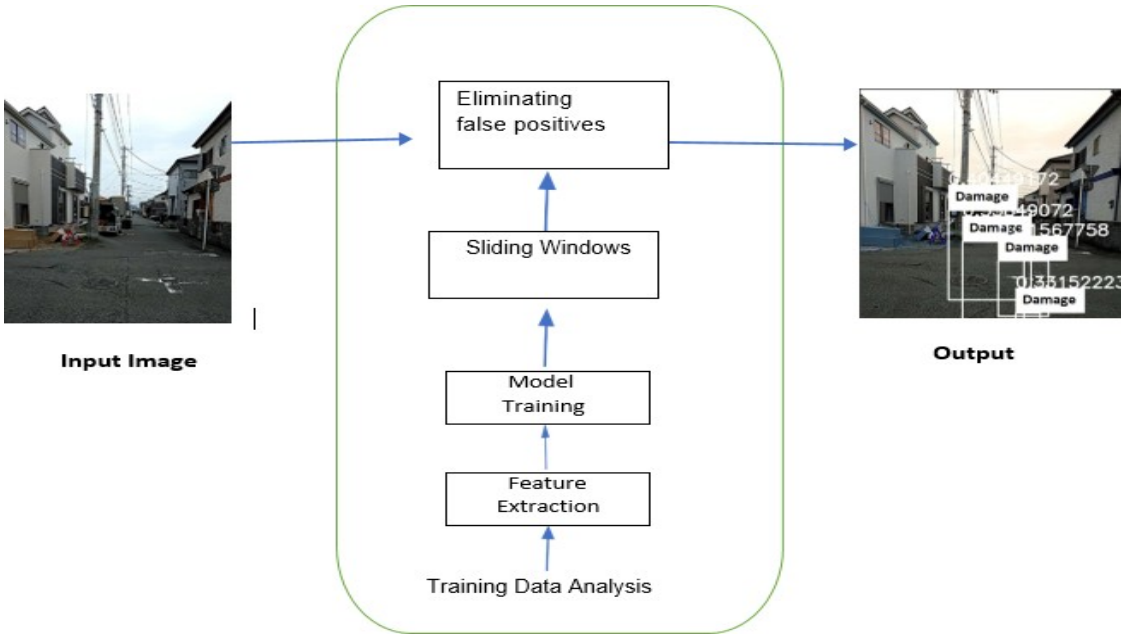



Fig 3.2. Process of proposed

## 3.2 UML DIAGRAMS

### 3.2.1 Use Case Diagram:

It is the most well-known diagram type of the behavioral UML types, Use-case diagrams give a graphic overview of the actors involved in a system, different functions needed by those actors and how these different functions interact.

It's a great starting point for any project discussion because you can easily identify the main actors involved and the main processes of the system.

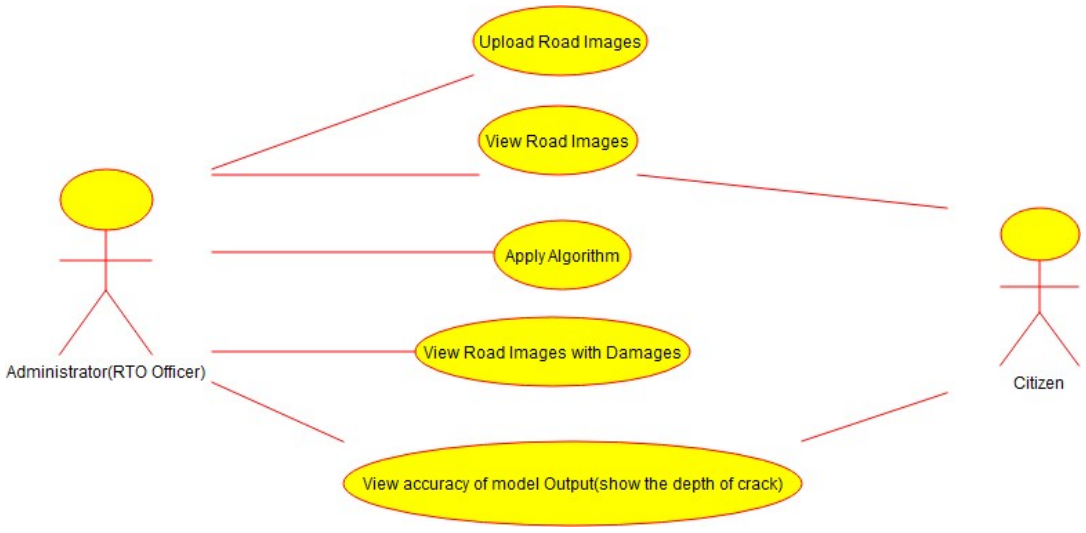



Fig 3.3. Use Case Diagram for Detecting Road damage

### 3.2.2 Class Diagram:

Class diagrams are the main building block of any object-oriented solution. It shows the classes in a system, attributes, and operations of each class and the relationship between each class. In most modelling tools, a class has three parts. Name at the top, attributes in the middle and operations or methods at the bottom. In a large system with many related classes, classes are grouped together to create class diagrams.

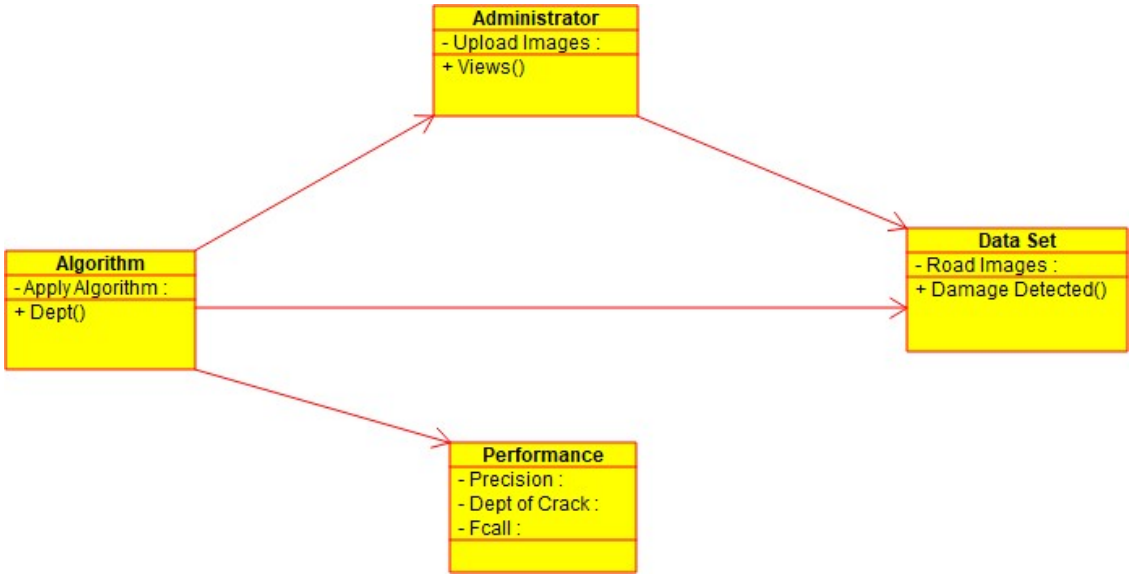



Fig 3.4. Class Diagram for detecting Road damage

### 3.2.3 Activity Diagram:

Activity diagrams represent workflows in a graphical way. They can be used to describe the business workflow or the operational workflow of any component in a system. Sometimes activity diagrams are used as an alternative to State machine diagrams.

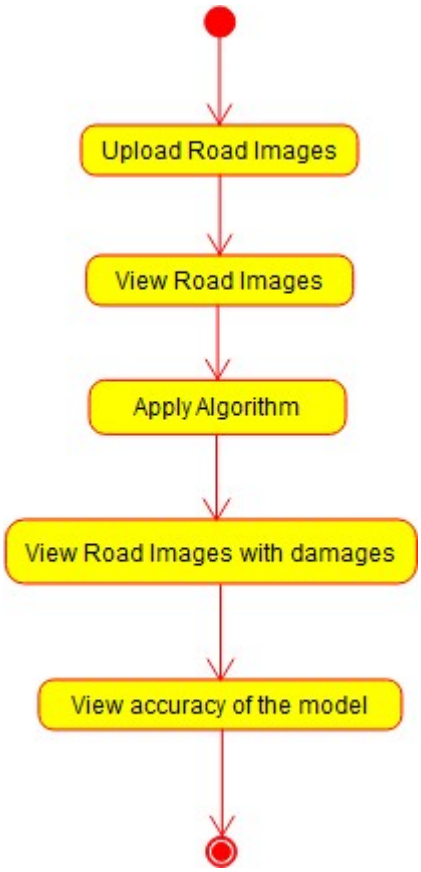



Fig 3.5. Activity Diagram for detecting Road damage

### 3.2.4 Deployment Diagram:

Deployment diagrams are useful when your software solution is deployed across multiple machines with each having a unique configuration. Below is a deployment diagram for Plant Disease Identification Using Leaf Images.

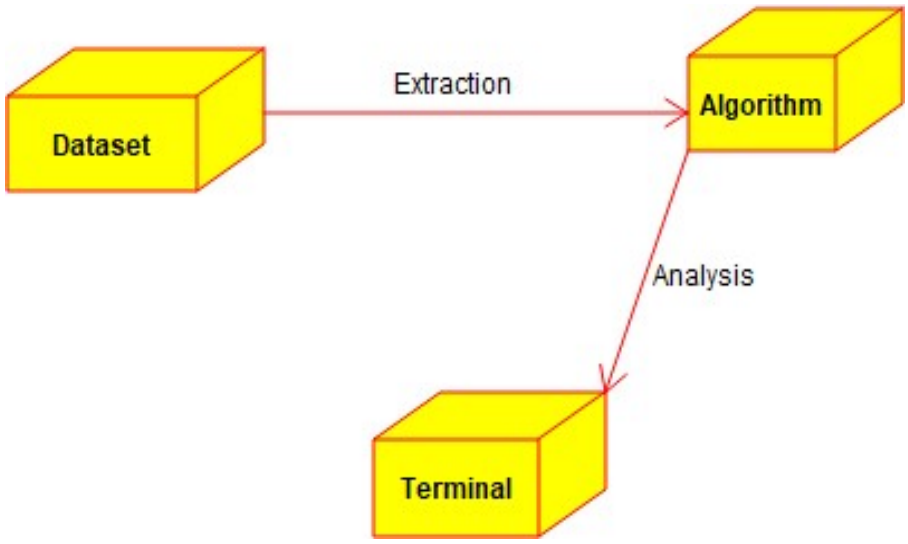



Fig 3.6. Deployment Diagram for detecting Road damage

### 3.2.5 Interaction Diagram:

Interaction overview diagrams are very similar to activity diagrams. While activity diagrams show a sequence of processes, Interaction overview diagrams show a sequence of interaction diagrams. They are a collection of interaction diagrams and the order they happen.

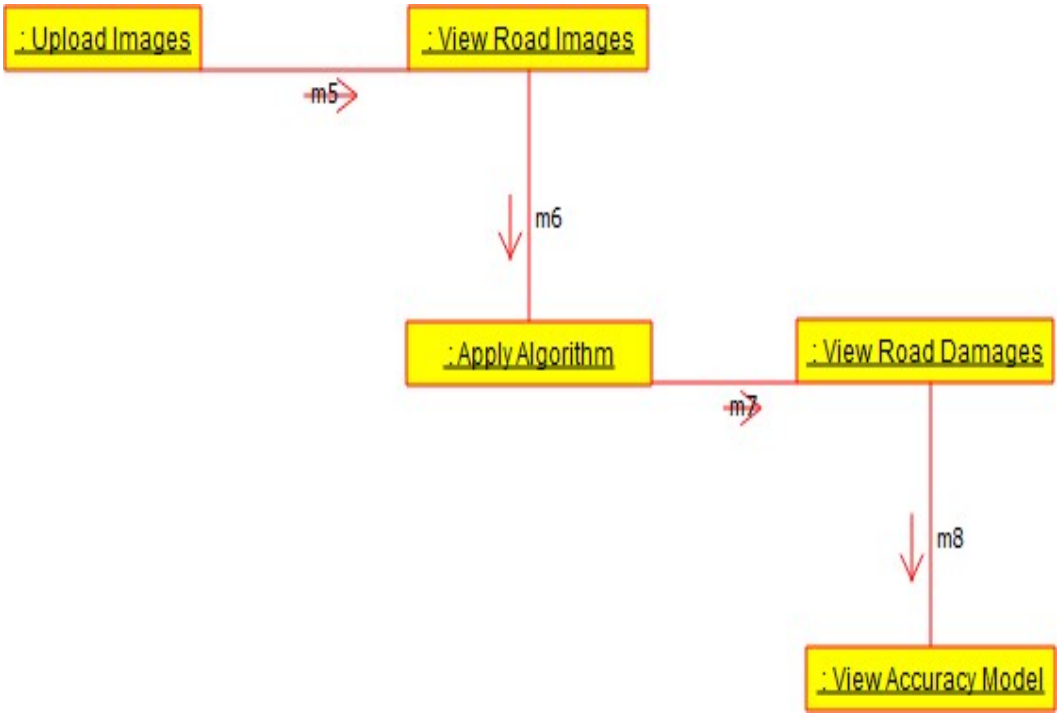



Fig 3.7. Interaction Diagram for detecting Road damage

### 3.2.6 Object Diagram:

Object Diagrams, sometimes referred to as Instance diagrams are very similar to class diagrams. Like class diagrams, they also show the relationship between objects, but they use real world examples.

They show how a system will look like at a given time. Because there is data available in the objects, they are used to explain complex relationships between objects.

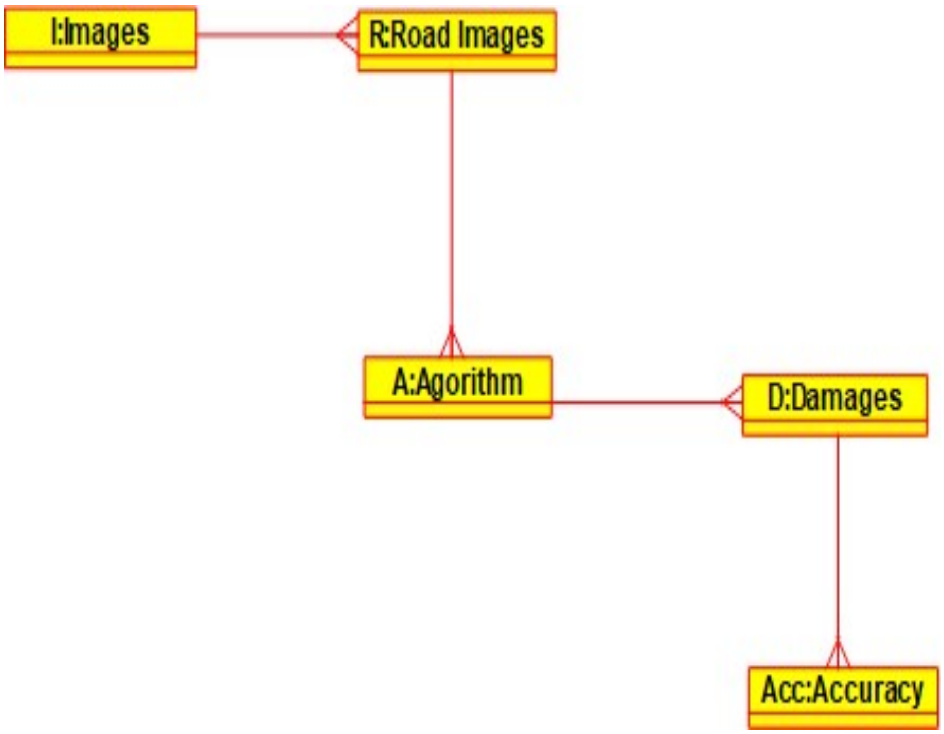



Fig 3.8. Object Diagram for detecting Road damage

### 3.2.7 Component Diagram:

A component diagram displays the structural relationship of components of a software system. These are mostly used when working with complex systems with many components. Components communicate with each other using interfaces. The image below shows a component diagram for Road damage detection.


Fig 3.9. Component Diagram for Road Damage

### 3.2.7 Component Diagram:

A component diagram displays the structural relationship of components of a software system. These are mostly used when working with complex systems with many components. Components communicate with each other using interfaces. The image below shows a component diagram for Road damage detection.

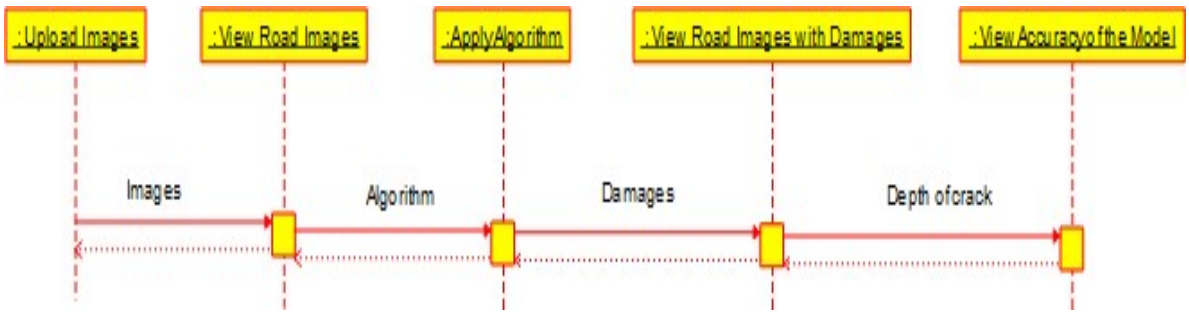



Fig 3.10. Sequence Diagram for Detecting Road Damage

## 3.3 Dependencies

## Pandas:

Pandas is an open-source Python library that provides high-performance data manipulation and analysis tools. It is built on top of NumPy and is widely used for handling structured data and performing data analysis tasks.

Key features of Pandas include:

1. **DataFrame:** The core data structure in Pandas is the DataFrame, which is a two-dimensional table-like data structure. It allows you to store and manipulate labeled and structured data, similar to a spreadsheet or SQL table. DataFrames can handle both homogeneous and heterogeneous data types.

2. **Data Manipulation**: Pandas offers a wide range of functions and methods for data manipulation tasks such as filtering, sorting, merging, grouping, reshaping, and aggregating data. These operations enable users to transform, clean, and reshape data efficiently.

3.**Data Alignment:** Pandas provides powerful alignment features, allowing data to be automatically aligned and aligned based on row and column labels. This makes it easy to work with and combine data from different sources.

4. **Missing Data Handling:** Pandas provides methods to handle missing or null values in datasets, allowing users to fill, drop, or interpolate missing values based on various strategies.

5. **Time Series and Date Functionality:** Pandas has extensive support for working with time series data and includes a range of date and time manipulation functions. It provides functionality for time-based indexing, resampling, time zone handling, and more.

6. **Integration with Other Libraries:** Pandas seamlessly integrates with other libraries in the scientific Python ecosystem, such as NumPy, Matplotlib, and Scikit-learn. It can efficiently read and write data in various file formats, including CSV, Excel, SQL databases, and more.

7. **Performance and Efficiency**: Pandas is designed for high-performance data manipulation and analysis. It leverages the underlying power of NumPy arrays and optimized algorithms to provide fast and efficient operations on large datasets.

Pandas is widely used in data analysis, data science, and machine learning tasks. It provides a convenient and expressive way to work with structured data, making it easier to extract insights, clean and preprocess data, perform statistical analysis, and prepare data for modeling. Its versatility and extensive functionality make it a fundamental tool in the Python data science ecosystem.

**OpenDataSets:**

`opendatasets` is a Python library that simplifies the process of downloading and working with publicly available datasets. It provides a convenient interface to access a wide range of datasets from various sources directly from your Python environment.

Key features of the `opendatasets` library include:

1. **Dataset Catalog:** `opendatasets` maintains a curated catalog of publicly available datasets. The catalog includes datasets from various domains, such as machine learning, natural language processing, computer vision, social sciences, and more.

2. **Simplified Download:** The library offers a simple API to download datasets with just a few lines of code. You can specify the dataset's name or URL, and `opendatasets` takes care of fetching and storing the data in your local environment.

3. **Data Exploration:** `opendatasets` provides functions to quickly explore and understand the downloaded dataset. You can view the dataset's metadata, browse through the file structure, and access sample records to get a glimpse of the data's structure and content.

4. **Version Management:** `opendatasets` supports version management, allowing you to download specific versions of a dataset. This is particularly useful when working with evolving or regularly updated datasets.

5. **Integration with Pandas:** The library seamlessly integrates with Pandas, a popular Python library for data manipulation and analysis. It provides functions to load the

downloaded dataset into a Pandas DataFrame, enabling you to leverage the full power of Pandas for data preprocessing and analysis.

Here's a simple example of using `open datasets` to download and explore a dataset:

```python
import opendatasets as od

# Download a dataset

dataset URL = 'https://www.kaggle.com/username/dataset-name'

od.download(dataset_url)


# Explore the dataset

od.info('dataset-name')


# Load the dataset into a Pandas DataFrame

import pandas as pd

df = pd.read_csv('dataset-name/data.csv')
```

With `opendatasets`, you can quickly access and work with a wide variety of datasets without leaving your Python environment. It simplifies the data acquisition process, making it easier to experiment, analyze, and build machine learning models using publicly available datasets.

## **Wandb:**

Wandb (short for Weights & Biases) is a platform that helps track, visualize, and analyze machine learning experiments. It provides tools and infrastructure for experiment

tracking, model visualization, hyperparameter tuning, and collaboration among team members.

Key features of Wandb include:

1. **Experiment Tracking:** Wandb allows you to log and track experiments, including metrics, hyperparameters, configurations, and code versions. It provides a dashboard where you can monitor and compare multiple runs, making it easier to understand the performance and behavior of your models.

2. **Visualization and Analysis:** Wandb offers interactive visualizations to explore and analyze experiment results. You can plot various metrics over time, compare different runs, and visualize the model's performance. It also provides tools for analyzing hyperparameter sweeps and understanding the impact of different configurations on the model's performance.

3. **Hyperparameter Optimization:** Wandb integrates with popular hyperparameter optimization libraries such as Optuna and Ray Tune. You can easily define search spaces and track the results of hyperparameter tuning experiments, helping you find optimal configurations for your models.

4. **Artifact Management:** Wandb allows you to save and version artifacts, such as model checkpoints, datasets, and preprocessing scripts. This ensures reproducibility and makes it easy to share and collaborate on projects.

5. **Team Collaboration:** Wandb provides features for collaboration within teams. You can share experiment results, dashboards, and artifacts with team members, facilitating communication and knowledge sharing.

6. **Integration with Machine Learning Frameworks:** Wandb seamlessly integrates with popular machine learning frameworks such as TensorFlow, PyTorch, and scikit-learn. It provides lightweight libraries and APIs that enable easy instrumentation of your code to log metrics, gradients, and visualizations during training and evaluation.

By using Wandb, you can gain better visibility into your machine learning experiments, track and compare results, and collaborate with team members more

effectively. It helps streamline the experimentation process and enables you to make informed decisions based on data-driven insights.

### 3.3.1 TensorFlow:

TensorFlow is an open-source machine learning framework developed by Google. It provides a comprehensive ecosystem of tools, libraries, and resources for building and deploying machine learning models. TensorFlow is widely used for various tasks, including deep learning, neural networks, natural language processing, computer vision, and more.

Key features of TensorFlow include:

1. **High-level APIs:** TensorFlow offers high-level APIs like Keras, which simplifies the process of building, training, and evaluating machine learning models. Keras provides an intuitive and user-friendly interface for defining neural networks.

2. **Flexibility:** TensorFlow provides a flexible architecture that allows users to define and execute computations efficiently on different hardware platforms, including CPUs, GPUs, and specialized accelerators like TPUs (Tensor Processing Units).

3. **Data Flow Graphs:** TensorFlow represents computations as data flow graphs. Nodes in the graph represent mathematical operations, while the edges represent the data that flows between the nodes. This graph-based approach enables TensorFlow to efficiently optimize and parallelize computations.

4. **Automatic Differentiation:** TensorFlow automatically computes gradients for optimization algorithms through a process called automatic differentiation. This feature simplifies the implementation of backpropagation, which is a key component in training neural networks.

5. **TensorBoard:** TensorFlow comes with TensorBoard, a visualization toolkit that allows users to analyze and debug models. TensorBoard provides visualizations of model architectures, training metrics, and more, making it easier to understand and monitor the training process.

6. **Model Serving:** TensorFlow provides tools for deploying and serving machine learning models in production environments. These tools enable developers to integrate trained models into real-world applications and serve predictions at scale.

7. **Community and Ecosystem:** TensorFlow has a vibrant and active community, with a wide range of resources, tutorials, and pre-trained models available. The ecosystem around TensorFlow includes libraries and extensions for specific tasks, such as TensorFlow Hub, TensorFlow.js, and TensorFlow Lite.

TensorFlow is written in Python and supports multiple programming languages, including Python, C++, Java, and more. It has become one of the most popular and widely used frameworks for machine learning and deep learning due to its versatility, scalability, and extensive community support.

### 3.3.2 NumPy:

NumPy is a powerful open-source library for numerical computing in Python. It stands for "Numerical Python" and provides a high-performance multidimensional array object, along with a collection of mathematical functions, linear algebra routines, and tools for working with arrays.

Key features of NumPy include:

1. **Multidimensional Array:** NumPy's main object is the ndarray (n-dimensional array), which is a homogeneous collection of elements with a fixed size. The ndarray allows efficient storage and manipulation of large datasets, such as matrices or tensors, and supports various numerical operations.

2. **Efficient Computations:** NumPy provides optimized routines for mathematical and logical operations on arrays. These operations are implemented in C or Fortran, which makes them significantly faster than equivalent Python operations.

3. **Broadcasting:** NumPy allows for implicit element-wise operations between arrays of different shapes and sizes. This broadcasting functionality eliminates the need for explicit loops and enables concise and efficient code for many array operations.

4. **Array Manipulation:** NumPy offers a wide range of functions for reshaping, slicing, and manipulating arrays. These functions allow you to extract specific elements, combine arrays, change dimensions, and perform other operations to manipulate the shape and content of arrays.

5. **Linear Algebra:** NumPy provides a comprehensive set of linear algebra functions, including matrix multiplication, decomposition, eigenvalues, and more. These functions are essential for many scientific and numerical applications.

6. **Integration with Python:** NumPy seamlessly integrates with other scientific computing libraries in Python, such as SciPy (Scientific Python) for advanced mathematical and scientific computations.

7. **NumPy's API:** NumPy provides a rich set of functions and methods for array manipulation, statistical analysis, random number generation, and more. It also allows users to create custom data types, handle missing values, and perform complex operations efficiently.

NumPy is widely used in various domains, including scientific research, data analysis, machine learning, and artificial intelligence. Its efficient array operations and powerful mathematical functions make it an essential tool for working with numerical data in Python. Many other libraries, such as Pandas, Scikit-learn, and TensorFlow, rely on NumPy as a fundamental building block.

### 3.3.3 Open CV:

OpenCV (Open Source Computer Vision Library) is released under a BSD license and hence it’s free for both academic and commercial use. It has C++, Python and Java interfaces and supports Windows, Linux, Mac OS, iOS and Android. OpenCV was designed for computational efficiency and with a strong focus on realtime applications. Written in optimized C/C++, the library can take advantage of multi-core processing. Enabled with OpenCL, it can take advantage of the hardware acceleration of the underlying heterogeneous compute platform.

### 3.3.4 OS:

The OS module in Python provides a way of using operating system dependent functionality. The functions that the OS module provides allows you to interface with the underlying operating system that Python is running on -- be that Windows, Mac or Linux. You can find important information about your location or about the process.

### 3.3.5 KERAS:

Keras is an open-source neural-network library written in Python. It can run on top of TensorFlow, Microsoft Cognitive Toolkit, Theano, or PlaidML. Designed to enable fast experimentation with deep neural networks, it focuses on being user-friendly, modular, and extensible. It was developed as part of the research effort of project ONEIROS (Open-ended Neuro-Electronic Intelligent Robot Operating System), and its primary author and maintainer is François Chollet, a Google engineer. Chollet also is the author of the XCeption deep neural network model.

## 3.4 MODULES:

### 3.4.1 Image acquisition:

The initial process is to collect the data from the public repository. It takes the image as input for further processing. Image acquisition is the step where the road image is taken as input.

### 3.4.2 Image Pre-processing:

As the images are acquired from the real field it may contain blocks or cracks. The purpose of data pre-processing is to eliminate the cracks in the image, to adjust the pixel values. It enhances the quality of the image.

### 3.4.3 Classification:

In this phase to detect the road damage, we are using python and some libraries like YOLO,Keras, TensorFlow etc.

# CHAPTER 4

# IMPLEMENTAION

## IMPLEMENTAION:

Implementing a road damage detection project typically involves several steps. Here's a high-level outline of the process:

**1. Data Collection:** Obtain a dataset of road images that contain various types of road damages, such as cracks, potholes, or other surface defects. This dataset can be collected through manual surveys, publicly available datasets, or using specialized vehicles equipped with cameras.

**2. Data Preprocessing:** Clean and preprocess the collected images to ensure consistency and remove any unnecessary artifacts. Preprocessing steps may include resizing, normalization, noise reduction, and color adjustments.

**3. Annotation and Labeling:** Manually label the collected images to indicate the presence and location of road damages. This step requires annotating the images with bounding boxes, polygons, or semantic segmentation masks to identify the damaged areas accurately.

**4. Training Data Preparation:** Split the annotated dataset into training, validation, and testing sets. The training set is used to train the road damage detection model, the validation set helps tune the model's hyperparameters, and the testing set evaluates the model's performance.

**5. Model Selection:** Choose an appropriate deep learning model for road damage detection. Popular choices include Convolutional Neural Networks (CNNs) like VGG, ResNet, or EfficientNet. You can also consider pre-trained models, such as those available in popular deep learning frameworks like TensorFlow or PyTorch.

**6. Model Training:** Train the selected model using the annotated training data. During training, the model learns to recognize road damages based on the labeled examples.

Adjust the model's parameters iteratively using an optimization algorithm (e.g., stochastic gradient descent) to minimize the difference between predicted and actual labels.

**7. Model Evaluation:** Assess the trained model's performance using the validation set. Measure metrics such as precision, recall, F1-score, and accuracy to gauge how well the model can detect road damages.

**8. Model Optimization:** Fine-tune the model and experiment with different architectures, hyperparameters, and data augmentation techniques to improve its performance. This step involves adjusting learning rates, regularization techniques, or adding additional layers to enhance the model's ability to detect road damages accurately.

**9. Model Testing:** Use the final trained model to detect road damages in unseen images from the testing set. Evaluate its performance using the same metrics as in the evaluation phase.

**10. Deployment:** Once you are satisfied with the model's performance, deploy it in a production environment. This may involve integrating the model into an application or system that can process real-time road images and provide detection results.

## 5.2 SAMPLE CODE :

```
Pip install opendatasets
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/ Collecting opendatasets Downloading opendatasets-0.1.22-py3-none-any.whl (15 kB)
  Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from opendatasets) (4.65.0) Requirement already satisfied: kaggle in /usr/local/lib/python3.10/dist-packages (from opendatasets) (1.5.13) Requirement already satisfied: click in /usr/local/lib/python3.10/dist-packages (from opendatasets) (8.1.3) Requirement already satisfied: six>=1.10 in /usr/local/lib/python3.10/dist-packages (from kaggle->opendatasets) (1.16.0)
  Requirement already satisfied: certifi in /usr/local/lib/python3.10/dist-packages (from
```

kaggle->opendatasets) (2023.5.7)

Requirement already satisfied: python-dateutil in /usr/local/lib/python3.10/dist-packages (from kaggle->opendatasets) (2.8.2)

Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (from kaggle->opendatasets) (2.27.1)

Requirement already satisfied: python-slugify in /usr/local/lib/python3.10/dist-packages (from kaggle->opendatasets) (8.0.1)

Requirement already satisfied: urllib3 in /usr/local/lib/python3.10/dist-packages (from kaggle->opendatasets) (1.26.16)

Requirement already satisfied: text-unencode>=1.3 in /usr/local/lib/python3.10/dist-packages (from python-slugify->kaggle->opendatasets) (1.3)

Requirement already satisfied: charset-normalizer~=2.0.0 in /usr/local/lib/python3.10/dist-packages (from requests->kaggle->opendatasets) (2.0.12)

Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests->kaggle->opendatasets) (3.4)

Installing collected packages: opendatasets

Successfully installed opendatasets-0.1.22

Pip install pandas

Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/ Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-packages (1.5.3)

Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.10/dist-packages (from pandas) (2.8.2)

Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas) (2022.7.1)

Requirement already satisfied: numpy>=1.21.0 in /usr/local/lib/python3.10/dist-packages (from pandas) (1.22.4)

Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.8.1->pandas) (1.16.0)

```
import opendatasets as od
import pandas
od.download("https://www.kaggle.com/datasets/rajdalsaniya/pothole-detection-dataset")
```

```
Please provide your Kaggle credentials to download this dataset. Learn more: http://bit.ly/kaggle-creds
Your Kaggle username: sivakona
Your Kaggle Key: ··········
Downloading pothole-detection-dataset.zip to ./pothole-detection-dataset

100%|██████████| 235M/235M [00:02<00:00, 101MB/s]
```

```
import numpy as np
import pandas as pd
import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

```
!nvidia-smi -L
```

```
GPU 0: Tesla T4 (UUID: GPU-c0e52a41-fde8-b00a-8d93-f0425a0f9095)
```

```
%%capture

!git clone https://github.com/RD191295/yolov7 # Downloading YOLOv7 repository and installing requirements
%cd yolov7
!pip3 install -qr requirements.txt
!pip3 install -q roboflow
```

```
!python3 -m venv yolov7-env
```

```
!source '/content/yolov7/yolov7-env/bin/python3'
```

```
!wget https://github.com/WongKinYiu/yolov7/releases/download/v0.1/yolov7-e6e.pt
```

```
--2023-06-26 06:16:50-- https://github.com/WongKinYiu/yolov7/releases/download/v0.1/yolov7-e6e.pt
Resolving github.com (github.com)... 140.82.113.3
Connecting to github.com (github.com)|140.82.113.3|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://objects.githubusercontent.com/github-production-release-asset-2e65be/511187726/5b2a5641-54d0-4dd0-a210-42bdc38235fa?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIAIWNJYAX4CSVEH53A%2F20230626%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20230626T061650Z&X-Amz-Expires=300&X-Amz-Signature=b0dfcf655818e785e9357fc8cf0ddf320ba02cb41aeccc81c073edf84ee08210&X-Amz-SignedHeaders=host&actor_id=0&key_id=0&repo_id=511187726&response-content-disposition=attachment%3B%20filename%3Dyolov7-e6e.pt&response-content-type=application%2Foctet-stream [following]
--2023-06-26 06:16:50-- https://objects.githubusercontent.com/github-production-release-asset-2e65be/511187726/5b2a5641-54d0-4dd0-a210-42bdc38235fa?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIAIWNJYAX4CSVEH53A%2F20230626%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20230626T061650Z&X-Amz-Expires=300&X-Amz-Signature=b0dfcf655818e785e9357fc8cf0ddf320ba02cb41aeccc81c073edf84ee08210&X-Amz-SignedHeaders=host&actor_id=0&key_id=0&repo_id=511187726&response-content-disposition=attachment%3B%20filename%3Dyolov7-e6e.pt&response-content-type=application%2Foctet-stream
Resolving objects.githubusercontent.com (objects.githubusercontent.com)...
```

```
185.199.108.133, 185.199.109.133, 185.199.110.133, …
Connecting to objects.githubusercontent.com (objects.githubusercontent.com)|185.199.108.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 304425133 (290M) [application/octet-stream]
Saving to: ‘yolov7-e6e.pt’

yolov7-e6e.pt 100%[===================>] 290.32M 130MB/s in 2.2s

2023-06-26 06:16:52 (130 MB/s) - ‘yolov7-e6e.pt’ saved [304425133/304425133]
```

```
pip install wandb
```

```
import os
import sys
import glob
import wandb
import torch
from roboflow import Roboflow


from IPython.display import Image, clear_output, display  # to display images


print(f"Setup complete. Using torch {torch.__version__} ({torch.cuda.get_device_properties(0).name if torch.cuda.is_available() else 'CPU'})")
```

```
Setup complete. Using torch 2.0.1+cu118 (Tesla T4)
```

```
try:
    user_secrets = UserSecretsClient()
    wandb_api_key = user_secrets.get_secret("wandb_api")
    wandb.login(key=wandb_api_key)
    anonymous = None
except:
    wandb.login(anonymous='must')
    print('To use your W&B account,\nGo to Add-ons -> Secrets and provide your W&B access token. Use the Label name as WANDB. \nGet your W&B access token from here: https://wandb.ai/authorize')



wandb.init(project="yolov7-R",name=f"run1")
```

```
wandb: Appending key for api.wandb.ai to your netrc file: /root/.netrc
wandb: Currently logged in as: anony-mouse-460138020999250252. Use `wandb login --relogin` to force relogin

To use your W&B account,
Go to Add-ons -> Secrets and provide your W&B access token. Use the Label name as WANDB.
Get your W&B access token from here: https://wandb.ai/authorize

Tracking run with wandb version 0.15.4
Run data is saved locally in /content/yolov7/wandb/run-20230626_061742-cafni5yk
Syncing run run1 to Weights & Biases (docs)

View project at https://wandb.ai/anony-mouse-460138020999250252/yolov7-R?apiKey=793323e0b6930e6429913277df7cd1b49b6ec284
View run at https://wandb.ai/anony-mouse-460138020999250252/yolov7-R/runs/cafni5yk?apiKey=793323e0b6930e6429913277df7cd1b49b6ec284
Do NOT share these links with anyone. They can be used to claim your runs.
```

Display W&B run

```
!ls
```

```
cfg         figure          paper                   tools          yolov7-e6e.pt
data        hubconf.py      README.md               train_aux.py   yolov7-env
deploy      inference       requirements.txt        train.py
Detect.py   LICENSE.md      scripts                 utils
export.py   models          test.py                 wandb
```

```
%cd ..
```

```
/content
```

```
!cp ../content/pothole-detection-dataset/data.yaml
!cp -R ../content/pothole-detection-dataset
```

```
cp: missing destination file operand after '../content/pothole-detection-dataset/data.yaml'
Try 'cp --help' for more information.
cp: missing destination file operand after '../content/pothole-detection-dataset'
 Try 'cp --help' for more information.
```

```
config_file_template = '''
train: ./pothole-detection-dataset/train/images
val: ./pothole-detection-dataset/valid/images


nc: 1
names: ['pothole']
'''
```

```
with open('data.yaml', 'w') as f:
    f.write(config_file_template)
```

```
!python yolov7/train_aux.py --batch 8 --cfg cfg/training/yolov7-e6e.yaml --epochs 20 --data ./data.yaml --weights 'yolov7/yolov7-e6e.pt' --device 0 --entity 'yolov7-R' --project 'yolov7-R' --name 'run1'
```

# CHAPTER 5

# TESTING

## 5.1 INTRODUCTION:

Welcome to our road damage detection project, where we harness the power of technology to improve road infrastructure and ensure safer journeys for all. Our project aims to address the critical issue of deteriorating road conditions by developing an intelligent system that can identify and classify various types of road damages. By leveraging cutting-edge computer vision algorithms and machine learning techniques, we strive to create a solution that can accurately detect and assess road defects such as potholes, cracks, and other structural abnormalities. Through this innovative approach, we hope to enable timely repairs and maintenance, ultimately enhancing the overall quality and safety of our roads. Join us on this exciting journey as we pave the way for smarter, more efficient road management and a smoother travel experience for everyone.

**Here is an outline of our approach:**

**1. Requirements Analysis:** Thoroughly understand the project requirements and specifications related to road damage detection. Identify the key functionalities, performance criteria, and expected outputs.

**2. Test Planning:** Develop a comprehensive test plan that outlines the testing objectives, scope, and strategies. Define the test environments, tools, and resources required. Establish testing timelines and milestones.

**3. Test Design:** Design test cases and scenarios based on the identified requirements. Consider various types of road damages, environmental conditions, and potential edge cases. Create a test data set that covers a wide range of scenarios.

**4. Test Execution:** Execute the test cases systematically. Utilize both manual and automated testing techniques. Monitor the system's behavior and performance during the testing process. Record and analyze the results.

**5. Defect Reporting and Tracking:** Document any issues or defects discovered during testing. Provide clear and detailed reports with steps to reproduce the problems. Track the defects throughout their lifecycle until resolution.

**6. Regression Testing:** Perform regression testing after fixing the reported defects to ensure that new changes or fixes do not introduce new issues or impact existing functionalities negatively.

**7. Performance Testing:** Assess the performance of the road damage detection system under different loads and stress conditions. Measure response times, resource utilization, and system stability.

**8. Usability Testing:** Evaluate the user-friendliness of the software interface. Gather feedback from potential end-users and incorporate necessary improvements to enhance the overall user experience.

**9. Integration Testing:** Verify the integration of the road damage detection system with other components or modules. Ensure seamless communication and interoperability between different parts of the software.

**10. Validation and Acceptance Testing**: Collaborate with stakeholders to validate the system's compliance with their requirements and expectations. Obtain their approval for the software's release and deployment.

**11. Continuous Monitoring**: Establish a mechanism for continuous monitoring and maintenance of the road damage detection system once it is deployed. Implement appropriate logging, error handling, and performance monitoring strategies.

## 5.2 Test Cases:

| S.no | Name of Scenario | Test case | Output | Description |
|---|---|---|---|---|
| 01 | Import dataset | import required dataset into colab | Accepted | Dataset must be organized and have annotation or labels and separate folders for training,testing and validation |
| 02 | Read dataset | Read the dataset | Accepted | Read the imported data using Pandas Library |
| 03 | Training and testing the dataset | Divide dataset into train and test data | Accepted | Train and test data are needed for the model |
| 04 | Testing the methods | Testing model using yolo v7 | Accepted | Expected output is obtained |
| 05 | Graph presentaion | Given correct accuracy values | Accepted | Expected graphhs obtained with accuracy values |

Table 5.2 Test Cases

# CHAPTER 6

# RESULTS AND DISCUSSIONS

## 6.1 Data Collection and Dataset Preparation:

Images can be downloaded from the Internet using the keyword road damage. The dataset is available from GitHub and dataset we have downloaded and applied it for our project. In that dataset, all the images are classified into different groups. From the dataset we collected 19,000 road images. Then, these images with road damage were visually confirmed and classified into eight classes; out of these, 9,053 images were annotated and released as a testing dataset. We trained and evaluated the damage detection model using our dataset. After testing stage, some of the images will be detected and divided into eight classes and recognize the road damage. The following Table 4.1 represents the type of damages that the model detects.

**Table 6.1. Types of damages**

| Damage type | | | Details | Class name |
|---|---|---|---|---|
| Crack | Linear Crack | Longitudinal | Wheel mark part | D00 |
| | | | Construction joint part | D01 |
| | | Lateral | Equal interval | D10 |
| | | | Construction joint part | D11 |
| | Alligator Crack | | Partial pavement, overall pavement | D20 |
| Other corruption | | | Rutting,bump,pothole,separaion | D40 |
| | | | Cross walk blur | D43 |
| | | | White line blur | D44 |

After running the “xml2Yolo.py’’ file in the system, 9053 images are detected as some damage types listed as some classes as shown in below figure.

After completing the training of the taken dataset, the detection of road damage is shown in the following


(a)

D01


(b)

D10

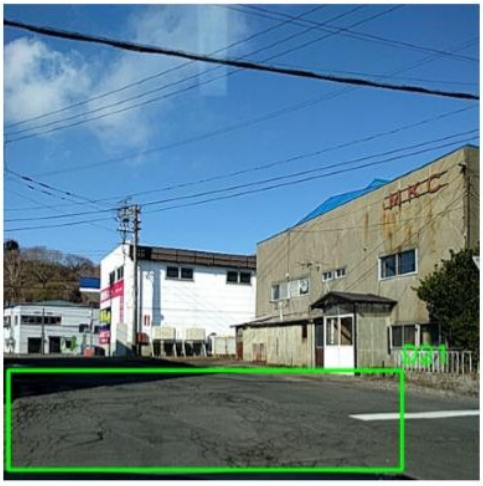

(c)

D11



(d)

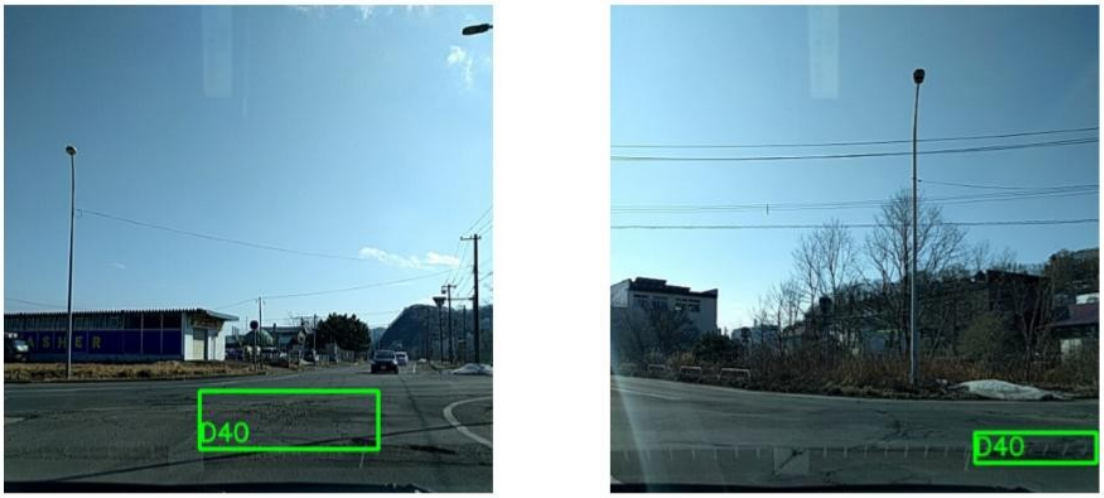
D20




(e)

D40

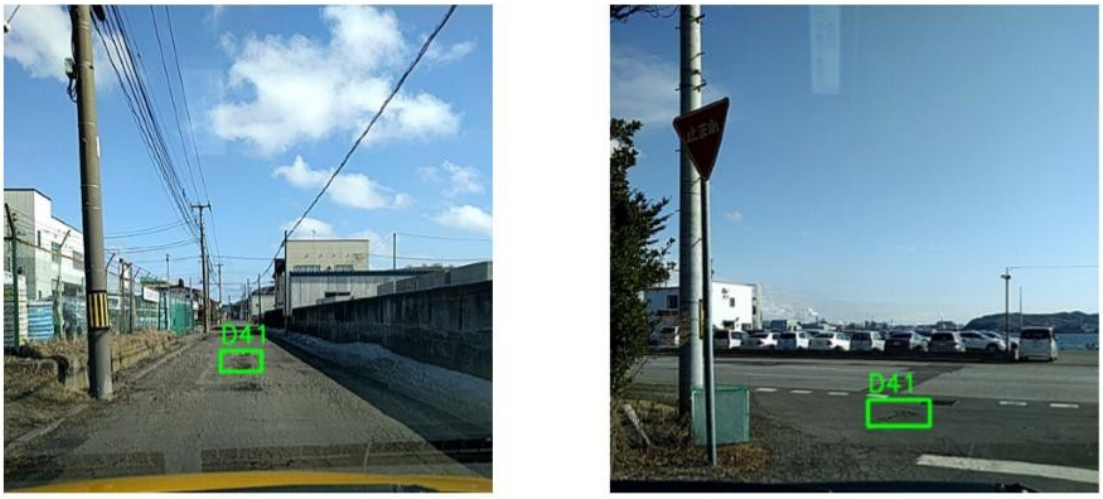
Bump




(f)

D40

Potholes




(g)

# D40

Separation



(h)

# D43



(i)


(j)

**Figure 6.1.** Sample images of our dataset: (a) to (j) correspond to each one of the eight categories, and (i) shows the legend. Our benchmark contains 163,664 road images and of these, 9,053 images include cracks. A total of 9,053 images were annotated with class labels and bounding boxes. The images were captured using a smartphone camera in realistic scenarios.

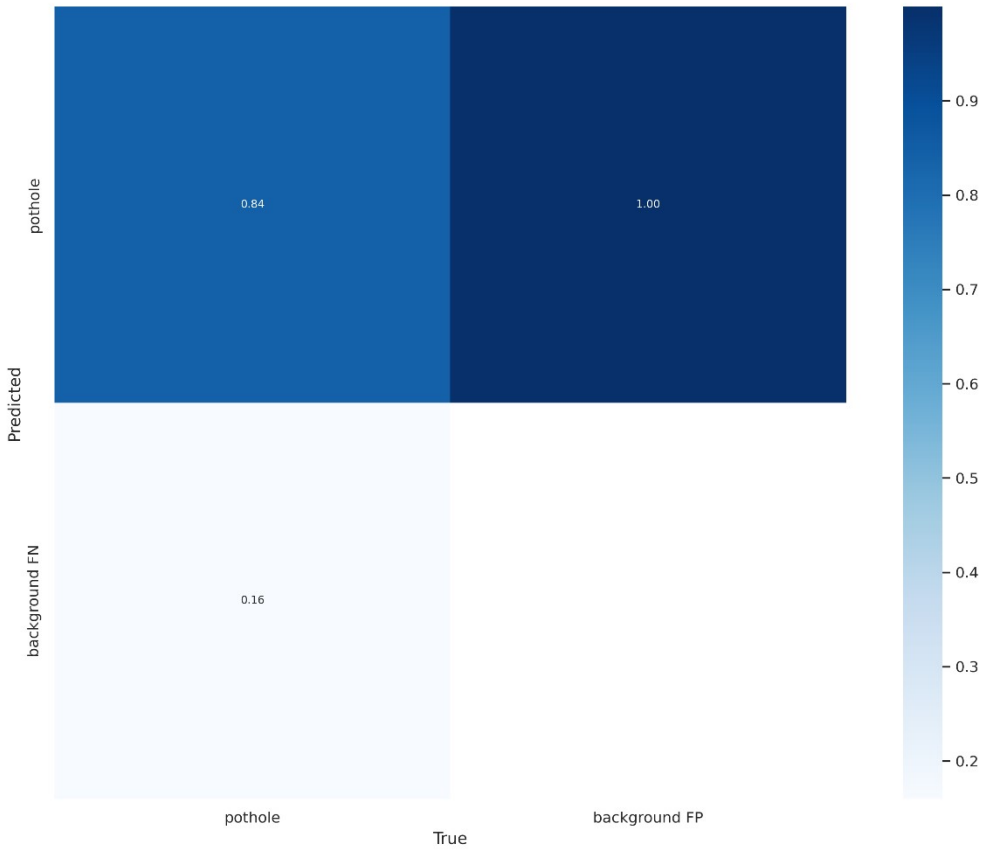



Fig 6.2 Confusion matrix

## RESULTS


Fig 6.3 results

# CHAPTER 7

# CONCLUSION & FUTURE SCOPE

In this study, we developed a new large-scale dataset for road damage detection and classification. From this dataset we collected 163,664 road images. Then, these images with road damage were visually confirmed and classified into eight classes; out of these, 9,053 images were annotated and released as a training dataset. To the best of our knowledge, this dataset is the first one for road damage detection. We strongly believe this dataset provides a new avenue for road damage detection. In addition, we trained and evaluated the damage detection model using our dataset. Based on the results, in the best-detectable category, we achieved recalls and precisions greater than 75% with an inference time of 1.5 s on a smartphone. We believe that a simple road inspection method using only a smartphone will be useful in regions where experts and financial resources are lacking. To support research in this field, we have made the dataset, trained models, source code, and smartphone application publicly available. In the future, we plan to develop methods that can detect rare types of damage that are uncommon in our dataset. The project can be linked to a government firm, cameras can be installed on busy roads and a time to time check on damage detection can be performed. In this way a lot of money can be saved by detecting damage as soon as possible. By enabling the model to detect damage in video, a device can be installed in vehicle which will capture video in real time and beware driver about cracks.

# BIBLIOGRAPHY

[1] AAoSHaT, O. (2008). Bridging the gap–restoring and rebuilding the nations bridges. Washington (DC): American Association of State Highway and Transportation Officials.

[2] Adeli, H. (2001). Neural networks in civil engineering: 1989–2000. ComputerAided Civil and Infrastructure Engineering, 16(2):126–142

[3] Akarsu, B., KARAKOSE, M., PARLAK, K., Erhan, A., ¨ and SARIMADEN, A. (2016). A fast and adaptive road defect detection approach using computer vision with real time implementation.

[4] Cha, Y.-J., Choi, W., and Buy¨ uk¨ ozt ¨ urk, O. (2017). Deep ¨ learning-based crack damage detection using convolutional neural networks. Computer-Aided Civil and Infrastructure Engineering, 32(5):361–378.

[5] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on, pages 248–255. IEEE.

[6] Everingham, M., Van Gool, L., Williams, C. K., Winn, J., and Zisserman, A. (2010). The pascal visual object classes (voc) challenge. International journal of computer vision, 88(2):303–338.

[7] Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., and Adam, H. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861.

[8] Zhang, L., Yang, F., Zhang, Y. D., and Zhu, Y. J. (2016). Road crack detection using deep convolutional neural network. In Image Processing (ICIP), 2016 IEEE International Conference on, pages 3708–3712. IEEE.

[9] Lixia Gong, Institute of Crustal Dynamics, China Earthquake Administration, Beijing 100085, China. Road damage detection from high-resolution RS image, 2012 IEEE International Geoscience and Remote Sensing Symposium.

[10] Road damage detection and classification with faster CNN, Wenzhe Wang ; Bin Wu ; Sixiong Yang ; Zhixiang Wang, 2018 IEEE International Conference on Big Data (Big Data)

[11] Detection of road surface damage using mobile robot equipped with 2D laser scanner, Proceedings of the 2013 IEEE/SICE International Symposium on System Integration.

[12] B. Douillard, et al. "Classification and semantic mapping of urban environments." The International Journal of Robotics Research 30.1 (2011): 5-32.

[13] C. Mertz. "Continuous Road Damage Detection Using Regular Service Vehicles." 18th World Congress on Intelligent Transport Systems. 2011.

[14] N. Tanaka, and K. Uematsu. "A crack detection method in road surface images using morphology." Workshop on Machine Vision Applications. 1998.

[15] Li Guangyao, Hu Yang. Road Feature Extraction from High Resolution Remote Sensing Images: Review and Prospects. Remote Sensing Information, 2008(1): 91-95.