Edge-Ready Road Damage Detection Using an Enhanced YOLO with Hyperparameter Tuning

Author 1: Baddam Soumya
Email: baddamsoumya0710@gmail.com

Author 2: B. Hruday Vikas
Email: hrudayvikasyadav008@gmail.com

Author 2: C. Sai Kumar
Email: chatlasaikumar5@gmail.com

Guide: Mr.Sreerama Sreekanth

Organization: Guru Nanak Institutions, India

ABSTRACT:

Efficient and accurate road damage detection is critical for the development of smart cities and maintaining safe transportation infrastructure. Manual inspection methods are slow, labor-intensive, and prone to errors, making automated detection necessary. In this project, we propose a YOLOv10n-based framework for real-time road damage detection optimized for edge devices. The system leverages advanced hyperparameter tuning to improve model performance while maintaining low computational requirements. The framework achieves high accuracy, with a precision of 0.986, recall of 0.973, mean average precision (mAP@0.5) of 0.988, and F1-score of 0.978. Deployment on NVIDIA Jetson Nano demonstrates an inference time of 0.13 seconds per frame at 7.5 FPS, while NVIDIA AGX Orin achieves 0.014 seconds per frame at 67 FPS, highlighting scalability and efficiency. This study demonstrates that YOLOv10n is highly effective for real-time, edge-based road damage recognition, enabling faster maintenance decisions and improved road safety.

keywords: Road damage detection, YOLOv10n, hyperparameter tuning, NVIDIA Jetson nano

CHAPTER-1

INTRODUCTION

Road infrastructure plays a vital role in ensuring the safety, efficiency, and sustainability of modern transportation systems. Well-maintained roads not only reduce the likelihood of accidents but also enhance driving comfort and decrease vehicle operating costs. However, road damage such as cracks, potholes, and surface wear is inevitable due to weather conditions, heavy traffic, and natural aging of materials. If not identified and repaired in time, such damages can lead to further deterioration, higher maintenance costs, and potential hazards for road users. Therefore, timely and accurate road damage detection is an essential component in the development of smart cities and intelligent transportation systems.

Traditional road inspection methods rely heavily on manual surveys conducted by experts or local authorities. While effective on a small scale, these methods are labor-intensive, costly, and prone to subjectivity, making them unsuitable

for large-scale urban environments. Recent advancements in computer vision and artificial intelligence have enabled the development of automated road monitoring systems. Several deep learning-based approaches have been proposed for damage detection; however, most of them require high computational resources, making them impractical for real-time deployment on resource-constrained edge devices. This gap highlights the need for a lightweight yet accurate framework that can balance detection performance with real-time efficiency.

To address these challenges, this study proposes a YOLOv10n-based road damage detection framework optimized for real-time applications on edge devices such as NVIDIA Jetson Nano and AGX Orin. The system leverages hyperparameter tuning and model optimization techniques to achieve high accuracy while maintaining low computational complexity. Extensive evaluation shows that the proposed model achieves a precision of 0.986, recall of 0.973, mAP@0.5 of 0.988, and an F1-score of 0.978. Furthermore, the deployment results demonstrate promising inference speeds of 7.5 FPS on Jetson Nano and 67 FPS on AGX Orin, proving the scalability and robustness of the approach. This work contributes toward the advancement of smart road maintenance systems, enabling faster decision-making and improved road safety.

1.2 SCOPE OF THE PROJECT

The scope of this project is centered on the development and deployment of an efficient, lightweight, and accurate road damage detection system that can operate in real-time on edge devices. The proposed YOLOv10n-based framework is designed to detect various types of road surface damages such as cracks, potholes, and surface wear, thereby enabling authorities to monitor road conditions more effectively. The project focuses on achieving a balance between detection accuracy and computational efficiency, making it suitable for large-scale deployment in smart city infrastructures. By validating the system on devices like NVIDIA Jetson Nano and AGX Orin, the project demonstrates its adaptability to both low-power and high-performance environments. The scope also extends to supporting proactive maintenance planning, reducing the reliance on manual inspection methods, and ensuring safer transportation systems. However, the project is limited to visual damage detection and does not address underground structural issues or predictive maintenance, which can be considered in future research.

1.3 OBJECTIVE

The primary objective of this project is to design and implement an automated road damage detection system using the YOLOv10n model, which is lightweight yet powerful, making it highly suitable for edge-based real-time applications. The goal is to leverage the efficiency of YOLOv10n along with advanced hyperparameter tuning to achieve high detection accuracy while keeping computational costs low. Specifically, the system aims to identify different categories of road damages such as cracks, potholes, and surface deterioration with precision and reliability. Another key objective is to evaluate the performance of the model on edge devices like NVIDIA Jetson Nano and NVIDIA AGX Orin, thereby ensuring scalability across hardware with varying resource capacities. By achieving these objectives, the project seeks to provide a cost-effective, scalable, and practical solution for road monitoring that enhances smart city infrastructure, supports proactive maintenance planning, and ultimately improves road safety.

1.4 EXISTING SYSTEM:

The existing system for automated road damage detection widely uses YOLOv7, a state-of-the-art object detection model that brought significant improvements in accuracy and inference speed compared to earlier versions such as YOLOv5. YOLOv7 introduced advanced training strategies, optimized anchor-free mechanisms, and improved architectural designs, making it highly effective for detecting road damages like cracks, potholes, and surface wear.

In road monitoring applications, YOLOv7 processes images or video frames captured from roadside cameras, vehicles, or drones, and predicts bounding boxes around damaged regions. The model's grid-based detection mechanism and optimized loss functions allow it to perform well on large-scale datasets while ensuring accurate localization of defects.

Although YOLOv7 has been successful in controlled environments and high-end computing platforms, its heavy architecture and resource demands pose significant challenges for deployment on real-time, resource-constrained

devices such as edge platforms. These limitations restrict the scalability of YOLOv7 for smart city and large-scale road monitoring applications.

1.4.1 EXISTINGSYSTEM DISADVANTAGES:

High computational requirements, making YOLOv7 unsuitable for low-power edge devices like NVIDIA Jetson Nano.

Slower inference speed on resource-constrained hardware, which prevents real-time performance in practical road monitoring scenarios.

Longer training time and higher complexity, requiring more computational resources and fine-tuning efforts.

Trade-off between model size and performance — lightweight YOLOv7 variants lose accuracy, especially when detecting small damages such as fine cracks.

Limited adaptability for large-scale deployments in smart cities, as continuous monitoring would require more efficient and lightweight models.

1.5 LITERATURE SURVEY

Title: Automatic Defect Detection of Pavement Diseases

Authors: L. Zhao, Y. Wu, X. Luo, Y. Yuan

Year: 2022

DESCRIPTION: Pavement disease detection is an important task for ensuring road safety. Manual visual detection requires a significant amount of time and effort. Therefore, an automated road disease identification technique is required to guarantee that city tasks are performed. However, due to the irregular shape and large-scale differences in road diseases, as well as the imbalance between the foreground and background, the task is challenging. Because of this, we created the deep convolution neural network—DASNet, which can be used to identify road diseases automatically. The network employs deformable convolution instead of regular convolution as the feature pyramid's input, adds the same supervision signal to the multi-scale features before feature fusion, decreases the semantic difference, extracts context information by residual feature enhancement, and reduces the information loss of the pyramid's top-level feature map. Considering the unique shape of road diseases, imbalance problems between the foreground and background are common, therefore, we introduce the sample weighted loss function. In order to prove the superiority and effectiveness of this method, it is compared to the latest method. A large number of experiments show that this method is superior in accuracy to other methods, specifically, under the COCO evaluation metric, compared with the Faster RCNN baseline, the proposed

Title: Real-Time Road Defect Monitoring from UAV Visual Data Sources

Author: I. Katsamenis, N. Bakalos, E. Protopapadakis, E. E. Karolou, G. Kopsiaftis, A. Voulodimos

Year: 2023

Description: The use of UAVs and artificial intelligence has emerged as a promising approach for monitoring road defects. This paper highlights the importance of these technologies in improving road inspection, maintenance, and safety. Traditional methods for inspecting roads are often time-consuming, expensive, and can put human inspectors in dangerous situations. However, drones equipped with high-resolution cameras and sensors can capture pavement image data quickly and safely. Deep learning algorithms can then analyze this data to identify and localize areas in need of repair. By leveraging these technologies, engineers and road construction experts can more efficiently monitor and maintain roads, reducing the costs associated with repairs and maintenance, while in parallel improving safety. To this end, this work emphasizes the potential of UAVs in conjunction with deep learning techniques to provide a more

comprehensive view of road conditions, allowing for targeted repairs and more effective maintenance strategies, such as prefabrication and robotic interventions. Experimental results using objective evaluation criteria, such as precision, recall, F1-score, and IoU are promising, which entails that this study advocates for the adoption of these technologies to enhance the monitoring and maintenance of road infrastructures.

Title: Road Damage Detection and Classification with Deep Learning Using Smartphone Sensor Data

Author: A. Basak, P. Kar, B. Kole, N. Dey

Year: 2023.

Description: Research on damage detection of road surfaces using image processing techniques has been actively conducted. This study makes three contributions to address road damage detection issues. First, to the best of our knowledge, for the first time, a large-scale road damage data set is prepared, comprising 9,053 road damage images captured using a smartphone installed on a car, with 15,435 instances of road surface damage included in these road images. Next, we used state-of-the-art object detection methods using convolutional neural networks to train the damage detection model with our data set, and compared the accuracy and runtime speed on both, using a GPU server and a smartphone. Finally, we demonstrate that the type of damage can be classified into eight types with high accuracy by applying the proposed object detection method. The road damage data set, our experimental results, and the developed smartphone application used in this study are publicly

Title: Digital Twin: Challenge Road Damage Detection on Edge Device

Author: H. Mahmudah, A. Musyafa, A. S. Aisjah, S. Arifin, C. A. Prastyanto

Year: 2024

Description: High efficiency and safety are critical factors in ensuring the optimal performance and reliability of systems and equipment across various industries. Fault monitoring (FM) techniques play a pivotal role in this regard by continuously monitoring system performance and identifying the presence of faults or abnormalities. However, traditional FM methods face limitations in fully capturing the complex interactions within a system and providing real-time monitoring capabilities. To overcome these challenges, Digital Twin (DT) technology has emerged as a promising solution to enhance existing FM practices. By creating a virtual replica or digital copy of a physical equipment or system, DT offers the potential to revolutionize fault monitoring approaches. This paper aims to explore and discuss the diverse range of predictive methods utilized in DT and their implementations in FM across industries. Furthermore, it will showcase successful implementations of DT in FM across a wide array of industries, including manufacturing, energy, transportation, and healthcare. The utilization of DT in FM enables a comprehensive understanding of system behavior and performance by leveraging real-time data, advanced analytics, and machine learning algorithms. By integrating physical and virtual components, DT facilitates the monitoring and prediction of faults, providing valuable insights into the system's health and enabling proactive maintenance and decision making.

Title: YOLOv7: Trainable Bag-of-Freebies Sets New State-of-the-Art for Real-Time Object Detectors

Author: C. Wang, A. Bochkovskiy, H.-Y. Mark Liao

Year: 2023.

Description: YOLOv7 surpasses all known object detectors in both speed and accuracy in the range from 5 FPS to 160 FPS and has the highest accuracy 56.8% AP among all known real-time object detectors with 30 FPS or higher on GPU V100. YOLOv7-E6 object detector (56 FPS V100, 55.9% AP) outperforms both transformer-based detector SWIN-L Cascade-Mask R-CNN (9.2 FPS A100, 53.9% AP) by 509% in speed and 2% in accuracy, and convolutional-based detector ConvNeXt-XL Cascade-Mask R-CNN (8.6 FPS A100, 55.2% AP) by 551% in speed and 0.7% AP in accuracy,

as well as YOLOv7 outperforms: YOLOR, YOLOX, Scaled-YOLOv4, YOLOv5, DETR, Deformable DETR, DINO-5scale-R50, ViT-Adapter-B and many other object detectors in speed and accuracy. Moreover, we train YOLOv7 only on MS COCO dataset from scratch without using any other datasets or pre-trained weights. Source code is released in https://github.com/WongKinYiu/yolov7.

1.6 PROPOSED SYSTEM

The proposed system introduces a YOLOv10n-based framework for real-time road damage detection, specifically optimized for deployment on edge devices. YOLOv10n is a lightweight yet powerful object detection model that offers improved efficiency compared to earlier versions like YOLOv7, while maintaining high detection accuracy. By incorporating hyperparameter tuning and optimization techniques, the system achieves a balance between computational efficiency and detection precision, making it well-suited for continuous road monitoring in smart city applications.

In this framework, YOLOv10n processes road images or live video feeds to identify and localize damages such as potholes, cracks, and surface wear. The model's lightweight architecture allows faster inference with reduced hardware requirements, enabling smooth deployment on devices like NVIDIA Jetson Nano and NVIDIA AGX Orin. The system achieves remarkable results with a precision of 0.986, recall of 0.973, mean average precision (mAP@0.5) of 0.988, and F1-score of 0.978, demonstrating both reliability and scalability.

Unlike traditional models or heavy architectures such as YOLOv7, the proposed system emphasizes real-time performance with low computational cost. This allows road authorities and smart city infrastructure to conduct proactive maintenance, minimize manual inspection efforts, and ensure safer road conditions for transportation systems.

1.6.1 PROPOSED SYSTEM ADVANTAGES:

Lightweight architecture of YOLOv10n ensures faster inference and lower hardware resource consumption.

Highly accurate detection performance with precision of 0.986, recall of 0.973, and mAP@0.5 of 0.988.

Supports real-time deployment on edge devices such as Jetson Nano (7.5 FPS) and AGX Orin (67 FPS).

Reduced training complexity compared to heavier models, while still maintaining high accuracy.

Scalable for large-scale road monitoring in smart cities, enabling proactive maintenance and improved road safety.

CHAPTER 2

PROJECT DESCRIPTION

2.1 GENERAL:

The project focuses on developing a lightweight and efficient framework for real-time road damage detection using the YOLOv10n model. Road damage such as potholes, cracks, and surface wear pose significant risks to traffic safety and increase vehicle maintenance costs. Traditional inspection methods, which rely heavily on manual surveys, are time-consuming, costly, and prone to human error, making them unsuitable for large-scale monitoring in smart city environments. To overcome these challenges, the project leverages deep learning and computer vision techniques to automate the detection process and provide faster, more reliable road condition assessments.

The core of the system is based on YOLOv10n, an advanced object detection model that offers a strong balance between computational efficiency and detection accuracy. Unlike heavier models such as YOLOv7, YOLOv10n is specifically optimized for edge deployment, ensuring high performance on low-power devices. Through hyperparameter tuning and

optimization, the model achieves excellent results with a precision of 0.986, recall of 0.973, and mean average precision (mAP@0.5) of 0.988. This ensures that the framework can accurately identify different categories of road damages in real-time without requiring high-end computing infrastructure.

The system is further validated by deploying it on NVIDIA Jetson Nano and NVIDIA AGX Orin, where it demonstrates inference speeds of 7.5 FPS and 67 FPS, respectively. These results prove that the proposed framework is not only scalable but also practical for real-world applications. By enabling proactive maintenance decisions, the project contributes to reducing road hazards, improving transportation safety, and supporting the development of smarter urban infrastructures. Ultimately, this project aims to provide a cost-effective, reliable, and scalable solution for intelligent road monitoring in smart cities.

2.2 METHODOLOGIES

The proposed system for road damage detection using YOLOv10n is implemented through a modular approach. Each module is designed to handle a specific task, and together they form a complete pipeline for efficient and accurate detection. The following modules describe the methodology adopted in this project

2.2.1MODULES NAME:

- Data Collection Preprocessing
- Model Selection
- Training and Validation (YOLO-based)
- Deployment on Edge devices
- Road Damage Detection and Output Generation

2.2.2 MODULES EXPLANATION:

1. Data Collection and Preprocessing

In this module, images of roads are collected from various sources such as open datasets, roadside cameras, or drone footage. The collected data is cleaned, annotated, and augmented to improve diversity. Preprocessing steps such as resizing, normalization, and noise reduction are performed to ensure that the images are suitable for training the YOLOv10n model.

2. Model Selection (YOLOv10n Framework):

This module focuses on selecting **YOLOv10n** as the detection model. YOLOv10n is chosen because of its lightweight architecture and ability to deliver high accuracy while maintaining real-time performance. Model hyperparameters are tuned to achieve an optimal balance between precision, recall, and inference speed.

3. Training and Validation:

In this module, the YOLOv10n model is trained using the prepared dataset. Training involves feeding annotated images to the model, adjusting weights, and optimizing loss functions. A portion of the dataset is reserved for validation to monitor the model's performance and prevent overfitting.

4. Deployment on Edge Devices:

Once trained, the model is deployed on edge devices such as **NVIDIA Jetson Nano** and **NVIDIA AGX Orin**. This module ensures that the model runs efficiently in real-world conditions with limited hardware resources. Inference time and frame rate are measured to evaluate real-time performance.

5. Road Damage Detection and Output Generation:

This module represents the final stage of the system. The deployed model processes live video streams or road images to detect and classify damages like cracks and potholes. The results are visualized with bounding boxes and labels, and can be further integrated into smart city maintenance dashboards for decision-making.

2.3 TECHNIQUE USED OR ALGORITHM USED

2.3.1 EXISTING TECHNIQUE: -

The existing system for road damage detection is based on the YOLOv7 (You Only Look Once, Version 7) algorithm, which is a state-of-the-art real-time object detection model. YOLOv7 follows the principle of single-stage object detection, where both object classification and localization are performed in a single forward pass of the network, making it faster than two-stage detectors like Faster R-CNN.

YOLOv7 improves upon its predecessors by introducing extended efficient layer aggregation networks (E-ELAN), which enhance feature extraction and gradient flow, thereby improving accuracy without increasing computational cost significantly.

It incorporates anchor-free detection heads, along with anchor-based heads, enabling better detection of objects of varying sizes, including small objects such as fine cracks in road images.

The algorithm also introduces advanced training techniques like coarse-to-fine lead loss and model re-parameterization, which improve convergence, reduce overfitting, and enhance inference efficiency.

In road damage detection, YOLOv7 processes images by dividing them into grids and predicting bounding boxes and class probabilities for potential damage regions. This enables the detection of multiple types of damages, such as potholes and cracks, in a single image. Despite its strong performance on high-end hardware, YOLOv7 is computationally heavy, making it less suitable for real-time deployment on low-power edge devices.

2.3.2 PROPOSED TECHNIQUE USED OR ALGORITHM USED:

The proposed system adopts the YOLOv10n (You Only Look Once, Version 10 – Nano variant) algorithm for real-time road damage detection. YOLOv10n is the lightweight version of the YOLOv10 family, specifically designed to achieve high detection accuracy with minimal computational requirements, making it highly suitable for deployment on edge devices. Similar to other YOLO models, YOLOv10n is a single-stage object detection algorithm, meaning it performs object classification and localization simultaneously in a single forward pass, ensuring both speed and efficiency.

YOLOv10n introduces task-aligned learning and decoupled head architecture, which improves the precision of object localization and classification, especially for small-scale damages like fine cracks. The model leverages anchor-free detection mechanisms, reducing the complexity of anchor box design and improving detection of varied shapes and sizes of road damages.

Advanced loss functions such as Distribution Focal Loss (DFL) and VariFocal Loss are incorporated to enhance the balance between precision and recall, leading to more robust predictions. The “Nano” version (YOLOv10n) ensures reduced model size and lower computational complexity, allowing smooth deployment on low-power devices without compromising accuracy.

In this project, YOLOv10n achieves outstanding performance with a precision of 0.986, recall of 0.973, mean average precision (mAP@0.5) of 0.988, and F1-score of 0.978. Its lightweight design allows real-time deployment on devices like NVIDIA Jetson Nano (7.5 FPS) and NVIDIA AGX Orin (67 FPS), making it far more scalable and efficient compared to YOLOv7. Thus, YOLOv10n provides the ideal balance between detection accuracy and computational efficiency for large-scale smart city road monitoring systems.

CHAPTER 3

REQUIREMENTS ENGINEERING

3.1 GENERAL

We can see from the results that on each database, the error rates are very low due to the discriminatory power of features and the regression capabilities of classifiers. Comparing the highest accuracies (corresponding to the lowest error rates) to those of previous works, our results are very competitive.

3.2 HARDWARE REQUIREMENTS

The hardware requirements may serve as the basis for a contract for the implementation of the system and should therefore be a complete and consistent specification of the whole system. They are used by software engineers as the starting point for the system design. It should what the system do and not how it should be implemented.

PROCESSOR : DUAL CORE 2 DUOS.

RAM : 4GB DD RAM

HARD DISK : 250 GB

3.3 SOFTWARE REQUIREMENTS

The software requirements document is the specification of the system. It should include both a definition and a specification of requirements. It is a set of what the system should do rather than how it should do it. The software requirements provide a basis for creating the software requirements specification. It is useful in estimating cost, planning team activities, performing tasks and tracking the teams and tracking the team's progress throughout the development activity.

Operating System : Windows 7/8/10

Platform : Spyder3

Programming Language : Python

Front End : Spyder3

3.4 FUNCTIONAL REQUIREMENTS

A functional requirement defines a function of a software-system or its component. A function is described as a set of inputs, the behavior, Firstly, the system is the first that achieves the standard notion of semantic security for data confidentiality in attribute-based deduplication systems by resorting to the hybrid cloud architecture.

3.5 NON-FUNCTIONAL REQUIREMENTS

The major non-functional Requirements of the system are as follows

Usability

The system is designed with completely automated process hence there is no or less user intervention.

Reliability

The system is more reliable because of the qualities that are inherited from the chosen platform python. The code built by using python is more reliable.

Performance

This system is developing in the high level languages and using the advanced back-end technologies it will give response to the end user on client system with in very less time.

Supportability

The system is designed to be the cross platform supportable. The system is supported on a wide range of hardware and any software platform, which is built into the system.

Implementation

The system is implemented in web environment using Jupyter notebook software. The server is used as the intellignce server and windows 10 professional is used as the platform. Interface the user interface is based on Jupyter notebook provides server system.

CHAPTER 4

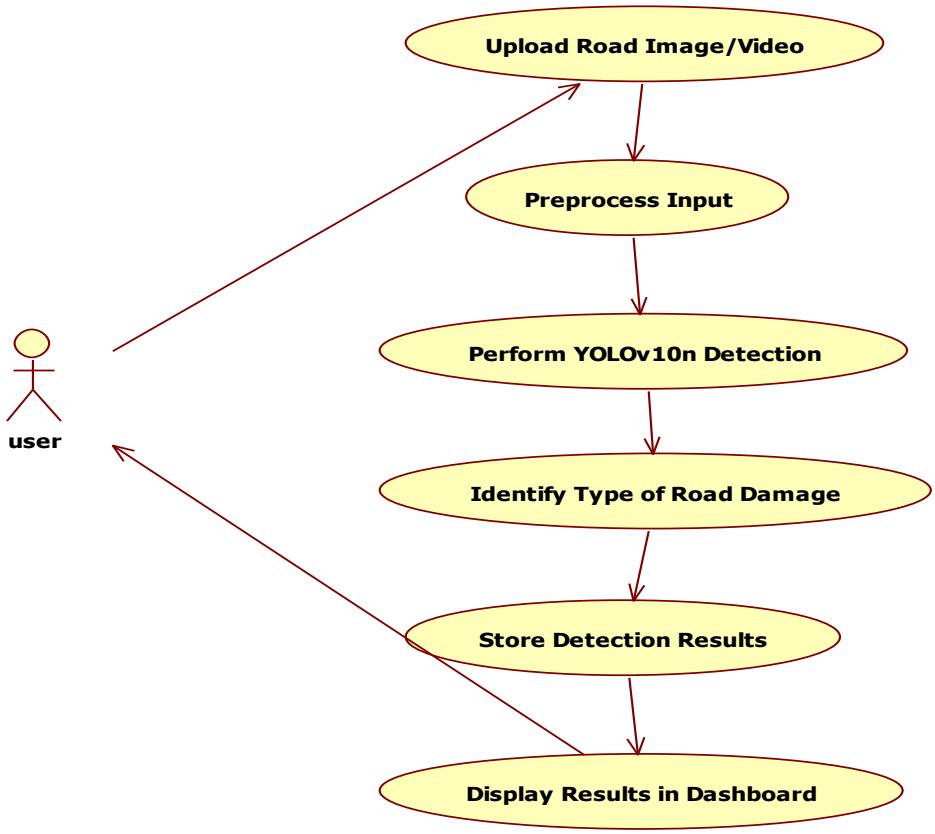
DESIGN ENGINEERING

4.1 GENERAL

Design Engineering deals with the various UML [Unified Modelling language] diagrams for the implementation of project. Design is a meaningful engineering representation of a thing that is to be built. Software design is a process through which the requirements are translated into representation of the software. Design is the place where quality is rendered in software engineering.

4.2 UML DIAGRAMS

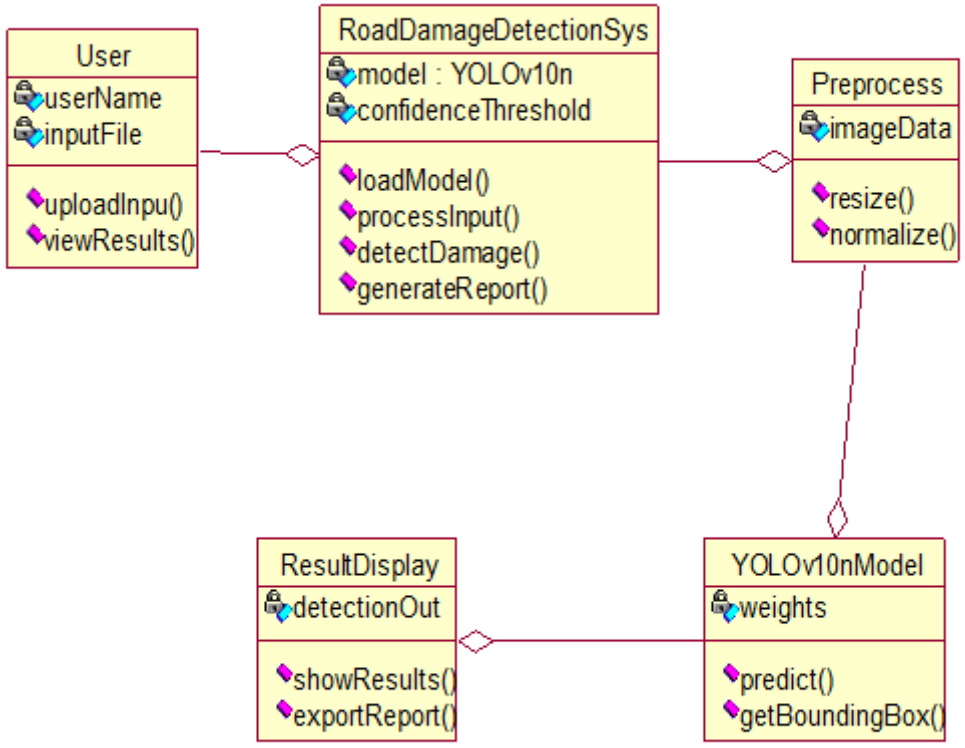
4.2.1 USE CASE DIAGRAM



EXPLANATION:

The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted. The above diagram consists of user as actor. Each will play a certain role to achieve the concept.

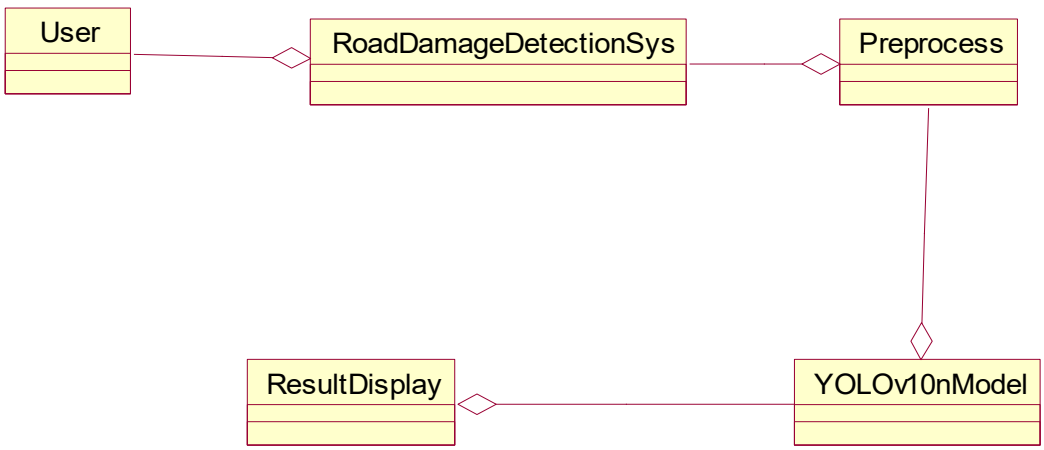
4.2.2 CLASS DIAGRAM



EXPLANATION

In this class diagram represents how the classes with attributes and methods are linked together to perform the verification with security. From the above diagram shown the various classes involved in our project.

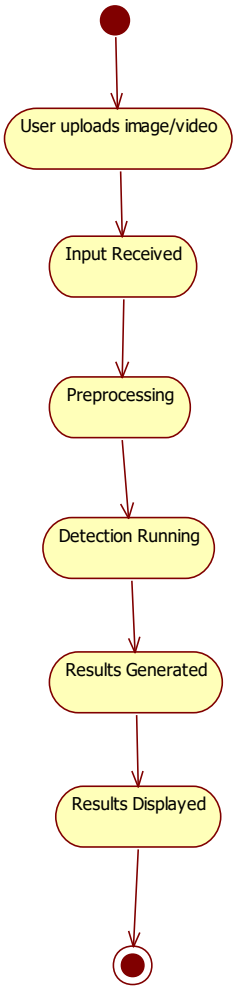
4.2.3 OBJECT DIAGRAM



EXPLANATION:

In the above digram tells about the flow of objects between the classes. It is a diagram that shows a complete or partial view of the structure of a modeled system. In this object diagram represents how the classes with attributes and methods are linked together to perform the verification with security.

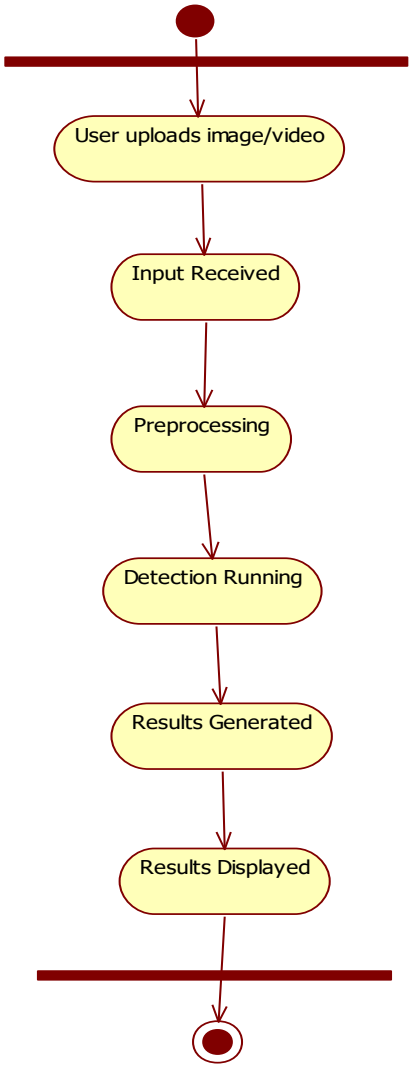
4.2.4 STATE DIAGRAM



EXPLANATION:

State diagram are a loosely defined diagram to show workflows of stepwise activities and actions, with support for choice, iteration and concurrency. State diagrams require that the system described is composed of a finite number of states; sometimes, this is indeed the case, while at other times this is a reasonable abstraction. Many forms of state diagrams exist, which differ slightly and have different semantics.

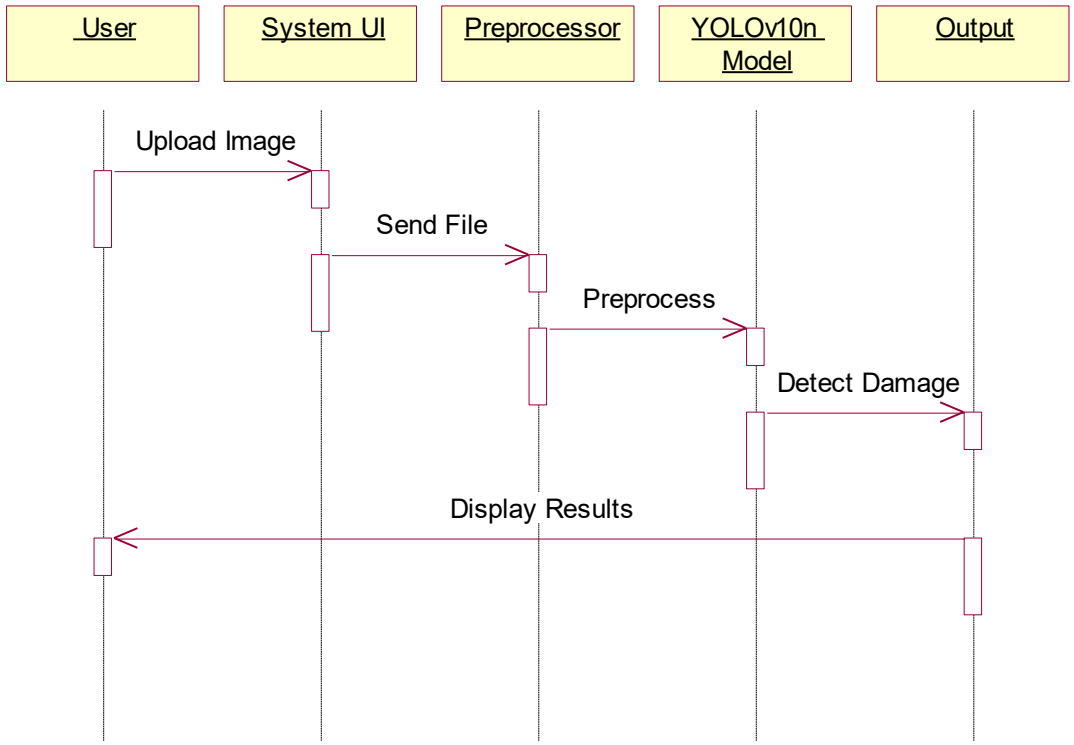
4.2.5 ACTIVITY DIAGRAM



EXPLANATION:

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

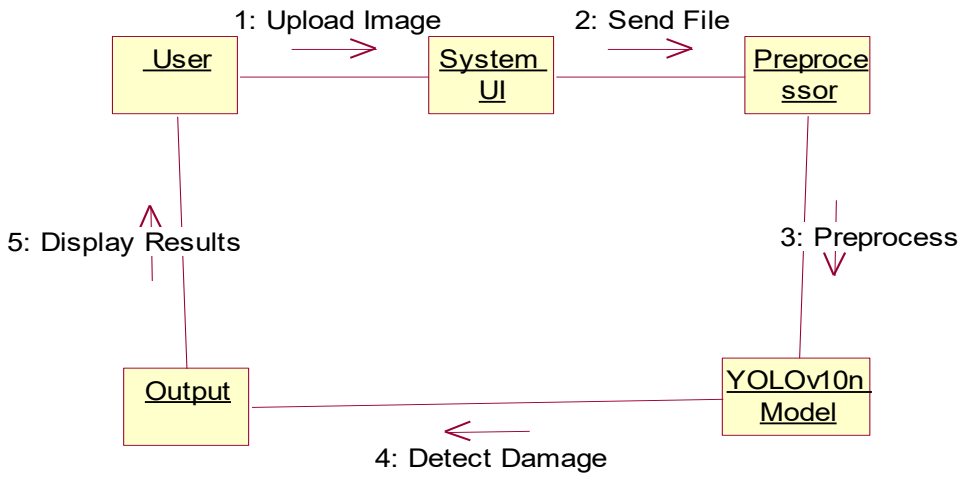
4.2.6 SEQUENCE DIAGRAM



EXPLANATION:

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario.

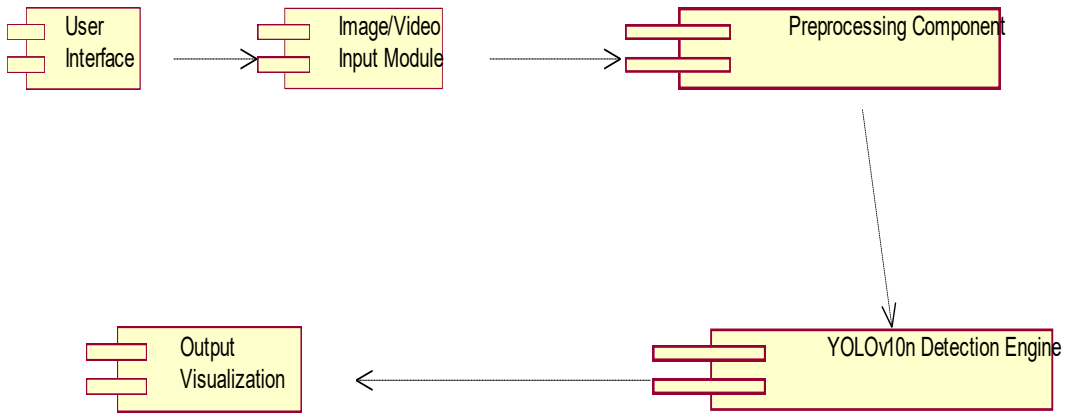
4.2.7 COLLABORATION DIAGRAM



EXPLANATION:

A collaboration diagram, also called a communication diagram or interaction diagram, is an illustration of the relationships and interactions among software objects in the Unified Modeling Language (UML). The concept is more than a decade old although it has been refined as modeling paradigms have evolved.

4.2.8 COMPONENT DIAGRAM

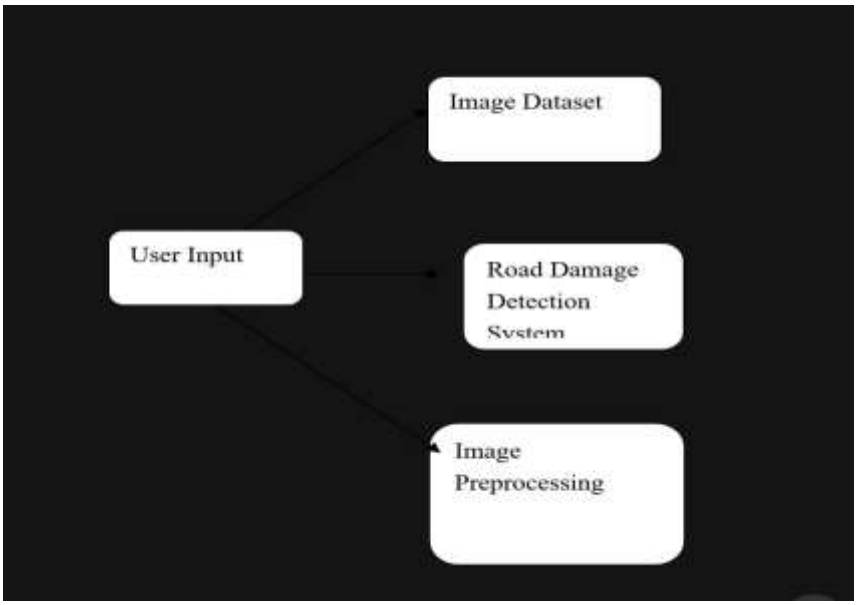


EXPLANATION

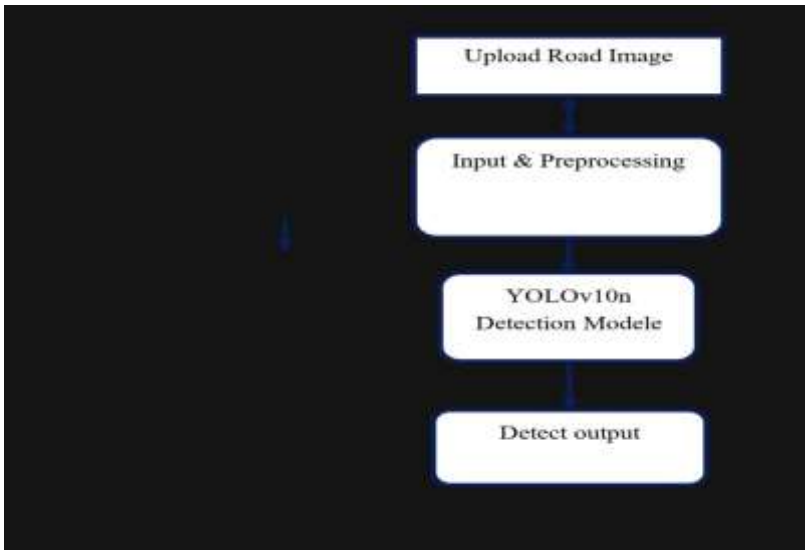
In the Unified Modeling Language, a component diagram depicts how components are wired together to form larger components and or software systems. They are used to illustrate the structure of arbitrarily complex systems. User gives main query and it converted into sub queries and sends through data dissemination to data aggregators. Results are to be showed to user by data aggregators. All boxes are components and arrow indicates dependencies.

4.2.9DATA FLOW DIAGRAM

Level 0



Level 1

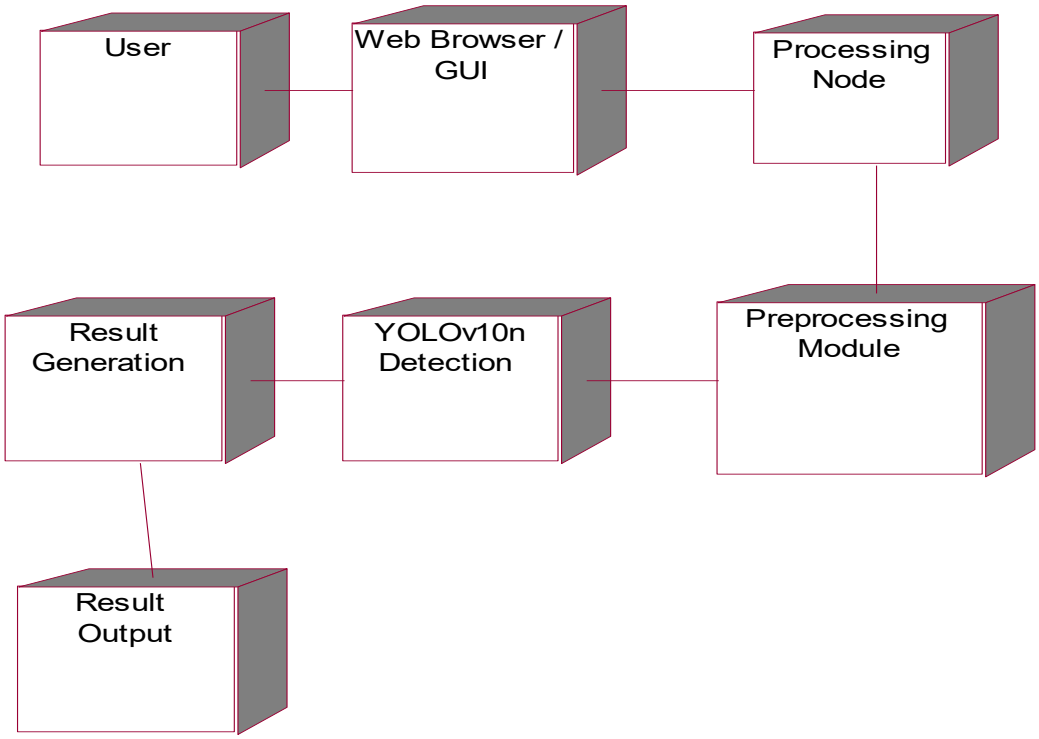


EXPLANATION:

A data flow diagram (DFD) is a graphical representation of the "flow" of data through an information system, modeling its process aspects. Often they are a preliminary step used to create an overview of the system which can later be elaborated. DFDs can also be used for the visualization of data processing (structured design).

A DFD shows what kinds of data will be input to and output from the system, where the data will come from and go to, and where the data will be stored. It does not show information about the timing of processes, or information about whether processes will operate in sequence or in parallel.

4.2.10 DEPLOYMENT DIAGRAM



EXPLANATION:

Deployment Diagram is a type of diagram that specifies the physical hardware on which the software system will execute. It also determines how the software is deployed on the underlying hardware. It maps software pieces of a system to the device that are going to execute it.

SYSTEM ARCHITECTURE:

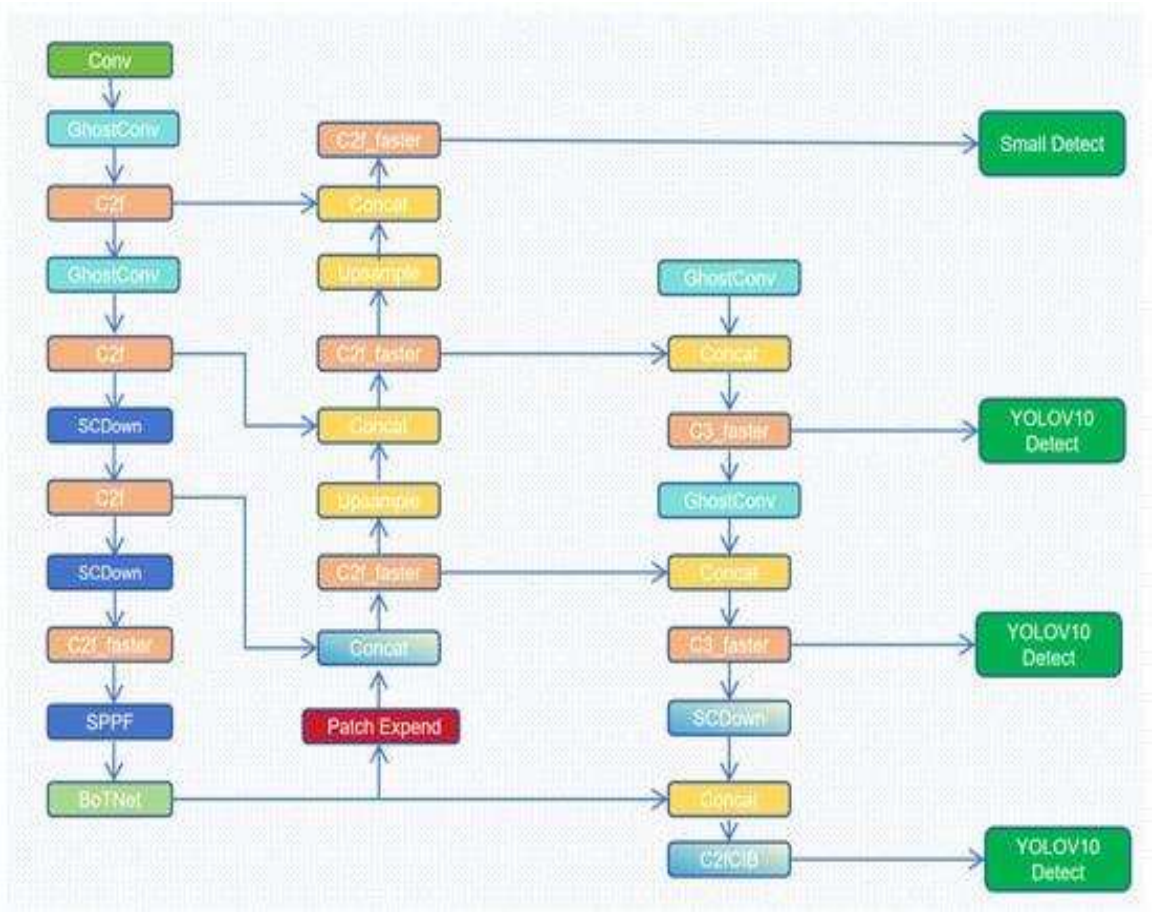


Fig 4.11: System Architecture

CHAPTER 5

DEVELOPMENT TOOLS

5.1 Python

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

5.2 History of Python

Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands.

Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, SmallTalk, and Unix shell and other scripting languages.

Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL).

Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

5.3 Importance of Python

- **Python is Interpreted** − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- **Python is Interactive** − You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
- **Python is Object-Oriented** − Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
- **Python is a Beginner's Language** − Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

5.4 Features of Python

- **Easy-to-learn** − Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
- **Easy-to-read** − Python code is more clearly defined and visible to the eyes.
- **Easy-to-maintain** − Python's source code is fairly easy-to-maintain.
- A broad standard library − Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
- **Interactive Mode** − Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.
- **Portable** − Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
- **Extendable** − You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
- **Databases** − Python provides interfaces to all major commercial databases.
- **GUI Programming** − Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.
- **Scalable** − Python provides a better structure and support for large programs than shell scripting.

Spyder3

- Spyder3 is an open-source Python IDE (Integrated Development Environment) designed mainly for scientists, engineers, and data analysts who work with Python. It provides a complete environment to write, run, debug, and analyze Python programs.
- You can imagine it as a digital laboratory for Python experiments 🧪🐍, where code, output, and data live together on one screen.

Key Features:

- **Code editor**: write python programs easily and Supports syntax highlighting and auto-completion.
- **IPython Consloe**: Allowes you to run Python commands interactively and see results instantly.
- **Variable Explorer:** Displays variables created in the program and helps in analyzing data and debugging code.
- **Debugger:** Helps find and fix errors step by step in a program
- **Integration with scientific Libraries:** works well with libraries like Numpy, Pandas, Matplotlib, and SciPy

.

CHAPTER 6

SOFTWARE TESTING

6.1 GENERAL

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

6.2 DEVELOPING METHODOLOGIES

The test process is initiated by developing a comprehensive plan to test the general functionality and special features on a variety of platform combinations. Strict quality control procedures are used. The process verifies that the application meets the requirements specified in the system requirements document and is bug free. The following are the considerations used to develop the framework from developing the testing methodologies

6.3Types of Tests

6.3.1 Unit testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program input produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

6.3.2 Functional test

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/Procedures: interfacing systems or procedures must be invoked.

6.3.3 System Test

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

6.3.4 Performance Test

The Performance test ensures that the output be produced within the time limits,and the time taken by the system for compiling, giving response to the users and request being send to the system for to retrieve the results.

6.3.5 Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

6.3.6 Acceptance Testing

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

Acceptance testing for Data Synchronization:

The Acknowledgements will be received by the Sender Node after the Packets are received by the Destination Node

The Route add operation is done only when there is a Route request in need

The Status of Nodes information is done automatically in the Cache Updation process

6.2.7 Build the test plan

Any project can be divided into units that can be further performed for detailed processing. Then a testing strategy for each of this unit is carried out. Unit testing helps to identity the possible bugs in the individual component, so the component that has bugs can be identified and can be rectified from errors.

CHAPTER 7

FUTURE ENHANCEMENT

7.1 FUTURE ENHANCEMENTS:

Although the proposed YOLOv10n-based system has proven to be efficient and highly accurate for real-time road damage detection, there are several areas where future enhancements can further improve its performance and applicability. One potential enhancement is the integration of severity classification, where the system not only detects the presence of damage but also evaluates its depth and size to prioritize maintenance activities. Another promising direction is the use of multi-sensor data fusion, combining visual inputs with LiDAR, thermal imaging, or IoT-enabled road sensors to improve detection accuracy under challenging conditions such as low lighting, rain, or heavy traffic. Additionally, the framework can be extended to incorporate predictive analytics using historical data and machine learning models, enabling authorities to forecast potential road failures before they occur and take preventive measures. Deployment of the system in a cloud-edge hybrid environment could also be considered, where critical detections are handled locally on edge devices while large-scale data analysis and model updates are performed in the cloud, ensuring both efficiency and scalability. Furthermore, the adoption of transfer learning and continual learning strategies would allow the system to adapt dynamically to new types of road damages and varying environmental conditions without the need for retraining from scratch. These enhancements will make the framework more robust, intelligent, and practical for large-scale smart city implementations, ultimately ensuring safer and more sustainable road networks.

CHAPTER 8

CONCLUSION AND REFERENCES

8.1 CONCLUSION

Road infrastructure maintenance is a critical requirement for ensuring transportation safety and supporting the growth of smart cities. Traditional road inspection methods, though reliable on a small scale, are highly labor-intensive, time-consuming, and prone to human errors, making them unsuitable for large-scale monitoring. This project addressed these

challenges by developing an automated road damage detection system using the YOLOv10n algorithm, which is lightweight, efficient, and optimized for real-time deployment on edge devices. The proposed system demonstrates that YOLOv10n can achieve high accuracy while maintaining low computational requirements, making it far more practical than earlier models such as YOLOv7. With a precision of 0.986, recall of 0.973, mean average precision (mAP@0.5) of 0.988, and an F1-score of 0.978, the framework reliably identifies road damages such as potholes, cracks, and surface wear. The system's deployment on NVIDIA Jetson Nano and NVIDIA AGX Orin further validates its adaptability, achieving real-time inference speeds of 7.5 FPS and 67 FPS, respectively. These results highlight the model's scalability and practical use in real-world smart city infrastructures.

Beyond the performance metrics, the project contributes significantly to the vision of intelligent transportation systems by enabling proactive maintenance planning and reducing the dependency on manual inspections. By providing timely insights into road conditions, this system can help reduce accidents, lower vehicle maintenance costs, and improve the overall safety and reliability of transportation networks.

While the current implementation focuses on visual damage detection using edge devices, there remains potential for further enhancement. Future work could integrate predictive analytics, IoT-enabled sensors, or satellite-based monitoring systems to provide deeper insights into road health. Additionally, expanding the system to classify the severity of damages and prioritize repair schedules would make it even more useful for municipal authorities. In conclusion, the proposed YOLOv10n-based framework proves to be an effective, scalable, and efficient solution for real-time road damage detection. By combining high accuracy with resource efficiency, the system lays the foundation for smarter, safer, and more sustainable urban transportation infrastructures.

8.2 REFERENCES

[1] T. Qiu, J. Chi, X. Zhou, Z. Ning, M. Atiquzzaman, and D. O. Wu, ''Edge computing in industrial Internet of Things: Architecture, advances and challenges,'' IEEE Commun. Surveys Tuts., vol. 22, no. 4, pp. 2462–2488, 4th Quart., 2020.

[2] H. Mahmudah, A. Musyafa, A. S. Aisjah, S. Arifin, and C. A. Prastyanto, ''Digital twin: Challenge road damage detection on edge device,'' Chem. Eng. Trans., vol. 109, pp. 601–606, 2024.

[3] L. Zhao, Y. Wu, X. Luo, and Y. Yuan, ''Automatic defect detection of pavement diseases,'' Remote Sens., vol. 14, no. 19, p. 4836, Sep. 2022.

[4] I. Katsamenis, N. Bakalos, E. Protopapadakis, E. E. Karolou, G. Kopsiaftis, and A. Voulodimos, ''Real time road defect monitoring from UAV visual data sources,'' in Proc. 16th Int. Conf. Pervasive Technol. Rel. Assistive Environ., Jul. 2023, pp. 603–609.

[5] M. Rathee, B. Bačí, and M. Doborjeh, ''Automated road defect and anomaly detection for traffic safety: A systematic review,'' Sensors, vol. 23, no. 12, p. 5656, Jun. 2023.

[6] B. Setiadji, Supriyono, and D. Purwanto, ''Surface distress index updates to improve crack damage evaluation,'' in Proc. 11th Asia–Pacific Transp. Environ. Conf. (APTE), 2019, pp. 274–281.

[7] W. Nur, B. S. Subagio, and E. S. Hariyadi, ''Relationship between the pavement condition index (PCI), present serviceability index (PSI), and surface distress index on soekarno hatta road, Bandung,'' Jurnal Teknik Sipil, vol. 26, no. 2, pp. 111–120, Aug. 2019.

[8] M. Surbakti, S. Samsuri, R. Anas, and A. P. M. Tarigan, ''Evaluation of road maintenance program based on international roughness index (IRI) and surface distress index (SDI),'' in ICCOEE: Proc. 6th Int. Conf. Civil, Offshore Environ. Eng. (ICCOEE), Dec. 2020, pp. 764–771. [9] M. Melyar, M. Isya, and S. M. Saleh, ''Pavement condition assessment using SDI and PCI method on geumpang road–west Aceh boundary,'' IOP Conf. Ser., Mater. Sci. Eng., vol. 1087, no. 1, Feb. 2021, Art. no. 012041.

[10] P. Paikun, E. Suminar, A. Irawan, and S. Bahri, ‘‘Determining road handling according to the level of damage using surface distress index (SDI) method,’’ Astonjadro, vol. 10, no. 1, pp. 135–149, Mar. 2021.

[11] K. De Zoysa, C. Keppitiyagama, G. P. Seneviratne, and W. W. A. T. Shihan, ‘‘A public transport system based sensor network for road surface condition monitoring,’’ in Proc. Workshop Netw. Syst. Developing Regions, Aug. 2007, pp. 1–6.

[12] J. Eriksson, L. Girod, B. Hull, R. Newton, S. Madden, and H. Balakrishnan, ‘‘The pothole patrol: Using a mobile sensor network for road surface monitoring,’’ in Proc. 6th Int. Conf. Mobile Syst., Appl., services, Jun. 2008, pp. 29–39.

[13] A. Mednis, G. Strazdins, R. Zviedris, G. Kanonirs, and L. Selavo, ‘‘Real time pothole detection using Android smartphones with accelerometers,’’ in Proc. Int. Conf. Distrib. Comput. Sensor Syst. Workshops (DCOSS), Jun. 2011, pp. 1–6.

[14] S. Rana and Asaduzzaman, ‘‘Vibration based pavement roughness monitoring system using vehicle dynamics and smartphone with estimated vehicle parameters,’’ Results Eng., vol. 12, Dec. 2021, Art. no. 100294.

[15] A. Martinelli, M. Meocci, M. Dolfi, V. Branzi, S. Morosi, F. Argenti, L. Berzi, and T. Consumi, ‘‘Road surface anomaly assessment using lowcost accelerometers: A machine learning approach,’’ Sensors, vol. 22, no. 10, p. 3788, May 2022.

[16] A. Basavaraju, J. Du, F. Zhou, and J. Ji, ‘‘A machine learning approach to road surface anomaly assessment using smartphone sensors,’’ IEEE Sensors J., vol. 20, no. 5, pp. 2635–2647, Mar. 2020.

[17] T. Shen, G. Schamp, and M. Haddad, ‘‘Stereo vision based road surface preview,’’ in Proc. 17th Int. IEEE Conf. Intell. Transp. Syst. (ITSC), Oct. 2014, pp. 1843–1849.

[18] N. Z. Nuzula, R. W. Sudibyo, H. Mahmudah, and N. Sa’Adah, ‘‘Implementation of LiDAR array system for damaged road characteristic identification on moving vehicle,’’ in Proc. Int. Electron. Symp. (IES), Aug. 2023, pp. 244–249.

[19] M. R. De Blasiis, A. Di Benedetto, M. Fiani, and M. Garozzo, ‘‘Assessing of the road pavement roughness by means of LiDAR technology,’’ Coatings, vol. 11, no. 1, p. 17, Dec. 2020.

[20] Z. Feng, A. El Issaoui, M. Lehtomäki, M. Ingman, H. Kaartinen, A. Kukko, J. Savela, H. Hyyppä, and J. Hyyppä, ‘‘Pavement distress detection using terrestrial laser scanning point clouds–accuracy evaluation and algorithm comparison,’’ ISPRS Open J. Photogramm. Remote Sens., vol. 3, Jan. 2022, Art. no. 100010.

[21] G. M. Jog, C. Koch, M. Golparvar-Fard, and I. Brilakis, ‘‘Pothole properties measurement through visual 2D recognition and 3D reconstruction,’’ Comput. Civil Eng., vol. 2012, pp. 553–560, Jun. 2012.

[22] E. Buza, S. Omanovic, and A. Huseinovic, ‘‘Pothole detection with image processing and spectral clustering,’’ in Proc. 2nd Int. Conf. Inf. Technol. Comput. Netw., vol. 810, Oman, 2013, pp. 48–53.

[23] L. Huidrom, L. K. Das, and S. K. Sud, ‘‘Method for automated assessment of potholes, cracks and patches from road surface video clips,’’ Proc.- Social Behav. Sci., vol. 104, pp. 312–321, Dec. 2013.

[24] M. B. S. G. Naik and V. Nirmalrani, ‘‘Detecting potholes using image processing techniques and real-world footage,’’ in Proc. Cognit. Informat. Soft Comput. (CISC), Jan. 2021, pp. 893–902. [25] R. Sharma, K. Singh, and L. Chand, ‘‘Analysis of image processing techniques for road anomalies detection,’’ Int. J. Emerg. Res. Manage. Technol., vol. 5, 2016.

[26] M. Gao, X. Wang, S. Zhu, and P. Guan, ‘‘Detection and segmentation of cement concrete pavement pothole based on image processing technology,’’ Math. Problems Eng., vol. 2020, pp. 1–13, Jan. 2020.

[27] H. Fang and N. He, ‘‘Detection method of cracks in expressway asphalt pavement based on digital image processing technology,’’ Appl. Sci., vol. 13, no. 22, p. 12270, Nov. 2023.

[28] M. H. Yousaf, K. Azhar, F. Murtaza, and F. Hussain, ‘‘Visual analysis of asphalt pavement for detection and localization of potholes,’’ Adv. Eng. Informat., vol. 38, pp. 527–537, Oct. 2018.

[29] N.-D. Hoang, ‘‘An artificial intelligence method for asphalt pavement pothole detection using least squares support vector machine and neural network with steerable filter-based feature extraction,’’ Adv. Civil Eng., vol. 2018, no. 1, Jan. 2018, Art. no. 7419058.

[30] H. Song, K. Baek, and Y.-C. Byun, ‘‘Pothole detection using machine learning,’’ Adv. Sci. Technol., vol. 2018, pp. 151–155, Feb. 2018.

[31] N.-D. Hoang, T.-C. Huynh, and V.-D. Tran, ‘‘Computer vision-based patched and unpatched pothole classification using machine learning approach optimized by forensic-based investigation Metaheuristic,’’ Complexity, vol. 2021, no. 1, Jan. 2021, Art. no. 3511375.

[32] C. Chun and S.-K. Ryu, ‘‘Road surface damage detection using fully convolutional neural networks and semi-supervised learning,’’ Sensors, vol. 19, no. 24, p. 5501, Dec. 2019.

[33] H. Maeda, Y. Sekimoto, T. Seto, T. Kashiyama, and H. Omata, ‘‘Road damage detection and classification using deep neural networks with smartphone images,’’ Comput.-Aided Civil Infrastruct. Eng., vol. 33, no. 12, pp. 1127–1141, 2018.

[34] Z. S. Hernanda, H. Mahmudah, and R. W. Sudibyo, ‘‘CNN-based hyperparameter optimization approach for road pothole and crack detection systems,’’ in Proc. IEEE World AI IoT Congr. (AIIoT), 2022, pp. 538–543.

[35] A. C. Aqsa, H. Mahmudah, and R. W. Sudibyo, ‘‘Detection and classification of road damage using CNN with hyperparameter optimization,’’ in Proc. 6th Int. Conf. Informat. Comput. Sci. (ICICoS), Sep. 2022, pp. 101–104.

[36] R. W. Sudibyo and A. Alimudin, ‘‘Design and implementation of artificial intelligence-based real-time pothole detection for IoT smart infrastructure,’’ Inf., Jurnal Ilmiah Bidang Teknologi Informasi Dan Komunikasi, vol. 8, no. 2, pp. 125–131, Jun. 2023.

[37] A. Zhang, K. C. P. Wang, Y. Fei, Y. Liu, S. Tao, C. Chen, J. Q. Li, and B. Li, ‘‘Deep learning–based fully automated pavement crack detection on 3D asphalt surfaces with an improved CrackNet,’’ J. Comput. Civil Eng., vol. 32, no. 5, Sep. 2018, Art. no. 04018041.

[38] Z. Liu, X. Gu, H. Yang, L. Wang, Y. Chen, and D. Wang, ‘‘Novel YOLOv3 model with structure and hyperparameter optimization for detection of pavement concealed cracks in GPR images,’’ IEEE Trans. Intell. Transp. Syst., vol. 23, no. 11, pp. 22258–22268, Nov. 2022.

[39] M. Sathvik, G. Saranya, and S. Karpagaselvi, ‘‘An intelligent convolutional neural network based potholes detection using YOLO-V7,’’ in Proc. Int. Conf. Autom., Comput. Renew. Syst. (ICACRS), Dec. 2022, pp. 813–819.

[40] J. J. Yebes, D. Montero, and I. Arriola, ‘‘Learning to automatically catch potholes in worldwide road scene images,’’ IEEE Intell. Transp. Syst. Mag., vol. 13, no. 3, pp. 192–205, Fall. 2021.

[41] Y. Cha, W. Choi, and O. Büyüköztürk, ‘‘Deep learning-based crack damage detection using convolutional neural networks,’’ Comput.-Aided Civil Infrastruct. Eng., vol. 32, no. 5, pp. 361–378, Mar. 2017.

[42] M. Faramarzi, ‘‘Road damage detection and classification using deep neural networks (YOLOv4) with smartphone images,’’ Jun. 2020. [Online]. Available: http://dx.doi.org/10.2139/ssrn.3627382

[43] J. Dharneeshkar, S. Aniruthan, R. Karthika, and L. Parameswaran, ‘‘Deep learning based detection of potholes in Indian roads using YOLO,’’ in Proc. Int. Conf. Inventive Comput. Technol. (ICICT), Feb. 2020, pp. 381–385.

[44] V. Pham, C. Pham, and T. Dang, ‘‘Road damage detection and classification with Detectron2 and faster R-CNN,’’ in Proc. IEEE Int. Conf. Big Data (Big Data), Dec. 2020, pp. 5592–5601.

[45] H. Samma, S. A. Suandi, N. A. Ismail, S. Sulaiman, and L. L. Ping, ‘‘Evolving pre-trained CNN using two-layers optimizer for road damage detection from drone images,’’ IEEE Access, vol. 9, pp. 158215–158226, 2021.

[46] R. Vishwakarma and R. Vennelakanti, ‘‘CNN model & tuning for global road damage detection,’’ in Proc. IEEE Int. Conf. Big Data (Big Data), Dec. 2020, pp. 5609–5615.

[47] D. Wang, Z. Liu, X. Gu, W. Wu, Y. Chen, and L. Wang, ‘‘Automatic detection of pothole distress in asphalt pavement using improved convolutional neural networks,’’ Remote Sens., vol. 14, no. 16, p. 3892, Aug. 2022.

[48] I. D. Pratama, H. Mahmudah, and R. W. Sudibyo, ‘‘Design and implementation of real-time pothole detection using convolutional neural network for IoT smart environment,’’ in Proc. Int. Electron. Symp. (IES), Sep. 2021, pp. 675–679.

[49] I. Moazzam, K. Kamal, S. Mathavan, S. Usman, and M. Rahman, ‘‘Metrology and visualization of potholes using the Microsoft Kinect sensor,’’ in Proc. 16th Int. IEEE Conf. Intell. Transp. Syst. (ITSC), Oct. 2013, pp. 1284–1291.

[50] H. Brunken and C. Gühmann, ‘‘Road surface reconstruction by stereo vision,’’ PFG – J. Photogramm., Remote Sens. Geoinf. Sci., vol. 88, no. 6, pp. 433–448, Dec. 2020.

[51] Y. Pan, X. Zhang, G. Cervone, and L. Yang, ‘‘Detection of asphalt pavement potholes and cracks based on the unmanned aerial vehicle multispectral imagery,’’ IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens., vol. 11, no. 10, pp. 3701–3712, Oct. 2018.

[52] S. Faghih-Roohi, S. Hajizadeh, A. Núñez, R. Babuska, and B. De Schutter, ‘‘Deep convolutional neural networks for detection of rail surface defects,’’ in Proc. Int. Joint Conf. Neural Netw. (IJCNN), Jul. 2016, pp. 2584–2589.

[53] L. Liu, W. Ouyang, X. Wang, P. Fieguth, J. Chen, X. Liu, and M. Pietikäinen, ‘‘Deep learning for generic object detection: A survey,’’ Int. J. Comput. Vis., vol. 128, no. 2, pp. 261–318, Feb. 2020.

[54] A. M. Roy and J. Bhaduri, ‘‘Real-time growth stage detection model for high degree of occultation using DenseNet-fused YOLOv4,’’ Comput. Electron. Agricult., vol. 193, Feb. 2022, Art. no. 106694.

[55] B. Jiang, S. Chen, B. Wang, and B. Luo, ‘‘MGLNN: Semi-supervised learning via multiple graph cooperative learning neural networks,’’ Neural Netw., vol. 153, pp. 204–214, Sep. 2022.

[56] A. M. Roy and J. Bhaduri, ‘‘DenseSPH-YOLOv5: An automated damage detection model based on DenseNet and swin-transformer prediction headenabled YOLOv5 with attention mechanism,’’ Adv. Eng. Informat., vol. 56, Apr. 2023, Art. no. 102007.

[57] A. M. Roy, J. Bhaduri, T. Kumar, and K. Raj, ‘‘WilDect-YOLO: An efficient and robust computer vision-based accurate object localization model for automated endangered wildlife detection,’’ Ecol. Informat., vol. 75, Jul. 2023, Art. no. 101919.

[58] A. Singh, K. Raj, T. Kumar, S. Verma, and A. Roy, ‘‘Deep learning-based cost-effective and responsive robot for autism treatment,’’ Drones, vol. 7, no. 2, p. 81, Jan. 2023.

[59] A. M. Roy, R. Bose, and J. Bhaduri, ‘‘A fast accurate fine-grain object detection model based on YOLOv4 deep neural network,’’ Neural Comput. Appl., vol. 34, no. 5, pp. 3895–3921, Mar. 2022.

[60] M. A. K. Raiaan, S. Sakib, N. M. Fahad, A. A. Mamun, M. A. Rahman, S. Shatabda, and M. S. H. Mukta, ‘‘A systematic review of hyperparameter optimization techniques in convolutional neural networks,’’ Decis. Anal. J., vol. 11, Jun. 2024, Art. no. 100470.

[61] S. Shekhar, A. Bansode, and A. Salim, ‘‘A comparative study of hyperparameter optimization tools,’’ in Proc. IEEE Asia–Pacific Conf. Comput. Sci. Data Eng. (CSDE), Dec. 2021, pp. 1–6.

[62] S. C. Smithson, G. Yang, W. J. Gross, and B. H. Meyer, ‘‘Neural networks designing neural networks: Multi-objective hyper-parameter optimization,’’ in Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD), Nov. 2016, pp. 1–8.

[63] H. Cui and J. Bai, ‘‘A new hyperparameters optimization method for convolutional neural networks,’’ Pattern Recognit. Lett., vol. 125, pp. 828–834, Jul. 2019.

[64] S. Watanabe, ‘‘Tree-structured Parzen estimator: Understanding its algorithm components and their roles for better empirical performance,’’ 2023, arXiv:2304.11127.

[65] S. Watanabe, N. Awad, M. Onishi, and F. Hutter, ‘‘Multi-objective treestructured Parzen estimator meets meta-learning,’’ in Proc. 6th Workshop Meta-Learn. Conf. Neural Inf. Process. Syst., 2022.

[66] Y. Ozaki, Y. Tanigaki, S. Watanabe, M. Nomura, and M. Onishi, ‘‘Multiobjective tree-structured Parzen estimator,’’ J. Artif. Intell. Res., vol. 73, pp. 1209–1250, Apr. 2022.

[67] S. Sieradzki and J. Mańdziuk, ‘‘Modified adaptive tree-structured Parzen estimator for hyperparameter optimization,’’ 2025, arXiv:2502.00871.

[68] S. Khessiba, A. G. Blaeich, A. B. Abdallah, A. Manzanera, K. B. Khalifa, and M. H. Bedoui, ‘‘A novel hybrid grid search and tree Parzen estimator for deep learning hyperparameters optimization,’’ in Proc. IEEE/ACS 21st Int. Conf. Comput. Syst. Appl. (AICCSA), Oct. 2024, pp. 1–8.

[69] R. Ishimwe, P. Iradukunda, and J. B. Kwizera, ‘‘Real-time road damage detection using deep convolutional neural networks and a smartphone: Project report,’’ Carnegie Mellon Univ., Africa, Tech. Rep., 2021. [Online]. Available: https://www.mininfra.gov.rw/fileadmin/user_upload/Mininfra/

[70] H. Mahmudah, S. Arifin, A. S. Aisjah, and C. A. Prastyanto, ‘‘Optimizers impact on RetinaNet model for detecting road damage on edge device,’’ in IEEE MTT-S Int. Microw. Symp. Dig., Feb. 2024, pp. 1–6.

[71] M. A. Qurishee, ‘‘Low-cost deep learning uav and raspberry pi solution to real time pavement condition assessment,’’ Univ. Tennessee at Chattanooga, Chattanooga, TN, USA, Tech. Rep., 2019. [Online]. Available: https://scholar.utc.edu/theses/601/

[72] S.-A. Hassan, T. Rahim, and S.-Y. Shin, ‘‘An improved deep convolutional neural network-based autonomous road inspection scheme using unmanned aerial vehicles,’’ Electronics, vol. 10, no. 22, p. 2764, Nov. 2021.

[73] L. Zhang, X. Du, R. Zhang, and J. Zhang, ‘‘A lightweight detection algorithm for unmanned surface vehicles based on multi-scale feature fusion,’’ J. Mar. Sci. Eng., vol. 11, no. 7, p. 1392, Jul. 2023.

[74] J. Jeon, J.-K. Kang, and Y. Kim, ‘‘Filter pruning method for inference time acceleration based on YOLOX in edge device,’’ in Proc. 19th Int. SoC Design Conf. (ISOCC), Oct. 2022, pp. 354–355.

[75] A. Demidovskij and E. Smirnov, ‘‘Effective post-training quantization of neural networks for inference on low power neural accelerator,’’ in Proc. Int. Joint Conf. Neural Netw. (IJCNN), Jul. 2020, pp. 1–7.

[76] M. M. H. Shuvo, S. K. Islam, J. Cheng, and B. I. Morshed, ‘‘Efficient acceleration of deep learning inference on resource-constrained edge devices: A review,’’ Proc. IEEE, vol. 111, no. 1, pp. 42–91, Jan. 2023.

[77] R. D. Rachmanto, Z. Sukma, A. N. L. Nabhaan, A. Setyanto, T. Jiang, and I. K. Kim, ‘‘Characterizing deep learning model compression with posttraining quantization on accelerated edge devices,’’ in Proc. IEEE Int. Conf. Edge Comput. Commun. (EDGE), Jul. 2024, pp. 110–120.