ENHANCED VEHICLE DETECTION AND CLASSIFICATION THROUGH OPTIMIZED YOLOV4 USING VISION-BASED TECHNOLOGY

A DISSERTATION

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE AWARD OF THE DEGREE

OF

MASTER OF TECHNOLOGY

IN

COMPUTER SCIENCE & ENGINEERING

Submitted By:

SAURAV SINGH RANA

2K21/CSE/18

Under the supervision of

DR. RAJESH KUMAR YADAV

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

DELHI TECHNOLOGICAL UNIVERSITY

(Formerly Delhi College of Engineering)

Bawana Road, Delhi-110042

May, 2023

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

DELHI TECHNOLOGICAL UNIVERSITY

(Formerly Delhi College of Engineering)

Bawana Road, Delhi-110042

CANDIDATE'S DECLARATION

I, Saurav Singh Rana, Roll No. 2K21/CSE/18 student of M. Tech (Computer Science and Engineering), hereby declare that the Project Dissertation titled "Enhanced vehicle detection and classification through optimized YOLOv4 using vision-based technology" which is being submitted by me to the Department of Computer Science & Engineering, Delhi Technological University, Delhi, in partial fulfilment of requirements for the award of the degree of Master of Technology, is original and not copied from any source without proper citation. This work has not previously formed the basis for the award of any Degree, Diploma Associateship, Fellowship or other similar title or recognition.

Place: Delhi **SAURAV SINGH RANA**

Date: (2K21/CSE/18)

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

DELHI TECHNOLOGICAL UNIVERSITY

(Formerly Delhi College of Engineering)

Bawana Road, Delhi-110042

CERTIFICATE

I, hereby certify that the Project Dissertation titled "Enhanced vehicle detection and classification through optimized YOLOv4 using vision-based technology"**,** which is submitted by Saurav Singh Rana, Roll No. 2K21/CSE/18, Department of Computer Science & Engineering, Delhi Technological University, Delhi in partial fulfilment of the requirement for the award of the degree of Master of Technology, is a record of the project work carried out by the student under my supervision. To the best of my knowledge this work has not been submitted in part or full for any Degree or Diploma to this University or elsewhere.

Place: Delhi

Date:

DR. RAJESH KUMAR YADAV

(SUPERVISOR)

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

DELHI TECHNOLOGICAL UNIVERSITY

(Formerly Delhi College of Engineering)

Bawana Road, Delhi-110042

ACKNOWLEDGEMENT

I wish to express our sincerest gratitude to Dr Rajesh Kumar Yadav for his continuous guidance and mentorship that he provided me during the project. He showed me the path to achieve our targets by explaining all the tasks to be done and explained to me the importance of this project as well as its industrial relevance. He was always ready to help me and clear our doubts regarding any hurdles in this project. Without his constant support and motivation, this project would not have been successful.

Place: Delhi **SAURAV SINGH RANA**

Date: (2K21/CSE/18)

ABSTRACT

This research aims to improve the detection and classification of vehicles in intelligent transportation systems. A novel model, called YOLOv4_AF, is proposed as an optimized version of YOLOv4. The model addresses challenges related to quick and accurate vehicle detection and identification, such as small gaps between vehicles and interference in images or video frames. It incorporates an attention mechanism to reduce interference features in both the channel and spatial dimensions of the images. Additionally, a modified version of the Feature Pyramid Network (FPN) from the Path Aggregation Network (PAN) is utilized to enhance the effectiveness of down-sampling and improve object positioning in 3D space. Experimental results on the BIT-Vehicle and UA-DETRAC datasets demonstrate the superiority of the proposed YOLOv4_AF model over the original YOLOv4 model, as well as two other state-of-the-art models, Faster R-CNN and EfficientDet. The proposed model achieves impressive mean average precision (mAP) values of 83.45% and 77.08%, along with F1 scores of 0.816 and 0.808, respectively, on the two datasets.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 OBJECTIVE

Object detection and classification are currently widely used in industrial, military, and intelligent transportation systems (ITSs). To efficiently analyze passing autos, for instance, ITSs use vehicle detection and categorization. This enables effective vehicle traffic management and urban development. Hardware-based and vision-based object identification techniques currently in use can be divided into two categories. Bounding boxes are used in vision-based techniques to precisely pinpoint items within still or moving picture frames.

1.2 SCOPE

After object classification, the image displays confidence scores and predicted class labels for each bounding box (BBox). Vision-based object detection techniques include additional categories such as logo-based, dimension-based, and feature-based methods. Logo-based methods focus on identifying objects based on their distinctive logos or brand marks. They are particularly useful in advertising or brand monitoring scenarios. Dimension-based methods analyze the spatial dimensions of objects for detection and localization, proving effective for objects with consistent size or aspect ratio patterns. Feature-based methods extract visual features or patterns from objects, including texture, shape, or color, to accurately differentiate and classify them. Incorporating these techniques enables vision-based object detection systems to cater to diverse application scenarios, ensuring comprehensive object recognition.

CHAPTER 2

LITERATURE SURVEY

2.1 A Comprehensive Study of the Effect of Spatial Resolution and Color of Digital Images on Vehicle Classification

AUTHORS: K. F. Hussain, M. Afifi, and G. Moussa,

ABSTRACT: Vehicle-type classification holds significant importance for various intelligent transportation applications, including speed monitoring, smart parking systems, and traffic analysis. This paper focuses on vision-based classification techniques utilizing only a digital camera, without any additional hardware requirements. The dimensions and colors of digital images affect the cost of the camera used for image acquisition. This study presents a comprehensive analysis of how these characteristics impact the accuracy and performance of the vehicle classification process. State-of-the-art image classifiers are applied to the BIT-Vehicle and LabelMe datasets, which are downscaled to different spatial resolutions. Additionally, the influence of color is examined by converting color versions to grayscale. Through over 46,000 individual experiments, we draw valid conclusions regarding the impact of dimension and color on the classification accuracy and performance of image classification methods. Experimental results reveal that color and spatial resolutions do not significantly affect the classification results obtained by most state-of-the-art image classifiers. However, a correlation is observed between spatial resolution and processing time required by these methods. These findings have practical implications for cost and time savings in vehicle-type classification systems.

2.2 Faster R-CNN: Towards realtime object detection with region proposal networks

AUTHORS: S. Ren, K. He, R. Girshick, and J. Sun

ABSTRACT: Cutting-edge object detection networks heavily rely on region proposal algorithms to identify potential object locations. Despite advancements in networks like SPPnet and Fast R-CNN, the computational bottleneck lies in region proposal computation, affecting the overall running time. To address this, we present a novel solution called the Region Proposal Network (RPN), which resolves the issue by sharing full-image convolutional features with the detection network. By doing so, region proposals become almost cost-free. The RPN functions as a fully convolutional network, simultaneously predicting object boundaries and objectness scores at each position. It is trained end-to-end to generate high-quality region proposals, subsequently utilized by Fast R-CNN for accurate detection. By merging the RPN and Fast R-CNN into a single network and leveraging the concept of 'attention' mechanisms in neural networks, the RPN guides the unified network on where to focus its attention. Our detection system, based on the VGG-16 model, achieves a frame rate of 5fps (including all steps) on a GPU, while attaining state-of-the-art accuracy in object detection on renowned datasets such as PASCAL VOC 2007, 2012, and MS COCO, with only 300 proposals per image. Notably, in competitions like ILSVRC and COCO 2015, Faster R-CNN and RPN have served as foundational components for the 1st-place winning entries in multiple tracks.

2.3 Histograms of oriented gradients for human detection

AUTHORS: N. Dalal and B. Triggs

ABSTRACT: In this study, we investigate feature sets for robust visual object recognition, specifically focusing on linear SVM based human detection. After examining existing edge and gradient based descriptors, we conduct experiments that demonstrate the superior performance of grids of histograms of oriented gradient (HOG) descriptors for human detection. We analyze the impact of each computational stage on the overall performance and find that fine-scale gradients, precise orientation binning, moderately coarse spatial binning, and high-quality local contrast normalization within overlapping descriptor blocks are all crucial for achieving favorable results. By applying the new approach, we achieve nearly perfect separation on the original MIT pedestrian database. To further challenge the detection system, we introduce a more diverse dataset comprising over 1800 annotated human images, incorporating various pose variations and backgrounds.

2.4 Road object detection: A comparative study of deep learning-based algorithms

AUTHORS: M. Haris and A. Glowacz,

ABSTRACT: Deep learning has revolutionized vision-based surround perception and emerged as a prominent area within Intelligent Transportation Systems (ITS). Deep learning algorithms utilizing two-dimensional images have become crucial for autonomous vehicles, offering object detection, tracking, and segmentation capabilities for various road targets like pedestrians, vehicles, traffic lights, and signs. Autonomous vehicles heavily rely on visual data to classify and understand target objects, ensuring the safety of pedestrians and other vehicles in their surroundings. Deep learning-based algorithms exhibit exceptional real-time performance in object detection. While several studies have extensively examined different types of deep learning-based object detection methods, few have conducted comprehensive comparisons of detection speed, accuracy, model size, and energy efficiency. This article aims to fill this gap by providing a detailed and systematic comparative analysis of five popular deep learning-based algorithms—R-FCN, Mask R-CNN, SSD, RetinaNet, and YOLOv4—on the large-scale Berkeley DeepDrive (BDD100K) dataset. The analysis includes evaluation metrics such as mean Average Precision (mAP) value, inference time, model size, computational complexity, and energy efficiency. Additionally, the performance of each algorithm is assessed under diverse road environmental conditions, including different times of the day and night. This comprehensive comparison offers valuable insights into the strengths, limitations, and real-time deployment feasibility of these popular deep learning-based algorithms under practical constraints.

2.5 Rapid object detection using a boosted cascade of simple features

AUTHORS: P. Viola and M. Jones

ABSTRACT: This research paper presents a machine learning approach for rapid visual object detection, achieving high detection rates. It introduces three key contributions. Firstly, the "integral image" is introduced as a new image representation, enabling quick computation of detector features. Secondly, a learning algorithm based on AdaBoost is employed to select critical visual features, resulting in highly efficient classifiers. The third contribution involves a "cascade" method that combines increasingly complex classifiers. This cascade effectively discards background regions while focusing computation on potential object-like regions. Unlike previous approaches, the cascade provides statistical guarantees that discarded regions are unlikely to contain the object of interest, acting as an object-specific focus-of-attention mechanism. In face detection, the system demonstrates comparable detection rates to top-performing previous systems. The detector operates in real-time applications at 15 frames per second without relying on image differencing or skin color detection techniques.

CHAPTER 3

ANALYSIS OF FRAMEWORKS

3.1 CURRENT FRAMEWORK:

The advantages of models for object recognition include high accuracy and precise localization. However, these models have drawbacks, particularly when considering the increasing importance of real-time object detection in practical applications. These drawbacks involve more challenging training requirements and slower operational speeds. The performance of single-stage object detection models, such as You Only Look Once (YOLO) and Single Shot MultiBox Detector (SSD), is increased in terms of operating speeds. For direct object detection, YOLO uses a regression approach, which speeds up processing and makes it possible to learn broad features. In contrast, SSD lacks the consideration of scale relationships, resulting in limitations when recognizing small objects. While both SSD and YOLO have their strengths, they struggle to handle complex graphic scenes, leading to significant detection errors and missed detections.

3.1.1 Drawback:

The primary disadvantages of these models are the requirement for more rigorous training and slower operational speeds. Additionally, sophisticated graphic scenes present a challenge for both SSD and YOLO, leading to higher rates of detection failures and missing data.

3.2 PROPOSED FRAMEWORK:

To tackle the challenges mentioned, this study introduces the YOLOv4 AF model, an optimized version of YOLOv4, for vehicle detection and classification. The suggested model includes an attention mechanism that successfully reduces visual interference in both the channel and spatial dimensions. Additionally, the model makes use of a

modified version of the Path Aggregation Network (PAN) utilized in YOLOv4's Feature Pyramid Network (FPN), which improves the extraction of pertinent features by down sampling. By accurately positioning objects in a 3D space, the YOLOv4 AF model improves object identification and classification performance.

3.2.1 Merits:

1. The proposed YOLOv4 AF model enhances vehicle detection performance.

2. On both of the datasets used in the performance comparison, the YOLOv4 AF model exceeds the three most recent models in terms of mean average precision (mAP) and F1 score.

3.3 SRS SOFTWARE REQUIREMENT SPECIFICATIONS

3.3.1 SDLC

Systems, information, and software engineering all employ the Systems Development Life Cycle (SDLC), referred as the Software Development Life Cycle, to construct or alter systems. It includes a range of models and development techniques used to create these systems. A framework for organising and managing the development of information systems through the software development process, the SDLC idea in software engineering forms the basis for several software development approaches.

3.3.2 Architecture

Structured project management methods, like the SDLC, provide you better control over your projects by dividing up difficult work into smaller, more manageable chunks. Frameworks for software development that are either descriptive or prescriptive are provided by software life cycle models. However, important topics like change management, incident management, and release management processes are not explicitly addressed by conventional SDLC models. Usually, these factors are taken care of at the level of overall project management.

To improve user-developer interaction in the traditional SDLC model, we suggest a threedimensional method in the hypothetical model that includes the user, owner, and developer. The constraints of a "one size fits all" approach to SDLC techniques are intended to be overcome by this concept. We have introduced a new hypothetical model for the SDLC, which is extensively discussed elsewhere, in an effort to address these flaws.

The lack of attention given to important technical concerns unique to the software development process is a disadvantage of addressing these management procedures purely within the context of overall project management. While these problems might be briefly highlighted in project management, they might not be fully handled on the ground.

3.3.3 Understanding SDLC

The planning, development, testing, and deployment of software are all included in the software development life cycle (SDLC). Each of these processes, collectively known as a Software Development Life Cycle Model (SDLC), can be used to carry out these tasks.

Depending on its purpose, a software life cycle model may be either descriptive or prescriptive. Descriptive models provide a historical account of how a specific software system was developed. They provide a basis for comprehending and improving software development procedures as well as building prescriptive models with an empirical basis. Prescriptive models, on the other hand, offer guidance on how software should be developed, providing a framework or methodology for effective software development practices.

3.3.4 Models

1. Linear Model (Waterfall):

- Sequential phases of specification and development.
- Activities are carried out in a linear fashion, with each phase completed before moving to the next.
- Emphasizes thorough planning and documentation.

2. Evolutionary Development:
- Specification and development go hand in one.
- Examples include Spiral, Incremental, Prototype-based development, and RAD.
- Focuses on developing a quality product in less time through iterative and incremental approaches.

3. Spiral Model:
- Development starts with a smaller module and gradually builds upon it in a spiral pattern.
- Also known as Component-based development.
- Emphasizes risk assessment and mitigation throughout the development process.

4. Formal Systems Development:
- Involves transforming a mathematical system model into an implementation.
- Utilizes formal methods and techniques for specification, verification, and validation.
- Ensures a high degree of correctness and reliability in the resulting system.

5. Agile Methods:
- Aim to induce flexibility and adaptability into the development process.
- Emphasize iterative development, continuous feedback, and collaboration.
- Examples include Scrum, Kanban, and Extreme Programming (XP).

6. Reuse-based Development:
- Assembles a system by leveraging existing components or software assets.
- Promotes efficiency, productivity, and consistency through reuse.
- Involves selecting and integrating appropriate components to build the desired system.

3.3.5 General Model

The phases of the software development process can be organized and sequenced using software life cycle models. While various models exist and companies may adopt their

own variations, they generally follow similar patterns. Each phase in the life cycle generates specific deliverables that serve as inputs for the subsequent phase. The process begins with requirements gathering, which is then translated into a design. The implementation phase involves writing the actual code based on the design specifications. Finally, testing is conducted to ensure that the implemented software meets the initial requirements.

3.3.6 Methodologies:

Similar to the incremental paradigm, the spiral model emphasises risk analysis in addition to iterative development. It is divided into four stages: planning, risk analysis, engineering, and evaluation. The software project progresses through these phases in iterations known as Spirals.

In planning phase, process begins with the baseline spiral, where needs are obtained and risk is assessed. Each following spiral builds on the previous one. Requirements are gathered during the planning phase. The focus of the risk analysis phase is on identifying hazards and finding potential solutions. At the end of this phase, a prototype is created.

The engineering phase entails software creation and, at the conclusion, testing. Finally, during the assessment phase, the client assesses the project's output thus far before determining whether or not to advance to the following spiral. The angular component of the spiral model represents progress, while the radius of the spiral represents expense. To efficiently manage software projects, the Spiral Life Cycle Model combines iterative development with risk analysis.

This document is critical in the Software Development Life Cycle (SDLC) since it contains the system's full requirements. It acts as a reference for developers as well as a basis for testing. Any future requirements modifications will require a formal change approval procedure.

Spiral Model, which presented by Barry Boehm in his 1987 essay "A Spiral Model of Software Development and Enhancement," is significant since it was one of the first models to emphasise iterative development. Although it wasn't the initial model to

encompass iterative development, it was the first to elucidate the objective and benefits of iterative development in the software development process.

The Spiral Model's iterations ranged in time from 6 months to 2 years in its initial conceptualization. Each iteration started with a specific design goal and ended with a customer appraisal of the work done thus far. Throughout the project, meticulous analytical and engineering efforts were used, with the final project aim in mind at all times.

3.3.7 Spiral Model steps:

- The new system requirements are carefully defined, aiming to gather comprehensive information. This process usually includes conducting interviews with different users who represent both internal and external stakeholders, while also taking into account other pertinent aspects of the existing system.
- A preliminary design is formulated for the new system.
- An initial prototype of the new system is developed based on the preliminary design. This prototype serves as a scaled-down version of the final product and provides an approximation of its key characteristics.
- A second prototype is evolved through a fourfold process:
- Evaluating the initial prototype by analyzing its strengths, weaknesses, and potential risks.
- Assessing the prerequisites for the ensuing prototype derived from the assessment of the initial prototype.
- Strategizing and formulating the subsequent prototype, including necessary steps and considerations.
- Assembling and examining the subsequent prototype to validate its functionality and address any identified issues.

- The customer has the choice to terminate the entire project if the identified risks are deemed excessively high, considering factors such as development cost overruns or inaccurate operating cost estimates that may lead to an unsatisfactory final product.
- The current prototype undergoes a similar evaluation process as the previous one, assessing its strengths, weaknesses, and risks. If necessary, another prototype is developed from the existing one, following the same four-step procedure mentioned earlier.
- Repetition of steps
- System is constructed based on refined prototype.
- Then the system undergoes comprehensive evaluation, testing, and regular maintenance to ensure optimal performance and minimize issues.

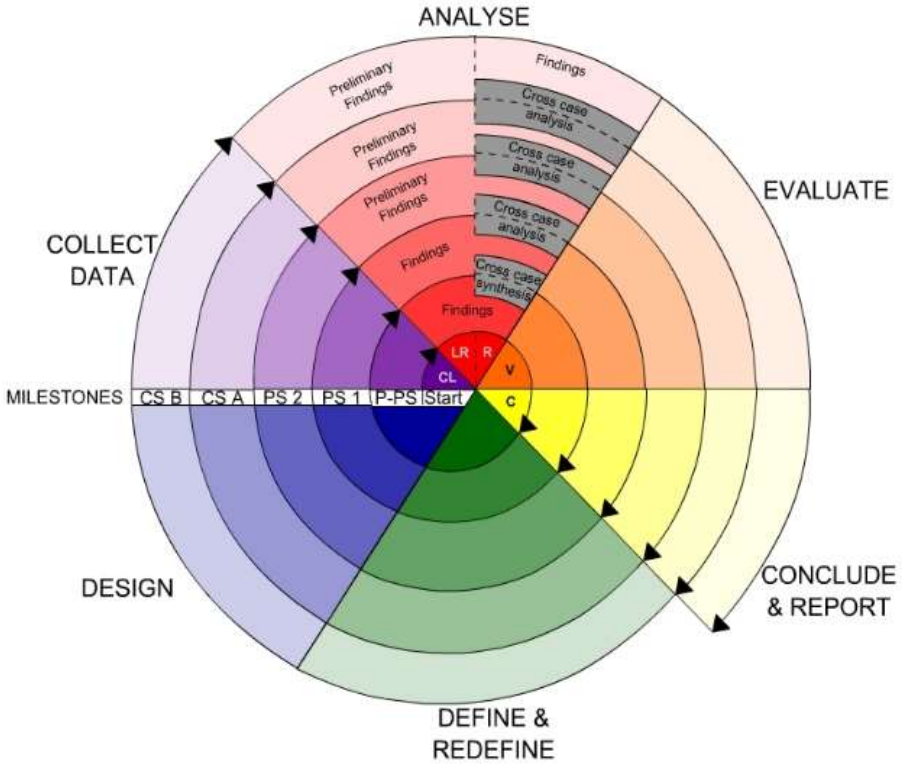


Fig. Spiral Model

CHAPTER 4

SYSTEM REQUIREMENT SPECIFICATION

4.1 HARDWARE REQUIREMENTS

The minimal hardware requirements for a software application differ based on factors such as the type of software being produced and the user's preference for programming tools such as Python, Google colab, or VS Code. Applications involving big arrays of objects may want additional RAM, but those requiring faster processing for complex computations or activities may necessitate a higher-performance processor.

Operating system : Windows 10/11

Processor : Intel i5/i7

Ram : 8 GB

Hard disk : 300 GB

4.2 SOFTWARE REQUIREMENTS

The practical prerequisites and comprehensive depiction documents include a broad array of subjects, encompassing product standpoint, characteristics, working framework, working environment, visuals prerequisites, design constraints, and consumer documentation. These publications give an in-depth assessment of the project, emphasising its merits, flaws, and execution difficulties. This information aids in the development process by highlighting areas that require attention and providing solutions.

Operating System - Windows8/10/11

Programming Language - Python 3.8

4.3 FUNCTIONAL REQUIREMENTS

Functional requirements are expressed in terms of the input to be provided to the system, the operations to be performed on that input, and the expected output. In the context of data collection and processing, the system should be capable of gathering data from various sources. Once collected, the data should undergo processing to ensure its suitability for use. This may involve performing analysis or applying algorithms to gain a better understanding of the data. The functional requirements define the specific operations and capabilities that the system needs to support in order to effectively collect, process, and analyze the data.

The following are the functional requirements for our application as listed below:

1. The system receives a testing image and resizes it to a predefined threshold size.
2. Vectors are generated by the system around the suspected objects in the image.
3. The resized image is then compared with the trained weights using the system's algorithms.
4. Based on the comparison, the system detects objects of interest and creates bounding boxes around them.

Examples:

1. The system enforces user authentication, allowing only users with valid IDs and passwords to access the system.
2. The system implements an authorization process that determines the level of access granted to each user based on their privileges.
3. Users have the option to log out of the system once they have completed their tasks or finished using it.

4.4 NON-FUNCTIONAL REQUIREMENTS

Usability

The "Improved Vision-Based Vehicle Detection and Classification by Optimized YOLOv4" places usability as the primary non-functional requirement. The user interface (UI) should be intuitively designed, ensuring that individuals can easily comprehend and access relevant information without requiring specialized training. Additionally, the option to provide different language support can be incorporated based on specific needs and preferences.

Accuracy

In addition to usability, accuracy is another critical non-functional requirement for the "Improved Vision-Based Vehicle Detection and Classification by Optimized YOLOv4." The dataset is utilized to train and test the model in Python, ensuring that predictions are correct, consistent, and reliable. In vehicle detection and classification, system tries for high precision and accuracy.

Availability

System's availability is crucial for the "Improved Vision-Based Vehicle Detection and Classification by Optimized YOLOv4." It should remain accessible while the user operates it and be recovered within an hour or less in case of a failure. Additionally, the system's response time should be two seconds or less to ensure prompt and efficient handling of user requests.

Maintainability

The software for the "Improved Vision-Based Vehicle Detection and Classification by Optimized YOLOv4" should possess ease of maintainability, allowing for effortless addition of new features and making changes to the software. Furthermore, it must exhibit portability, ensuring that it can be easily transferred or adapted to different environments or platforms.

Examples:

1. Upon the first successful login, users are required to change their initially assigned password immediately. Additionally, the system enforces the policy of not allowing the reuse of previous passwords.
2. The system maintains an audit trail to record every unsuccessful attempt made by a user to access specific data items.
3. The website is designed and optimized to handle a large number of users, with the capacity to accommodate up to 20 million users without compromising its performance.
4. The software is developed with portability in mind, ensuring that it can seamlessly transition from one operating system to another without any compatibility issues.
5. The system incorporates auditing mechanisms to ensure compliance with privacy regulations, export restrictions on sensitive technologies, protection of intellectual property rights, and other relevant legal and security considerations.

4.5 SYSTEM STUDY

4.5.1 VIABILITY STUDY

As part of the system analysis phase, a viability study is conducted to assess the feasibility of the proposed project, including the preparation of a business proposal outlining the project plan and estimated costs. Ensuring the system does not overly burden the company is a key objective of the viability analysis, which requires a comprehensive understanding of the system's essential requirements.

The viability analysis encompasses three primary considerations:

4.5.1.1 ECONOMIC VIABILITY

This evaluation focuses on the economic impact of the system on the organization, ensuring that the allocated funds for research and development are justified. The developed system remained within budget by leveraging freely available technologies and only requiring the purchase of customized products.

4.5.1.2 TECHNICAL VIABILITY

This assessment aims to determine if the system can be implemented without excessive demands on available technical resources, avoiding unnecessary strain on the client. The developed system has modest requirements, necessitating minimal or no changes for its implementation.

4.5.1.3 SOCIETAL VIABILITY

The objective of the investigation is to evaluate the consumer's degree of embrace of the framework. This encompasses the procedure of instructing the consumer on the proper utilization of the framework proficiently. The framework ought not to be dreaded by the consumer, but instead acknowledged as an indispensable requirement. The degree of reception by consumers is chiefly ascertained by the approaches employed to instruct and acquaint them with the framework. The consumer's assurance must be enhanced so that they can provide productive input, which is urged since they are the framework's ultimate consumer.

Fig. Viability Study

CHAPTER 5

SYSTEM DESIGN

5.1 YOLOv4 ARCHITECTURE

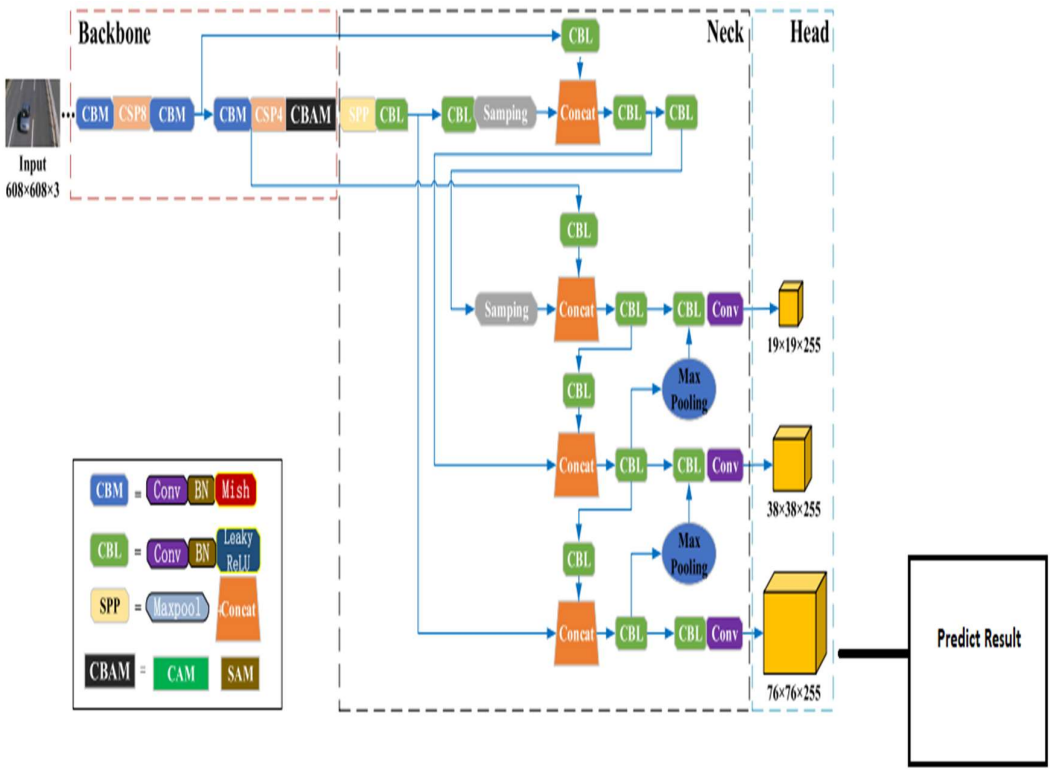
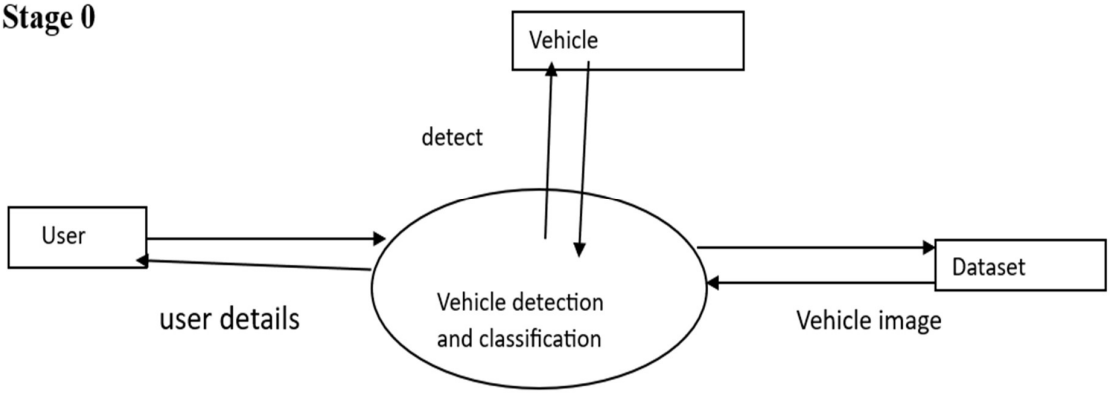


Fig. Architecture

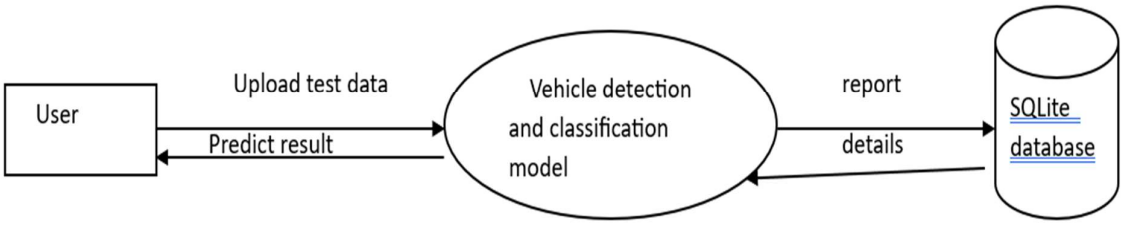
5.2 DATA FLOW DIAGRAM:

1. The data flow diagram (DFD), known as a bubble chart, is a graphical technique used to represent a system at different levels of hidden abstracts, showcasing input data, processing steps, and output data without control flow, loops, or decision rules.
2. DFD models system components, including processes, data utilized by process, external entities interacting with system, and the information flows in the system.
3. By illustrating movement of info that undergoes transformations, DFDs depict the flow of information and the modifications it undergoes as it progresses from input to output.
4. DFDs can be expanded with more details to explain specific operations based on the type of data, allowing for the representation of increasing information flow and functional detail as the diagram is partitioned into levels.

Stage 0



Stage-1



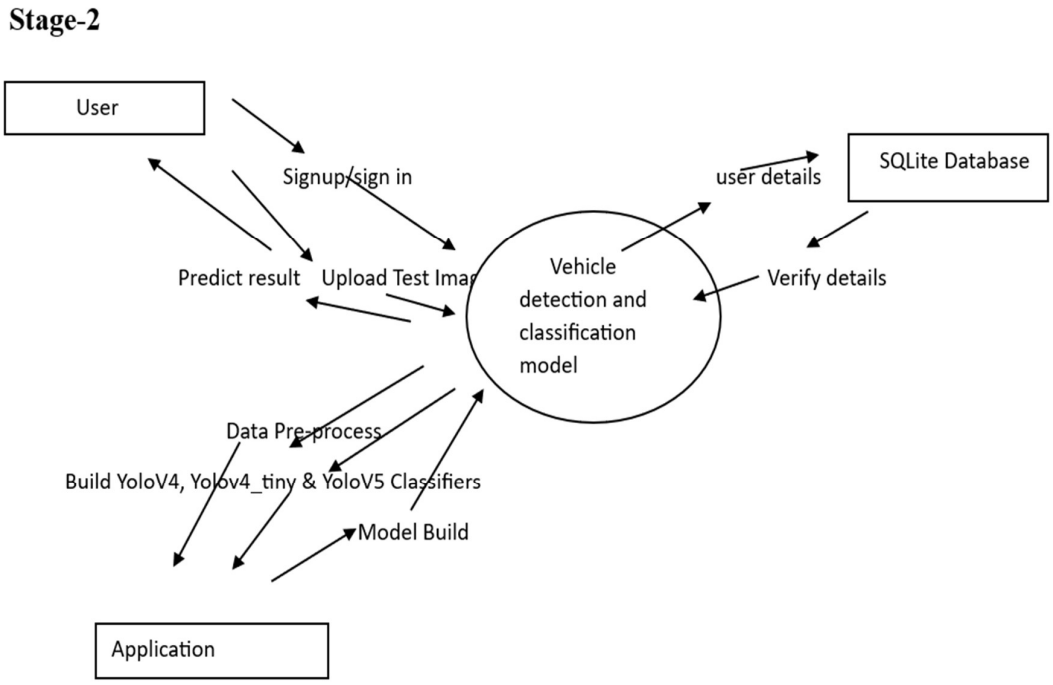


Fig. Data Flow Stages

5.3 UML DIAGRAMS

UML, which stands for Unified Modelling Language, is a standardised modelling language that is extensively used in object oriented software engineering. The Object Management Group established and manages it. UML's goal is to offer a standard language for modelling object oriented computer software. The present version of UML is built from two primary parts notation and meta model. UML might include a kind of process. It's a standard language for describing, visualising, building, and documenting software system artefacts, as well as business modelling and other non-software systems.

The Unified Modelling Language (UML) is a significant set of engineering practises that have proven effective in modelling big and complex systems. It is critical in the creation of object-oriented software as well as the software development process. The UML is largely used to illustrate the design of software projects using graphical notations.

Goals:

The UML is designed with several primary goals in mind:

- Provide users with an easy to use and expressive visual modelling language that allows them to construct and share meaningful models.
- To offer tools for extendibility and specialization, which will allow the key principles to be developed on.
- To be independent of certain programming languages and development methods, allowing it to be used in a variety of contexts.
- To lay a formal foundation for understanding the modelling language and improve its clarity and precision.
- Promote the acceptance and usage of UML to help develop the market for object-oriented (OO) technologies.
- To embrace correct practises and industry standards, allowing the UML to reflect and incorporate the most successful ways in the industry.

5.3.1 Use case

A UML use case diagram illustrates the behaviour of a system by capturing interactions between actors and the system, showcasing the actors' goals through use cases and depicting interactions and dependencies among them, with the main aim of showcasing system functions performed by each actor and emphasizing player roles within the system.

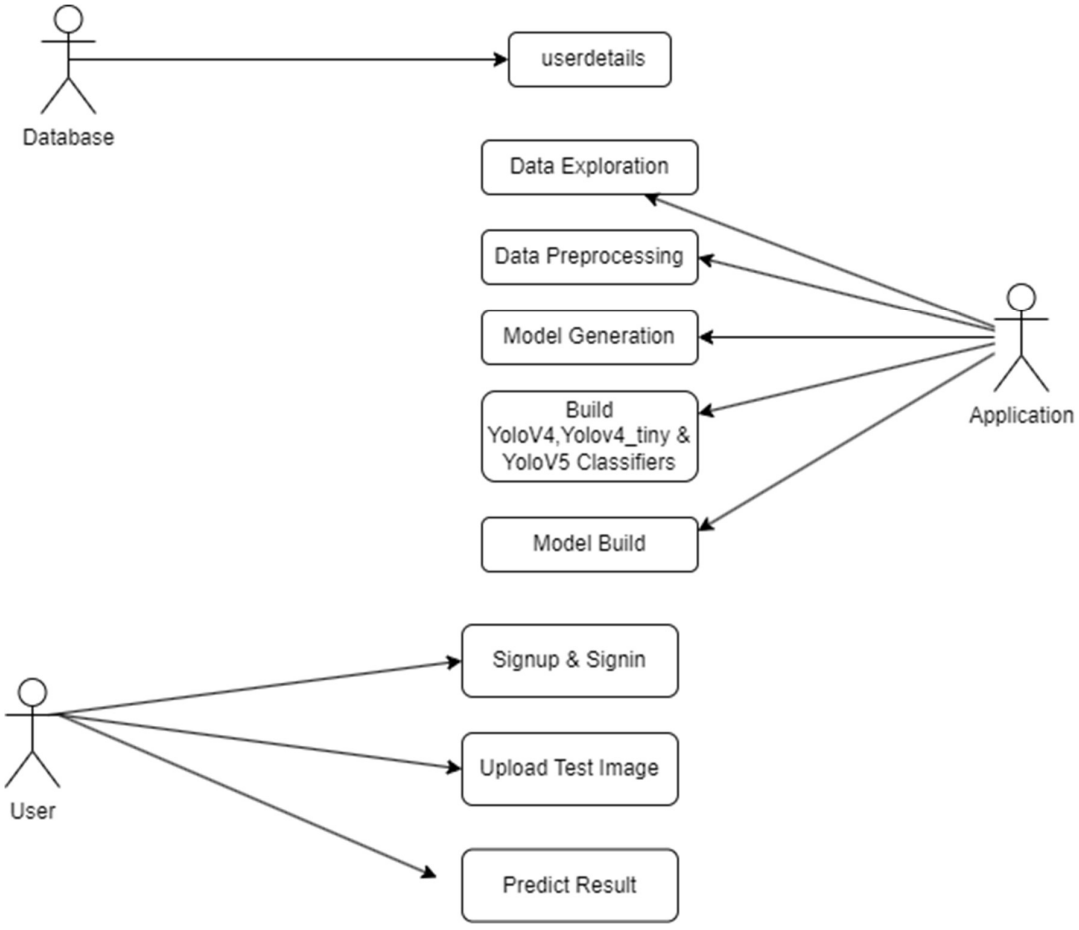


Fig. Use Case Diagram

5.3.2 Class

The UML class diagram is a useful tool for refining the use case diagram and developing a thorough system architecture. It divides the participants in use case diagram into a number of interconnected classes. Class connections can be characterised as "is-a" or "has-a" relationships, signifying inheritance or composition, respectively. Each class in the class diagram represents a distinct entity that may offer specific functionalities, referred to as "methods." Additionally, each class can have attributes that serve as unique identifiers for the class. The class diagram provides a structured representation of the system classes, their relationships, and their characteristics, facilitating a deeper understanding of system's design.

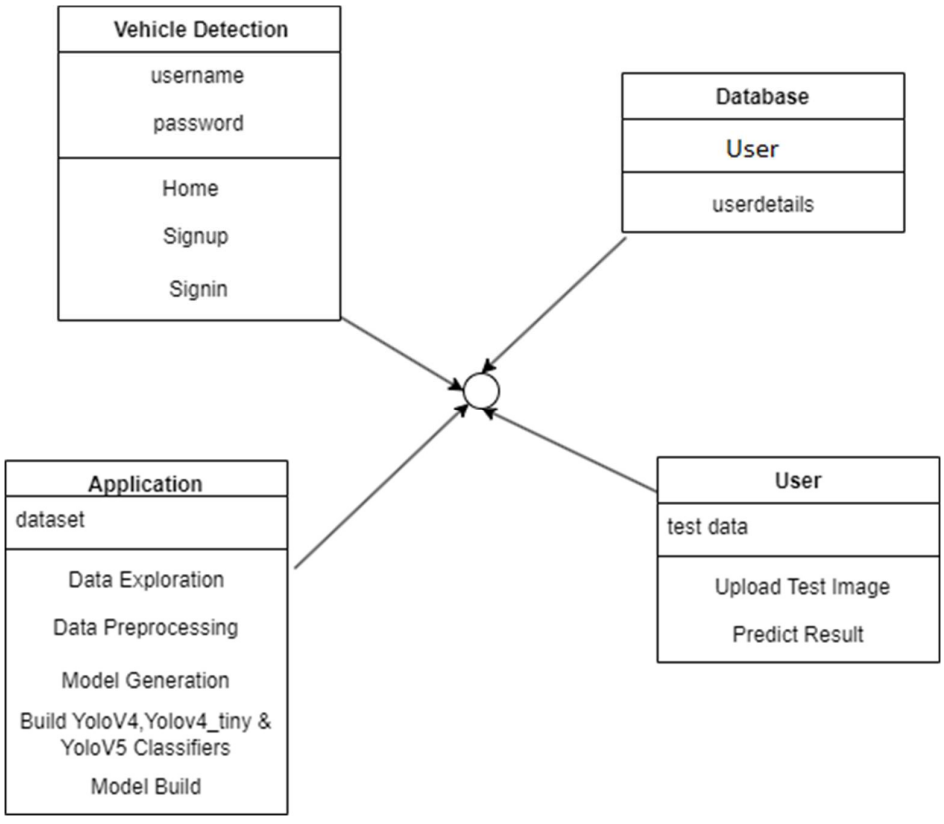


Fig. Class Diagram

5.3.3 Object

The object diagram, a specialized type of class diagram in UML, focuses on representing the state of objects within a system. An object is an instance of a class, representing the specific values and attributes of that class at a particular moment during system execution. The object diagram provides a snapshot of the system, illustrating the state and relationships of various classes and their corresponding objects at a specific point in time. By visualizing the objects and their associations, the object diagram helps in understanding how different objects interact and collaborate within a system during runtime.

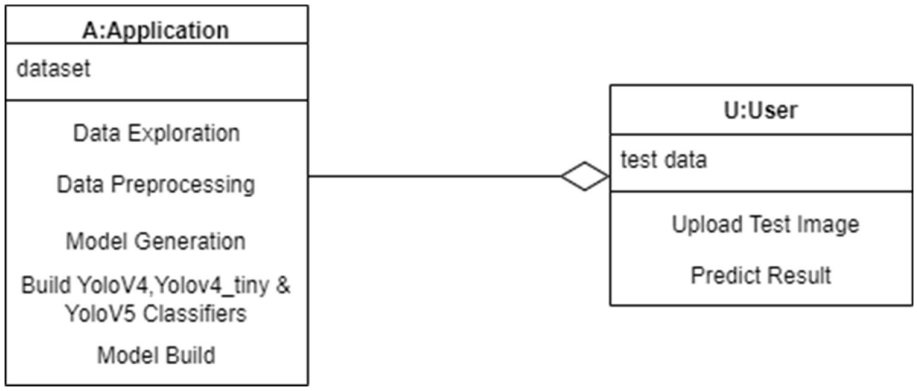


Fig. Object Diagram

5.3.4 State

State diagram or state machine diagram, is a UML behavioral diagram that shows the many states and state transitions of objects in a system. It depicts the life cycle of an object by representing the various states it can be in and the events that trigger transitions between these states. Each state represents a specific behavior or condition of the object, and the transitions illustrate how the object moves from one state to another based on the occurrence of events. State diagrams help to visualize and understand the dynamic behavior of objects and the sequences of state changes that occur within a system.

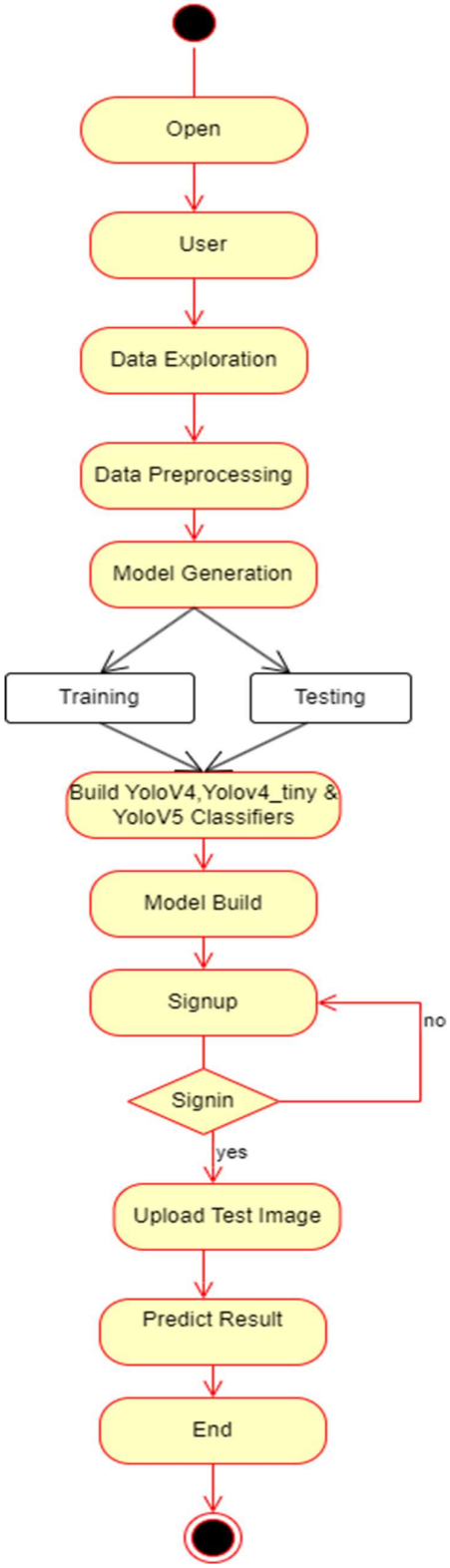


Fig. States Diagram

5.3.5 Activity

Activity diagrams in UML depict the flow of activities within a system, showing the sequence of actions, decisions, and parallel activities. They model business processes, workflows, or complex algorithms, representing steps, actions, and order. Activity diagrams visualize activity flow, including concurrency and looping, providing a clear representation.

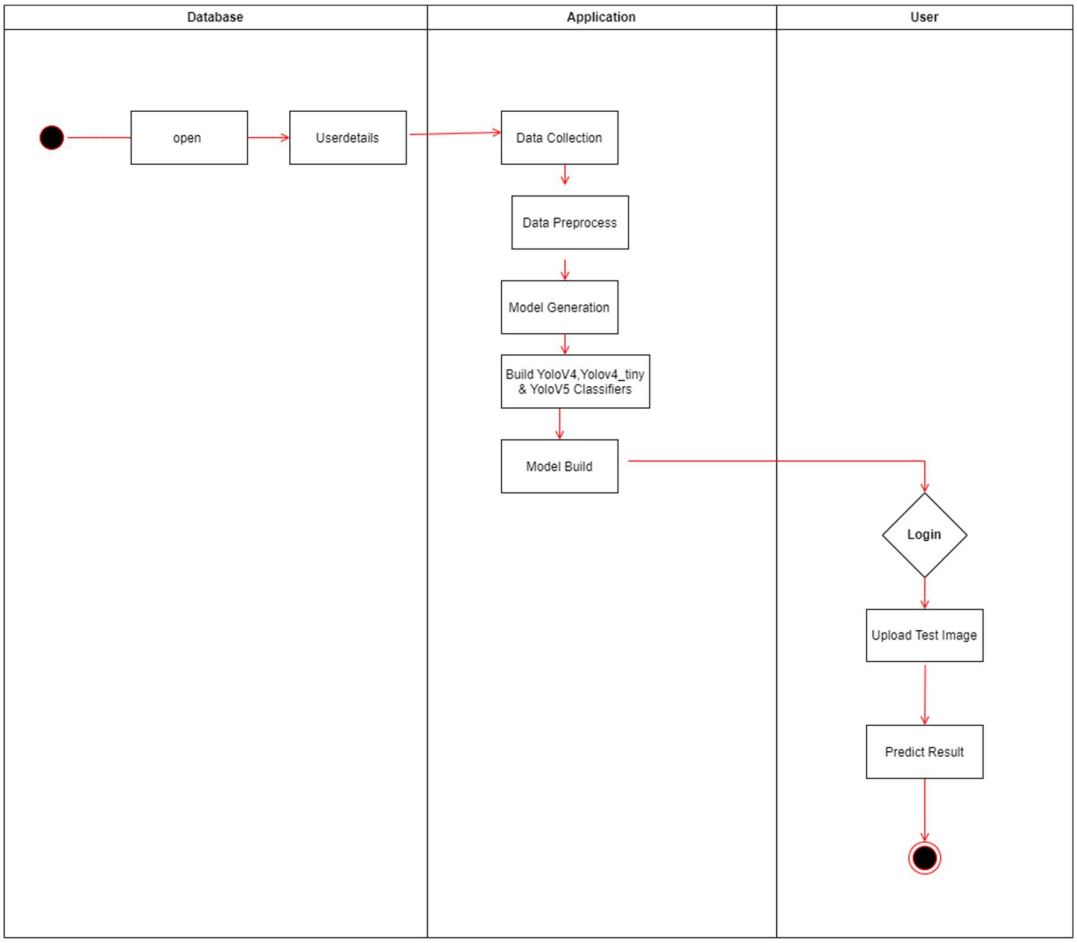


Fig. Activity Diagram

5.3.6 Sequence

A sequence diagram depicts the step-by-step sequence of interactions between items in the system that are time-ordered. Objects communicate through messages, enabling a clear representation of their interactions and the flow of information.

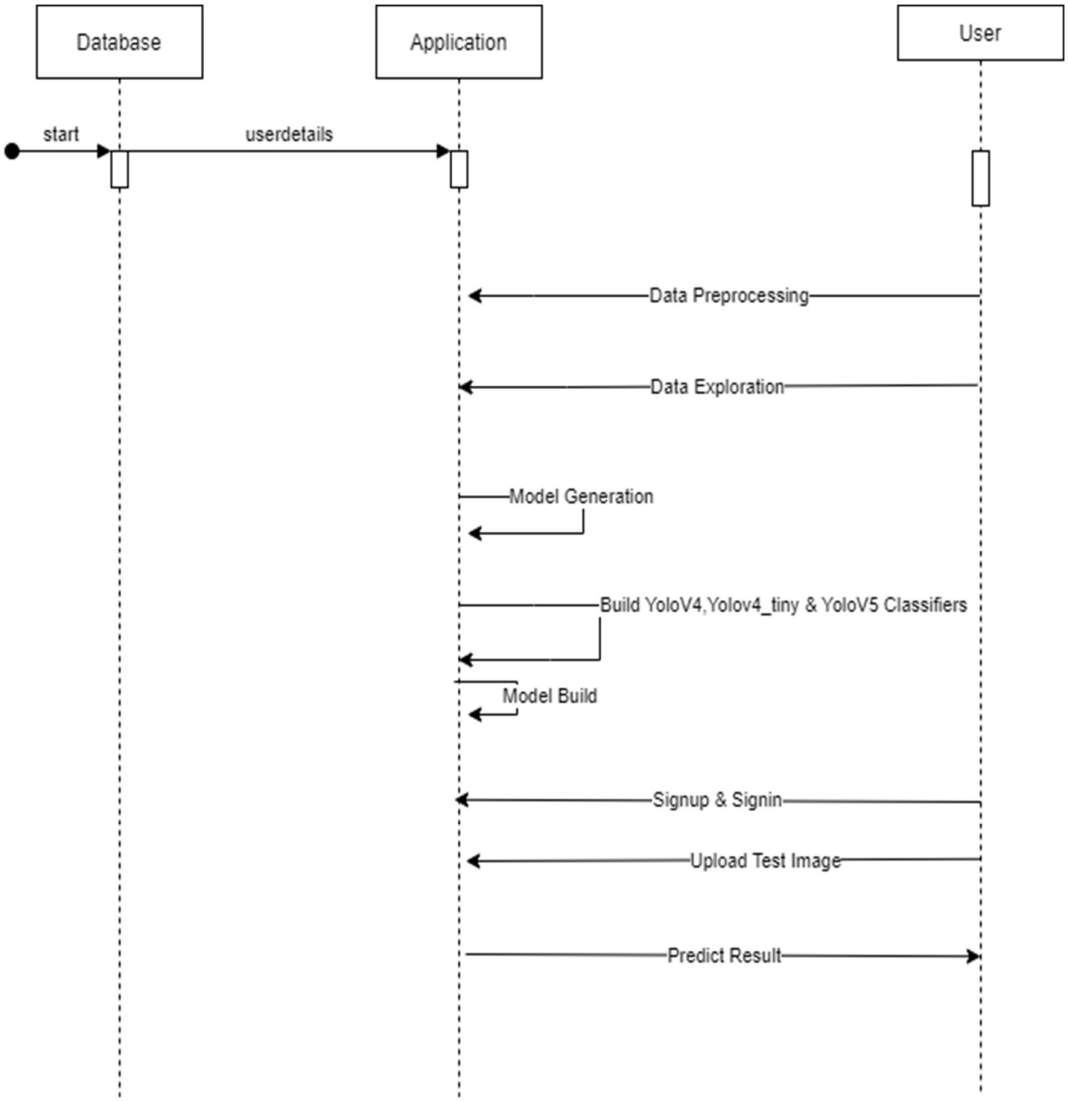


Fig. Sequence Diagram

5.3.7 Collaboration

A collaboration diagram organizes object interactions by grouping them together and assigning numbers to trace their sequence. It provides a comprehensive view of all possible interactions between objects, enabling a thorough understanding of their relationships and interactions.

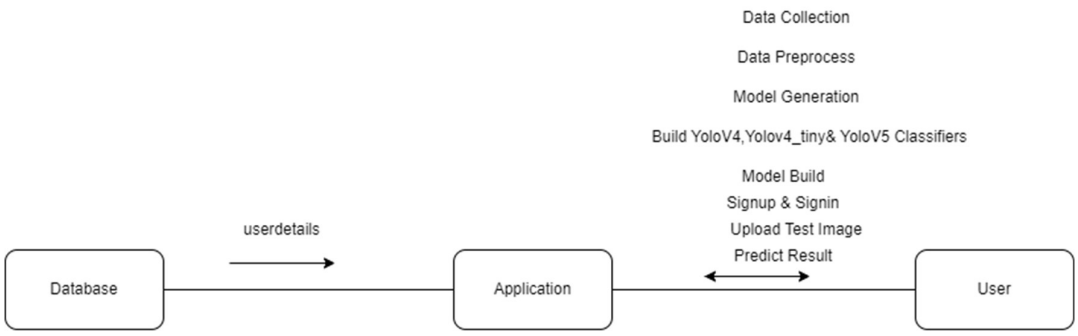


Fig. Collaboration Diagram

5.3.8 Component

A component diagram illustrates the high-level composition of the system by showing its components and their interrelationships. It provides an overview of the components that make up the system, highlighting their interdependencies. The diagram is often prepared following the system's development or construction phase.

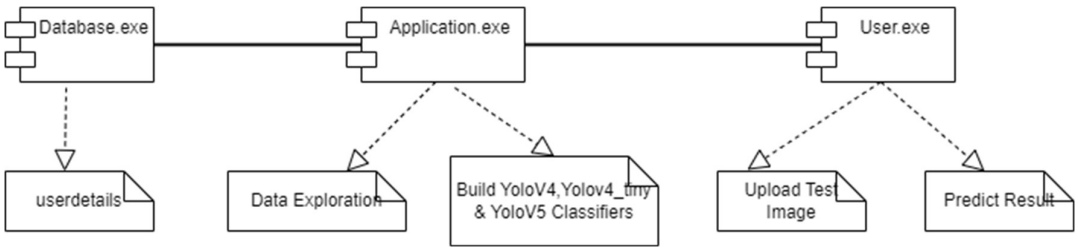


Fig. Component Diagram

5.3.9 Deployment

The deployment diagram depicts the setup of the application's runtime parts. It visually illustrates how the software components and hardware devices are deployed and interconnected in the system. This diagram is particularly valuable when the system is ready for deployment, as it provides a clear understanding of the runtime environment.

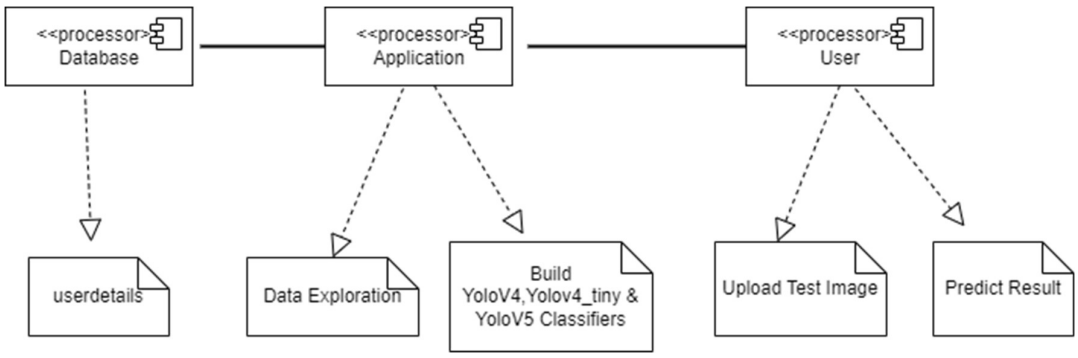


Fig. Deployment Diagram

5.4 MODULES DESCRIPTION:

Data loading: This module is responsible for loading data into the system for further processing.

Data processing: This module processes the loaded data, performing necessary operations and transformations.

Data splitting: With the help of this module, To ease model evaluation, the data is separated into training and testing sets.

Model generation: This module focuses on building YOLOv4, YOLOv4-tiny, and YOLOv5 classifiers for object detection.

User signup and login: This module handles user registration and login functionalities, ensuring secure access to the system.

User input: Through this module, users can provide input for prediction, enabling personalized results.

Prediction: The final predictions are generated based on the user input and displayed to the user.

CHAPTER 6

IMPLEMENTATION

6.1 ALGORITHMS:

6.1.1 Convolution Neural Network:

We will develop a small 6-layer network to discriminate one image from others to show the creation of a convolutional neural network (CNN) for image classification. This network is optimized for CPU execution, making it suitable for practical use. While larger, more complex CNNs are typically employed for image classification tasks, they often require more parameters and extensive training time when run on a CPU. Our focus here is to provide a practical example of building a CNN using TENSORFLOW, showcasing the process of constructing a real-world image classifier.

Neural networks are mathematical models designed to solve optimization problems by utilizing interconnected neurons as their fundamental computation units. A neuron accepts an input (for example, x), conducts calculations (for ex, multiply weight w and adding b), and produces an output result (for example, z = wx + b). This number is then processed through a non-linear function known as an activation function (f) to create the neuron's final output or activation. There are several forms of activation functions, one of which is the sigmoid function. Neurons that employ the sigmoid function as their activation function are referred to as sigmoid neurons. Depending on the choice of activation function, neurons can have various names, such as RELU and TanH.

In neural networks, when neurons are arranged in a sequential manner, it forms a layer, which is another essential component of neural networks. In the context of predicting image classes, multiple layers work together to achieve the best possible match

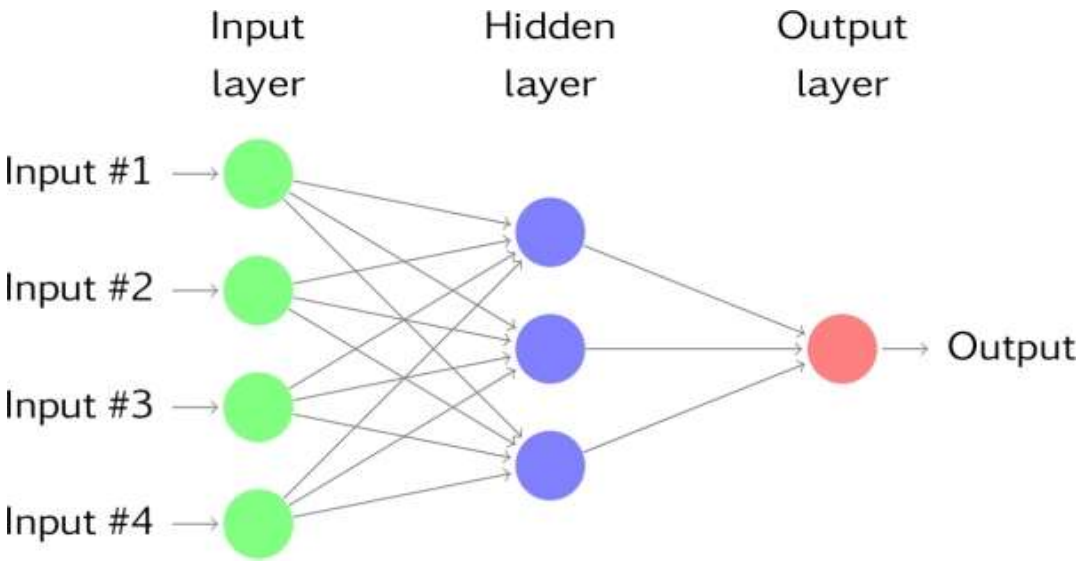


Fig. Convolutional Neural Network

This process of layer-by-layer computation continues until further improvements cannot be obtained. By stacking and connecting layers, neural networks can learn complex patterns and make accurate predictions for image classification tasks.

6.1.2 YOLO

Introduced by Joseph Redmon in 2016, YOLO (You Only Look Once) revolutionized object detection by analyzing the entire image in a single pass to detect and locate objects, treating it as a regression problem rather than repurposing classifiers. By utilizing a single convolutional neural network (CNN) applied to entire image, YOLO predicts spatially separated bounding boxes and class probabilities. It is trained on complete images and optimized for superior object identification performance.

Regardless of development technique or application area, software design is critical in the software engineering process since it is the first stage in producing any designed product or system. The primary goal of the designer is to create a model or depiction of the intended outcome that will be developed later, following the specification and analysis of system requirements, thus constituting one of the essential processes in software construction and validation alongside coding and testing.

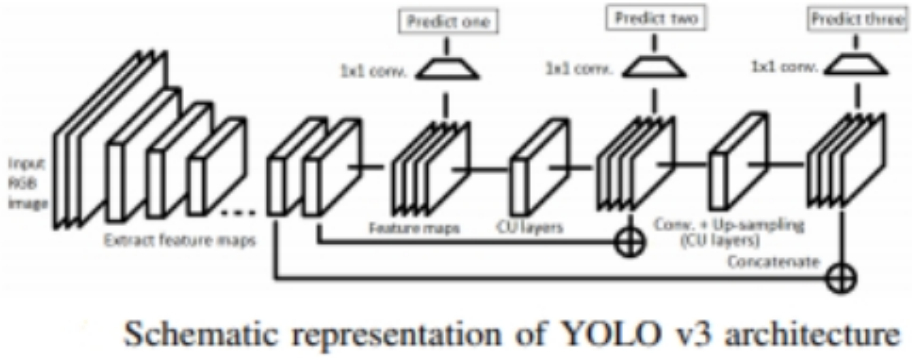


Fig. YOLO Architecture

6.2 PYTHON SOFTWARE

Python is now the most popular multi-purpose, high-level programming language, supporting both the Object Oriented and Procedural paradigms. Python's emphasis on code readability is one of its advantages, with its indentation requirement making programs more comprehensible. Its popularity can be seen in the adoption by major tech firms like as Facebooke, Amezon, Instegram, Ubar.

Python's biggest advantage is its library, containing a plethora of pre-built modules for a variety of applications. Machine learning, GUI development with frameworks such as Kivy, and PyQt, web development with Django (used by YouTube, Netflix, Instagram), BeautifulSoup, and Selenium, as well as test frameworks and multimedia applications, are all examples. Python is a flexible language for a wide range of software development demands due to the availability of such a vast ecosystem of tools.

6.2.1 Advantages

1. Libraries

Python has a large library with pre-built code for a variety of functions such as documentation generation, regular expressions, online browsing, unit testing, threading, database access, CGI programming, email handling, image processing, and much more. This avoids the need to rewrite all of the code.

2. Extensible

Python supports seamless integration with other programming languages, allowing you to incorporate code written in languages like C++ or C. This flexibility proves useful in projects that require combining different language components.

3. Integrable

Python is not only extendable, but it can also be embedded. It allows you to embed Python code is in the source code of another language, like C++. This enables for the incorporation of scripting features into host-language code.

4. Improved Productivity

Python's simplicity and vast library ecosystem contribute to increased productivity compared to languages like Java and C++. With Python, programmers can achieve more by writing fewer lines of code, thereby accelerating development time.

5. IOT Opportunities

Python's compatibility with platforms like Raspberry Pi opens up opportunities for IoT development. Its seamless integration with IoT devices makes Python a favorable language for connecting software with the physical world.

6. Simple

Unlike Java, where creating a class is required to print "Hello World," Python achieves the same result with just a simple print statement. Its straightforward syntax and minimalistic approach make it easy to learn, understand, and write code. This simplicity often makes transitioning to other more verbose languages challenging for Python users.

7. Readable

Python's syntax is compact and clear, similar to natural language, making it very readable and intuitive. Python code is visually organised and easy to understand since it does not require curly brackets to create code blocks or enforced indentation.

8. Object-Oriented

Python is compatible with both functional and OOPS paradigm. Whereas code reuse is used in functions objects and classes make it easier to model real-world notions. Classes enable the encapsulation of data and functions, promoting code organization and modularity.

9. Free Open-Source

Python is freely available, and its open-source nature allows users to access the source code, modify it, and distribute it as needed. Additionally, Python provides an extensive library collection that assists in various tasks, enhancing productivity.

10. Portable

Unlike languages such as C++, Python offers platform portability. Once you write your code, it can be run on any platform without significant modifications, following the "Write Once, Run Anywhere" (WORA) principle. However, care must be taken to avoid relying on system dependent features to ensure true portability.

6.2.2 Disadvantages

While Python offers numerous advantages, it's important to consider its limitations before choosing it for a project. Here are some downsides to keep in mind:

1. Speed Limitations

Due to its interpreted nature, Python can be slower compared to languages like C++ or Java. This may become a concern for performance-critical applications. However, for most general-purpose tasks, the benefits of Python outweigh its limitations.

2. Browsing capabilities

Python is mostly utilized on the server side and is less frequent on the client side or in mobile app development. While frameworks like Brython exist for running Python in browsers, their adoption is limited due to security concerns. This restricts Python's usage in certain domains.

3. Design Restrictions

Python is dynamically-typed, allowing variables to be assigned without specifying their types explicitly. While this flexibility can enhance programmer productivity, it can also lead to runtime errors if not handled properly. Static type checking tools like MyPy can mitigate this issue.

4. Insufficient Database Connectivity

Python's database access layers are not as mature or widely adopted as technologies like JDBC or ODBC. This can be a consideration for large enterprises with complex database requirements, as they may prefer more established frameworks and technologies.

6.3 MACHINE LEARNING

It's crucial to define machine learning and its limitations before diving in different ML techniques. Although ML few times seen as a subset of AI, its usage in data science is more concerned with creating models to comprehend data than with imitating human intellect.

Machine learning is fundamentally the creation of mathematical models that facilitate the understanding of data. When these models have changeable parameters that can change based on collected data, the word "learning" is used. The program can learn from the data it encounters thanks to its versatility. Predictions and fresh data analysis may be done using the models after they have been fitted to the available data.

It's important to note that there is philosophical disagreement over how closely this mathematical, model-based "learning" mimics the learning that occurs in the human brain. However, mastering machine learning's issue setup is essential for making the most of these techniques.

Let's start with some general categorizations of the machine learning methodologies we'll be looking at to provide the groundwork for our discussion.

6.3.1 Categories Of Machine Learning

Modelling the correlation between measured data characteristics and related labels underlies supervised learning. The model that results from this association may subsequently be used to classify brand-new, unlabeled data. There are two further categories of supervised learning classification and regression tasks. While labels in regression are continuous values, those in classification are discrete categories. We will look at instances of both kinds of supervised learning in the next section.

Unsupervised learning, on the other hand, entails modelling dataset characteristics without making use of labels that have already been assigned. It may be viewed as giving the dataset the opportunity to exhibit its innate structure. Unsupervised learning includes activities like dimensionality reduction and grouping. While dimensionality reduction techniques look for more condensed representations of the data, clustering algorithms strive to detect unique groupings or clusters within data. Next examine instances of all kinds of unsupervised learning in the given section.

6.3.2 Need

Machine learning serves a crucial purpose in enabling efficient and scalable decision-making based on data. While human intelligence is unparalleled in its complexity, Organizations may get useful insights and make data-driven choices thanks to AI technologies like machine learning, deep learning, and artificial intelligence. By automating processes and leveraging data-driven decisions, organizations can tackle real-world problems that cannot be easily programmed using traditional logic. While human intelligence remains essential, machine learning fills the gap by addressing the need for efficiency and scalability in problem-solving at a large scale.

6.3.3 Challenges

Machine Learning, although making advancements in areas like cybersecurity and autonomous cars, still encounters various challenges that hinder its progress. These challenges include:

1. Quality of data: ML algorithms require high-quality data, and the use of low-quality data can lead to issues during data preprocessing and feature extraction.
2. Time-consuming tasks: ML models often consume a significant amount of time for data acquisition, feature extraction, and retrieval, posing a challenge in terms of efficiency.
3. Lack of specialist resources: As ML technology is still relatively new, finding skilled experts in the field can be challenging.
4. Unclear business objectives: ML models require clear and well-defined goals to address specific business problems, and the lack of clarity in formulating objectives can impede progress.
5. Overfitting and underfitting: If a model is overfitting or underfitting, it fails to accurately represent the problem, leading to suboptimal results.
6. Curse of dimensionality: ML models can struggle when dealing with high-dimensional data, where the number of features becomes overwhelming.
7. Difficulty in deployment: The complexity of ML models can make their deployment in real-life applications challenging and resource-intensive.

6.3.4 Applications

Machine Learning has emerged as a rapidly growing technology, particularly in the field of Artificial Intelligence (AI). It offers solutions to complex real-world problems that cannot be effectively addressed using traditional approaches. Here are some noteworthy examples of machine learning in practice:

1. Analyzing and interpreting human emotions from textual or audio data.
2. Determining the sentiment or opinion expressed in text or speech.
3. Identifying and preventing errors or anomalies in systems or processes.
4. Using data analysis to forecast and predict weather conditions.
5. Analyzing and predicting stock market trends and performance.
6. Generating artificial human speech from text or data.
7. Converting spoken language into written text or commands.
8. Dividing customers into distinct groups based on common characteristics or behaviors.
9. Identifying and classifying objects or patterns within images or videos.
10. Identifying fraudulent activities or behaviors within systems or transactions.
11. Implementing measures to proactively prevent fraudulent activities or behaviors.
12. Providing personalized recommendations to customers based on their preferences and browsing history in online shopping.

Advantages

1. **Trend and pattern identification**: Large data sets may be examined via machine learning to find hidden trends and patterns that may not be visible to people right away. This gives companies the ability to comprehend client behaviour and preferences and to make data-driven decisions for tailored services and ads.

2. **Automation**: Machine Learning eliminates the need for constant human intervention by enabling machines to learn and make predictions on their own. Tasks such as spam filtering and antivirus software rely on ML algorithms that continuously learn and adapt to new threats without human intervention.

3. **Continuous improvement**: ML algorithms improve over time as they gain more experience and exposure to data. They become more accurate and efficient in making predictions or providing insights. For example, weather forecasting models become more precise as they process larger datasets and learn from historical weather patterns.

4. **Handling complex data**: Even in dynamic and unpredictable contexts, machine learning algorithms are able to handle multidimensional and multi-variety data. They may thus analyse and draw conclusions from a variety of data sources, including sensor data, pictures, text, and more..

5. **Wide range of applications**: Machine Learning finds applications across various industries and sectors, including e-commerce, healthcare, finance, and more. It enables businesses to deliver personalized experiences, target specific customer segments, optimize processes, and improve decision-making based on data-driven insights.

Disadvantages

1. Data Acquisition: Machine Learning relies on large, inclusive, unbiased, and high-quality datasets for effective training. Waiting for new data to be generated can also be a challenge in some cases.

2. Time and Resources: Machine Learning algorithms require sufficient time and resources to learn and develop accurate models. This may involve significant computational power and processing capabilities.

3. Interpretation of Results: Accurately interpreting the results generated by Machine Learning algorithms can be challenging. It requires careful analysis and understanding of the output to make informed decisions.

4. High Error-susceptibility: Machine Learning algorithms are autonomous but prone to errors. Biased or incomplete training data can lead to biased predictions and irrelevant outcomes. Detecting and correcting these errors can be time-consuming and complex.

6.4 Project modules

Scikit learn

Python's Scikit-learn module provides a wide range of supervised and unsupervised learning techniques. It offers a dependable and approachable interface for using these algorithms on your data. The permissive simplified BSD licence that Scikit-learn is published under makes it simple to use for both academic and commercial purposes. It has broad support and is accessible on many Linux distributions, which further encourages adoption and growth within the Python community.

Matplotlib

Matplotlib is a powerful Python toolkit for creating 2D charts of the finest quality. It offers a wide range of options for creating publishing quality figures in various hardcopy formats and interactive settings across several platforms. Whether you are working with Python scripts, the Python and IPython shells, Jupyter Notebook, web application servers, or graphical user interface toolkits, Matplotlib may be included into your workflow. Matplotlib's goal is to enable both the feasibility of complicated operations and the simplicity of basic chores. Its straightforward syntax allows you to build a wide range of visualisations, including plots, histograms, power spectra, charts, scatter plots with only a some code lines.

Tensorflow

Dataflow and differentiable programming are made easier by the open-source software package TensorFlow in a variety of activities. It functions as a library for symbolic maths and has several uses in machine learning, including neural networks. TensorFlow, which grew out of internal requirements for the Google Brain team, is used in both Google research and production. TensorFlow was made available under the Apache 3.0 open-source licence on Octorber 10, 2015.

Pandas

A robust open-source Python package called Pandas provides high-performance capabilities for data analysis and manipulation. Pandas came into existence as a solution to the problem that Python's data analysis capabilities had historically been constrained. Data loading, preparation, manipulation, modelling, and analysis are the five phases that users can carry out with ease using Pandas. This adaptable library has found use in a number of industries, including education, business, economics, statistics, and more. Users may efficiently handle and analyse data from various fields by combining Python and Pandas, creating new opportunities for research and decision-making.

Numpy

Python's Numpy is a flexible tool created for array processing. In addition to a variety of tools for working with these arrays, it provides a high performance multidimensional array object. Numpy is a foundational package for scientific computing and includes a number of key components. A reliable array object of N-dimensional, broadcasting abilities, the ability to combine C/C++ code, and helpful utilities for linear algebra, the Fourier transform, random number operations are some of these. Numpy's flexibility to generate any data types makes it possible to seamlessly integrate with other databases in addition to its primary scientific uses as an effective container for generic data.

CHAPTER 7

SYSTEM TESTING

7.1 TESTING STRATEGIES

Testing is a crucial phase in the software development process where test data is prepared and used to validate and verify the functionality of individual modules. It involves various stages, starting with unit testing to ensure the correctness of individual components. System testing is then conducted to ensure that all system components function effectively together. During testing, it is important to select appropriate test data that covers all possible scenarios and conditions. The main objective of testing is to ensure the accuracy and efficiency of the system before it is deployed for actual operation. In the following description, we will outline the testing strategies employed during this testing phase.

7.1.1 System Testing

Any system or project now includes testing, especially those in the information technology sector. In judging preparation and capacity to bear diverse conditions, it is quite important. To make sure that the software can achieve its goals, testing is crucial before development. It uses a variety of testing techniques to guarantee the software's dependability. The program is put through logical testing, which repeats the execution patterns for a certain collection of data. This rigorous testing strategy thoroughly evaluates the code for all potential proper data and validates the resulting results.

7.1.2 Module Testing

Each module is examined separately to find faults, enabling error identification and remediation without affecting other modules. The program is modified to provide the desired results when it does not have the necessary functionality. A bottom-up methodology is used in testing, starting with the simplest and lowest modules and working up to the next level. Each module is separately tested. For example, the job

categorization module is evaluated seperately utilizing several jobs and determining the rough execution time for each task. The test results are then compared with manually prepared results. This comparison demonstrates the efficient results of the system in contrast to existing one. Similarly, resource classification and job scheduling modules are tested separately in this system, leading to reduced process waiting time based on the obtained results.

7.1.3 Integration Testing

Following the module testing, integration testing is conducted to address any potential errors that may arise during the linking of modules. This testing ensures the proper functioning of all interconnected modules within the system. The testing results indicate a high level of accuracy, demonstrating that the system has correctly mapped jobs to resources.

7.1.4 Acceptance Testing

Once the user confirms that there are no significant accuracy issues, the system proceeds to undergo a final acceptance test. Without actually running the system, this test ensures that it satisfies the initial objectives, requirements, and goals established during the analysis process. By eliminating the need for users and management to perform acceptance tests, this saves time and resources. Once the system successfully passes the acceptance test, it is deemed acceptable and ready for operation.

7.2 ANALYSIS OF TEST CASES

S.NO	I/P	O/P	RESULT
Test Case 1 **(Dataset)**	Vehicle Dataset is the form of input provided by user.	An output is Detect vehicle Result.	A result is Detect vehicle result.
Test Case 2 **(Accuracy)**	The user enters info in the form of a vehicle Dataset.	An output is Detect vehicle Result.	A result is Detect vehicle using YoloV5 algorithm got Accuracy up to 100%.
Test Case 3 **(Machine Learning Algorithms)**	Vehicle Dataset is provided by the user as input.	An output is Detect vehicle Result.	A result is Detect vehicle using YoloV5 algorithm got Accuracy up to 100%.
Test Case 4 **(Integration)**	Test data are provided by the user as input.	An output is Detect vehicle Result.	A result is Detect vehicle using YoloV5 algorithm got Accuracy up to 100%.
Test Case 5 **(Big Bang testing)**	Test data are provided by the user as input.	An output is Detect vehicle Result.	A result prediction using DL algorithm like Yolov4, YoloV4_tiny & YoloV5 for Detected vehicle Result.

Test Case 6 **(Data Flow Testing)**	Test data are provided by the user as input.	An output is Detect vehicle Result.	A result is Detect vehicle Result.
Test Case 7 **(User interface Testing)**	Test data are provided by the user as input.	An output is Detect vehicle Result.	A result is Detect vehicle Result.
Test Case 8 **(User interface Testing-Event based)**	Username and password are used for application login.	An output is user login successfully.	A result is user successfully login to application.
Test Case 9 **(User interface Testing-Event based)**	The user uploads dataset in to application.	An output is user upload dataset Successfully.	A result is user successfully upload dataset in to application.
Test Case 10 **(User interface Testing-Event based)**	The user enters information by clicking Upload Skin Disease Dataset.	An output is Detect vehicle Result.	A result is Detect vehicle using YoloV5 algorithm got Accuracy up to 100%.

Table 7.2 Test Cases

7.3 RESULTS

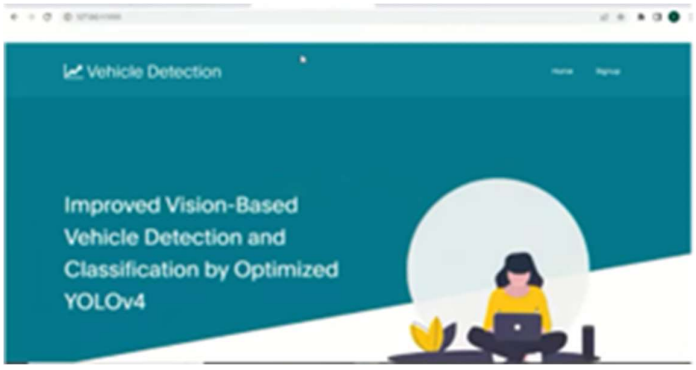


Fig. Home Screen

Fig. Registration

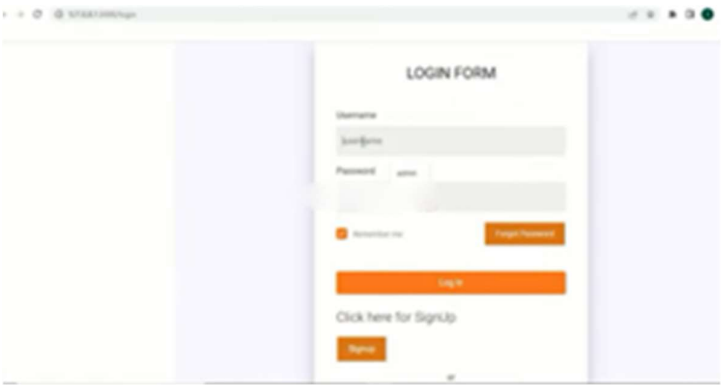


Fig. Login

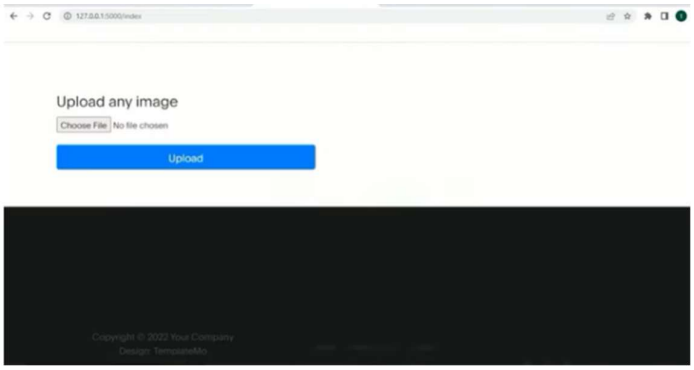


Fig. Main screen

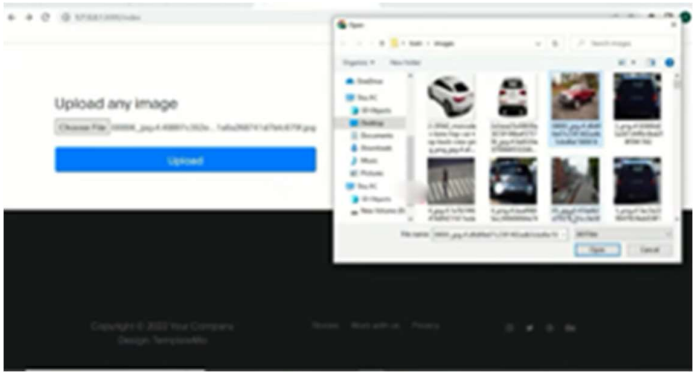


Fig. User input

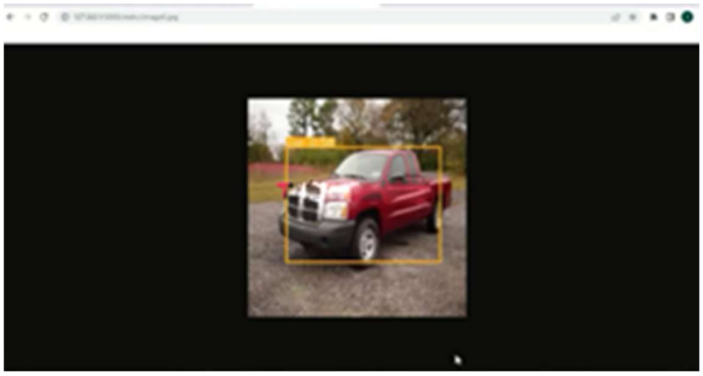


Fig. Prediction Result

CHAPTER 8

CONCLUSION

This study introduces YOLOv4_AF, an improved version of YOLOv4, a more accurate vehicle detection and classification model. In order to broaden the receptive field in both channel and spatial dimensions, the model includes an attention mechanism in the form of a CBAM module. To enhance detection performance, the feature fusion and upsampling techniques in the FPN component are also changed. In terms of mean average accuracy (mAP) and F1 score, experimental testing on the BIT-Vehicle and UA-DETRAC datasets reveals that YOLOv4_AF beats the original YOLOv4 model, as well as Faster R-CNN and EfficientDet. The suggested model may be used to detect different kinds of objects and enhance regression techniques. However, compared to the original, the CBAM module's introduction increases calculation complexity and time than YOLOv4.

8.1 FUTURE WORK:

In the future, we can include expanding the capabilities of the proposed model. We aim to incorporate object tracking for running objects, perform traffic stats analysis, explore the detection and classification abilities of the model on other types of things beyond vehicles. These efforts will further enhance the applicability and versatility of the model in various domains and pave the way for more advanced research in object detection and classification.

8.2 EXTENSION

The paper proposed the use of the YOLOv4_AF model with a residual network for vehicle detection and classification. As an extension to this project, we plan to explore the utilization of two additional models, YOLOv4_tiny and YOLOv5, for vehicle detection and classification purposes. Furthermore, we intend to implement a front-end interface using the Flask framework, enhancing the user experience and interaction with the system. To ensure secure access, we will incorporate user authentication using an SQLite3 database. These extensions aim to improve the system's performance, expand its capabilities, and provide a more robust and user-friendly solution for vehicle detection and classification.

REFERENCES

[1] K. F. Hussain, M. Afifi, and G. Moussa, ‘‘A comprehensive study of the effect of spatial resolution and color of digital images on vehicle classification,’’ IEEE Trans. Intell. Transp. Syst., vol. 20, no. 3, pp. 1181–1190, Mar. 2019.

[2] M. Haris and A. Glowacz, ‘‘Road object detection: A comparative study of deep learning-based algorithms,’’ Electronics, vol. 10, no. 16, p. 1932, Aug. 2021.

[3] P. Viola and M. Jones, ‘‘Rapid object detection using a boosted cascade of simple features,’’ in Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR), Kauai, HI, USA, vol. 1, Dec. 2001, pp. I–I.

[4] W. T. Freeman and M. Roth, ‘‘Orientation histograms for hand gesture recognition,’’ in Proc. Int. Workshop Autom. Face Gesture Recognit., Zurich, Switzerland, 1995, pp. 296–301.

[5] N. Dalal and B. Triggs, ‘‘Histograms of oriented gradients for human detection,’’ in Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR), San Diego, CA, USA, vol. 1, Jun. 2005, pp. 886–893.

[6] S. Woo, J. Park, J.-Y. Lee, and I.-S. Kweon, ‘‘CBAM: Convolutional block attention module,’’ in Proc. Eur. Conf. Comput. Vis. (ECCV), 2018, pp. 3–19.

[7] S. Ren, K. He, R. Girshick, and J. Sun, ‘‘Faster R-CNN: Towards realtime object detection with region proposal networks,’’ IEEE Trans. Pattern Anal. Mach. Intell., vol. 39, no. 6, pp. 1137–1149, Jun. 2017.

[8] R. Girshick, ‘‘Fast R-CNN,’’ in Proc. IEEE Int. Conf. Comput. Vis. (ICCV), Santiago, Chile, Dec. 2015, pp. 1440–1448.

[9] K. He, G. Gkioxari, P. Dollár, and R. Girshick, ‘‘Mask R-CNN,’’ in Proc. IEEE Int. Conf. Comput. Vis. (ICCV), Venice, Italy, 2017, pp. 2980–2988.

[10] G. Gkioxari, J. Malik, and J. Johnson, ‘‘Mesh R-CNN,’’ presented at the IEEE/CVF Int. Conf. Comput. Vis. (ICCV), Seoul, South Korea, 2019.

[11] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, ‘‘You only look once: Unified, real-time object detection,’’ in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Las Vegas, NV, USA, Jun. 2016, pp. 779–788.

[12] W. Kong, J. Hong, M. Jia, J. Yao, W. Cong, H. Hu, and H. Zhang, ‘‘YOLOv3-DPFIN: A dual-path feature fusion neural network for robust real-time sonar target detection,’’ IEEE Sensors J., vol. 20, no. 7, pp. 3745–3756, Apr. 2020.

[13] A. Bochkovskiy, C.-Y. Wang, and H.-Y. Mark Liao, ‘‘YOLOv4: Optimal speed and accuracy of object detection,’’ 2020, arXiv:2004.10934.

[14] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, ‘‘Gradient-based learning applied to document recognition,’’ Proc. IEEE, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[15] L. Xiao, Q. Yan, and S. Deng, ‘‘Scene classification with improved AlexNet model,’’ in Proc. 12th Int. Conf. Intell. Syst. Knowl. Eng. (ISKE), Nanjing, China, Nov. 2017, pp. 1–6.