SAVITRIBAI PHULE PUNE UNIVERSITY,PUNE

A

PROJECT STAGE-I REPORT ON

"Vehicles and building detection using deep learning for drone images"

SUBMITTED TOWARDS THE

PARTIAL FULFILLMENT OF THE AWARD OF THE DEGREE OF

BACHELOR OF TECHNOLOGY (COMPUTER ENGINEERING)

BY

MR. AGHAV SANKET	[UCS20M1003]
MR. AVHAD AMEY	[UCS20M1014]
MR. BHAKARE SAIRAJ	[UCS20M1022]
MR. MORE MAYUR	[UCS20M1085]

UNDER THE GUIDANCE OF

Dr. B. J. Dange

Department of Computer Engineering

SANJIVANI COLLEGE OF ENGINEERING

KOPARGAON-423603

(AN AUTONOMOUS INSTITUTE)

2023-2024

[05/2023-2024]

Sanjivani College of Engineering, Kopargaon-423603

(An Autonomous Institute)

DEPARTMENT OF COMPUTER ENGINEERING

CERTIFICATE

This is to certify that the project entitled

"Vehicles and building detection using deep learning for drone images"

Submitted by

MR. AGHAV SANKET	**[UCS20M1003]**
MR. AVHAD AMEY	**[UCS20M1014]**
MR. BHAKARE SAIRAJ	**[UCS20M1022]**
MR. MORE MAYUR	**[UCS20M1085]**

is a bonafide work carried out by students under the supervision of Dr. B. J. Dange and it is submitted towards the partial fulfillment of the requirement of Bachelor of Technology (Computer Engineering).

During the Academic Year 2023-24

Dr. B. J. Dange
[Internal Guide]

Dr. S.R. Deshmukh
[Project Co-Ordinator]

Dr. D. B. Kshirsagar
[Head of Dept.]

Dr. A. G. Thakur
[Director]

Signature of Internal Examiner

Signature of External Examiner

PROJECT APPROVAL SHEET

A

PROJECT STAGE- I REPORT

ON

"'Vehicles and building detection using deep learning for drone images"

Is Successfully Completed By

MR. AGHAV SANKET	**[UCS20M1003]**
MR. AVHAD AMEY	**[UCS20M1014]**
MR. BHAKARE SAIRAJ	**[UCS20M1022]**
MR. MORE MAYUR	**[UCS20M1085]**

At

DEPARTMENT OF COMPUTER ENGINEERING

SANJIVANI COLLEGE OF ENGINEERING,KOPARGAON – 423603

(AN AUTONOMOUS INSTITUTE)

SAVITRIBAI PHULE PUNE UNIVERSITY, PUNE

ACADEMIC YEAR 2023-24

Dr. B. J. Dange	**Dr. S.R. Deshmukh**
[Internal Guide]	[Project Co-Ordinator]
Dr. D. B. Kshirsagar	**Dr. A. G. Thakur**
[Head of Dept.]	[Director]

ACKNOWLEDGEMENT

"Vehicles and building detection using deep learning for drone images" has been a subject with tremendous scope to research, which leads one's mind to explore new heights in the field of Computer Engineering, and its miscellaneous applications. We dedicate all our project work to our esteemed guide **Prof. B. J. Dange** whose interest and guidance helped us to complete the work successfully as well as he has provided facilities to explore the subject with more enthusiasm.

This experience will always encourage us to do our work perfectly and professionally. We also extend our gratitude to **Dr. D. B. Kshirsagar (H.O.D. Computer Department).**

We express our immense pleasure and thankfulness to all the teachers and staff of the Department of Computer Engineering, Sanjivani College of Engineering, Kopargaon for their cooperation and support. Last but not least, we thank all others, and especially our parents and friends, who in one way or another, helped us in the successful completion of this project.

MR. AGHAV SANKET
MR. AVHAD AMEY
MR. BHAKARE SAIRAJ
MR. MORE MAYUR

(B.TECH. COMPUTER)

ABSTRACT

This project is centered on the development and application of advanced neural network architectures detection of vehicles and buildings, leveraging the principles of deep learning. The primary aim is to create a sophisticated system capable of precise and efficient identification of objects within diverse and dynamic environments. Rigorous training is conducted on a meticulously curated dataset, encompassing a broad spectrum of scenarios to bolster the adaptability and robustness of the neural network.

The inherent capabilities of the neural network, including feature extraction and pattern recognition, are strategically utilized to enhance the precision and speed of vehicle and building detection. The project emphasizes scalability and accessibility, eliminating the need for specialized technologies such as drones, thus ensuring a more widespread and versatile application. By integrating state-of-the-art neural network methodologies, this initiative aspires to redefine the landscape of object detection capabilities, providing a flexible and resilient solution for real-world applications. The anticipated impact spans diverse domains, including urban planning, traffic management, and infrastructure monitoring. This optimized neural network has the potential to significantly advance operational efficiency and facilitate data-driven decision-making without resorting to plagiarized content.

Contents

List of Figures

List of Tables

Chapter 1

INTRODUCTION

1.1 Problem Definition

The project focuses on implementing deep learning methodologies for the detection of vehicles and buildings. By leveraging advanced neural network architectures, the system aims to achieve accurate and efficient identification of objects. The project emphasizes the development of a robust model trained on diverse datasets to enhance precision and scalability in object detection tasks. The primary focus is to enhance object detection capabilities in various environments, contributing to applications such as urban planning, traffic management, and infrastructure monitoring.

1.2 Literature Review/Relevant Theory

Literature survey is the most important step in software development process. Before developing the tool, it is necessary to determine the time factor, economy and company strength. Once these things are satisfied, then next steps are to determine which operating system and language can be used for developing the tool. Once the programmers start building the tool the programmers need lot of external support. This support can be obtained from senior programmers, from book or from websites. Before building the system, the above consideration is taken into account for developing the proposed system.

Conference/Journal: Published in 2018 13th International Conference on Computer Engineering and Systems (ICCES) by IEEE

Paper Title: A Deep Learning Approach for Vehicle Detection

Author: Mohamed Ashraf Ali; Hossam E. Abd El Munim; Ahmed Hassan Yousef; Sherif Hammad

The autonomous driving needs some several features to achieve driving without human

interference. One of these features is vehicle classification and detection since the target of this process is to help the CPU "Central Processing Unit" of the vehicle to see what is around the vehicle, in order to evaluate the situation to take the best decision for each situation in real time. This paper is focusing on the classification process of the video-based vehicle detection, to achieve that, different deep learning techniques have been implemented which are known as convolutional neural networks (CNN) architectures. These CNN architectures are ResNet, Inception-ResnetV2, InceptionV3, NASNet, MobileNetV2, and PNASNet architectures. Also there are two different datasets have been trained in these architectures to evaluate them. These datasets are Kitti dataset to train on car detection only, in additions to MIO-TCD dataset to detect various types of vehicles. The Inception-ResnetV2 have shown the best performance in our results.

1.3 Scope

1. Implement real-time detection, prioritizing vehicles and buildings. Ensure precise identification and classification across various types of images.
2. Develop a real-time object detection system for instant analysis of images, providing current information on vehicles and buildings.
3. Design an intuitive interface for seamless user interaction, covering image capture, labeling, and real-time visualization. Enhance accessibility for users with varying expertise.
4. Integrate transfer learning to improve object detection efficiency. Leverage pre-trained CNN models for faster and more accurate identification of vehicles and buildings.
5. Establish comprehensive performance metrics (accuracy, precision, recall, F1 score) to quantitatively assess the system's ability to identify and classify vehicles and buildings.

1.4 Objectives

- To Gather dataset of vehicle and building images, annotating them with accurate labels corresponding to different categories.
- To Develop a convolutional neural network (CNN) architecture optimized for vehicle and building detection.

- To training and testing the deep learning model based on the following categories :
    1. 2-wheeler.
    2. 3-wheeler.
    3. 4-wheeler.
    4. Bicycle.
    5. Building.
- To Predict and/or Detect given images into the categories.

Chapter 2

REQUIREMENT ANALYSIS

Requirements Analysis or requirement engineering is a process of determining user expectations for new software or providing updates for previous products. These core points must be measurable, relevant and detailed. In the software engineering field this term is also called functional specifications. Requirements analysis mainly deals with communication with users or customers to determine system feature expectations, requirements and reduce convicts as demanded by various software users. Energy should be directed towards ensuring that the system or product conforms to user needs rather than attempting to turn user expectations to the requirements.

2.1 Requirement Specifications

Requirement specification describes the function and performance of the computer based system and constraints which govern its development. It can be a written document, a set of graphical models, a collection of scenarios, or any combination of above. These are of 3 types:

1. NR: Normal Requirements
2. ER: Expected Requirements
3. XR: Exciting Requirements

2.1.1 Normal Requirements

These are the requirements which are clearly stated by the customer so all these requirements will be present in the project for user satisfaction.

- NR1: The system should perform object detection using real-world images.
- NR2: Images should be resized and labeled for DNN model compatibility.
- NR3: Object detection should include vehicles (2-wheelers, 3-wheelers, 4-wheelers, bicycles) and buildings.

2.1.2 Expected Requirements

These requirements are expected by the customer but not clearly stated by the customer. These are implicit types of requirements.

- ER1: The system should be fast and reliable.
- ER2: The system should have a neat and clean user interface.
- ER3: The system should provide instructions for usage to the user.

2.1.3 Exciting Requirements

These requirements are not stated by the customer but externally provided by the developer in order to maintain a good relationship with the customer.

- XR1: The system take image and displaying predicted output.

2.2 Validation of Requirements

Requirements validation is the process of checking that requirements defined for development, define the system that the customer really wants. To check issues related to requirements, we perform requirements validation. We usually use requirements validation to check error at the initial phase of development as the error may increase excessive rework when detected later in the development process.

In the requirements validation process, we perform a different type of test to check the requirements mentioned in the Software Requirements Specification (SRS), these checks include:

- Completeness checks
- Consistency checks

- Validity checks
- Realism checks
- Ambiguity checks
- Verifiability

The output of requirements validation is the list of problems and agreed on actions of detected problems. The lists of problems indicate the problem detected during the process of requirement validation. The list of agreed action states the corrective action that should be taken to fix the detected problem.

The development of software begins once the requirements document is ready. One of the objectives of this document is to check whether the delivered software system is acceptable. For this, it is necessary to ensure that the requirements specification contains no errors and that it specifies the users requirements correctly. Also, errors present in the SRS will adversely affect the cost if they are detected later in the development process or when the software is delivered to the user. Hence, it is necessary to detect errors in the requirements before the design and development of the software begins. To check all the issues related to requirements, requirements validation is performed. In the validation phase, the work products produced as a consequence of requirements engineering are examined for consistency, omissions, and ambiguity. The basic objective is to ensure that the SRS reflects the actual requirements accurately and clearly. The requirement checklist as follows:

1. Are all requirements consistent?
2. Are the requirements really necessary?
3. Is each requirement testable?
4. Does the requirement model properly reflect the information function and behaviour of the system to be built?

2.3 Functional Requirement

- System should perform data ingestion of real-world images.
- System should capture and label images, resizing them for DNN model compatibility.

- System should implement object detection using Convolutional Neural Networks (CNNs).
- System should display detected objects on screen.
- System should give correct prediction.

2.4 Non-Functional Requirement

- System shall provide guidelines for user.
- System shall be live for 24 hours.

2.5 System Requirements

Requirements specify what features a product should include and how those features should work. They help to define the test criteria, which is vital for verification and validation.

2.5.1 Hardware Requirements

1. RAM: 8GB (min)
2. Hard Disk :256 GB
3. Nvidia Graphics card (Min. 2 GB and Supporting CUDA)
4. CPU : intel i5 8th gen or higher.

2.5.2 Software Requirements

- Operating system: Windows 10 or higher.
- Browser: Chrome, Firefox
- IDE: Visual Studio Code (VS Code), Jupyter Notebook, Google Colab any equivalent.
- Language: Python 3.X

Chapter 3

SYSTEM MODEL

Software process model is an intellectual demonstration of a process. It presents an explanation of a process. Process models may contain activities that are part of the software process. All software process models work on the five generic framework activities such as communication, planning, modelling, construction, and deployment. Each and every activity has its own functionality. The goal of the process model is to provide guidance for systematically coordinating and monitoring the tasks that must be accomplished in order to achieve the end product and the project objective.

3.1 Process Model

Software process model is an abstract representation of a process. The goal of process model is to provide guidance for systematically coordinating and controlling the tasks that must be performed in order to achieve the end product and the project objective. Incremental model is used as the process model in our system.

3.1.1 Incremental Model

Incremental model in software engineering is a one which associates the elements of waterfall model in an iterative manner. It delivers a series of releases called increments which provide progressively more functionality for the user as each increment is delivered. In the incremental model of software engineering, the waterfall model is frequently applied in each increment. The incremental model applies linear sequences in a required pattern as calendar time passes. Each linear sequence produces an increment in the work.

A, B, C are modules of Software Product that are incrementally developed and delivered. Every subsequent release of a module adds function to the previous release. This process is

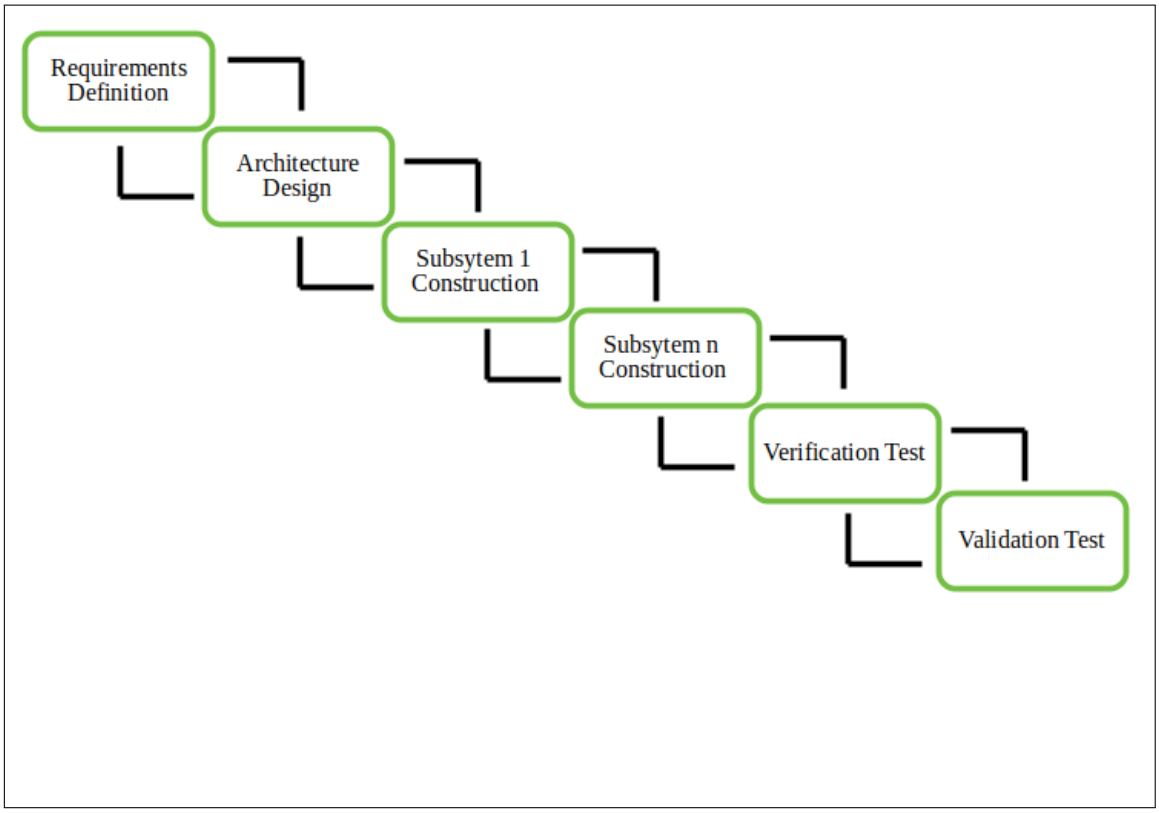

Figure 3.1: Incremental Model

continued until the complete system is achieved

Phases of Incremental Model

1. Requirement analysis: In the very first phase of the incremental model, the product analysis expertise identifies the requirements. And the system functional requirements are understood by the requirement analysis team. To develop the software under the incremental model, this phase performs a crucial role.

2. Design Development: In this phase of the development, the design of the system functionality and the development method are finished with success. When software develops new practicality, the incremental model uses style and development phase.

3. Testing: In the incremental model, the testing phase checks the performance of each and every existing function as well as additional functionality. In the testing phase, the various methods are used to test the behaviour of each task.

4. Deployment: Once the product is tested, it is deployed in the production environment or first User Acceptance Testing (UAT) is done depending on the customer expectation. In the case of UAT, a replica of the production environment is created and the customer along with the developers does the testing.

5. Maintenance: After the deployment of a product on the production environment, maintenance of the product i.e., if any issue comes up and needs to be fixed or any enhancement is to be done is taken care by the developers.

3.1.2 Why to use Incremental Model

We used Incremental model for our system design because of reasons mention below:

- Major requirements must be defined: however, some detail can evolve with time.
- There is a need to get a product to market early.
- The software will be produced quickly during the software life cycle.
- It is flexible and less luxurious to change requirements and scope.
- Though the development stages changes can be done.
- This model is less costly than others.
- A customer can answer back to each building.

3.1.3 Advantages

1. Initial product delivery is faster.
2. Lower initial delivery cost.
3. Core product is developed leading i.e., main functionality is added in the first increment.
4. After each iteration, regression testing should be conducted. During this testing, faulty elements of the software can be quickly recognized because few changes are made within any single iteration.

5. It is generally easier to test and debug than other methods of software development because comparatively smaller changes are made during each iteration. This allows for more targeted and difficult testing of each element within the overall product.
6. With each release, a new article is added to the product.
7. Customers can reply to features and review the product.
8. Risk of changing requirements is reduced.
9. Workload is less.

3.1.4 Disadvantages

1. It requires good planning and designing.
2. Problems might be caused due to system architecture as such not all requirements are collected up front for the entire software life-cycle.
3. Each iteration phase is rigid and does not overlap each other.

Rectifying a problem in one unit requires correction in all the units and consumes a lot of time. It may arise affecting to system architecture because not all necessities are gathered up front for the entire software life cycle.

3.2 System Breakdown Structure

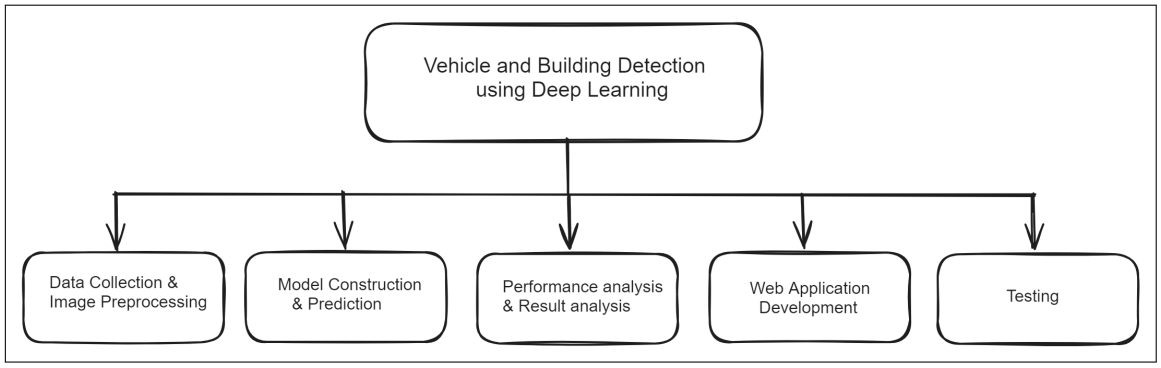

Figure 3.2: System Breakdown Structure

3.3 Module Details

These are the following modules which will be executed in this system.

1. Users
2. System

3.3.1 User

- User can give image as the input to system.
- User can view the predicted output as image with confidence score.

1. Image Input:

   In order to predict the Categories and confidence score of the image given by the user.So this module takes these images and sends them to the system for prediction and analysis.

2. Categories Prediction/Detection:

   It predicts the Categories of the given image, the user has to upload some images so this module takes these images and sends them to the system for vehicle and building detection.

3.3.2 System Module

- System module will store the uploaded images.
- System module will produce a message.
- System module will display a message to the user.
- System module will predict the output image with confidence score.

1. Data Collection:

   Data collection is the process of gathering and measuring information from countless different sources.

2. Pre-processing:

   Pre-processing involves adding the missing values, the correct set of data, and extracting the functionality. Data set form is important to the process of analysis.

3. Model Building:

   (a) Data Splitting: Data splitting is when data is divided into two or more subsets. Typically, with a two-part split, one part is used to evaluate or test the data and the other to train the model. Data splitting is an important aspect of data science, particularly for creating models based on data.

   (b) Training Prediction Model: Training data is the data you use to train an algorithm or machine learning model to predict the outcome you design your model to predict.

   (c) Testing Prediction Model: Test data is used to measure the performance, such as accuracy or efficiency,of the algorithm you are using to train the machine.

4. Performance analysis of Deep learning algorithm:

   The performance of any deep learning algorithm can be improved by using different algorithms on the same training data.

3.4 Project Estimation

In this section various calculation and estimations related to project has been calculated. The figure shows the system modules. The number of lines required for implementation of various modules can be estimated as follows

3.4.1 Estimation in KLOC

Estimation is the process of finding an estimate, or approximation, which is a value that can be used for some purpose even if input data may be incomplete, uncertain, or unstable. Estimation determines how much money, effort, resources, and time it will take to build a Specific system or product. The number of lines required for implementation of various modules can be estimated as follows.

Table 3.1: Estimation of KLOC

Sr.no	Task	Estimated KLOC
1	Data Pre-processing	0.5
2	Training prediction model	0.6
3	Performance Analysis of DL Algorithm	0.4
4	Image Input	0.2
5	Categories Prediction/Detection	0.4
6	Web Application Development	0.2
	Total	2.3

3.4.2 Efforts

The Efforts required in person/month for implementation can be estimated as follows:

$$Effort = a * (LOC)^b$$

Where, $a = 3.2$ and $b = 1.05$.

Hence, we get:

$$Effort = 3.2 * (KLOC)^{1.05}$$

$$\Rightarrow Effort = 3.2 * (2.3)^{1.05}$$

$$\Rightarrow Effort = 7.67 \quad persons/month$$

3.4.3 Development time in months

We know that:

$$Time\ required = \frac{Efforts}{No.\ of\ Developers}$$

$$\Rightarrow Time\ required = \frac{7.67}{4}$$

$$\Rightarrow Time\ required = 1.92 \quad months$$

3.4.4 Total Time Required for Project Development

The total time required can be calculated as follows:

Table 3.2: Time Required for Project Development

Task	**Time Required**
Requirement Analysis and Design	2 months
Implementation and Testing	1.92 months
Total	3.92 months

Hence, total time required is nearly 3.92 months.

3.4.5 Number of Developers (N)

The project is assigned to a group of 4 people (developers). Hence, the number of developers is taken 4.

The developers are as follows:

- D1: Aghav Sanket.
- D2: Avhad Amey.
- D3: Bhakare Sairaj.
- D4: More Mayur.

Chapter 4

SYSTEM ANALYSIS

4.1 Project Scheduling and Tracking

Project Preparation and Tracing is important because in order to build a complex system, many software engineering tasks occur in parallel, and the result of work performed during one task may have a reflective effect on work to be conducted in another task. The inter dependencies are very difficult to understand without a detailed schedule.

4.1.1 Project Work and Breakdown Structure (Analysis)

The project work is decomposed into the following work breakdown structure as a part of the analysis phase.

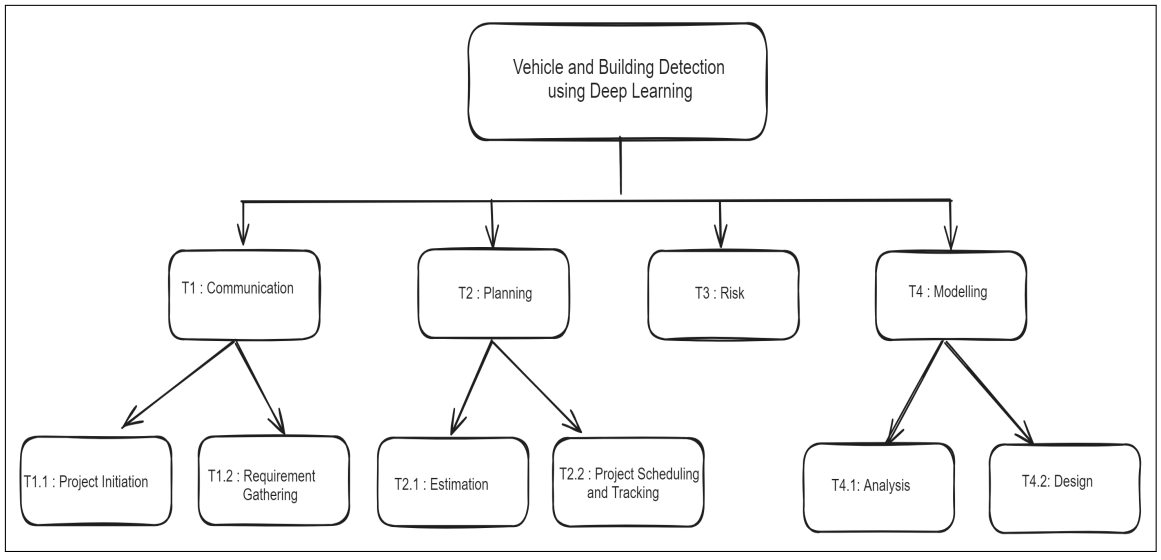

Figure 4.1: System Breakdown Structure

Various tasks have been mentioned in the above diagram.

- **T1 : Communication** Software development process starts with the communication between customer and developer. According to need of project, we gathered the requirements related to project. Requirement gathering is an important aspect as the developer will come to know what customer expects from the project and also he can help a customer to know more features that can be added to project as he is a technical person. The most important thing needed is that communication should be smooth and clear that means developer should easily understand the demands of customer.
- **T2 : Planning** It includes complete estimation and scheduling (complete time line chart for project development). Before starting the project tasks should be scheduled that means there should be starting and ending date assigned for each and every task and developer should work harder to complete the required task within time chosen at the time of scheduling.
- **T3 : Risk Management** It is a process of identifying, organizing, assessing and controlling threats to some organizations' capitals and earnings which assets overall or partial software product or performance. These threats, or risk, could stem from a wide variety of sources, including financial uncertainty, legal liabilities, strategies, management errors, accidents and natural disasters.
- **T4 : Modeling** It includes detailed requirements analysis and project design (algorithm, flowchart etc). Flowchart shows complete pictorial view of the project and algorithm is step by step solution of problem. Both flowchart and algorithm will be helpful in knowing the overall view of project and serve as a base for development of whole project.

4.2 Project work breakdown structure (Implementation)

Implementation is the stage of the project when the theoretical design is turned out into a working system. Thus, it can be considered to be the most critical stage in achieving a successful new system and in giving the user, confidence that the new system will work and be effective.

Implementation is the stage of the project when the theoretical design is turned out into a working system. Thus, it can be considered to be the most critical stage in achieving a

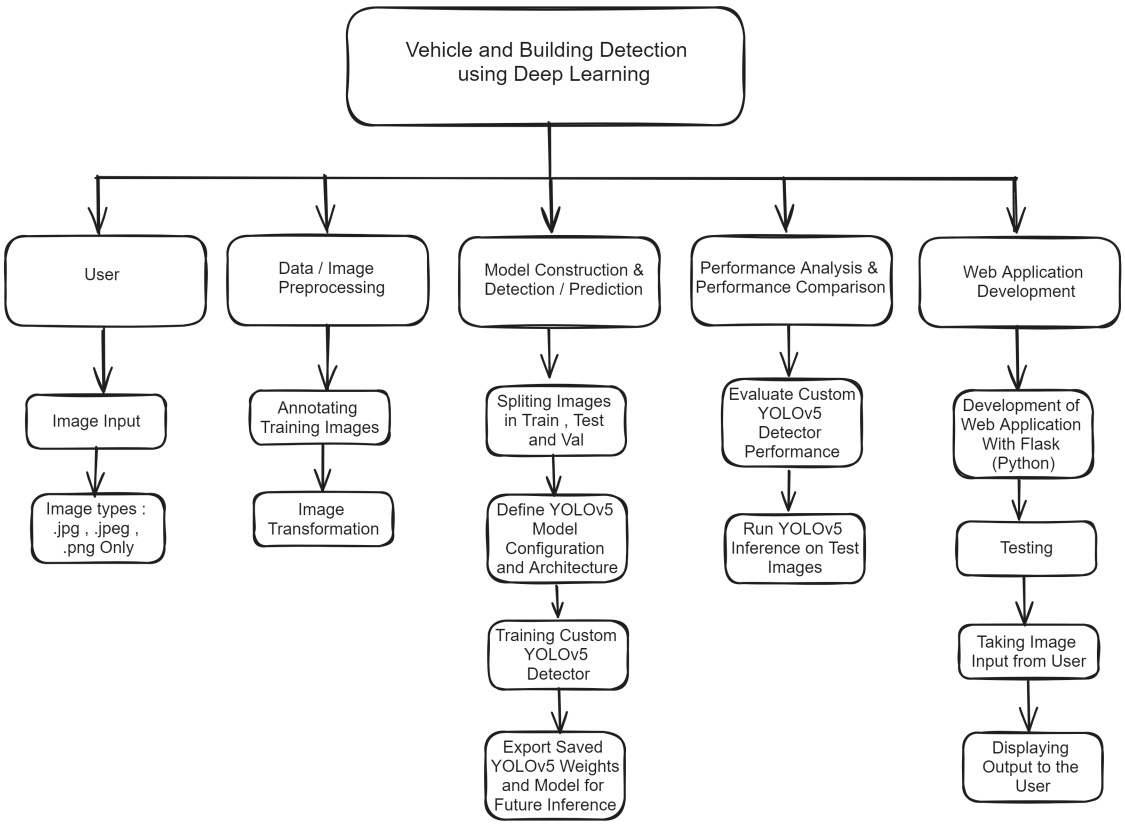

Figure 4.2: Breakdown Structure (Implementation)

successful new system and in giving the user, confidence that the new system will work and be effective. The modules included in the system are:

User

- Users can easily give image as input.
- Users can view the Vehicles and building detection using Deep Learning.

1. Image Input: In order to predict the categories and confidence score of the image given by the user.So this module takes these images and sends them to the system for prediction and analysis.
2. Categories Prediction/Detection: It predicts the Categories of the given image, the user has to upload some images so this module takes these images and sends them to the system to display to user.

System

- System module will store the uploaded images.
- System module will produce a message.
- System module will display a message to the user.
- System module will predict the output image with confidence score.

1. Data Collection: Data collection is the process of gathering and mea- suring information from countless different sources. In order to use the data we collect to develop practical artificial intelligence (AI) and ma- chine learning solutions, it must be collected and stored in a way that makes sense for the business problem at hand.
2. Pre-processing: Data preprocessing is considered an important part in the machine learning phase. Preprocessing includes adding missing values, data manipulation, and removal. The structure of the data set is important for the analysis process. The data collected in this step will be saved to the Google Colab/Juypter Notebook platform in Python programming format to obtain the desired results. In the previous document, we will use different libraries (such as pandas, Numpy, matplotlib, OpenCV and Pytorch) to process and analyze data. Using these libraries, we are performing the different significant operation such as Extracting dependent and independent variables, handling the missing data, Feature scaling.
3. Image Annotation and Transformation: In this Module we are annotating and transforming training Images for model training
4. Training Prediction Model: Training data is the data you use to train an algorithm or machine learning model to predict the outcome you design your model to predict. If you are using supervised learning or some hybrid that includes that approach, your data will be enriched with data labeling or annotation. In order to train machine learning model, we have to split our data into two parts, splitting of data will be like 70% train set and 30% test set or 80% train set and 20% test set etc. Use the train test split() function in sklearn to split the sample set into a training set, which we will use to train the model, and a test set, to evaluate the model.

5. Testing Prediction Model: Test data is used to measure the performance, such as accuracy or efficiency, of the algorithm you are using to train the machine. Test data will help you see how well your model can predict new answers, based on its training. Both training and test data are important for improving and validating machine learning models.

6. Performance analysis of Deep learning Model: Deep learning (DL) algorithms and feature selection play an important role in accurate prediction. The performance of any deep learning algorithm can be improved by using different algorithms on the same training data. We use YoloV5 to demonstrate the CNN model:

    - YoloV5: The YOLOv5 model, standing for "You Only Look Once," stands at the forefront of real-time object detection frameworks, offering a potent blend of efficiency and accuracy. YOLOv5 builds upon its predecessors with a streamlined architecture, featuring a CSPDarknet53-based backbone neural network and a sophisticated detection head. The inclusion of a Cross-Stage Partial (CSP) connection in the backbone network enhances feature representation by capturing information from multiple stages. This architecture facilitates end-to-end training and empowers the model to excel in real-time object detection tasks. YOLOv5 is available in various model variants, catering to a spectrum of computational resources and application demands. Ranging from YOLOv5s (small) to YOLOv5x (extra-large), users can choose a model size that aligns with their specific requirements. Notable improvements in training techniques, such as grid-size inference, autoanchor generation, and multi-scale training, contribute to faster convergence and increased adaptability to diverse object scales. With its efficiency, speed, and applicability to a wide range of fields, YOLOv5 has become a cornerstone in object detection, empowering applications such as autonomous vehicles, surveillance systems, and industrial automation with real-time and accurate object recognition capabilities

7. Web Application Development :Create an application located on a remote server and distribute it to clients over the network. Upload images and the web application will provide output for vehicle and building detection/prediction

4.3 Task Identification

Following analysis and design tasks are to be carried out in the process of analysis and design of the project. All project modules are divided into following tasks.

- T1: Project Definition Searching
- T2: Project Definition Preparation
- T3: Literature Collection
- T4: Project Definition Finalization
- T5: Dataset Collection
- T6: Synopsis Preparation
- T7: Requirement Analysis and Validation
- T8: Determine Process Model (Incremental Model)
- T9: System Breakdown Structure
- T10: Project Estimation in KLOC
- T11: Project Scheduling and Tracking
- T12: Analysis Modelling using Behavioral, Functional and Architectural Modelling
- T13: Feasibility Management of Project using Mathematical Modelling
- T14: Risk Analysis, Project Management and Risk Management
- T15: Report Preparation
- T16: Data Preprocessing using Preprocessing Techniques
- T17: Extracting the Features from the Preprocessed Dataset
- T18: Creating Graphical User Interface
- T19: : Model Training using Scratch CNN layers and YoloV5 Pre-Trained Model.
- T20: Predicting / Detecting the vehicle and building with confidence Score

- T21: Testing of the Generated Model
- T22: Deployment of Project

4.3.1 Project Schedule

Table 4.1 describes the schedule for project development and also highlight all the task to be carried out along with their duration, dependency and developer(s) assign to accomplish the task.

Table 4.1: Project Task Table

Task	Days	Dependencies	Developer Assigned
T1	3	-	D1,D2,D3,D4
T2	5	T1	D1,D2,D3,D4
T3	3	T2	D1,D2,D3,D4
T4	6	T1, T2,T3	D1,D2,D3,D4
T5	3	T2	D1,D2,D3,D4
T6	7	T3,T4	D1,D2,D3,D4
T7	8	T6	D1,D2,D3,D4
T8	3	-	D1,D2,D3,D4
T9	4	T5	D1,D2,D3,D4
T10	2	-	D1,D2,D3,D4
T11	2	-	D1,D2,D3,D4
T12	4	T7, T8, T9	D1,D2,D3,D4
T13	5	T11	D1,D2,D3,D4
T14	6	T8	D1,D2,D3,D4
T15	12	T7,T8	D1,D2,D3,D4
T16	12	T14	D1,D2,D3,D4
T17	12	T15	D1,D2,D3,D4
T18	14	-	D1,D2,D3,D4
T19	13	T17	D1,D2,D3,D4
T20	13	T18	D1,D2,D3,D4
T21	7	T18, T19	D1,D2,D3,D4
T22	4	T15,T19,T21	D1,D2,D3,D4
Total	**148**		

4.4 Project Table and Time-line Chart

4.4.1 Project Schedule Time

Table 4.2: Project Schedule Time Table

Test ID	Exp. Start Time	Act. Start Time	Exp. End Time	Act. End Time	Developers
T1	24/08/23	24/08/23	27/08/23	27/08/23	D1,D2,D3,D4
T2	27/08/23	27/08/23	01/09/23	01/09/23	D1,D2,D3,D4
T3	02/09/23	02/09/23	05/09/23	05/09/23	D1,D2,D3,D4
T4	06/09/23	06/09/23	13/09/23	13/09/23	D1,D2,D3,D4
T5	13/09/23	13/09/23	17/09/23	17/09/23	D1,D2,D3,D4
T6	17/09/23	17/09/23	24/09/23	24/09/23	D1,D2,D3,D4
T7	25/09/23	25/09/23	04/10/23	04/10/23	D1,D2,D3,D4
T8	05/10/23	05/10/23	09/10/23	09/10/23	D1,D2,D3,D4
T9	10/10/23	10/10/23	14/10/23	14/10/23	D1,D2,D3,D4
T10	15/10/23	15/10/23	17/10/23	17/10/23	D1,D2,D3,D4
T11	18/10/23	18/10/23	21/10/23	21/10/23	D1,D2,D3,D4
T12	21/10/23	21/10/23	25/10/23	25/10/23	D1,D2,D3,D4
T13	25/10/23	25/10/23	31/10/23	31/10/23	D1,D2,D3,D4
T14	01/11/23	01/11/23	07/11/23	07/11/23	D1,D2,D3,D4
T15	07/11/23	07/11/23	19/11/23	19/11/23	D1,D2,D3,D4
T16	19/11/23	19/11/23	01/12/23	01/12/23	D1,D2,D3,D4
T17	01/12/23	01/12/23	13/02/23	13/12/23	D1,D2,D3,D4
T18	13/12/23	13/12/23	14/03/23	14/12/23	D1,D2,D3,D4
T19	14/12/23	14/12/23	15/12/23	15/12/23	D1,D2,D3,D4
T20	15/12/23	15/12/23	16/12/23	16/12/23	D1,D2,D3,D4
T21	16/012/23	16/12/23	17/12/23	17/12/23	D1,D2,D3,D4
T22	17/12/23	17/12/23	18/12/23	18/12/23	D1,D2,D3,D4

4.4.2 Time-Line Chart

Timeline chart shows the progress of project development in various phases.

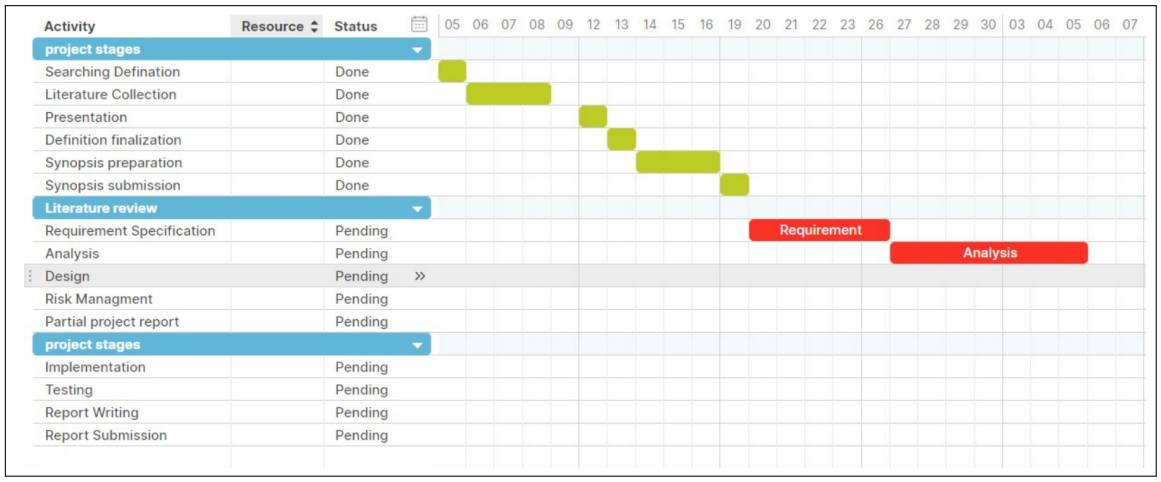

Figure 4.3: Expected Timeline Chart 1

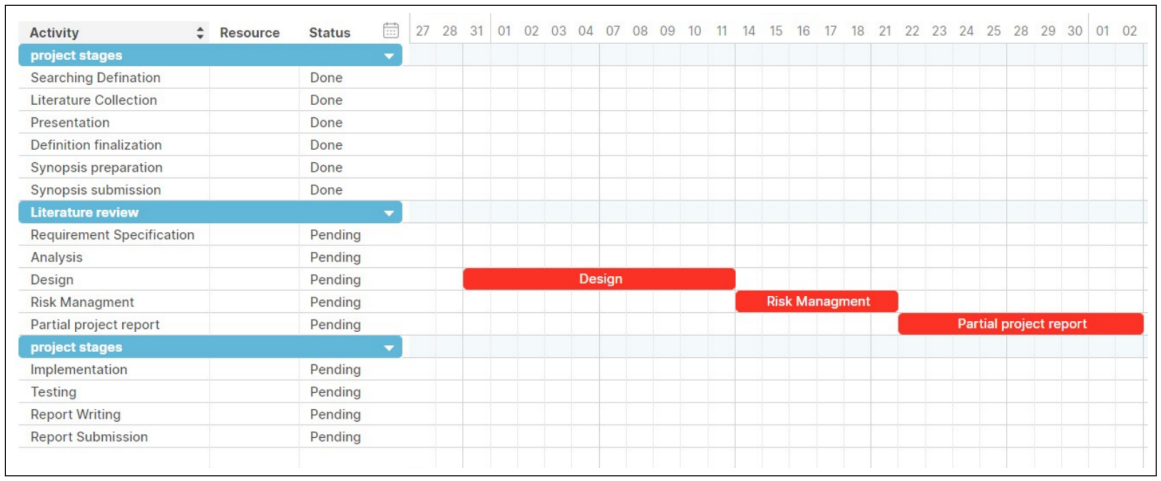

Figure 4.4: Expected Timeline Chart 2

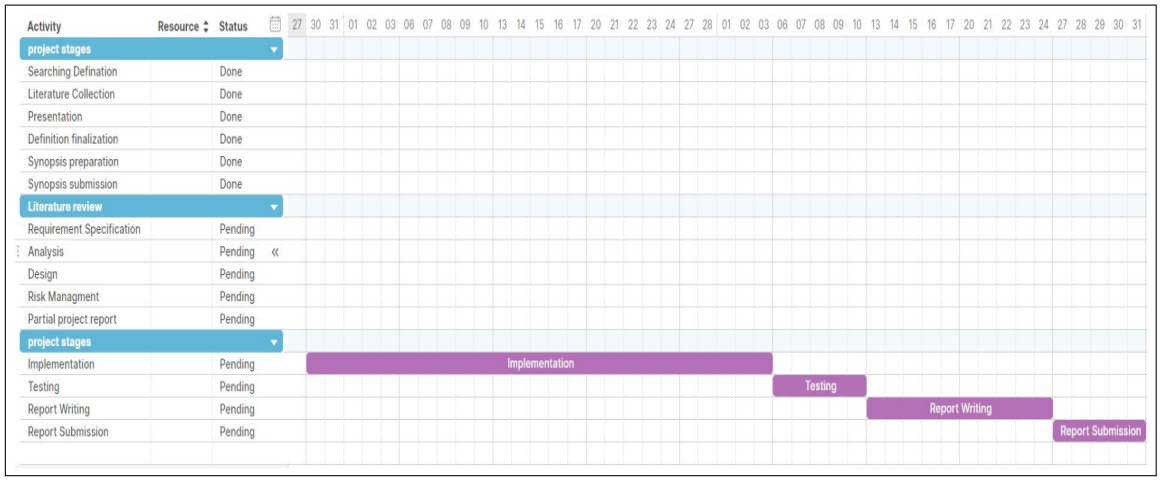

Figure 4.5: Expected Timeline Chart 3

4.5 Analysis Modeling

The system analysis model is made up of class diagram, sequence or collaboration diagrams and state chart diagrams. Between them they constitute a logical, implementation free view of computer system that includes a detail definition of every aspect of functionality. Analysis model contains following modeling:

1. Behavioral modeling.
2. Functional modeling.
3. Architectural modeling.

Analysis modeling uses a combination of text and diagrammatic form to depict requirement for data, function and behaviour in a way that is relatively easy to understand and more important, straightforward to review for correctness, completeness and consistency.

4.5.1 Behavioral modeling

Use case diagram

A use case involves a sequence of interactions between the motivator and the system, possibly including other actors.

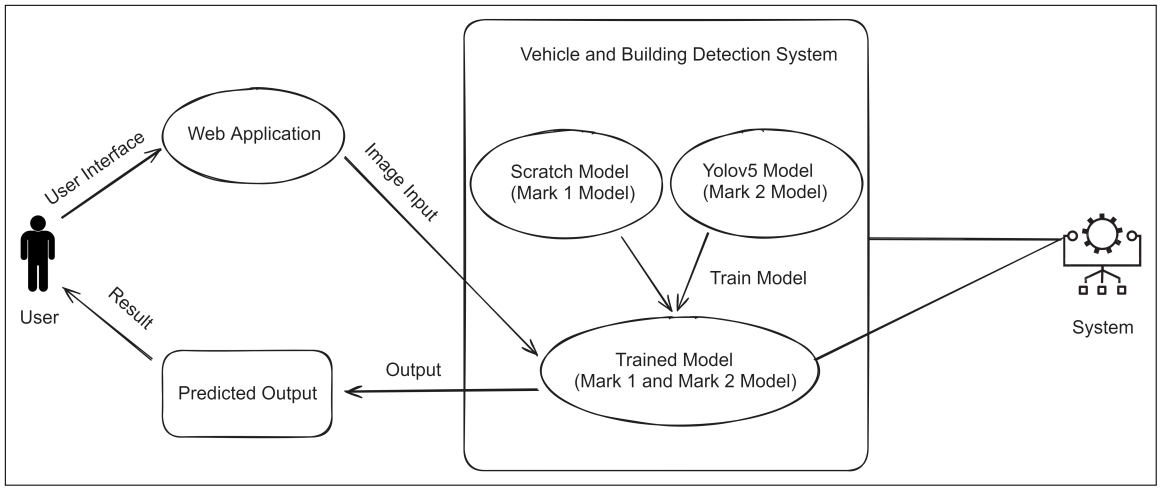

Figure 4.6: Use Case Diagram

Sequence Diagram

A sequence diagram is a graphical view of a state that shows object interface in a time-based sequence.

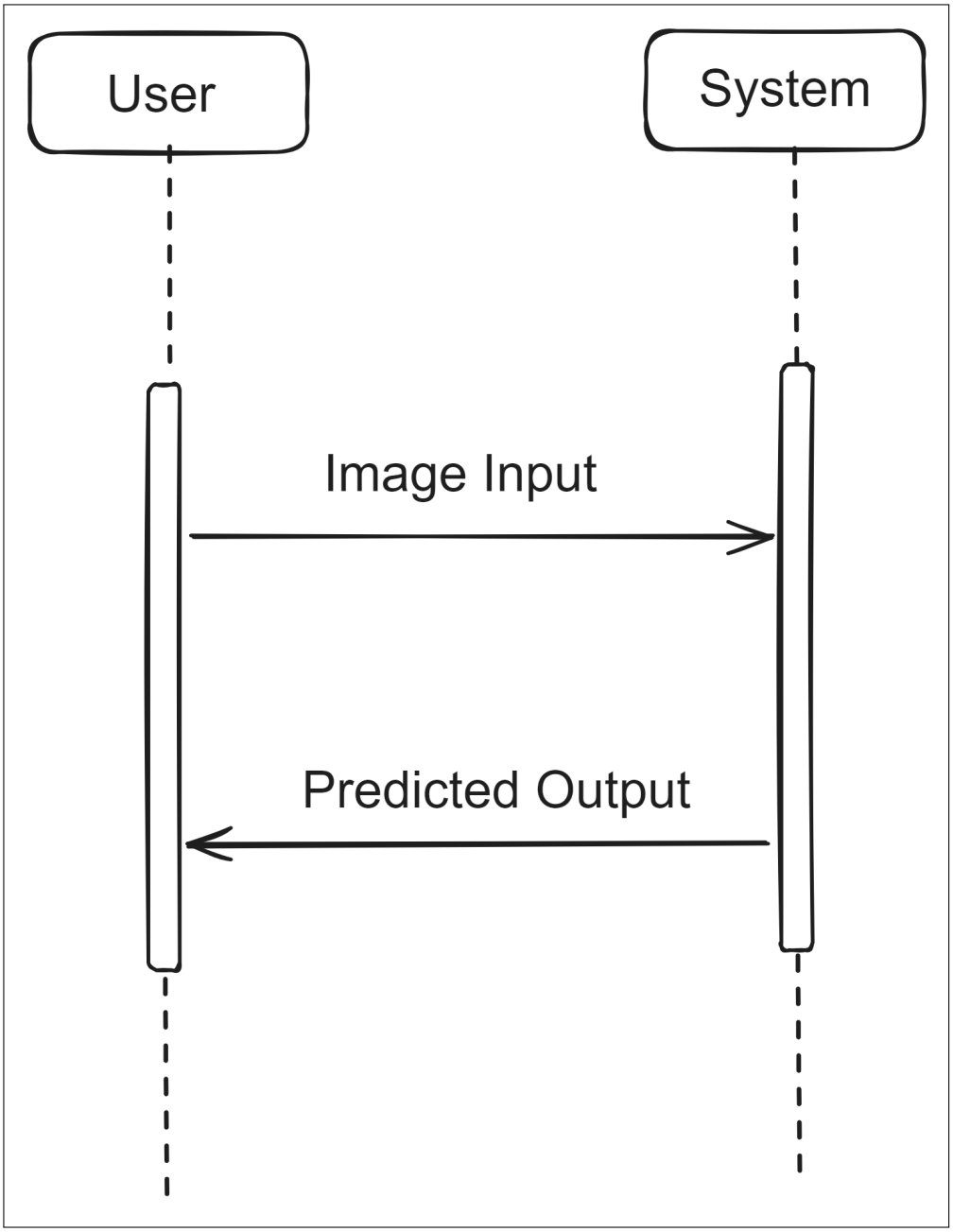

Figure 4.7: Sequence Diagram

Class Diagram

A Class is an Explanation of a set of objects that have the same attributes and methods Attribute.

- Attribute
- Operation
- Relationship

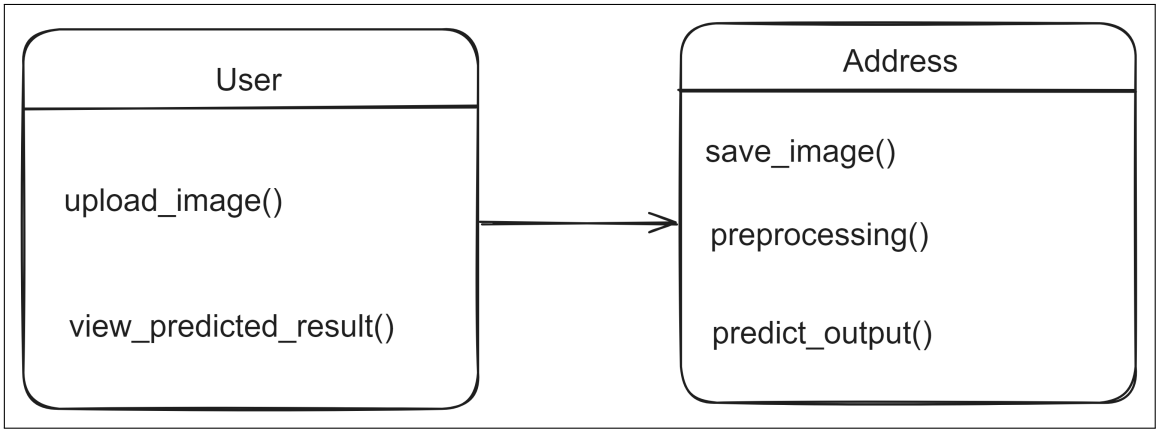

Figure 4.8: Class Diagram

Activity Diagram

Activity diagram is an important diagram to describe the dynamic aspects of the system. Activity diagram is essentially a flowchart to represent the flow from one activity to another activity.

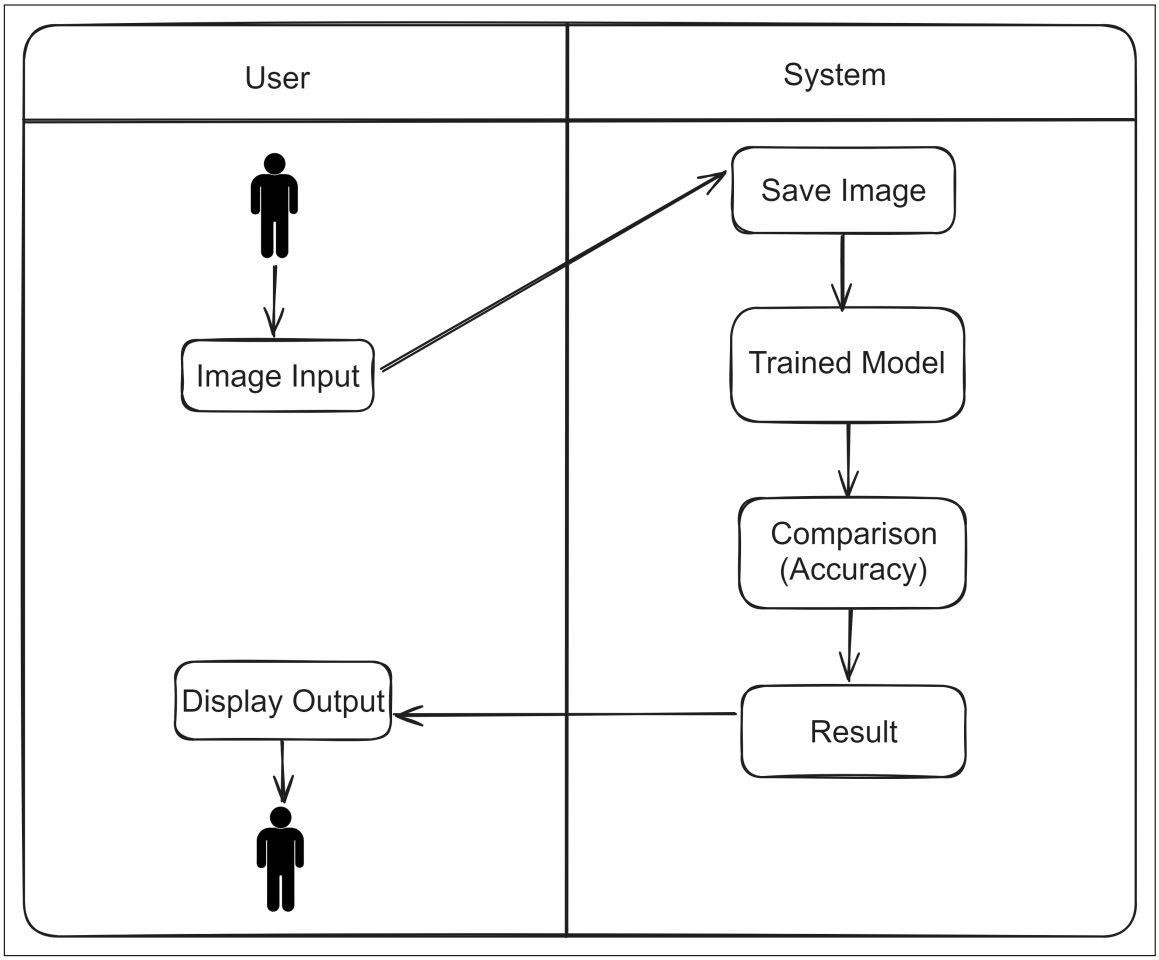

Figure 4.9: Activity Diagram

State Chart Diagram

State chart Diagram shows the state machine that consists of states, transitions, events and activities. This diagram shows how the system makes the transition from one state to another state on occurrence of a particular event.

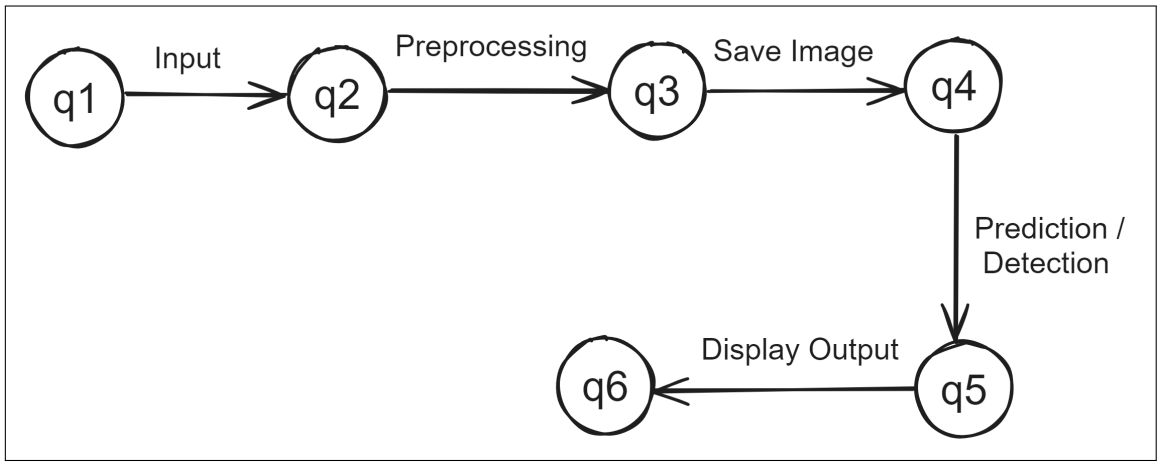

Figure 4.10: State Chart Diagram

4.5.2 Functional Modeling

Data Flow Diagram

Data Flow Diagram: Data flow diagram is also called Bubble Chart is a graphical technique, which is used to represent information flow, and converters are applied when data moves from input to output. DFD represents system requirements clearly and identifies modifiers that become programs in design. DFD may be further separated into different levels to show detailed information flow.

- DFD - 0: The data flow diagram 0 is the abstract diagram of the proposed system that consists of only input, main method and output.

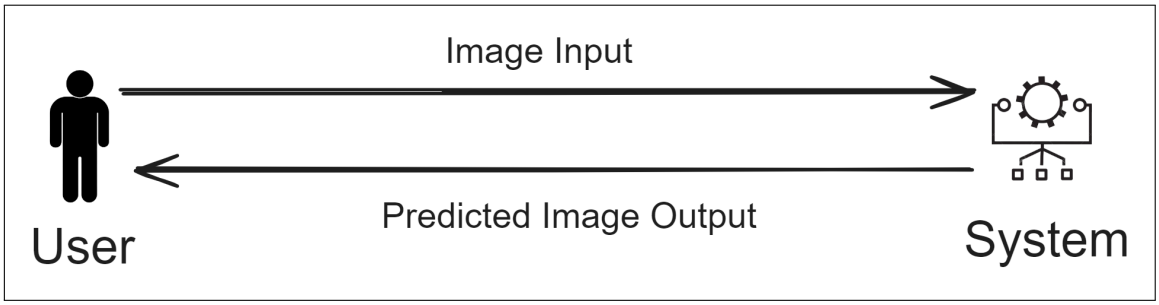

Figure 4.11: Data Flow Diagram level 0

- DFD - 1: The data flow diagram 1 contains some additional methods in the system including inputs, methods or modules and outputs as shown in below figure.

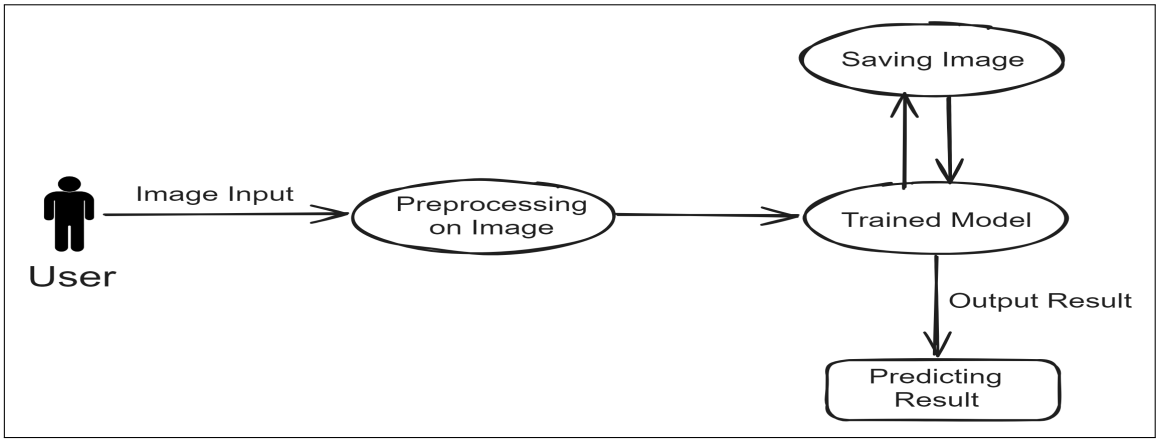

Figure 4.12: Data Flow Diagram level 1

- DFD - 2: The data flow diagram 2 contains some additional methods in the system.

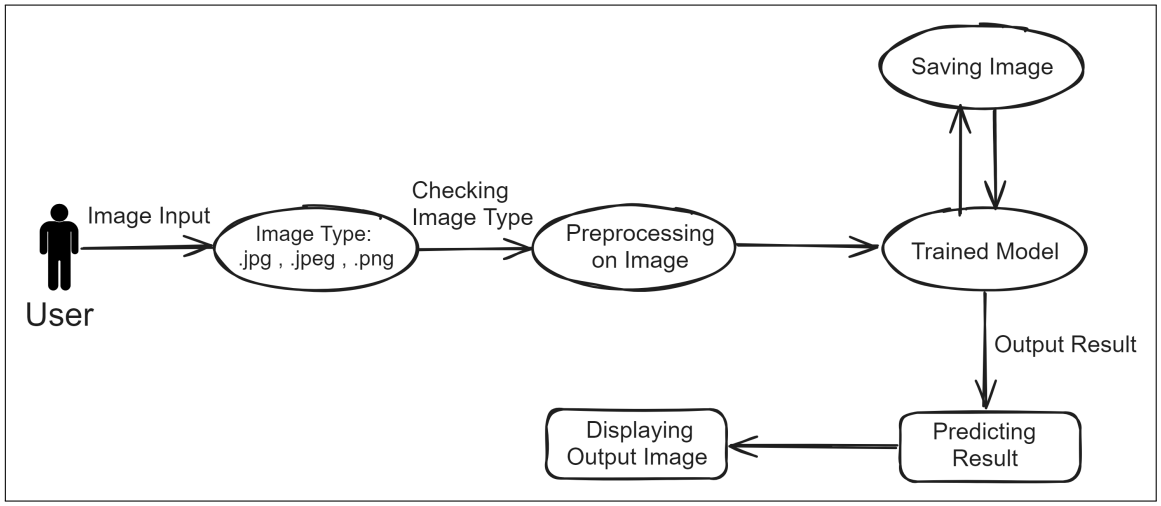

Figure 4.13: Data Flow Diagram level 2

Control Flow Diagram

The large class of applications having following characteristics requires control flow modeling. The applications that are driven by events rather than data. The applications that produce control flow material rather than reports or Displays.

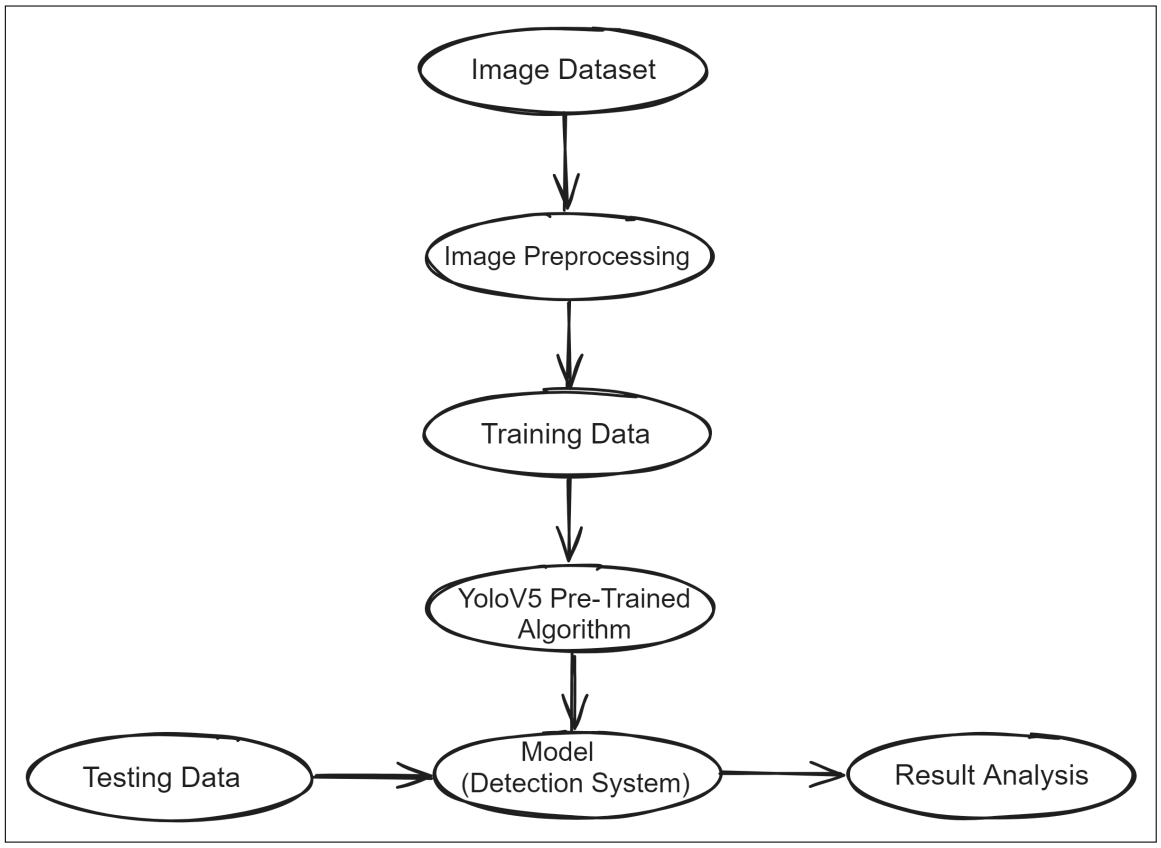

Figure 4.14: Control Flow Diagram

4.5.3 Architectural Modeling

Component Diagram

A Component diagram shows a set of components with their relationships. The component diagrams are useful to represent the static implementation view of a system.

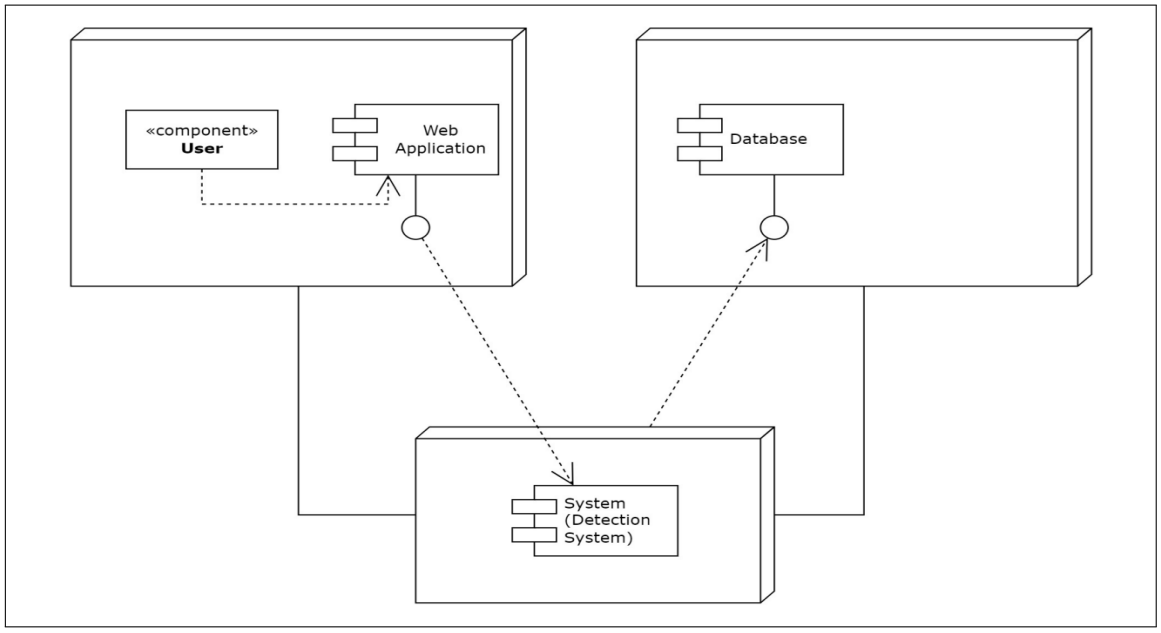

Figure 4.15: Component Diagram

Deployment Diagram

Deployment figures are used to visualize the topology of the physical components of a system where the software components are organized. Deployment diagrams are used for describing the hardware components where software components are deployed.

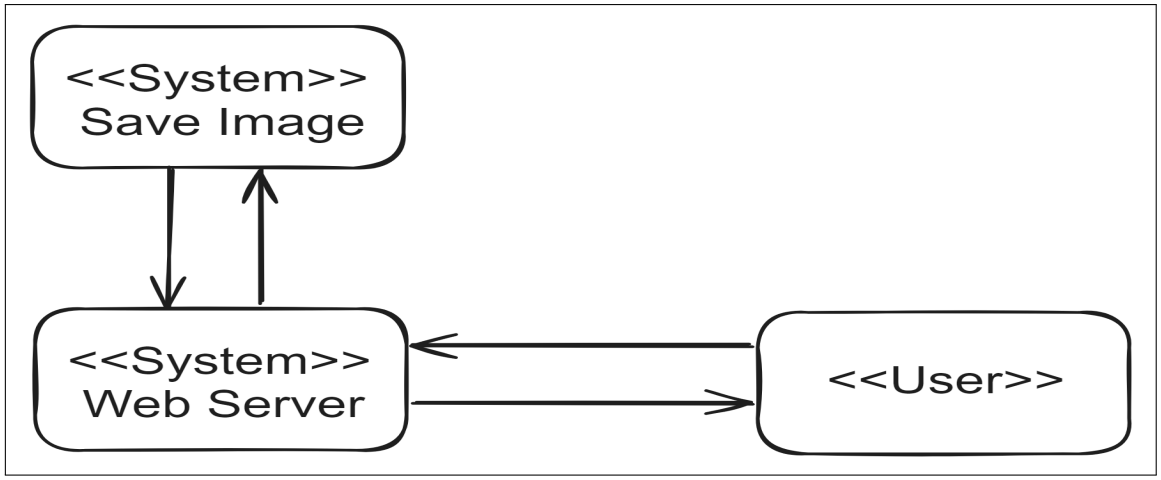

Figure 4.16: Deployment Diagram

4.6 Mathematical Modeling

NP-Hard Problem

These problems need not have any certainty on their running time. Intuitively, these are the problems that are even harder than the NP- complete problems Note that NP-hard problems do not have to be in NP, and they do not have to be decision problems. The precise definition here is that a problem X is NP- hard, if there is an NP-complete problem Y, such that Y is reducible to X in polynomial time. But since any NP- complete problem can be reduced to any other NP-complete problem in polynomial time, all NP-complete problems can be reduced to any NP-hard problem in polynomial time. Then, if there is a solution to one NP-hard problem in polynomial time, there is a solution to all NP problems in polynomial time. From the above theory we can conclude that our problem definition comes under the 'P' class.

Set Theory

Set theory is a branch of calculated logic that studies set, which informally are collections of objects. Although any type of object can be composed into a set, set theory is applied most often to objects that are appropriate to mathematics. The language of set theory can be used in the meanings of nearly all mathematical objects.

System representation is given as,

$$S = \{I, P, R, O\}$$

Where,
$\{I\}$ is set of all inputs given to system.
$\{P\}$ is set of all processes in system.
$\{R\}$ is set of rules that drives your input set.
$\{O\}$ is set of output expected from system.
Input $\{I\}$ is set of inputs giving to system.

$$I = \{I1\}$$

Where,
I1 = Required Image Input.

Process

$$P = \{P1, P2, P3, P4\}$$

Where,

P1 = Taking Image input from user.

P2 = Storing/Saving Image.

P3 = Predicting/Detecting using DL Model.

P4 = Displaying output/result Image to User.

Rules

$$R = \{R1\}$$

Where,

R1 = User must provide Image in .jpg , .jpeg , .png format only.

Output

$$O = \{O1\}$$

Where,

O1 = Provide generated result/output to the user..

4.6.1 Venn Diagram

A diagram representing mathematical or logical sets pictorially as circles or closed curves within an enclosing rectangle (the universal set), common elements of the sets being represented by intersections of the circles.

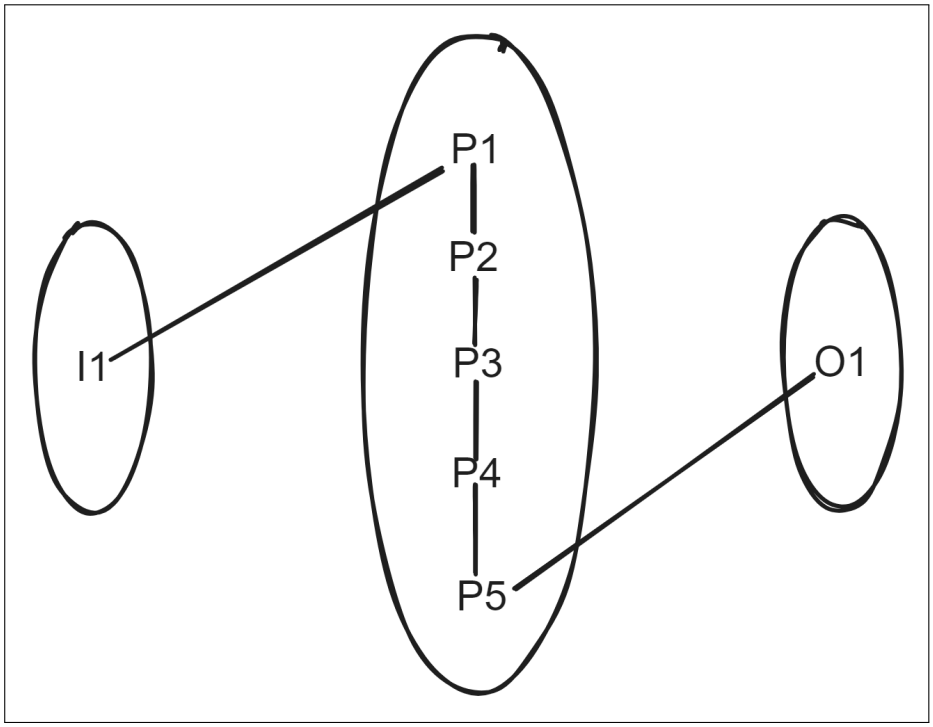

Figure 4.17: Venn Diagram

Chapter 5

RISK MANAGEMENT

Project risk management is the process of identifying, analyzing and then responding to any risk that arises over the life cycle of a project to help the project remain on track and meet its goal. Managing risk isn't reactive only, it should be part of the planning process to figure out risk that might happen in the project and how to control that risk if it in fact occurs. A risk is anything that could potentially impact your projects timeline, performance or budget. Risks are potentialities, and in a project management context, if they become realities, they then become classified as issues that must be addressed. So risk management, then, is the process of identifying, categorizing, prioritizing and planning for risks before they become issues .This article gives us ten golden rules to apply risk management successfully in our project.

5.1 Risk Identification

- **Product size risk**

  R1: Is the development team able to decompose the required program into smaller, highly cohesive modules.

  R2: Are there enough development team members available relative to the size of the project.

- **Business impact**

  R3: Delay in project delivery (violation in time constraints) can hamper the customer economically.

  R4: If system is not more efficient than the existing system, it will cause economic losses.

- **Customer related risk**

  R5: User or service provider is a non-technical person; if proper guide- lines were not mentioned then it will create ambiguity.

  R6: If a user wants any modifications that lead to changing the entire System.

- **Process risk** R7: The risk of technology errors or security incidents that disrupt or invalid processes.

  R8: Quality of a process itself that leads to failures. A low quality process may not properly work and may break down the system.

- **Technical Risk**

  R9: Lack of database stability and concurrency.

  R10: Module integration fails.

  R11: Wrong trained data may lead to wrong results.

- **Development Environment Related risk**

  R12: Lack of proper training and less knowledge of programming leads to a moderate risk. It will delay product development and deployment.

5.2 Strategies Used to Manage Risk

- S1: Formulation and follow up of the project plan on a regular basis.
- S2: Keep assigned work under certain deadlines.
- S3: Web Development cycle should be used.
- S4: Regular meetings with users reduce the risk to some extent, design systems with flexibility and maintain necessary documentation for the same.
- S5: Re-defined software process at higher degree.
- S6: Proper training on required technical tools for development of projects reduces risk.
- S7: Make certain rules that each one the members are taking part in the design.

- S8: Study and understanding of project definition, programming language.
- S9: Take a look at model development and all its associated used software.
- S10: Time constraints must be followed to avoid economical risks.
- S11: Each and every module must be tested for its functioning.
- S12: After unit testing, the system must be integrated and validated accordingly.
- S13: Integration testing for authentication hierarchy.
- S14: Use of standard database technology which supports concurrency more.

5.3 Risk Projection

5.3.1 Preparing Risk Table

The risk table shown below lists all possible risks which may occur at any stage during development of a project. Table also clearly shows the impact of the risks and RMMM (Risk Mitigation Monitoring and Management) planto deal with any such risks.

Table 5.1: Risk Management Table

Risk	Category	Probability	Impact	Plan
R1	Product Size Risk	More	High	S1,S3,S4
R2	Product Size Risk	Less	Less	S4,S6,S1
R3	Business Impact Risk	More	High	S2
R4	Business Impact Risk	More	High	S4
R5	Customer Related Risk	More	High	S11,S12,S13
R6	Customer Related Risk	More	Less	S11,S12,S13
R7	Process Risk	More	High	S14
R8	Process Risk	More	High	S14
R9	Technical Risk	Less	High	S1,S6
R10	Technical Risk	More	High	S7
R11	Development Environment Related Risk	Less	Less	S4,S5,S7,S8
R12	Product Size Risk	More	High	S9,S10

5.4 Feasibility

Feasibility is defined as an evaluation or analysis of the potential impact of a proposed project.

- Technical feasibility: - It is disturbed with specifying equipment and software that will successfully preserve the task required.
- SAT (Satisfiability): - Boolean formula is satisfiability if there exists at least one way of assigning value to its variable so as to make it true and we denote it by using SAT. The problem of deciding whether a given formula is satisfiability or not.
- Facility to produce output in given times.
- Response time under certain conditions.
- Operational Feasibility: -It is related to human organization.
- What changes will be brought in with the system.
- How organizational Structure will be distributed.
- What new skills are required.
- Economic Feasibility: -It is the most frequently used technique for evaluating the effectiveness of proposed systems. Most usually identified as cost/benefit analysis.

Chapter 6

TECHNICAL SPECIFICATION

6.1 Software requirement specification

6.1.1 Operating System (OS)

- Windows 10 or higher.

6.1.2 Integrated Development Environment (IDE)

- Visual Studio Code (VS Code), Jupyter Notebook, Google Colab any equivalent.

6.1.3 Programming Language

- Python 3.X

6.2 Hardware Requirement Specifications

The algorithms used in the project are computationally intensive and thus require high computing capabilities in terms of hardware. For the testing and demo purpose we will need good hardware usually found in desktop type of computers. Thus, to reduce the computation time of the project we recommend to use high performance system.

- Hard Disk: 256 GB
- Ram: 8GB
- Processor: Intel core i5 8th Gen
- Nvidia Graphics card (Min. 2 GB and Supporting CUDA)

Chapter 7

IMPLEMENTATION DETAILS

In this chapter implementation details are explained as follow:

Main Code of Mark-1 Model

```
import os
import numpy as np
import cv2

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

import PIL
from PIL import Image

import torch
import torchvision
import torch.nn as nn
from torch.utils.data import DataLoader
import torch.nn.functional as F
from torchvision import transforms
from torch.optim import Adam
from torch.autograd import Variable

from tqdm import tqdm
```

```
import glob


train_path = 'F:/Mayur/DL/Dataset'
test_path = 'F:/Mayur/DL/Test-Dataset'

class_name = os.listdir(train_path)
print(class_name)

def get_count(each_class):
    return len(os.listdir(train_path + '/' + each_class))

for each_class in class_name:
    print("Number of samples in {} category {}".format(each_class,
        ↪ get_count(each_class)))


directory=os.listdir(train_path)
for each in directory:
    plt.figure(figsize=(10, 10))
    currentFolder = train_path + '/' + each
    for i, file in enumerate(os.listdir(currentFolder)[0:5]):
        fullpath = train_path + '/' + each + "/" + file
        img=mpimg.imread(fullpath)
        ax=plt.subplot(1,5,i+1)
        ax.set_title(each)
        plt.imshow(img)


device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
device

train_transformer = transforms.Compose([transforms.Resize((150,150)),
```

```
                                transforms.RandomHorizontalFlip(),
                                transforms.ToTensor()])
test_transformer = transforms.Compose([transforms.Resize((150,150)),
                                transforms.ToTensor()])

train_loader = DataLoader(torchvision.datasets.ImageFolder(train_path,
    ↪ transform=train_transformer),batch_size=10,shuffle=True)
test_loader = DataLoader(torchvision.datasets.ImageFolder(test_path,
    ↪ transform=test_transformer),batch_size=10,shuffle=True)


class NeuralNet(nn.Module):
    def __init__(self,num_classes):
        super(NeuralNet,self).__init__()

        self.conv1 = nn.Conv2d(in_channels=3,out_channels=12,kernel_size
            ↪ =3,stride=1,padding=1)
        self.relu1 = nn.ReLU()
        self.pool = nn.MaxPool2d(kernel_size=2)

        self.conv2 = nn.Conv2d(in_channels=12,out_channels=20,kernel_size
            ↪ =3,stride=1,padding=1)
        self.relu2 = nn.ReLU()

        self.conv3 = nn.Conv2d(in_channels=20,out_channels=32,kernel_size
            ↪ =3,stride=1,padding=1)
        self.relu3 = nn.ReLU()

        self.fc = nn.Linear(in_features=32*75*75,out_features=num_classes
            ↪ )

    def forward(self,input):
        output = self.conv1(input)
```

```
        output = self.relu1(output)

        output = self.pool(output)

        output = self.conv2(output)
        output = self.relu2(output)

        output = self.conv3(output)
        output = self.relu3(output)

        output = output.view(-1,32*75*75)
        output = self.fc(output)

        return output


model = NeuralNet(num_classes=4).to(device)

optimizer = Adam(model.parameters(),lr=0.001,weight_decay=0.0001)
loss_function = nn.CrossEntropyLoss()

num_epochs = 10

train_count = len(glob.glob(train_path+'/**/*.jpg'))
test_count = len(glob.glob(test_path+'/**/*.jpg'))

print(train_count,test_count)

best_accuracy = 0.0
for epoch in range(num_epochs):
    model.train()
    train_accuracy = 0.0
    train_loss = 0.0
```

```
for i ,(images,labels) in enumerate(train_loader):
    if torch.cuda.is_available():
        images = Variable(images.cuda())
        labels = Variable(labels.cuda())

    optimizer.zero_grad()

    outputs = model(images)
    loss = loss_function(outputs,labels)
    loss.backward()
    optimizer.step()

    train_loss += loss.cpu().data*images.size(0)
    _,prediction = torch.max(outputs.data,1)
    train_accuracy += int(torch.sum(prediction==labels.data))

train_accuracy = train_accuracy/train_count
train_loss = train_loss/train_count

model.eval()

test_accuracy = 0.0
for i ,(images,labels) in enumerate(test_loader):
    if torch.cuda.is_available():
        images = Variable(images.cuda())
        labels = Variable(labels.cuda())

    outputs = model(images)
    _,prediction = torch.max(outputs.data,1)
    test_accuracy += int(torch.sum(prediction==labels.data))

test_accuracy = test_accuracy/test_count
```

```
    print('Epoch: '+str(epoch)+'Train Loss : '+str(int(train_loss))+'
        ↪ Train Accuracy : '+str(train_accuracy)+' Test Accuracy : '+str
        ↪ (test_accuracy))

    if test_accuracy>best_accuracy:
        torch.save(model.state_dict(),'best_checkpoint.model')
        best_accuracy = test_accuracy

net = NeuralNet(num_classes=4)
net.load_state_dict(torch.load('F:/Mayur/DL/best_checkpoint.model'))


classess = ['2-wheeler', '4-wheeler', 'Bicycle', 'Buildings']
classess

loader = transforms.Compose([transforms.Resize((150,150)),
                             transforms.ToTensor()])

def image_loader(image_name):
    image = Image.open(image_name)
    image = loader(image).float()
    image = Variable(image, requires_grad=True)
    image = image.unsqueeze(0)
    return image

imgpath = 'F:/Mayur/DL/t.jpg'
image = image_loader(imgpath)


prediction = net(image).detach().numpy()[0]
print(prediction)
index = np.where(prediction>=prediction.max())[0][0]
```

```
print("Predicted Class: ",classess[index])


plt.figure(figsize=(8, 8))
img=mpimg.imread(imgpath)
ax=plt.subplot(1,1,1)
ax.set_title(classess[index])
plt.imshow(img)
```

Main Code of Mark-2 Model

```
import torch
import cv2
import numpy as np
import matplotlib.pyplot as plt

!cd yolov5 && python train.py --img 640 --batch 1 --epochs 50 --data
    ↪ dataset.yaml --weights yolov5s.pt --workers 2

model = torch.hub.load('ultralytics/yolov5','custom',path='C:/Users/
    ↪ Admin/Desktop/Final Year/YoloV5 Model/yolov5/runs/train/exp14/
    ↪ weights/best.pt',force_reload=True)

model

img = 'test.jpg'
result = model(img)
result.print()
%matplotlib inline
plt.imshow(np.squeeze(result.render()))
plt.show()
```

Chapter 8

APPLICATIONS OF THE PROJECT

- **Smart Parking Systems:** Vehicle detection technology is used in smart parking systems to monitor parking space occupancy. This helps drivers find available parking spots quickly, reducing traffic congestion and enhancing the overall parking experience in urban areas.
- **Surveillance and Security:** Building detection is crucial for surveillance and security applications. It allows us to monitoring of public spaces, commercial areas, and critical infrastructure, helping to identify potential security threats and ensure public safety.
- **Traffic Management and Control:** Vehicle detection is integral to traffic management systems, optimizing traffic flow at intersections and along roadways. It enables adaptive traffic signal control, reducing congestion and improving overall transportation efficiency in urban environments.
- **Industrial Site Monitoring:** In industrial settings, vehicle and building detection is employed for monitoring and securing facilities. This ensures compliance with safety regulations, tracks the movement of vehicles within the site, and enhances overall security.
- **Real Estate and Property Management:** Building detection is applied in real estate for property management and assessment. It aids in the identification and classification of various types of buildings, supporting property valuation and development planning.
- **Transportation and Logistics:** Vehicle detection plays a crucial role in transportation and logistics, enabling the tracking and monitoring of vehicles throughout the supply chain. This enhances efficiency in logistics operations, including inventory management and shipment tracking.

Chapter 9

CONCLUSION & FUTURE SCOPE

9.1 Conclusion

This system has successfully tackled the challenge of real-time detection and classification of vehicles and buildings using advanced deep learning methodologies. Through the creation of a diverse and annotated dataset, the development of an optimized convolutional neural network architecture, and the application of transfer learning with models like YOLOv5, a robust system has been established for precise object identification. The user-friendly interface facilitates easy image uploads and delivers insightful results, making the technology accessible for various applications beyond the agricultural sector. The comprehensive evaluation metrics validate the system's reliability and efficacy in practical scenarios, showcasing its potential impact in diverse domains.

9.2 Future Scope

1. **Semantic Segmentation:** Extend the project to incorporate semantic segmentation techniques for a more detailed understanding of spatial layouts within images.
2. **Multi-Object Tracking:** Explore the integration of multi-object tracking algorithms to monitor the movement and interactions of vehicles and buildings over time, providing a dynamic perspective.
3. **Edge Computing Implementation:** Investigate deploying the model on edge devices for on-site processing, reducing reliance on centralized computing resources and improving real-time performance.
4. **Continuous Model Improvement:** Implement mechanisms for continuous learning,

allowing the model to adapt and improve over time with additional annotated data and user feedback.

5. **Expansion to Other Domains:** Extend the application to domains such as urban planning, traffic management, and disaster response by incorporating additional object categories and refining the model accordingly.

6. **Integration with GIS Technology:** Incorporate geographic information system (GIS) technology for spatial analysis, enabling users to visualize the distribution of identified vehicles and buildings on maps.

7. **Accessibility Features:** Enhance the user interface with accessibility features, ensuring the system is inclusive and user-friendly for individuals with diverse needs and expertise levels.

8. **Collaboration with Domain Experts:**Collaborate with experts in specific domains to tailor the system to their needs, incorporating domain-specific knowledge and ensuring practical relevance in various contexts.

References

[1] SurSingh Rawat; Jyoti Gautam; Sukhendra Singh; Vimal Gupta; Gynendra Kumar; Lal Pratap Verma, "Car Detection and Recognition using Deep Learning Techniques," in 2021 4th International Symposium on Advanced Electrical and Communication Technologies (ISAECT)

[2] Daniel Widjojo; Endang Setyati; Yosi Kristian, "Integrated Deep Learning System for Car Damage Detection and Classification Using Deep Transfer Learning," in 2022 IEEE 8th Information Technology International Seminar (ITIS)

[3] "Deep Learning Techniques for Vehicle Detection and Classification from Images/Videos: A Survey" by Michael Abebe Berwo et al., published in Sensors, 2023

[4] "Vision-based vehicle detection and counting system using deep learning in highway scenes" by Huansheng Song et al., published in European Transport Research Review, 2019

[5] "An Effective Approach of Vehicle Detection Using Deep Learning" by Yidan Chen and Zhenjin Li, published in Computational Intelligence and Neuroscience, 2022

[6] "Road object detection: a comparative study of deep learning-based algorithms" published in Multimedia Tools and Applications, 2022

[7] "Vehicle Speed Estimation and Tracking Using Deep Learning" published in Springer, 2022