International Research Journal of Education and Technology

Peer Reviewed Journal, **ISSN 2581-7795**

ROAD OBJECT DETECTION IN FOGGY COMPLEX SCENES ON IMPROVED YOLOv10

BACHELOR OF TECHNOLOGY
in
COMPUTER SCIENCE & ENGINEERING – CYBER SECURITY

Submitted by

K MANEESHA	**21WJ1A6221**
CHAKALI NARESH	**22WJ5A6201**
CHILUKA GANGA	**22WJ5A6203**
NUKAM DEEPTHI	**22WJ5A6206**

Under the Guidance of

Mr.Ch.Upendar.,M.Tech.,(Ph.D)

Associate Professor in CSE (Cyber Security)

Department of CSE – Cyber Security

School of Engineering and Technology

GURU NANAK INSTITUTIONS TECHNICAL CAMPUS

Ibrahimpatnam, Hyderabad, R.R. District – 501506

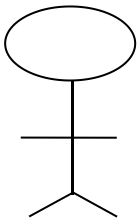
2024 - 2025

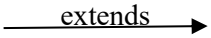
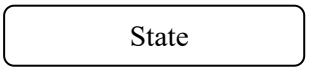
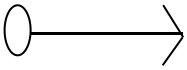
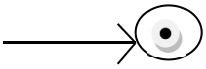
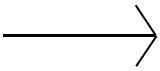
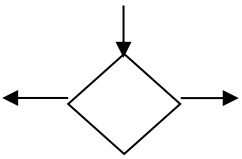
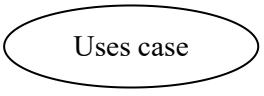
ABSTRACT

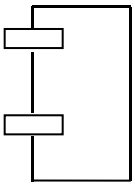
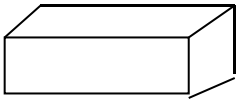
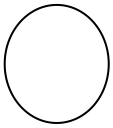
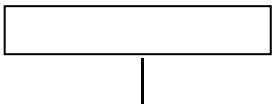
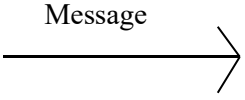
Foggy weather presents substantial challenges for vehicle detection systems due to reduced visibility and the obscured appearance of objects. To overcome these challenges, a novel vehicle and Humans detection algorithm based on an improved lightweight YOLOv10 model is introduced. The proposed algorithm leverages advanced preprocessing techniques, including data transformations, Dehaze Formers, and dark channel methods, to improve image quality and visibility. These preprocessing steps effectively reduce the impact of haze and low contrast, enabling the model to focus on meaningful features. An enhanced attention module is incorporated into the architecture to improve feature prioritization by capturing long-range dependencies and contextual information. This ensures that the model emphasizes relevant spatial and channel features, crucial for detecting small or partially visible vehicles in foggy scenes. Furthermore, the feature extraction process has been optimized, integrating an advanced lightweight module that improves the balance between computational efficiency and detection performance. This research addresses critical issues in adverse weather conditions, providing a robust framework for foggy weather vehicle and Humans detection.

LIST OF FIGURES

LIST OF SYMBOLS

S.NO	NOTATION NAME	NOTATION	DESCRIPTION
1.	Class	*Class Name* *-attribute* *-attribute* *+operation* *+operation* *+operation* + *public* - *private* # *protected*	Represents a collection of similar entities grouped together.
2.	Association	Class A — NAME — Class B Class A — Class B	Associations represents static relationships between classes. Roles represents the way the two classes see each other.
3.	Actor		It aggregates several classes into a single classes.
4.	Aggregation	Class A ↑ Class B Class A ↑ Class B	Interaction between the system and external environment

5.	Relation (uses)	uses	Used for additional process communication.
6.	Relation (extends)		Extends relationship is used when one use case is similar to another use case but does a bit more.
7.	Communication		Communication between various use cases.
8.	State		State of the processes.
9.	Initial State		Initial state of the object
10.	Final state		Final state of the object
11.	Control flow		Represents various control flow between the states.
12.	Decision box		Represents decision making process from a constraint.
13.	Use case		Interact ion between the system and external environment.

14.	Component		Represents physical modules which are a collection of components.
15.	Node		Represents physical modules which are a collection of components.
16.	Data Process/State		A circle in DFD represents a state or process which has been triggered due to some event or action.
17.	External entity		Represents external entities such as keyboard, sensors, etc.
18.	Transition		Represents communication that occurs between processes.
19.	Object Lifeline		Represents the vertical dimensions that the object communications.
20.	Message	Message 	Represents the message exchanged.

CHAPTER - 1

INTRODUCTION

1.1 GENERAL

Foggy weather significantly hinders the performance of vehicle detection systems, as fog reduces visibility, obscures object details, and creates low-contrast conditions. Accurate detection of vehicles and pedestrians in such weather conditions is critical for improving the safety and reliability of autonomous driving systems. Traditional detection methods often fail to deliver satisfactory results under foggy conditions due to the blurring and reduction in contrast that fog causes. In this paper, we propose a novel vehicle and human detection algorithm based on an improved lightweight YOLOv10 model designed to overcome these challenges. By incorporating advanced preprocessing techniques such as Dehaze Formers and dark channel methods, we aim to enhance image quality and restore visibility, enabling the model to focus on critical features despite the fog. Furthermore, an enhanced attention module is introduced to prioritize important spatial and channel features, ensuring better performance in detecting small or partially visible objects, which are common in foggy scenes. Our approach leverages the power of YOLOv10, optimized for both computational efficiency and high detection accuracy, to deliver reliable results even in the most challenging conditions.

1.2 SCOPE OF THE PROJECT

The scope of this research is focused on improving object detection in foggy road conditions, specifically targeting vehicles and humans. By integrating state-of-the-art preprocessing techniques with an advanced YOLOv10 model, the project aims to develop a robust detection system suitable for real-world autonomous driving applications. The method is designed to operate effectively in environments where visibility is compromised due to adverse weather conditions, including fog. This research not only targets improving detection performance in foggy conditions but also strives to enhance the general capabilities of YOLOv10 for use in similar low-visibility scenarios, making it adaptable to other weather-related challenges, such as rain and snow.

1.3 OBJECTIVE

- Improved Image Preprocessing: To enhance the quality of foggy images using techniques like Dehaze Formers and dark channel methods, thereby improving visibility and contrast.
- Feature Enhancement: To integrate an attention module that improves the model's focus on relevant spatial and channel features, ensuring accurate detection of partially visible or small objects.
- Computational Efficiency: To optimize the YOLOv10 architecture for real-time performance, balancing detection accuracy with computational resources to ensure fast and efficient processing.
- Robust Detection in Foggy Conditions: To address the unique challenges posed by fog, ensuring that the system can reliably detect vehicles and humans in low-visibility environments.

1.4 EXISTING SYSTEM:

- YOLOv8 is an evolution of the YOLO object detection family, continuing its core design of a single-stage architecture that simultaneously predicts both object classes and bounding boxes. It is an end-to-end deep learning model that uses convolutional layers for feature extraction, followed by a series of dense layers for classification and localization.
- YOLOv8 builds on the improvements from previous versions by optimizing the backbone architecture, enhancing the detection head, and introducing various techniques to reduce computational costs while improving detection accuracy. The model has become widely used in real-time applications due to its efficiency and ability to handle high-speed processing tasks.

1.4.1 EXISTING SYSTEM DISADVANTAGES:

- Despite improvements, YOLOv8 may still struggle with detecting small objects, especially when they are surrounded by large, complex backgrounds.
- YOLOv8 might not perform optimally under challenging conditions like fog, heavy rain, or low-light environments unless specifically trained on such data.
- YOLOv8 can have difficulty detecting objects that are partially occluded, which is common in real-world scenarios such as crowded scenes or vehicles hidden by other objects.

CHAPTER - 2

LITERATURE SURVEY

2.1 LITERATURE SURVEY

Title: S2-MLP: Spatial-shift MLP architecture for vision
Author: T. Yu, X. Li, Y. Cai, M. Sun, and P. Li,
Year: 2022.
Description: Recently, visual Transformer (ViT) and its following works abandon the convolution and exploit the self-attention operation, attaining a comparable or even higher accuracy than CNN. More recently, MLP-mixer abandons both the convolution and the self-attention operation, proposing an architecture containing only MLP layers. To achieve crosspatch communications, it devises an additional token-mixing MLP besides the channel-mixing MLP. It achieves promising results when training on an extremely large-scale datasetsuch as JFT-300M. But it cannot achieve as outstanding performance as its CNN and ViT counterparts when training on medium-scale datasets such as ImageNet-1K. The performance drop of MLP-mixer motivates us to rethink the token-mixing MLP. We discover that token-mixing operation in MLP-mixer is a variant of depthwise convolution with a global reception field and spatial-specific configuration. In this paper, we propose a novel pure MLP architecture,spatial-shift MLP (S2 -MLP). Different from MLP-mixer, our S2-MLP only contains channel-mixing MLP. We devise a spatial-shift operation for achieving the communication between patches. It has a local reception field and is spatialagnostic. Meanwhile, it is parameter-free and efficient for computation. The proposed S2 -MLP attains higher recognition accuracy than MLP-mixer when training on ImageNet1K dataset. Meanwhile, S2-MLP accomplishes as excellent performance as ViT on ImageNet-1K dataset with considerably simpler architecture and fewer FLOPs and parameters.

Title: PConv: Simple yet effective convolutional layer for generative adversarial network.
Author: S. Park, Y.-J. Yeo, and Y.-G. Shin,
Year: 2022.
Description: This paper presents a novel convolutional layer, called perturbed convolution (PConv), which performs not only a convolution operation but also a dropout one. The PConv focuses on achieving two goals simultaneously: improving the generative adversarial network (GAN) performance and

alleviating the memorization problem in which the discriminator memorizes all images from a given dataset as training progresses. In PConv, perturbed features are generated by randomly disturbing an input tensor before performing the convolution operation. This approach is simple but surprisingly effective. First, to produce a similar output even with the perturbed tensor, each layer in the discriminator should learn robust features having a small local Lipschitz value. Second, since the input tensor is randomly perturbed during the training procedure like the dropout in neural networks, the memorization problem could be alleviated. To show the generalization ability of the proposed method, we conducted extensive experiments with various loss functions and datasets including CIFAR-10, CelebA, CelebA-HQ, LSUN, and tiny-ImageNet. The quantitative evaluations demonstrate that PConv effectively boosts the performance of GAN and conditional GAN in terms of Frechet inception distance (FID).

Title: DINO: DETR with improved de noising anchor boxes for end-to-end object detection
Author: H. Zhang, F. Li, S. Liu, L. Zhang, H. Su, J. Zhu, L. M. Ni, and H.-Y. Shum,
Year: 2022
Description: — e present DINO (DETR with Improved deNoising anchOr boxes), a strong end-to-end object detector. DINO improves over previous DETR-like models in performance and efficiency by using a contrastive way for denoising training, a look forward twice scheme for box prediction, and a mixed query selection method for anchor initialization. DINO achieves 49.4AP in 12 epochs and 51.3AP in 24 epochs on COCO with a ResNet-50 backbone and multi-scale features, yielding a significant improvement of +6.0AP and +2.7AP, respectively, compared to DN-DETR, the previous best DETR-like model. DINO scales well in both model size and data size. Without bells and whistles, after pre-training on the Objects365 dataset with a SwinL backbone, DINO obtains the best results on both COCO val2017 (63.2AP) and test-dev (63.3AP) with model size under 1 billion parameters. Compared to other models on the leaderboard, DINO significantly reduces its model size and pre-training data size while achieving better results. The code will be available.

Title: Run, don't walk: Chasing higher FLOPS for faster neural networks
Author: J. Chen, S.-H. Kao, H. He, W. Zhuo, S. Wen, C.-H. Lee, and S.-H.-G. Chan,
Year: 2023.
Description: To design fast neural networks, many works have been focusing on reducing the number of floating-point operations (FLOPs). We observe that such reduction in FLOPs, however, does not necessarily lead to a similar level of reduction in latency. This mainly stems from inefficiently low floating-point operations per second (FLOPS). To achieve faster networks, we revisit popular operators and demonstrate that such low FLOPS is mainly

due to frequent memory access of the operators, especially the depthwise convolution. We hence propose a novel partial convolution (PConv) that extracts spatial features more efficiently, by cutting down redundant computation and memory access simultaneously. Building upon our PConv, we further propose FasterNet, a new family of neural networks, which attains substantially higher running speed than others on a wide range of devices, without compromising on accuracy for various vision tasks. For example, on ImageNet1k, our tiny FasterNet-T0 is 2.8×, 3.3×, and 2.4× faster than MobileViT-XXS on GPU, CPU, and ARM processors, respectively, while being 2.9% more accurate. Our large FasterNet-L achieves impressive 83.5% top-1 accuracy, on par with the emerging Swin-B, while having 36% higher inference throughput on GPU, as well as saving 37% compute time on CPU.

Title: Low-illumination object detection method based on dark-YOLO.

Author:] J. Zetao, X. Yun, and Z. Shaoqin,

Year: 2023**.**

Description: Low-light object detection is an important research area in computer vision, but it is also a difficult issue. This research offers a low-light target detection network, NLE-YOLO, based on YOLOV5, to address the issues of insufficient illumination and noise interference experienced by target detection tasks in low-light environments. The network initially preprocesses the input image with an improvement technique before suppressing high-frequency noise and enhancing essential information with C2fLEFEM, a unique feature extraction module. We also created a multi-scale feature extraction module, AMC2fLEFEM, and an attention mechanism receptive field module, AMRFB, which are utilized to extract features of multiple scales and enhance the receptive field. The C2fLEFEM module, in particular, merges the LEF and FEM modules on top of the C2f module. The LEF module employs a low-frequency filter to remove high-frequency noise; the FEM module employs dual inputs to fuse low-frequency enhanced and original features; and the C2f module employs a gradient retention method to minimize information loss. The AMC2fLEFEM module combines the SimAM attention mechanism and uses the pixel relationship to obtain features of different receptive fields, adapt to brightness changes, capture the difference between the target and the background, improve the network's feature extraction capability, and effectively reduce the impact of noise. The AMRFB module employs atrous convolution to enlarge the receptive field, maintain global information, and adjust to targets of various scales. Finally, for low-light settings, we replaced the original YOLOv5 detection head with a decoupled head. The Exdark dataset experiments show that our method outperforms previous methods in terms of detection accuracy and performance.

2.2 PROPOSED SYSTEM

- YOLOv10 is the latest advancement in the YOLO series, with improvements that make it better suited for real-time detection in various environments, including low-visibility or adverse weather conditions. It introduces several innovations in architecture and feature extraction to handle small, occluded, or distorted objects more effectively. YOLOv10 also integrates enhancements that reduce computational cost, making it more efficient without compromising on accuracy.
- The model's architecture allows it to process images faster, enabling its use in time-sensitive applications where rapid decision-making is critical. YOLOv10's increased robustness, especially in complex and challenging environments, makes it a superior choice for vehicle detection, surveillance, and other computer vision tasks.

2.2.1 PROPOSED SYSTEM ADVANTAGES:

- YOLOv10 enhances object detection capabilities, particularly for small or occluded objects, making it more robust in challenging environments such as fog, nighttime, or busy scenes.
- YOLOv10 is designed with features that enable it to handle difficult scenarios like low visibility or cluttered scenes, making it more reliable in real-world applications.
- YOLOv10 incorporates enhancements in feature extraction and model architecture, allowing it to handle a wide range of detection tasks effectively.

CHAPTER - 3

DESIGN AND DEVELOPMENT

3.1 PROJECT DESCRIPTION

This project proposes a robust detection framework that enhances vehicle and human detection in foggy conditions using an improved YOLOv10 model. By leveraging advanced image preprocessing techniques such as Dehaze Formers and dark channel methods, the model improves image visibility, reducing the impact of haze and low contrast. An advanced attention module is integrated into the model to ensure that it captures long-range dependencies and contextual information, prioritizing relevant features and improving detection performance, particularly for small or occluded objects. Additionally, the feature extraction process has been optimized to create a lightweight yet efficient detection system that strikes a balance between performance and computational cost. The proposed method is evaluated on challenging foggy road scenes, demonstrating its effectiveness in accurately detecting vehicles and pedestrians, thereby improving safety and reliability for autonomous systems in adverse weather conditions.

3.2 METHODOLOGIES

3.2.1 MODULES NAME:

Modules Name:

- Data Collection
- Data Labels Analysis
- Annotations
- Data Preprocessing
- Model Apply
- UI Design
- Detection

3.2.2 MODULES EXPLANATION:

1. **Data Collection**

This module involves gathering a comprehensive dataset of road scenes captured under foggy weather conditions. The dataset should include images of vehicles, pedestrians, and other objects typically present on roads. Data collection can be done using cameras in real-world environments or sourced from publicly available datasets like Foggy Cityscapes. Ensuring diversity in fog density, object types, and scene layouts is crucial for developing a robust detection system.

2. **Data Labels Analysis**

Data labeling involves categorizing the objects in the dataset, such as vehicles, humans, and other road elements. Analysis of these labels ensures balanced representation of all categories and helps identify potential issues like class imbalance or mislabeling. Accurate and well-distributed labels are essential for training a model that generalizes well across different scenarios.

3. **Annotations**

Annotations define the exact location and category of objects in the dataset by marking bounding boxes around vehicles and pedestrians. This module uses tools like Labeling or VIA (VGG Image Annotator) to manually or semi-automatically annotate images. High-quality annotations are critical for training object detection models, as they directly influence the accuracy of predictions.

4. **Data Preprocessing**

Preprocessing prepares the dataset for model training by improving the quality of images and ensuring consistency. Techniques include:

Dehazing : Using Dehaze Formers or dark channel methods to remove fog and improve visibility.

Normalization : Scaling pixel values to a uniform range to optimize model performance.

Augmentation : Enhancing dataset diversity by applying transformations such as rotation, flipping, and scaling to simulate various real-world scenarios.

5. **Model Apply**

In this module, the improved YOLOv10 model is applied for detecting vehicles and humans in foggy scenes. The model is trained on the preprocessed and annotated dataset, with the inclusion of an enhanced attention module to prioritize relevant features. Training involves optimizing hyperparameters and using techniques like transfer learning to achieve high detection accuracy even in challenging conditions.

6. **UI Design**

A user interface (UI) is designed to enable seamless interaction with the detection system. The UI allows

users to upload images or provide real-time video feeds for processing. It displays the detection results, including bounding boxes and labels for identified objects, in a user-friendly manner. The design focuses

on simplicity and efficiency to cater to diverse users, including researchers and developers.

7. **Detection**

The detection module represents the core functionality of the system. It processes input images or videos through the trained YOLOv10 model to identify and localize vehicles and pedestrians. The system outputs bounding boxes, class labels, and confidence scores, highlighting detected objects in the foggy scenes. The effectiveness of this module is evaluated using metrics like mean Average Precision (mAP) and Intersection over Union (IoU), ensuring reliable performance in real-world scenarios.

3.3 TECHNIQUE USED OR ALGORITHM USED

3.3.1 EXISTING TECHNIQUE:

- **YOLOv8 (You Only Look Once version 8)**

➢ YOLOv8 (You Only Look Once version 8) is a cutting-edge object detection algorithm that is part of the YOLO (You Only Look Once) series of models. It is designed to perform real-time object detection with a single pass through the network, making it both fast and efficient. YOLOv8 aims to strike a balance between speed and accuracy, making it suitable for a wide range of applications in computer vision, such as surveillance, autonomous vehicles, and robotics

➢ The model is optimized to work with varying object sizes and complex backgrounds, providing accurate results even under challenging conditions. YOLOv8 is known for its lightweight nature, which allows it to be deployed on devices with limited computational power, such as mobile phones and embedded systems.

3.3.2 PROPOSED TECHNIQUE USED OR ALGORITHM USED:

- **YOLOv10 (You Only Look Once version 10)**

➢ YOLOv10 (You Only Look Once version 10) is an advanced version of the YOLO architecture, designed for even greater efficiency and accuracy in real-time object detection tasks. YOLOv10 builds upon the YOLOv8 architecture by introducing further optimizations and innovations that improve its performance, especially in challenging environments and complex scenarios.

➢ Like previous versions, YOLOv10 is based on a one-stage detection model that performs both localization (bounding boxes) and classification of objects in a single pass. YOLOv10 is optimized for real-time applications, providing faster inference times without sacrificing detection accuracy, making it suitable for tasks such as autonomous driving, security surveillance.

3.3.3 HARDWARE REQUIREMENTS

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

- PROCESSOR : DUAL CORE 2 DUOS.
- RAM : 4GB DD RAM
- HARD DISK : 250 GB

3.3.4 SOFTWARE REQUIREMENTS

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

- OPERTAING SYSTEM : WINDOWS 7/8/10
- PLATFORM : SPYDER 3
- PROGRAMMING LANGUAGE : PYTHON
- FRONT END : SPYDER 3

3.3.5 FUNCTIONAL REQUIREMENTS

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

3.3.6 NON-FUNCTIONAL REQUIREMENTS

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

- **Usability :** The system is designed with completely automated process hence there is no or less user intervention.
- **Reliability :** The system is more reliable because of the qualities that are inherited from the chosen platform python. The code built by using python is more reliable.
- **Performance :** This system is developing in the high level languages and using the advanced back-end technologies it will give response to the end user on client system with in very less time.
- **Supportability :** The system is designed to be the cross platform supportable. The system is

supported on a wide range of hardware and any software platform, which is built into the system.

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

3.4 UML DIAGRAMS

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

3.4.1 USE CASE DIAGRAM

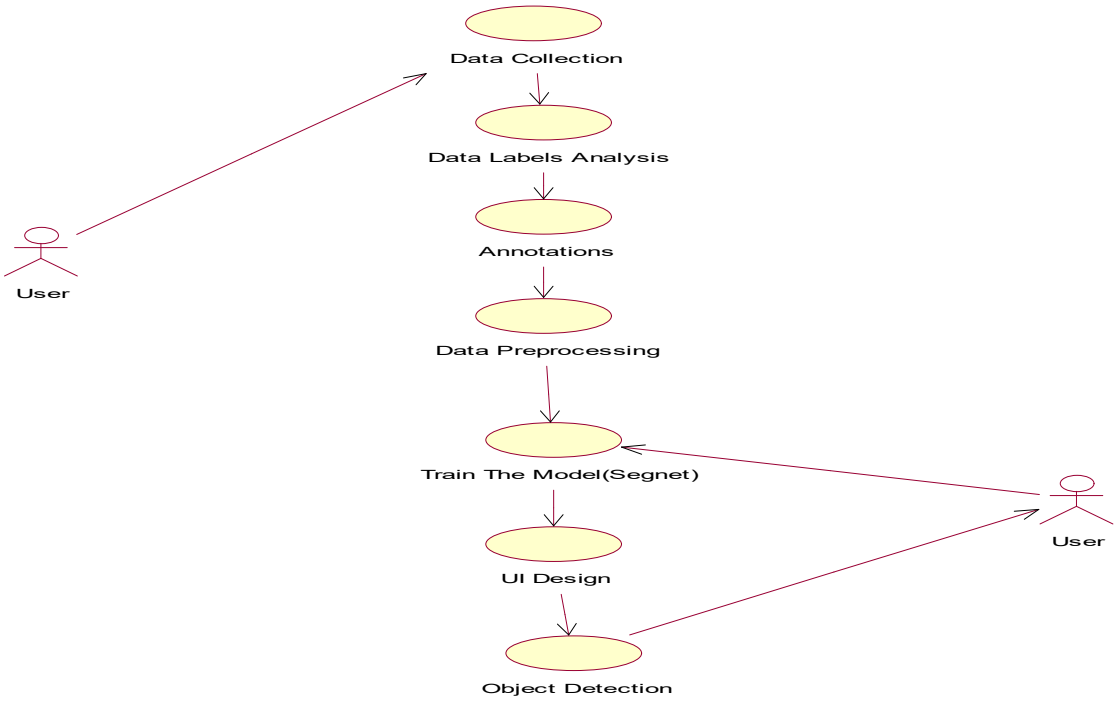


Fig 3.4.1 : Use Case Diagram

EXPLANATION:

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

3.4.2 CLASS DIAGRAM

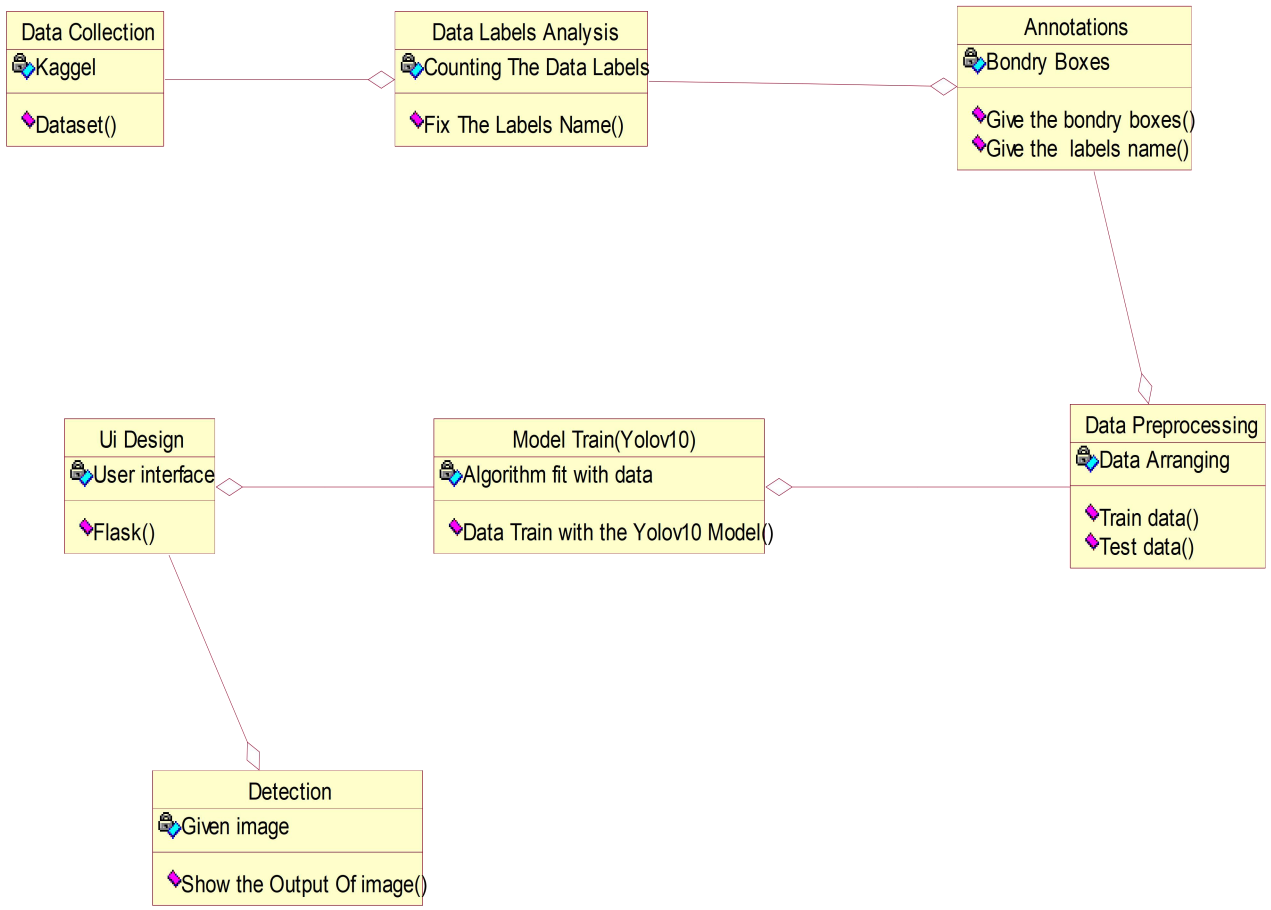


Fig 3.4.2 : Class Diagram

EXPLANATION:

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

3.4.3 OBJECT DIAGRAM

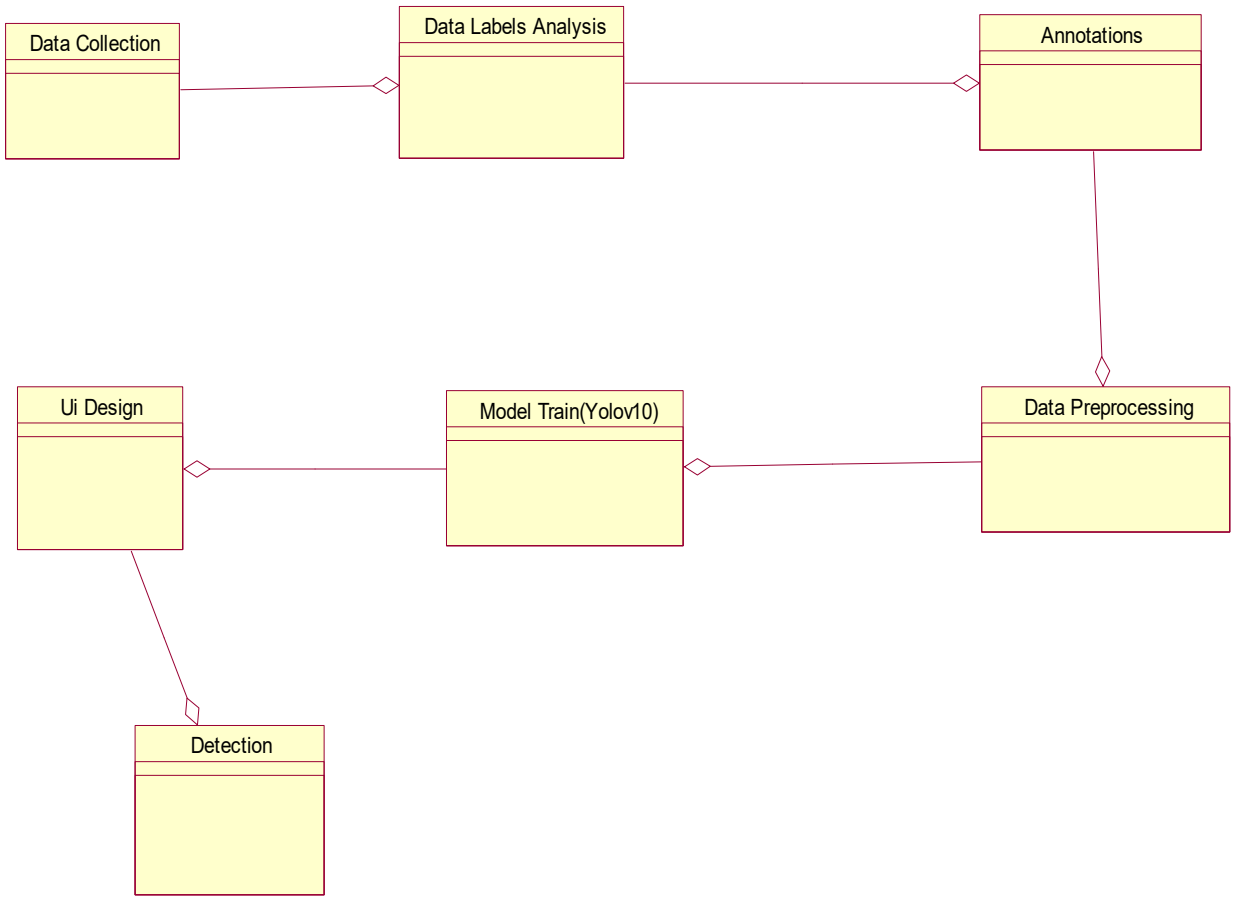


Fig 3.4.3 : Object Diagram

EXPLANATION:

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

3.4.4 STATE DIAGRAM

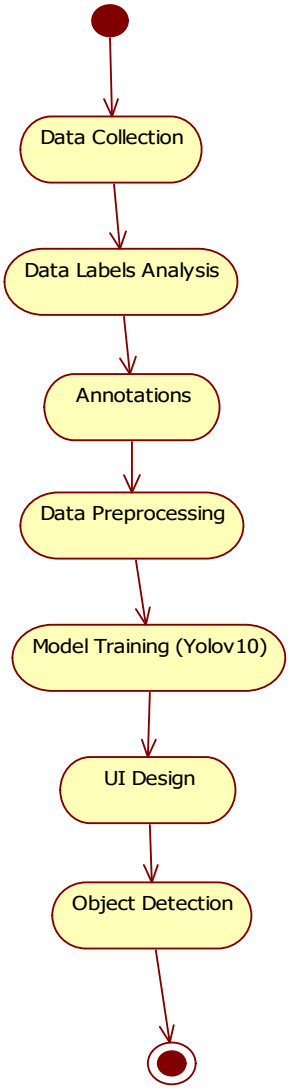


Fig 3.4.4 : State Diagram

EXPLANATION:

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

3.4.5 ACTIVITY DIAGRAM

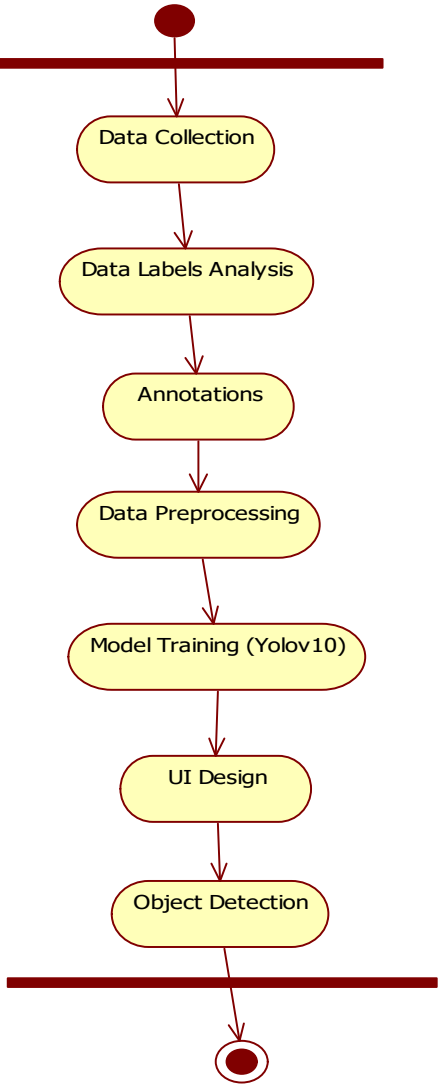


Fig 3.4.5 : Activity Diagram

EXPLANATION:

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

3.4.6 SEQUENCE DIAGRAM

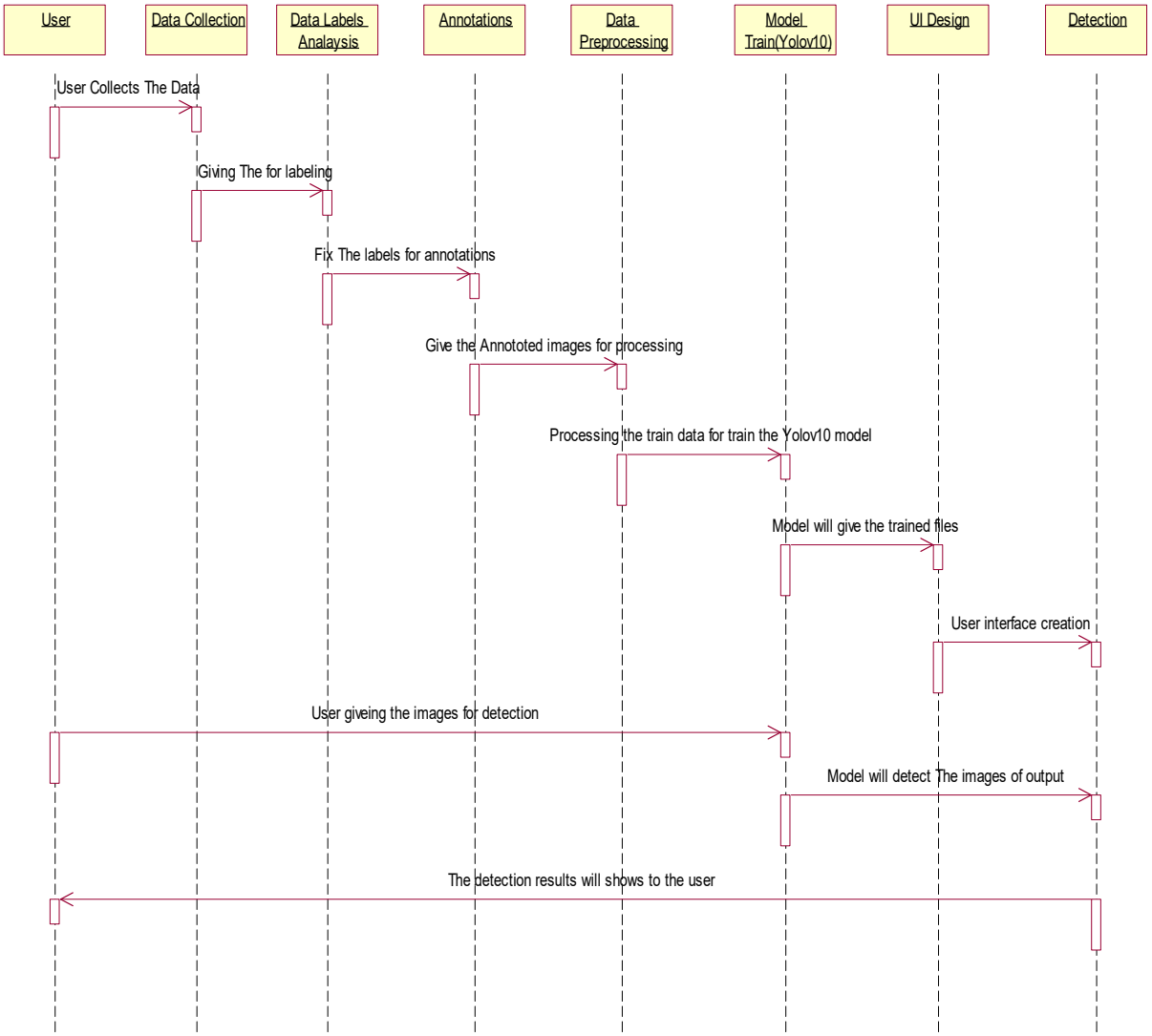


Fig 3.4.6 : Sequence Diagram

EXPLANATION:

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

3.4.7 COLLABORATION DIAGRAM

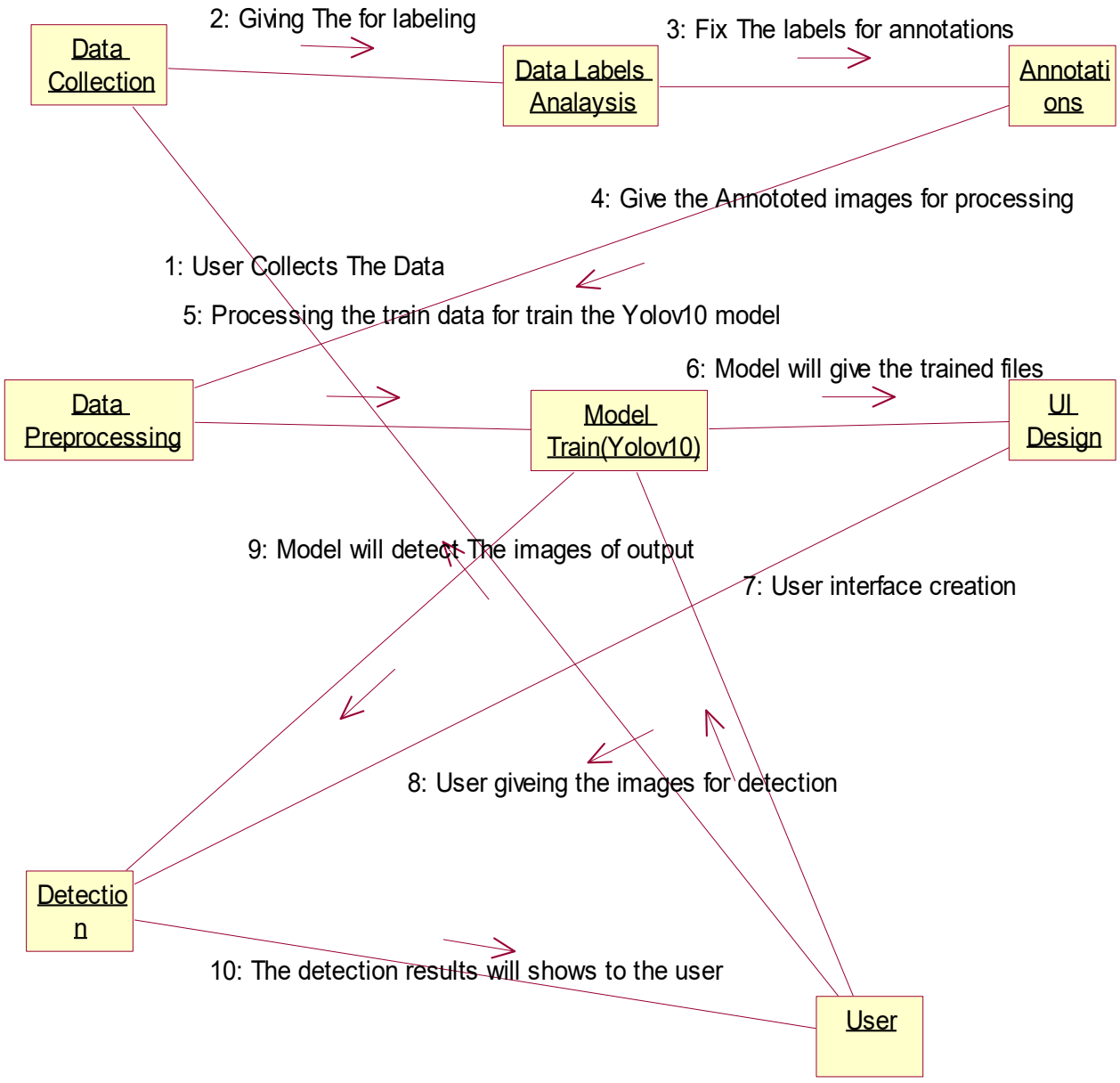


Fig 3.4.7 : Collaboration Diagram

EXPLANATION:

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

3.4.8 COMPONENT DIAGRAM

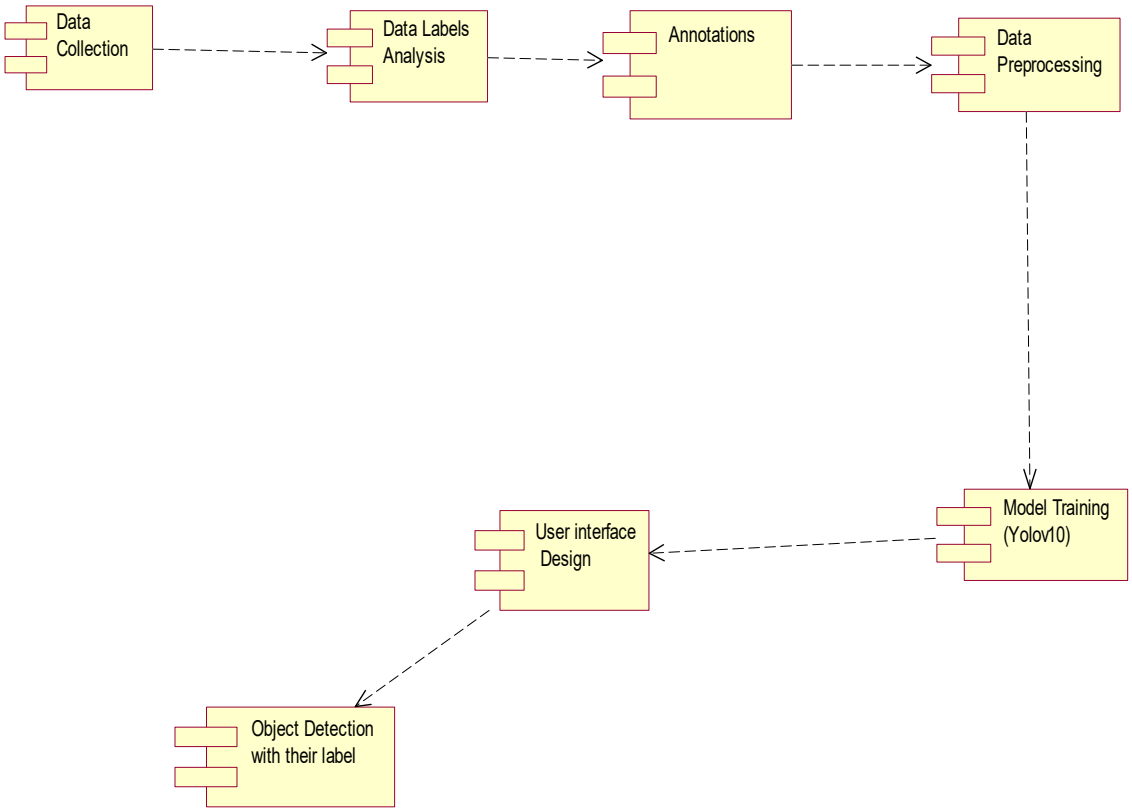


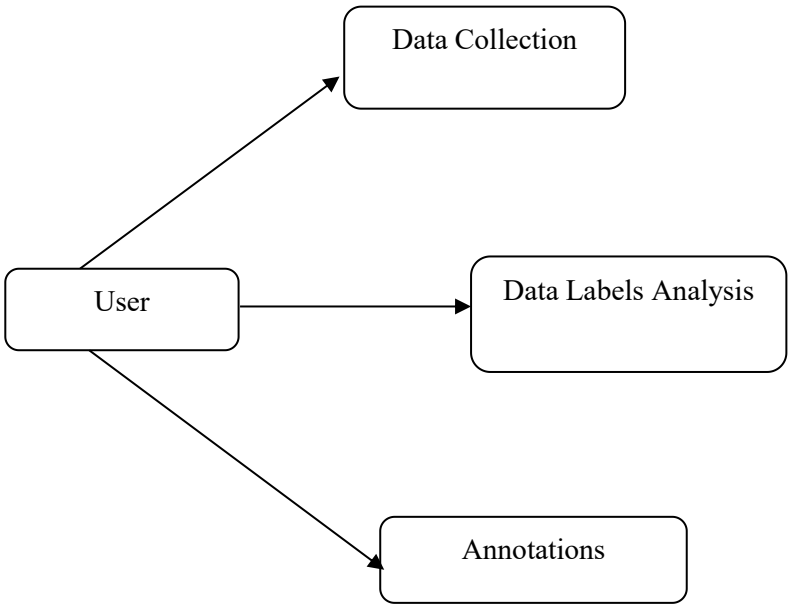
Fig 3.4.8 : Component Diagram

EXPLANATION

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

3.4.9 DATA FLOW DIAGRAM

Level 0



Level 1

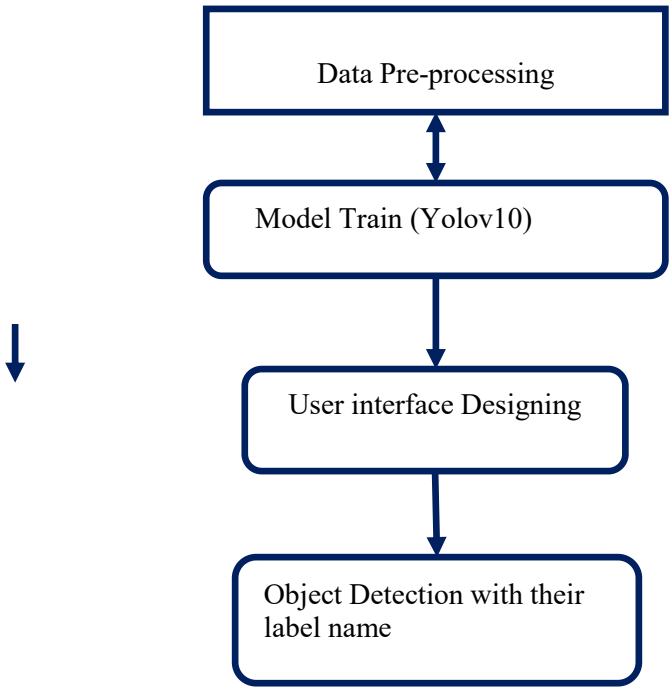


Fig 3.4.9 : Data Flow Diagrams

EXPLANATION:

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

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

3.4.10 DEPLOYMENT DIAGRAM

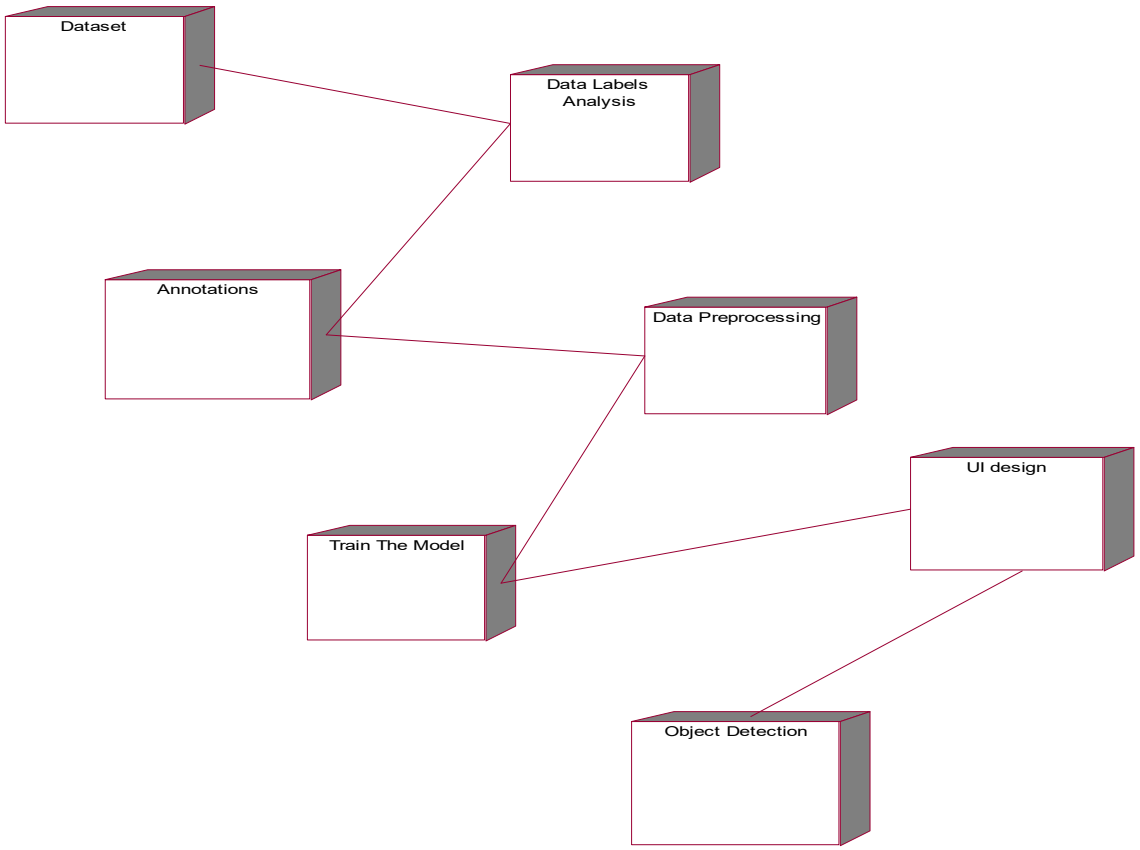


Fig 3.4.10 : Deployment Diagram

EXPLANATION:

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

3.4.11 SYSTEM ARCHITECTURE:

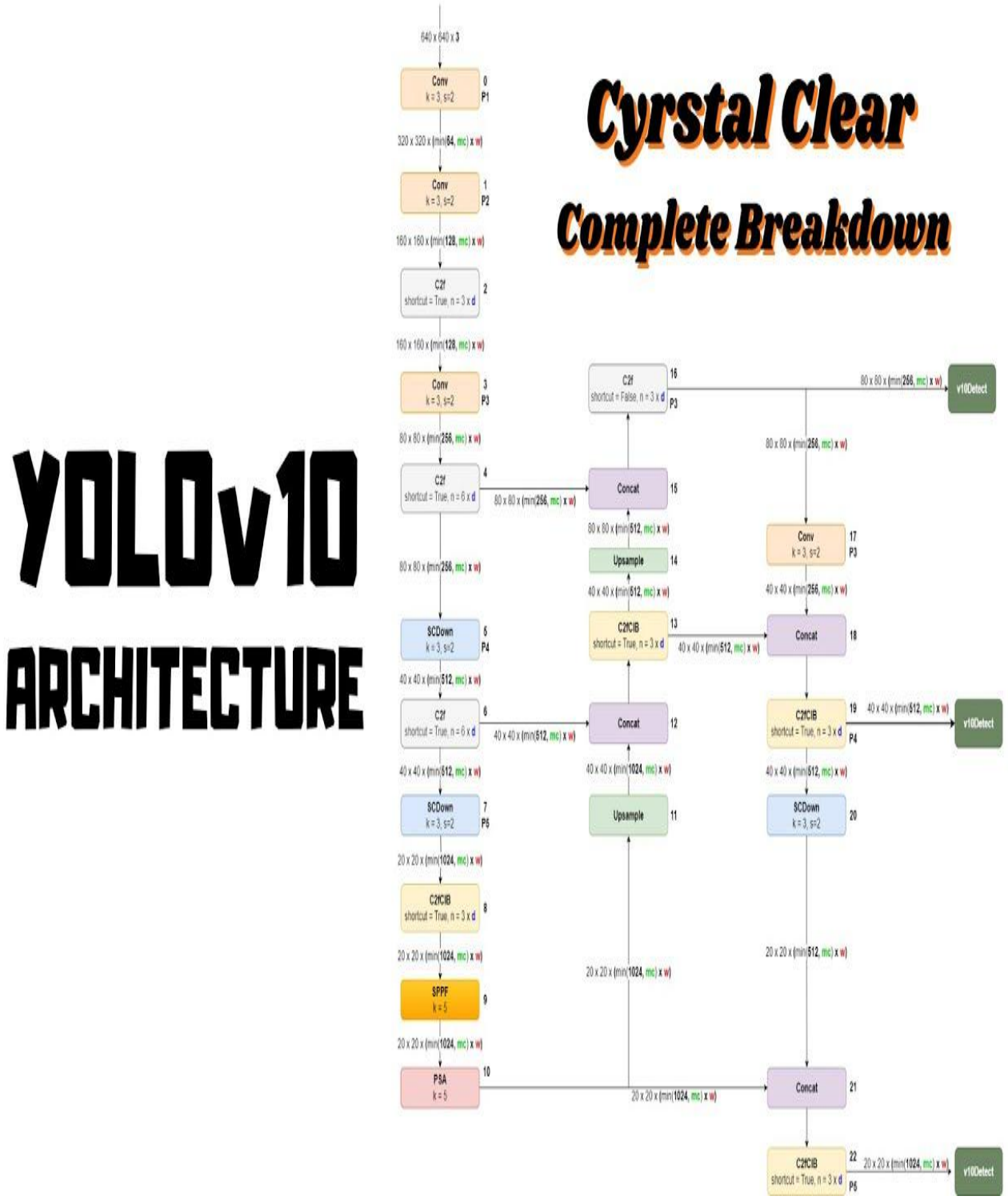


Fig: 3.4.11: System Architecture

3.5 DEVELOPMENT TOOLS

3.5.1 Python

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

3.5.2 History of Python

Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands.Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, SmallTalk, and Unix shell and other scripting languages.Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL).Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

3.5.3 Importance of Python

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

3.5.4 Features of Python

- **Easy-to-learn** − Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
- **Easy-to-read** − Python code is more clearly defined and visible to the eyes.
- **Easy-to-maintain** − Python's source code is fairly easy-to-maintain.
- **A broad standard library** − Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
- **Interactive Mode** − Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.

- **Portable** − Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
- **Extendable** − You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
- **Databases** − Python provides interfaces to all major commercial databases.
- **GUI Programming** − Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.
- **Scalable** − Python provides a better structure and support for large programs than shell scripting.

Apart from the above-mentioned features, Python has a big list of good features, few are listed below −

- It supports functional and structured programming methods as well as OOP.
- It can be used as a scripting language or can be compiled to byte-code for building large applications.
- It provides very high-level dynamic data types and supports dynamic type checking.
- IT supports automatic garbage collection.
- It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

3.5.5 Libraries used in python

- numpy - mainly useful for its N-dimensional array objects.
- pandas - Python data analysis library, including structures such as dataframes.
- matplotlib - 2D plotting library producing publication quality figures.
- scikit-learn - the machine learning algorithms used for data analysis and data mining tasks.

3.6 IMPLEMENTATION

```
from flask import Flask, render_template, request, redirect, url_for, send_from_directory, session
from ultralytics import YOLO
import os
import cv2
from werkzeug.utils import secure_filename
# Initialize the Flask app
app = Flask(_name_)
app.secret_key = 'your_secret_key'
# Change this to a strong secret key
# Folder for uploaded images and detection results
UPLOAD_FOLDER = 'uploads'
RESULTS_FOLDER = 'results'
# Create directories if they don't exist
os.makedirs(UPLOAD_FOLDER, exist_ok=True)
os.makedirs(RESULTS_FOLDER, exist_ok=True)
app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER
app.config['RESULTS_FOLDER'] = RESULTS_FOLDER
# Load the YOLO model
model = YOLO('best.pt')  # Path to your trained YOLO weights
users = {'admin': 'admin'}
@app.route('/')
def home():
```

```
    return render_template('home.html')
@app.route('/about')
def about():
    return render_template('about.html')
@app.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        username = request.form['username']
        password = request.form['password']
        if username in users and users[username] == password:
            session['username'] = username
            return redirect(url_for('index'))
        else:
            return "Login failed. Please check your username and password."
    return render_template('login.html')
@app.route('/register', methods=['GET', 'POST'])
def register():
    if request.method == 'POST':
        username = request.form['username']
        password = request.form['password']
        if username not in users:
            users[username] = password
            return redirect(url_for('login'))
```

```
else:
            return "Registration failed. User already exists."
    return render_template('register.html')
@app.route('/index')
def index():
    if 'username' not in session:
        return redirect(url_for('login'))
    return render_template('index.html')
# @app.route('/')
# def home():
#     return render_template('index.html')
@app.route('/upload', methods=['POST'])
def upload_file():
    if 'file' not in request.files:
        return redirect(url_for('home'))
    file = request.files['file']
    if file.filename == '':
        return redirect(url_for('home'))
    if file:
        # Save the uploaded file
        filename = secure_filename(file.filename)
        file_path = os.path.join(app.config['UPLOAD_FOLDER'], filename)
        file.save(file_path)
```

```
    # Perform object detection
        results = model(file_path)
        # Save the annotated image
        result_img_path = os.path.join(app.config['RESULTS_FOLDER'], f"result_{filename}")
        annotated_frame = results[0].plot()  # Annotate the image
        cv2.imwrite(result_img_path, annotated_frame)
        # Render result.html with paths to the original and detected images
        return render_template('result.html',
                               original_file=f'uploads/{filename}',
                               result_file=f'results/result_{filename}')
@app.route('/uploads/<filename>')
def uploaded_file(filename):
    return send_from_directory(app.config['UPLOAD_FOLDER'], filename)
@app.route('/results/<filename>')
def result_file(filename):
    return send_from_directory(app.config['RESULTS_FOLDER'], filename)
if _name_ == '_main_':
    app.run(debug=True)
```

3.7 SNAPSHOTS

3.7.1: Login Page

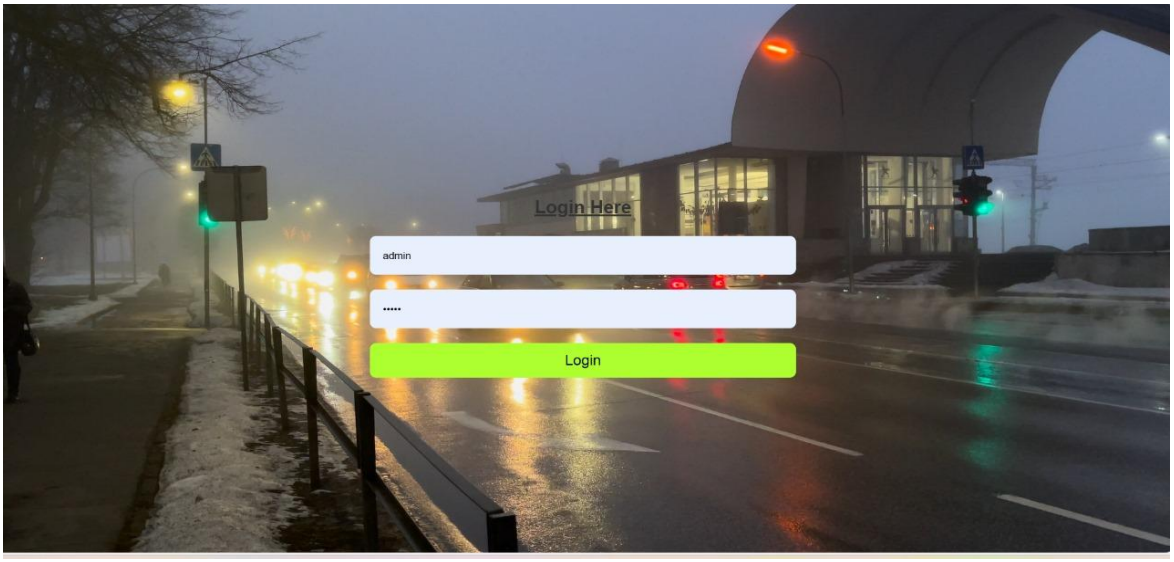


Fig 3.7.1: Login Page

3.7.2 : Image Upload Page

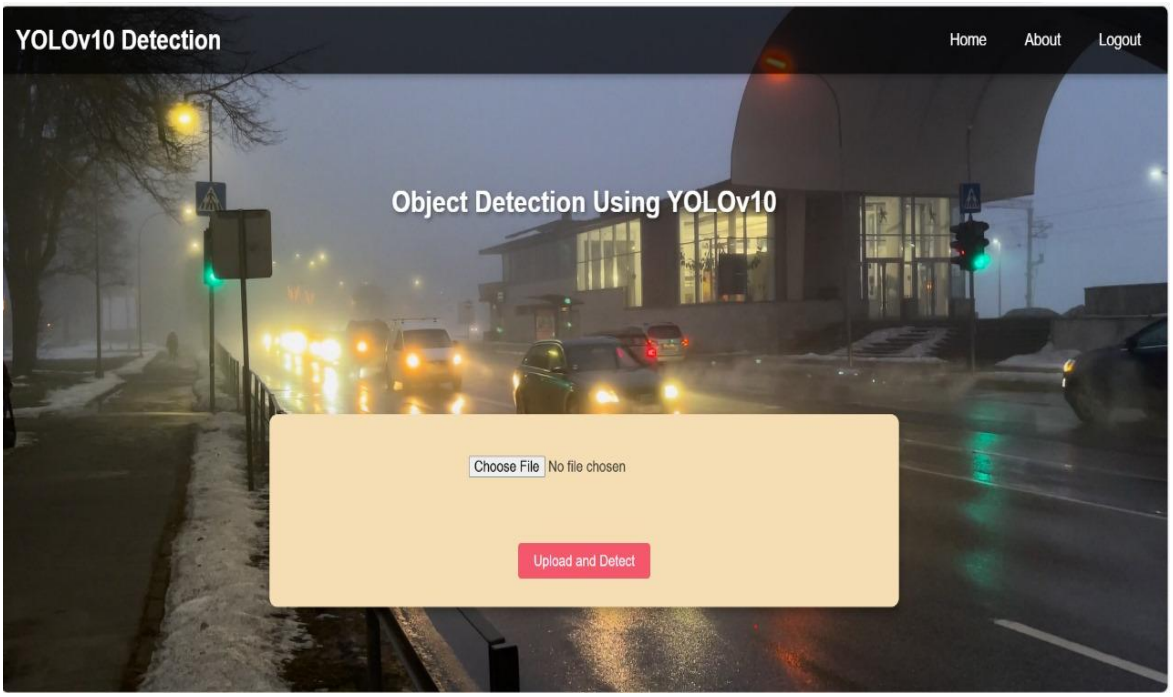


Fig 3.7.2: Image Upload Page

3.7.3 : Image Selection Page

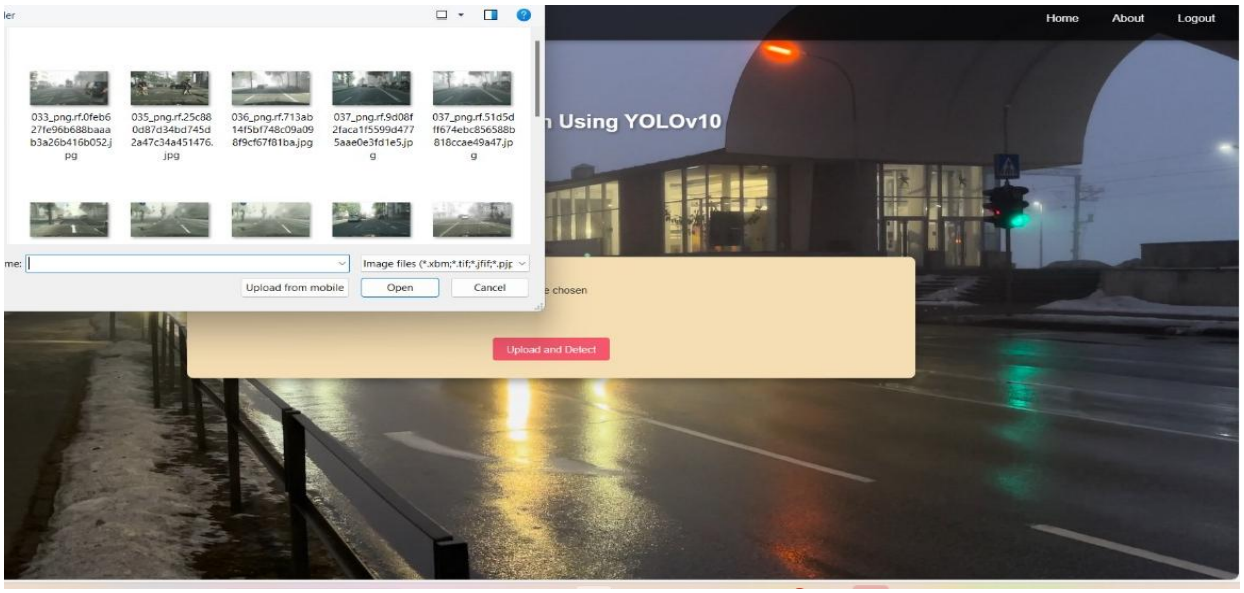


Fig 3.7.3: Image Selection Page

3.7.4 : Detection Results Page

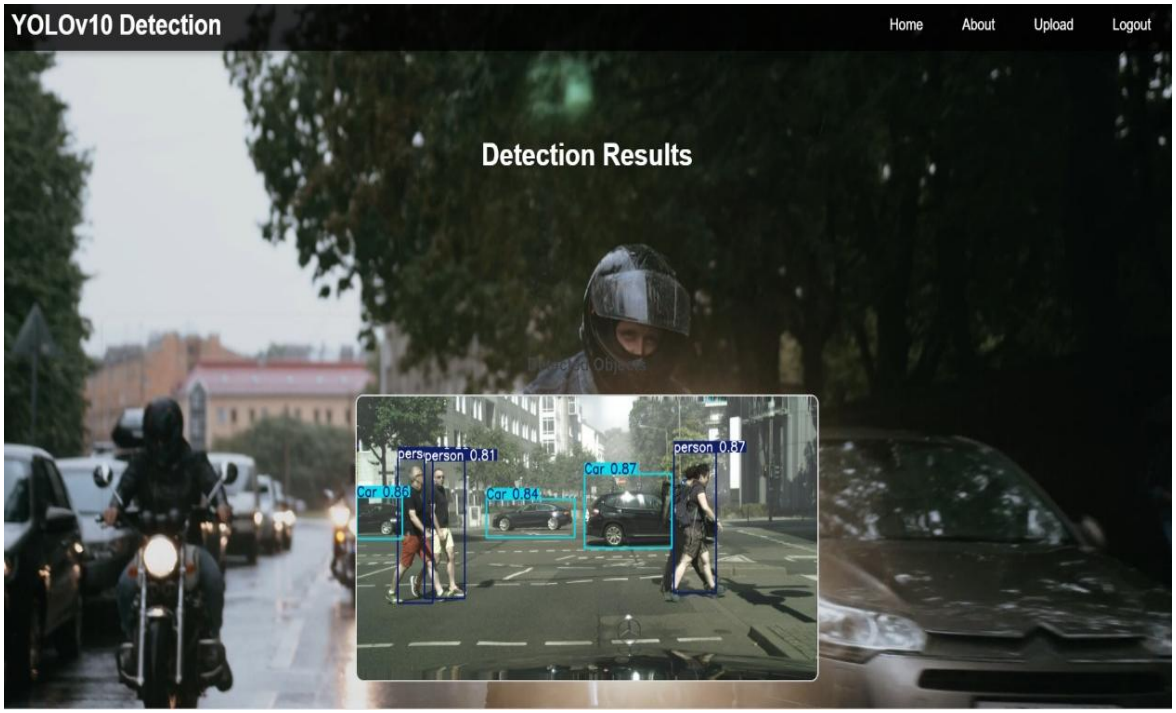


Fig 3.7.4: Detection Results Page

3.8 TESTS AND VALIDATION

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

TYPES OF TESTS

3.8.1 Unit testing

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

3.8.2 Functional test

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

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.
Invalid Input : identified classes of invalid input must be rejected.
Functions : identified functions must be exercised.
Output : identified classes of application outputs must be exercised.
Systems/Procedures : interfacing systems or procedures must be invoke

3.8.3 System Test

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

3.8.4 Performance Test

The Performance test ensures that the output be produced within the time limits,and the time taken

by the system for compiling, giving response to the users and request being send to the system for to retrieve the results.

3.8.5 Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects. The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

3.8.6 Acceptance Testing

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

Acceptance testing for Data Synchronization:

- The Acknowledgements will be received by the Sender Node after the Packets are received by the Destination Node
- The Route add operation is done only when there is a Route request in need
- The Status of Nodes information is done automatically in the Cache Updation process

3.8.7 Build the test plan

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

CHAPTER - 4

RESULTS & DISCUSSION

1.1 RESULT

The research successfully developed an efficient and lightweight YOLOv10 model optimized for traffic target detection in foggy conditions. By integrating advanced modules such as DCN, Involution, and FasterNex, the model demonstrated enhanced detection accuracy and performance while reducing computational complexity. The introduction of the S5 attention module improved feature fusion, and an additional small target detection layer significantly boosted the accuracy for detecting smaller objects and refining boundary box regression.

4.2 FUTURE ENHANCEMENTS

Future research will focus on optimizing the YOLOv10 network to enhance detection accuracy while maintaining high processing speeds. Efforts will include developing advanced feature extraction techniques to improve performance in extreme foggy or low-visibility conditions. Additionally, the model's detection capabilities will be extended to include other traffic-related objects, such as road signs, obstacles, and lane markings, to enhance its versatility. To ensure real-world applicability, a more compact and efficient version of the YOLOv10 model will be designed for deployment on edge devices and mobile platforms. Practical field tests will be conducted under diverse conditions to validate the model's robustness and reliability, and innovative lightweight attention modules will be explored to further improve feature prioritization and reduce computational overhead.

CHAPTER - 5

CONCLUSION

5.1 CONCLUSION

The research we proposed an efficient and lightweight YOLOv10 network model tailored for traffic target detection in foggy weather. By integrating advanced modules such as DCN, Involution, and FasterNex, we successfully reduced model parameters and size while enhancing detection accuracy and performance. A novel S5 attention module was introduced to improve feature fusion, and an additional small target detection layer significantly boosted the accuracy of detecting small objects and refined boundary box regression.

Experimental results show that the proposed model outperforms existing state-of-the-art methods in terms of detection accuracy and computational efficiency. This makes it an attractive solution for real-world applications, such as autonomous driving and intelligent transportation systems, where accurate object detection is crucial for safety and reliability. The optimized YOLOv10 model has significant potential to enhance traffic monitoring and management, and future work can explore further optimizations and applications of the proposed model.

CHAPTER - 6

REFERENCES

6.1 REFERENCES

[1] C. Li, C. Guo, J. Guo, P. Han, H. Fu, and R. Cong, ‘‘PDR-Net: Perception-inspired single image dehazing network with refinement,’’ IEEE Trans. Multimedia, vol. 22, no. 3, pp. 704–716, Mar. 2020, doi: 10.1109/TMM.2019.2933334.

[2] X. Xiaomin and L. Wei, ‘‘Two stages end-to-end generative network for single image defogging,’’ J. Comput.-Aided Des.Comput. Graph., vol. 32, no.1, pp.164–172, 2020, doi: 10.3724/SP.J.1089.2020.17856.

[3] J. Wu, Z. Kuang, L. Wang, W. Zhang, and G. Wu, ‘‘Context-aware RCNN: A baseline for action detection in videos,’’ 2020, arXiv:2007.09861.

[4] L. Jiang, J. Chen, H. Todo, Z. Tang, S. Liu, and Y. Li, ‘‘Application of a fast RCNN based on upper and lower layers in face recognition,’’ Comput. Intell. Neurosci., vol. 2021, pp. 1–12, Sep. 2021.

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

[6] K. He, G. Gkioxari, P. Dollár, and R. Girshick, ‘‘Mask R-CNN,’’ 2017, arXiv:1703.06870.

[7] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, ‘‘You only look once: Unified, real-time object detection,’’ 2015, arXiv:1506.02640.

[8] J. Redmon and A. Farhadi, ‘‘YOLO9000: Better, faster, stronger,’’ 2016, arXiv:1612.08242.

[9] J. Redmon and A. Farhadi, ‘‘YOLOv3: An incremental improvement,’’ 2018, arXiv:1804.02767.

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

[11] M. Wang, W. Yang, L. Wang, D. Chen, F. Wei, H. KeZiErBieKe, and Y. Liao, ‘‘FE-YOLOv5: Feature enhancement network based on YOLOv5 for small object detection,’’ J. Vis. Commun. Image Represent., vol. 90, Feb. 2023, Art. no. 103752, doi: 10.1016/j.jvcir.2023.103752.

[12] C.-Y. Wang, A. Bochkovskiy, and H.-Y.-M. Liao, ‘‘YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors,’’ in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Vancouver, BC, Canada, Jun. 2023, pp. 7464–7475.

[13] L. Wei, A. Dragomir, E. Dumitru, S. Christian, R. Scott, F. ChengYang, B. Alexander, ‘SSD: Single shot MultiBox detector,’’ in Computer Vision—ECCV 2016, vol. 9905. Cham, Switzerland:

Springer,2016.

[14] B. Gao, Z. Jiang, and J. Zhang, ‘‘Traffic sign detection based on SSD,’’ in Proc. 4th Int. Conf. Autom., Control Robot. Eng., Jul. 2019, p. 16, doi: 10.1145/3351917.3351988.

[15] L. Xuan, L. Jing, and W. Haiyan, ‘‘Study on traffice scene object detection algorithm under complex meteorological conditions,’’ Comput. Simul., vol. 38, no. 2, pp. 87–90, 2021.

[16] W. Qi-Ming, Z. He, D. Zhang, and Z. Mao, ‘‘Research on pedestrian and vehicle detection method based on YOLOv3 in foggy scene,’’ Control Eng. China, vol. 1, pp. 1–8, Sep. 2023.

[17] J. Zetao, X. Yun, and Z. Shaoqin, ‘‘Low-illumination object detection method based on dark-YOLO,’’ J. Comput.-Aided Des. Comput. Graph., vol. 35, no. 3, pp. 441–451, 2023.

[18] J. Yin, S. Qu, Z. Yao, X. Hu, X. Qin, and P. Hua, ‘‘Traffic sign recognition model in haze weather based on YOLOv5,’’ J. Comput. Appl., vol. 42, no. 9, pp. 2876–2884, 2022.

[19] K. Li, Y. Wang, and Z. Hu, ‘‘Improved YOLOv7 for small object detection algorithm based on attention and dynamic convolution,’’ Appl. Sci., vol. 13, no. 16, p. 9316, Aug. 2023, doi: 10.3390/app13169316.

[20] C. Bhagya and A. Shyna, ‘‘An overview of deep learning based object detection techniques,’’ in Proc. 1st Int. Conf. Innov. Inf. Commun. Technol. (ICIICT), Chennai, India, Apr. 2019, pp. 1–6.

[21] X. Zhu, W. Su, L. Lu, B. Li, X. Wang, and J. Dai, ‘‘Deformable DETR: Deformable transformers for end-to-end object detection,’’ 2020, arXiv:2010.04159.

[22] B. Li, W. Ren, D. Fu, D. Tao, D. Feng, W. Zeng, and Z. Wang, ‘‘Benchmarking single-image dehazing and beyond,’’ IEEE Trans. Image Process., vol. 28, no. 1, pp. 492–505, Jan. 2019.

[23] K. He, X. Zhang, S. Ren, and J. Sun, ‘‘Spatial pyramid pooling in deep convolutional networks for visual recognition,’’ IEEE Trans. Pattern Anal. Mach. Intell., vol. 37, no. 9, pp. 1904–1916, Sep. 2015.

[24] S. Ioffe and C. Szegedy, ‘‘Batch normalization: Accelerating deep network training by reducing internal covariate shift,’’ 2015, arXiv:1502.03167.

[25] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, ‘‘Attention is all you need,’’ in Proc. NIPS, 2017, pp. 1–11.

[26] J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, and Y. Wei, ‘‘Deformable convolutional networks,’’ 2017, arXiv:1703.06211.

[27] X. Zhu, H. Hu, S. Lin, and J. Dai, ‘‘Deformable ConvNets v2: More deformable, better results,’’ 2018, arXiv:1811.11168.

[28] Y. Guo, Y. Li, R. Feris, L. Wang, and T. Rosing, ‘‘Depthwise convolution is all you need for

learning multiple visual domains,'' 2019, arXiv:1902.00927.

[29] J. Chen, S.-H. Kao, H. He, W. Zhuo, S. Wen, C.-H. Lee, and S.-H.-G. Chan, ''Run, don't walk: Chasing higher FLOPS for faster neural networks,'' in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2023, pp. 12021–12031.

[30] S. Park, Y.-J. Yeo, and Y.-G. Shin, ''PConv: Simple yet effective convolutional layer for generative adversarial network,'' Neural Comput. Appl., vol. 34, no. 9, pp. 7113–7124, May 2022, doi: 10.1007/s00521-021- 06846-2.

[31] T. Yu, X. Li, Y. Cai, M. Sun, and P. Li, ''S2-MLP: Spatial-shift MLP architecture for vision,'' in Proc. IEEE/CVF Winter Conf. Appl. Comput. Vis. (WACV) Jan. 2022, pp. 3615–3624.

[32] H. Zhang, F. Li, S. Liu, L. Zhang, H. Su, J. Zhu, L. M. Ni, and H.-Y. Shum, ''DINO: DETR with improved de noising anchor boxes for end-to-end object detection,'' 2022, arXiv:2203.03605.