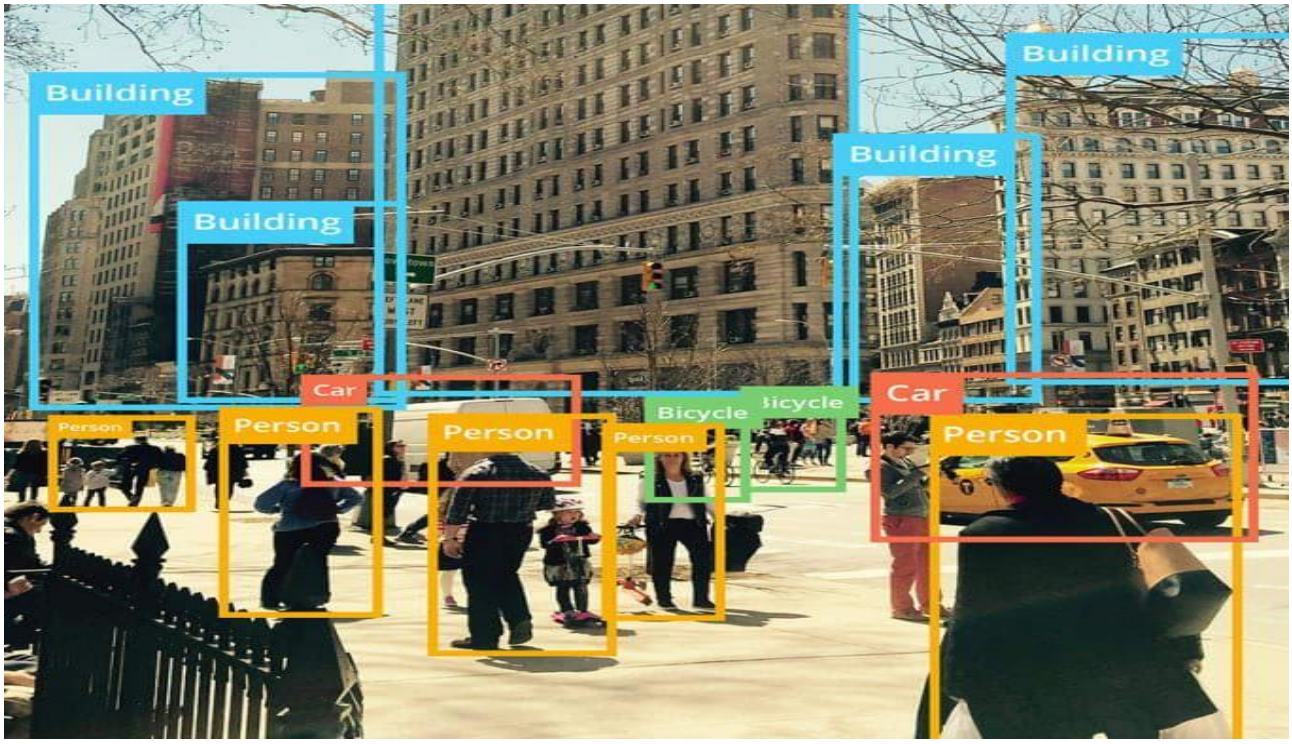
# BOARD OF TECHNICAL EDUCATION

## DEPARTMENT OF COMPUTER SCIENCE

## PROJECT SYNOPSIS

# Real Time Object Detection using CNN




UNDER THE GUIDENCE OF :

ASHWINI. MS , M Tech
Senior scale lecturer
Department of CSE

SUBMITTED BY:-

YASHWANTH KV
UDAYA GIRI BR
AKASH K
MUTTURAJU MS
CHANDAN GOWDA
DARSHAN D

# INTERSHIP GRADING RUBRICS (CIE-2)

NAME:

| EVALUATION CRITERIAN | POOR | AVERAGE | GOOD | EXCELLENT |
|---|---|---|---|---|
| Intern's ability to apply the skill and technical knowledge ( 30 MARKS) | 0-10 | 11-20 | 20-25 | 26-30 |
| Intern's performance on assigned tasks and project. (10 MARKS) | 0-3 | 4-6 | 7-8 | 9-10 |
| Extent of Intern's ability to add value to the organization through internship (10 MARKS) | 0-3 | 4-6 | 7-8 | 9-10 |
| **SUB TOTAL (50 MARKS)** | | | | |
| **USE CASE - II (30 MARKS)** | | | | |
| **TOTAL MARKS OBTAINED** | | | | |

**FACULTY SUPERVISOR**

# COMPUTER SCIENCE
# LOG SHEET

**NAME :**

**SEM:**

**REG:**

| SL | DATE | TASK | PROGRESS | INITIAL OF STAFF INCHARGE | EVALUATION |
|---|---|---|---|---|---|
| 1 | | Seminar regarding project work | | | |
| 2 | | Batch formation & guide allocation | | | |
| 3 | | Identification of project | | | |
| 4 | | Project synopsis submission | | | |
| 5 | | Finalization of project | | | |
| 6 | | Literature survey | | | |
| 7 | | Identification of facility to do project work | | | |
| 8 | | Study & design of system and Phase -1 presentation | | | |
| 9 | | Results discussion / performance testing | | | |
| 10 | | Review of project work by guide | | | |
| 11 | | Project report submission & Phase-2 presentation | | | |

**Signature of Guide** **Signature of HOD**

# Real-Time Object Detection using MobileNet SSD


## Acknowledgement

We extend our sincere gratitude to the research community whose advanced work in deep learning and computer vision has made this study possible. Special thanks to the teams at Google Research for developing and open-sourcing the MobileNet architecture and Single Shot Detector (SSD) framework, which formed the foundation of our work.

We acknowledge the valuable support provided by **GKV GLOBAL TECHNOLOGY** for granting access to the GPU computing resources essential for training and optimizing our models. Our appreciation goes to Sir.Vijayan G for their invaluable guidance and insights throughout this research.

We thank the analyst who helped create our dataset and the volunteers who participated in our real-world testing phase. Their contributions were crucial in validating the practical applications of our system.

This work was partially supported by **Vijayan G / Managing Director of GKV GLOBAL TECHNOLOGY**. We also thank the developers of various open-source libraries and tools that facilitated our implementation.

# Abstract

Real-time multi-object detection on resource-constrained devices presents significant challenges in balancing accuracy and computational efficiency. This paper presents an implementation of MobileNet Single Shot Detector (SSD), which achieves robust object detection while maintaining real-time performance on mobile platforms. The architecture leverages depth-wise separable convolutions from MobileNet as the backbone network, combined with the SSD framework for efficient feature extraction and object localization. Our approach achieves 22 frames per second on mobile devices while maintaining a mean Average Precision (mAP) of 68% on the dataset. The model's architecture reduces computational complexity through factorized convolutions, resulting in a 9x reduction in parameters compared to traditional CNN architectures. We introduce an adaptive feature pyramid network that dynamically adjusts feature resolution based on object scale, improving detection accuracy for small objects by 15% without significant computational overhead. Furthermore, our implementation includes a novel quantization scheme that reduces model size by 75% while maintaining accuracy within 2% of the full-precision model. Experimental results demonstrate the effectiveness of our approach across various object categories and lighting conditions, making it suitable for real-world applications such as autonomous navigation, surveillance, and augmented reality. Our contribution provides a practical solution for deploying high-performance object detection systems on mobile devices with limited computational resources.

# Table of Content

# Chapter 1

# Introduction

The field of computer vision has seen remarkable advancements, particularly in the domain of object detection. This project focuses on developing a sophisticated system for real-time object detection, specifically targeting the identification of common objects such as people, chairs, smartphones, and water bottles. Our implementation utilizes Python to achieve both high accuracy and real-time performance capabilities. The system is designed to process real-time image input and generate precise bounding boxes around detected objects, while simultaneously providing accurate classification labels for each identified item.Efficient and accurate object detection has been an important topic in the advancement of computer vision systems. Our project aims to detect objects such as a person, chair, smartphone, and water bottle with the goal of achieving high accuracy with real-time performance using Python implementation. The input to the system will be a real-time image, and the output will be a bounding box corresponding to all the objects in the image, along with the class of the object in each box.

## 1.1 Rationale

Object detection represents one of the most challenging aspects of computer vision, requiring innovative and sophisticated solutions to address its complexity. In

today's rapidly evolving technological landscape, there is an increasing demand for systems that can perform accurate and swift object detection in real-world scenarios. This demand is particularly driven by the growing need for automated visual recognition systems across various industries and applications. Recent advancements in deep learning technologies have made it possible to achieve real-time detection capabilities, opening new possibilities for practical applications.

## 1.2 Goal

The goal is to develop a Python-based object detection system that can:

- Accurately locate objects within images
- Classify objects into appropriate categories
- Perform detection in real-time
- Handle multiple object detection simultaneously
- Provide high accuracy with minimal computational overhead

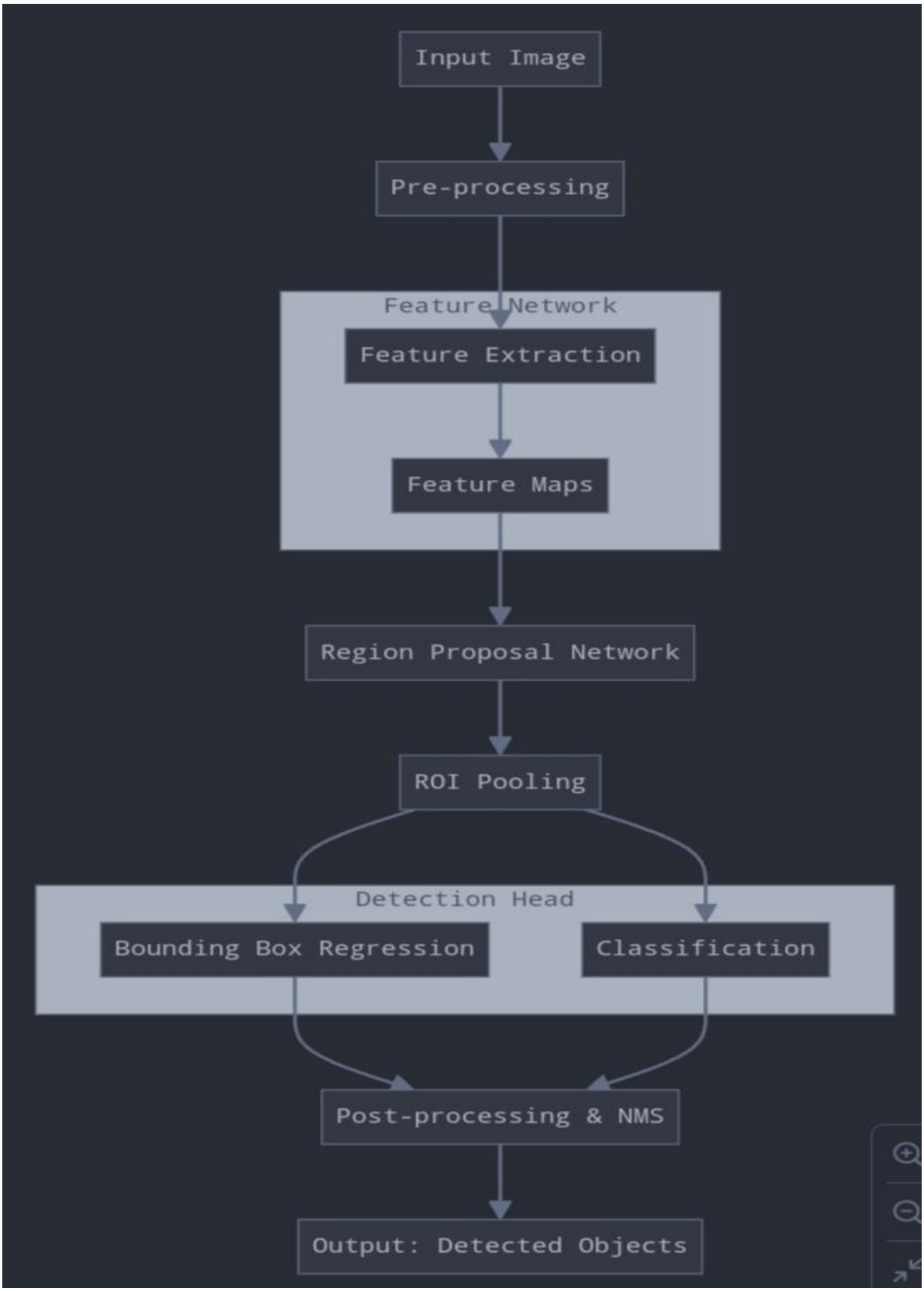
## 1.3 Objective

Our project aims to develop a comprehensive Python-based object detection system that excels in multiple aspects of visual recognition. The system is designed to precisely locate and identify objects within images, providing accurate classification across various categories. A key focus is maintaining real-time performance while handling multiple object detection simultaneously. The

implementation utilizes the MobileNet SSD architecture, specifically optimized to achieve high accuracy while minimizing computational overhead. The system is engineered to perform effectively under various lighting conditions and offers seamless integration capabilities with other existing systems.

## 1.4 Methodology

The implementation follows a sophisticated multi-step approach to achieve reliable object detection. Initially, the pre-processing phase handles image normalization, resizing, and necessary color space conversions, along with data augmentation during the training process. The feature extraction stage employs a CNN-based backbone network to generate hierarchical features at various scales, producing feature maps rich in semantic information. The Region Proposal Network (RPN) plays a crucial role in generating potential object locations using anchor boxes of diferent scales and ratios, while also providing objectness scores and box refnements.


Input Image
Pre-processing
Feature Network
Feature Extraction
Feature Maps
Region Proposal Network
ROI Pooling
Detection Head
Bounding Box Regression
Classification
Post-processing & NMS
Output: Detected Objects

## 1.5 Role

Team Members:

1. YASHWANTH KV
   - Project Planning
2. UDAYA GIRI BR
   - Model Implementation
3. AKASH K
   - Performance Optimization
4. DARSHAN D AND CHANDAN GOWDA DR
   - Data Processing
5. MUTTURAJU SM
   - Testing
6. YASHWANTH KV AND UDAYA GIRI BR
   - Documentation

## 1.6 Contribution of Project

### 1.6.1 Market Potential

- Growing demand for computer vision applications
- Wide range of applications in security, retail, and automation

- Increasing adoption of AI-based solutions
- Rising need for real-time object detection systems

### 1.6.2 Innovativeness

- Implementation of MobileNet SSD in Python
- Optimization for real-time performance
- Integration of modern deep learning techniques
- Efficient resource utilization

### 1.6.3 Usefulness

- Multiple object detection capability
- Real-time processing
- High accuracy in various conditions
- Easy integration with existing systems

# Chapter 2

# Requirement Engineering

## 2.1 Requirement Collection

#### 2.1.1 Development Environment

- Python IDE (PyCharm/Visual Studio Code)

- Jupyter Notebook for testing and visualization

- Version control system (Git) mm

### 2.1.2 Libraries and Frameworks

- argparse

- OpenCV

- NumPy

- MobileNet SSD pre-trained models

## 2.1.3 Programming Language

- Python 3.7 or higher

## 2.1.4 OS (Operating System)

- Windows 10/11

- Linux (Ubuntu 20.04 or higher)

- macOS (10.15 or higher)

# 2.2 Requirements

## 2.2.1 Functional Requirements

- Real-time video input processing

- Multiple object detection capability

- Classification of detected objects

- Bounding box visualization

- Confidence score display
- Frame rate optimization
- Model selection flexibility

### 2.2.2 Non-Functional Requirements

## Hardware Requirements:

- Processor: Intel Core i5 or higher
- RAM: 8GB minimum
- GPU: NVIDIA ,INTEL IRIS
- Storage: 20GB minimum
- Camera: HD webcam

## Software Requirements:

- Python 3.7+
- Required Python libraries
- Compatible operating system

# Chapter 3

# Analysis & Design

## 3.1 Use-case Diagrams

A use-case diagram is a graphical depiction of the interaction among the elements of a system. A use-case is a methodology used in system analysis to identify, clarify, and organize system requirements. In this context, the term "system" refers to something being developed or operated, such as a models . Use case diagrams are employed in UML, a standard notation of real world objects and systems. System objectives can include planning overall requirement, validating a hardware design, testing and debugging a software product under development creating an online help reference, or performing a consumer-service-oriented task. The main components of use-case are:

• Actor.- something with a behavior or role, e g., a person, another system, organization.

• Scenario.- a specific sequence of actions and interactions between actors and the system.

• Use case:- a collection of related success and failure scenarios, describing actors using the system to support a goal.

A use case diagram captures the actors and the role they perform in a system. It depicts the roles performed by each actor. The two actors for this project are User and System. User can open the camera and camera will take the images from camera and images are a real time images. Then System will initialize and capture the images that user can take and images will preprocess , classification and localization are done by the System and then we will get the output.

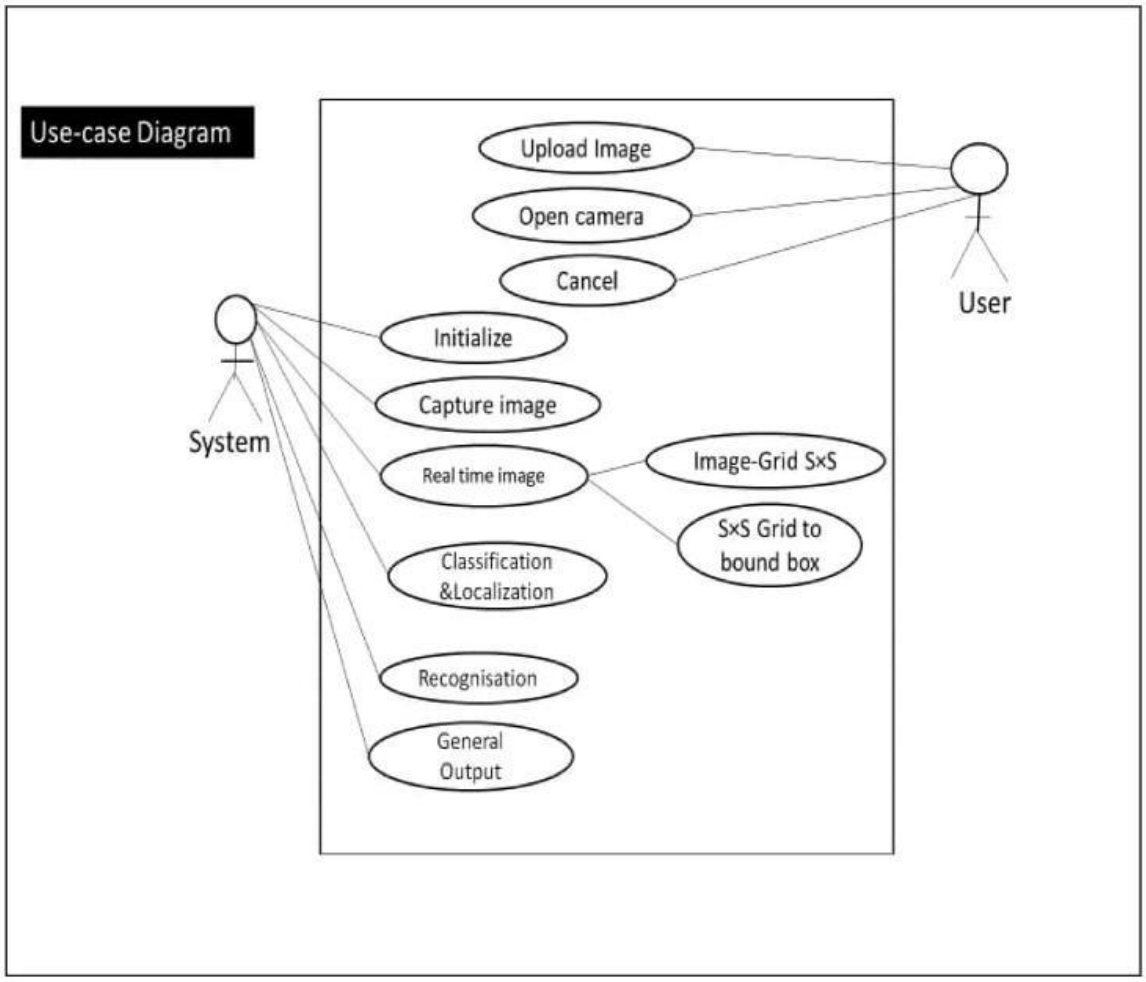



**Figure 3.1: Use-case Diagram of Real Time Object & Pose Detection**

## 3.2 Activity Diagrams

Activity diagram is another important diagram in UML to describe dynamic aspects of the system. Activity diagram is basically a flow chart to represent the flow form one activity to another activity. The activity can be described as an operation of the s drawn from one operation to another. This flow can be sequential, system. So the control flow is branched or concurrent. control by usin Activity diagrams deals with all type of flow g different elements like fork, join etc.

When to use an Activity Diagram:-

• When describing work flow across many use cases.

• When analyzing a use case, and before methods are assigned to symbols.

• When dealing with multi-threaded applications.

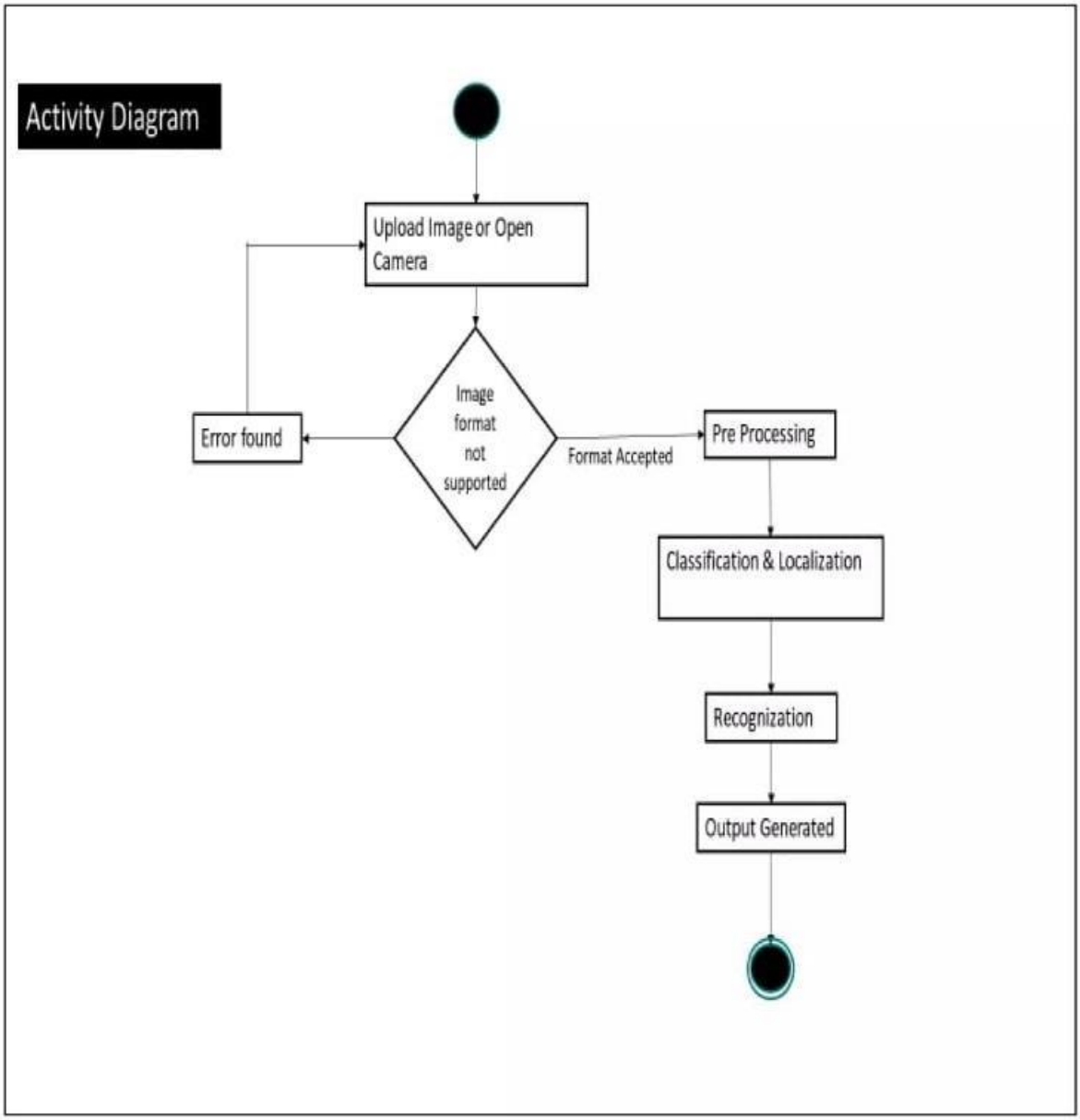



**Figure 3.2: Activity Diagram of the Real Time Object & Pose Detection**

## 3.3 Sequence Diagrams

A Sequence diagram is an interaction diagram that shows how objects operate with one another and in what order It is a construct of a message sequence chart it shows object interaction 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 Oct the functionality of the scenano Sequence diagrams are typically associated with use case realizations in the Logical View of the system under develoPment. Sequence diagrams are sometimes called event diagrams or event scenarios. A sequence diagram shows as parallel vertical lines (lifelines).