A Mini Project Synopsis on

Object Detection

B.E. - I.T Engineering

Submitted By

Rahul Sharma	**22104107**
Swayam Shah	**22104115**
Nitin Vishwakarma	**22104093**
Hamza Shaikh	**22104162**

Under The Guidance Of
Ms. Pranali Vhora

DEPARTMENT OF INFORMATION TECHNOLOGY
A.P.SHAH INSTITUTE OF TECHNOLOGY
G.B. Road, Kasarvadavali, Thane (W), Mumbai-400615
UNIVERSITY OF MUMBAI

Academic year: 2025-26

CERTIFICATE

This to certify that the Mini Project report on **"Object Detection "** has been submitted by **Rahul Sharma (22104107), Swayam Shah (22104115), Nitin Vishwakarma (22104093) and Hamza Shaikh (22104162)** who are a Bonafide students of A. P. Shah Institute of Technology, Thane, Mumbai, as a partial fulfilment of the requirement for the degree in **Information Technology**, during the academic year **2025-2026** in the satisfactory manner as per the curriculum laid down by University of Mumbai.

Ms. Pranali Vhora
Guide

External Examiner(s)
1.
2.

Place: A.P. Shah Institute of Technology, Thane
Date:

ACKNOWLEDGEMENT

This project would not have come to fruition without the invaluable help of our **Ms. Pranali Vhora**. Expressing gratitude towards our HoD, **Dr. Kiran Deshpande**, and the Department of Information Technology for providing us with the opportunity as well as the support required to pursue this project.

TABLE OF CONTENTS

1. Introduction. 1

    1.1. Purpose 1

    1.2. Objectives. 2

    1.3. Scope 2

2. Literature Survey 3

3. Problem Definition… 4

4. Proposed System… 5

    4.1. Algorithm…….. 6

5. Technology Stack 7

6. Implementation 8

7. Results… 10

8. Conclusion…. 12

9. Future Scope 13

References

Chapter 1

Introduction

Our project is an mobile application that detects everyday objects in real time. The app focuses on categories like **food, plants, home goods, and fashion items**, enabling users to recognize objects simply by capturing an image. By leveraging **deep learning**, this application brings intelligent recognition capabilities directly to smartphones, making technology more accessible for daily use. It bridges the gap between AI and real-world usability by combining accuracy, speed, and mobile optimization.

1.1 Purpose:

The main purpose of this project is to develop an intelligent object detection system capable of recognizing multiple real-world objects through a camera interface using deep learning. It aims to empower users with an easy-to-use application that can accurately detect and identify everyday items, making technology more accessible and practical in daily scenarios. Additionally, the project demonstrates how deep learning techniques can be effectively optimized for mobile platforms to ensure efficiency and performance. A key focus is to provide real-time detection capabilities that work seamlessly even without an internet connection, offering users an offline-capable, responsive, and efficient object recognition experience.

1.2 Objectives

- To design and develop an accurate yet lightweight model that runs efficiently on mobile devices.
- To train the model on diverse datasets covering food, plant, fashion, and home goods**.**
- To ensure real-time performance and smooth user experience during detection.
- To create a user-friendly interface for image capture and object visualization.
- To explore mobile deployment using TensorFlow Lite or Core ML**.**

1.3 Scope:

The scope of this project includes:

- Building and deploying an object detection system for four major categories: food, plant, fashion, and home goods**.**
- Implementation on Android with functionalities such as image capture, detection, and display of results with bounding boxes.
- Future extensions include integration with e-commerce platforms**,** voice assistance**,** and augmented reality for interactive detection and recommendations.

Chapter 2

Literature Review

Author(s)	Title	Contribution	Advantage	Disadvantage	Findings
Joseph Redmon et al.	You Only Look Once (YOLO)	Proposed real-time object detection framework	Extremely fast, single-shot detection	Requires large training data	YOLO architecture effective for real-time applications
Shaoqing Ren et al.	Faster R-CNN	Introduced region proposal networks	High detection accuracy	Computationally expensive	Achieves high mAP accuracy on multiple datasets
Tsung-Yi Lin et al.	COCO Dataset Paper	Provided benchmark dataset for object detection	Standardized dataset	Complex annotation	COCO remains widely used for CNN benchmarking
Andrew Howard et al.	MobileNet: Efficient CNN for Mobile Vision	Lightweight CNN optimized for mobile	Runs efficiently on phones	Slight trade-off in accuracy	Ideal for mobile AI applications

Table 2.1 Literature Review of Existing System

Chapter 3

Problem Definition

While several tools such as Google Lens provide object detection capabilities, they either tend to be too generic or restricted to specific domains, such as applications focused solely on food detection or plant recognition. Moreover, most of these systems are cloud-dependent**,** requiring constant internet connectivity and high computational resources to function effectively. This dependence limits their usability in low-network or offline environments.There is also the challenge of balancing performance with accessibility. Many state-of-the-art object detection models like YOLO, Faster R-CNN, or SSD achieve high accuracy but are computationally intensive, making them unsuitable for mobile devices with limited processing power and memory. As a result, users lack a lightweight, fast, and accurate tool that can provide real-time object identification without the need for high-end hardware or continuous data connectivity.To address these challenges, there is a clear need for **a** reliable, mobile-friendly, and offline-capable object detection system that can recognize multiple categories of objects with minimal latency. Such a solution should integrate deep learning techniques optimized for deployment on mobile platforms using frameworks like TensorFlow Lite or Core ML**.**

Chapter 4

Proposed System

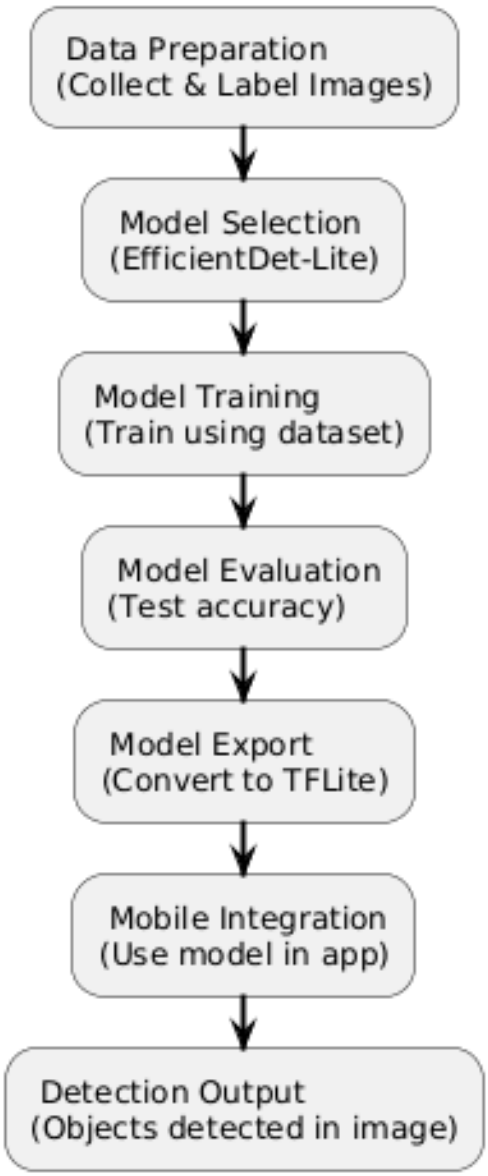

Fig 4.1 Block Diagram

4.1 Algorithm

The core algorithm is **EfficientDet-Lite**.

Steps:

Input Layer: Accepts RGB image.

1. **Convolutional Layers:** Extract features (edges, shapes, patterns).
2. **ReLU Activation:** Adds non-linearity.
3. **Pooling Layers:** Reduce spatial size and computational load.
4. **Fully Connected Layers:** Interpret extracted features.
5. **Softmax Layer:** Outputs probability of object categories.

Bounding Box Regression: Determines location of detected objects. The model is trained on labeled datasets using **cross-entropy loss** and optimized via **Adam optimizer**.

Chapter 5

Technology Stack

Frontend:

- Android for UI and image capture.

Backend / AI Model:

- **Python** with **TensorFlow** for model training.
- **TensorFlow Lite** for mobile deployment.
- **Dataset Format :** CSV with image paths and bounding box annotations.
- **Development Tools:** Android Studio
- **OpenCV** for image preprocessing.
- **tflite-model-maker -** TensorFlow Lite Model Maker for simplified mobile model training/export

Chapter 6

Implementation

```
class MainActivity : AppCompatActivity(), View.OnClickListener {
    companion object {
        const val TAG = "TFLite - ODT"
        const val REQUEST_IMAGE_CAPTURE: Int = 1
        private const val MAX_FONT_SIZE = 96F
    }

    private lateinit var captureImageFab: Button
    private lateinit var inputImageView: ImageView
    private lateinit var imgSampleOne: ImageView
    private lateinit var imgSampleTwo: ImageView
    private lateinit var imgSampleThree: ImageView
    private lateinit var tvPlaceholder: TextView
    private lateinit var currentPhotoPath: String

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        captureImageFab = findViewById(R.id.captureImageFab)
        inputImageView = findViewById(R.id.imageView)
        imgSampleOne = findViewById(R.id.imgSampleOne)
        imgSampleTwo = findViewById(R.id.imgSampleTwo)
        imgSampleThree = findViewById(R.id.imgSampleThree)
        tvPlaceholder = findViewById(R.id.tvPlaceholder)

        captureImageFab.setOnClickListener(this)
        imgSampleOne.setOnClickListener(this)
        imgSampleTwo.setOnClickListener(this)
        imgSampleThree.setOnClickListener(this)
    }

    override fun onActivityResult(requestCode: Int, resultCode: Int, d
        super.onActivityResult(requestCode, resultCode, data)
        if (requestCode == REQUEST_IMAGE_CAPTURE &&
            resultCode == Activity.RESULT_OK
        ) {
            setViewAndDetect(getCapturedImage())
        }
    }
```

Fig 6.1 Object Detection Code

This Kotlin code defines the **main activity** of an Android app that captures images and processes them using **TensorFlow Lite** for object detection. It initializes UI elements, handles button clicks, and processes the captured image in `onActivityResult()` to perform detection and display results.

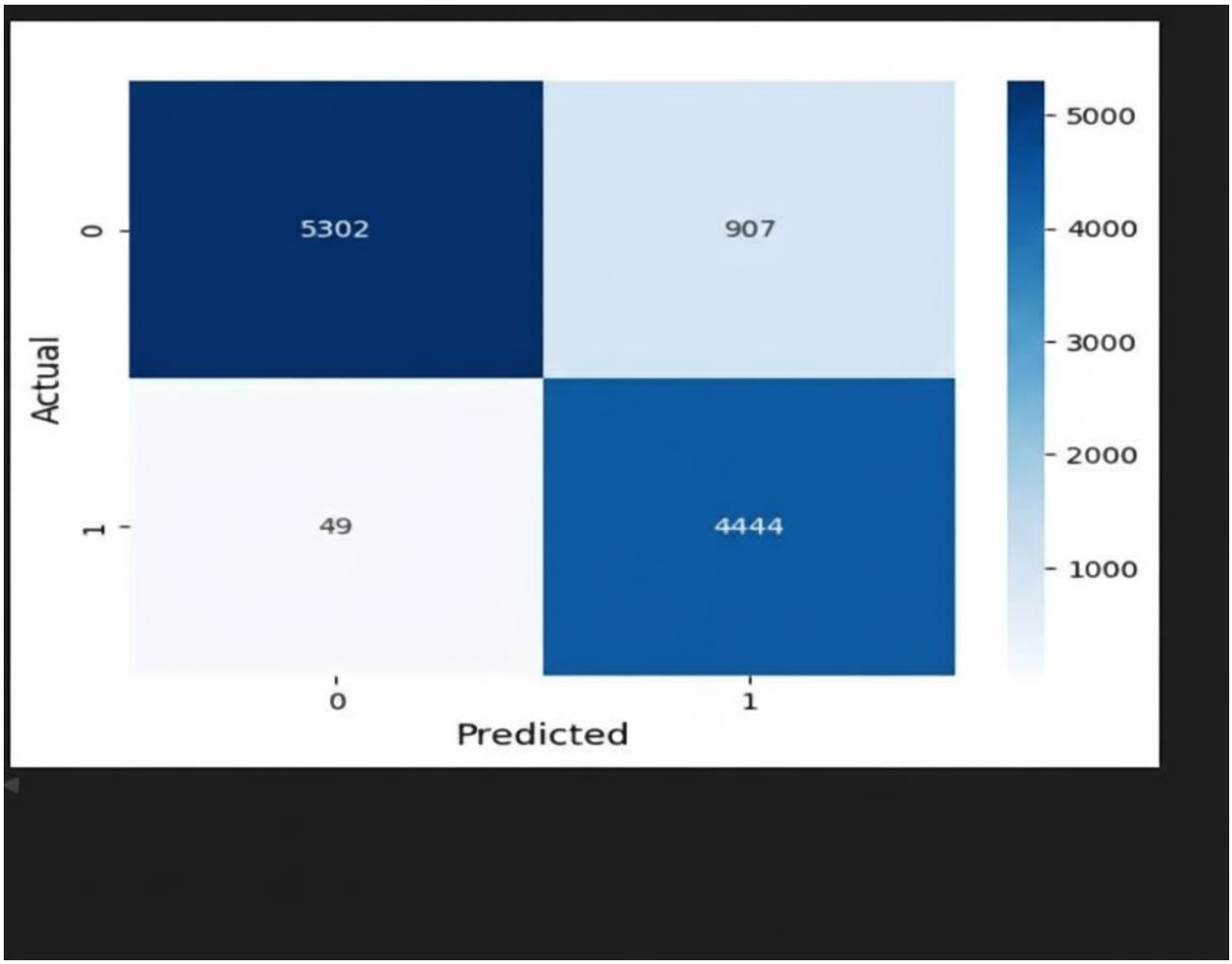

Fig 6.1 Object Detection Confusion Matrix

The confusion matrix for object detection, visually presenting true and false positives and negatives. The figure demonstrates model performance in distinguishing object labels from images, supporting the project's high accuracy metrics achieved in binary gender classification.

Chapter 7

Results

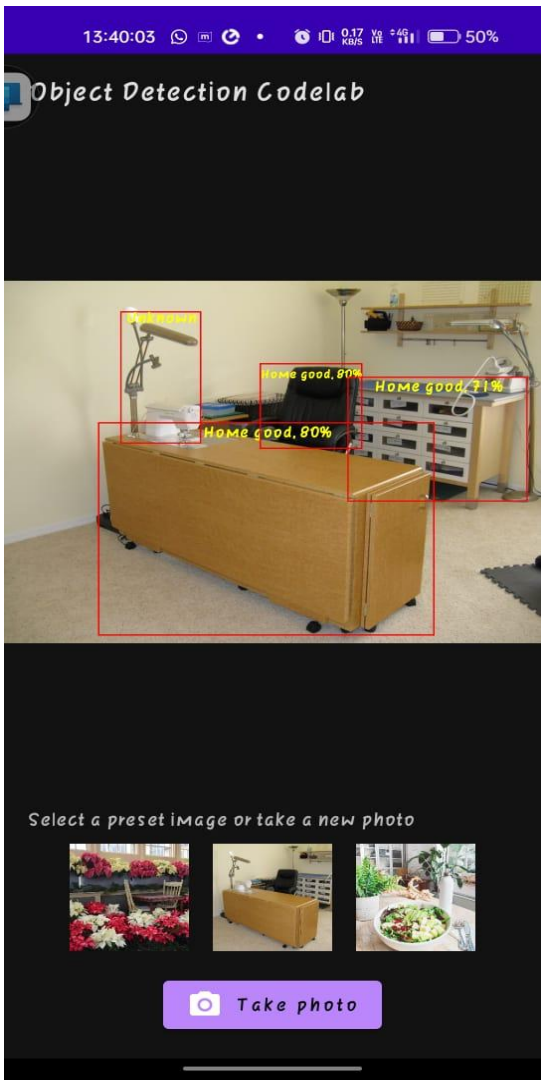

Fig 7.1 Image Analysis

Fig 7.2 Object Detection Page

Chapter 8

Conclusion

The given Kotlin code serves as a crucial component of a mobile-based object detection system built using TensorFlow Lite (TFLite) . It provides the essential functionality to capture images through the device's camera, handle user interactions, and initiate object detection once an image is obtained.

By efficiently linking the user interface with backend AI processing, this code ensures a smooth workflow — from image capture to real-time object recognition and display of results. The modular structure and event-driven design make the app responsive, lightweight, and easily extendable for future enhancements such as adding multiple detection modes, improving accuracy, or integrating cloud-based analytics.

Overall, this implementation demonstrates how deep learning models can be seamlessly deployed on mobile devices for real-time, offline object detection, combining both performance and accessibility.

Chapter 9

Future Scope

- Integrate voice assistance and augmented reality overlays.
- Enable product recommendation systems linked to detected objects.
- Implement cross-platform support for iOS devices.
- Introduce cloud synchronization for storing detection history and analytics.
- Expand to more object categories and domains.

References

1. Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun, "*Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks*," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2017.

2. Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick, "*Microsoft COCO: Common Objects in Context*," *European Conference on Computer Vision (ECCV)*, 2014.

3. Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam, "*MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications*," *arXiv preprint arXiv:1704.04861*, 2017.

4. TensorFlow Developers, "*TensorFlow Documentation*," Google Research, 2024.

5. OpenCV Developers, "*OpenCV-Python Documentation*," OpenCV.org, 2025.