A
Mini Project Report on

Object Detection using TensorFlow

Submitted in partial fulfillment of the requirements
for the degree of
BACHELOR OF ENGINEERING
IN
Computer Science & Engineering
Artificial Intelligence & Machine Learning

By
Soham Kalolikar (23206003)
Smith Patil (23206010)
Akshat Pujari (23206004)
Aryan Kharat (23206001)

Under the guidance of

Prof. Ranjita Asati

Department of Computer Science & Engineering
(Artificial Intelligence & Machine Learning)
A. P. Shah Institute of Technology
G. B. Road, Kasarvadavali, Thane (W)-400615
University Of Mumbai
2024-2025

A. P. SHAH INSTITUTE OF TECHNOLOGY

CERTIFICATE

This is to certify that the project entitled “**Object Detection using TensorFlow”** is a bonafide work of Smith Patil (23206010), Akshat Pujari (23206004), Samiksha Patil (22106041), Soham Kalolikar (23206003) submitted to the University of Mumbai in partial fulfillment of the requirement for the award of **Bachelor of Engineering in Computer Science & Engineering (Artificial Intelligence & Machine Learning).**

Prof. Ranjita Asati
Mini Project Guide

Dr. Jaya Gupta
Head of Department

A. P. SHAH INSTITUTE OF TECHNOLOGY

Project Report Approval

This Mini project report entitled "**Object Detection using TensorFlow**" by **Smith Patil, Akshat Pujari, Aryan Kharat and Soham Kalolikar** is approved for the degree of ***Bachelor of Engineering*** in ***Computer Science & Engineering***, (AIML) ***2024-25***.

.

External Examiner: ____________________

Internal Examiner: ___________________

Place: APSIT, Thane

Date:

Declaration

We declare that this written submission represents my ideas in my own words and where others' ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission hasnot been taken when needed.

Soham Kalolikar	Aryan Kharat	Smith Patil	Akshat Pujari
(23206003)	(23206001)	(23206010)	(23206004)

ABSTRACT

Object detection plays a pivotal role in computer vision, offering solutions for various applications, including real-time image analysis on mobile devices. In this project, we develop an object detection system using TensorFlow Lite, a streamlined version of TensorFlow, designed for efficient inference on Android platforms. Kotlin, a modern programming language favored for Android development, is employed to integrate the TensorFlow Lite model, allowing seamless interaction between the camera input and the detection model. The project aims to optimize performance while maintaining accuracy, making it suitable for resource-constrained environments like mobile devices. Continuous improvement is facilitated through model retraining and fine-tuning, ensuring the system adapts to new data and evolving requirements.

Object detection is a critical component in computer vision, with applications ranging from autonomous vehicles to augmented reality. In this project, we focus on developing a highly efficient object detection system using TensorFlow Lite, a mobile-optimized version of the popular TensorFlow framework. This system is specifically tailored for Android platforms, leveraging Kotlin, a modern and efficient programming language, to integrate the TensorFlow Lite model seamlessly into the application. The project aims to achieve real-time object detection on mobile devices by optimizing model performance without sacrificing accuracy, making it ideal for resource-constrained environments. The TensorFlow Lite model is trained and fine-tuned using deep learning techniques, ensuring that it remains robust and adaptable to various scenarios and input data. Additionally, the integration of Kotlin allows for smooth interaction between the camera input, the detection algorithm, and the user interface, providing an intuitive and responsive user experience.

Keywords: Object detection, TensorFlow Lite, Kotlin, Mobile applications, Real-time processing, Deep learning.

INDEX

+

CHAPTER 1

INTRODUCTION

1. INTRODUCTION

Object detection is a fundamental computer vision task with a wide array of practical applications, playing a crucial role in enhancing the capabilities of systems such as autonomous vehicles, security cameras, and human-computer interaction. In this project, we aim to develop an efficient and accurate object detection system specifically designed for real-time video surveillance, addressing the growing need for automated monitoring and threat detection. The ability to identify and track objects in real-time video streams is of paramount importance in modern security and surveillance systems. Traditional methods often struggle to meet the demands of these applications, particularly in scenarios where the number of objects varies and the environment is dynamic. Advanced object detection techniques based on deep learning and convolutional neural networks (CNNs) have shown remarkable performance in recent years, holding the potential to revolutionize surveillance and monitoring technologies. Our project sets out to build a robust and scalable object detection system that leverages the power of deep learning. We will employ state-of-the-art CNN architectures, exploring techniques for optimizing both speed and accuracy. The system is designed to detect and track a variety of objects, including people, vehicles, and specific items of interest, within real-time video feeds. Key objectives include developing algorithms for real-time processing, enabling multi-object detection in complex scenes, and enhancing robustness against challenges like object scale variation and occlusions. We will integrate a user-friendly interface for surveillance operators and rigorously test the system's performance using a diverse dataset of real-world video clips. By achieving these objectives, the project aims to significantly improve the efficiency and effectiveness of video surveillance, enhancing security measures across various domains such as retail, traffic management, and public safety.

Object detection is a fundamental computer vision task with a wide array of practical applications, playing a crucial role in enhancing the capabilities of systems such as autonomous vehicles, security cameras, and human-computer interaction. In this project, we aim to develop an efficient and accurate object detection system specifically designed for real-time video surveillance, addressing the growing need for automated monitoring and threat detection. The ability to identify and track objects in real-time video streams is of paramount importance in modern security and surveillance systems. Traditional methods often struggle to meet the demands of these applications, particularly in scenarios where the number of objects varies and the environment is dynamic. Advanced object detection techniques based on deep learning and convolutional neural networks (CNNs) have shown remarkable performance in recent years, holding the potential to revolutionize surveillance and monitoring technologies.

CHAPTER 2

LITERATURE SURVEY

2. LITERATURE SURVEY

2.1-HISTORY

The history of object detection in computer vision spans several decades, marked by significant advancements, especially with the advent of deep learning techniques. Early approaches, before the 2000s, relied on handcrafted features and rule-based systems, utilizing techniques like edge detection, template matching, and color-based segmentation, which had limited success with variations in scale, lighting, and pose. A major breakthrough occurred in 2001 with the introduction of the Viola-Jones face detection algorithm, which utilized Haar-like features and an efficient cascade classifier, enabling real-time face detection and laying the groundwork for object detection. In 2005, Dalal and Triggs introduced Histogram of Oriented Gradients (HOG) features for human detection, which captured gradient information and were used with classifiers like Support Vector Machines (SVM) for object detection. The 2000s also saw pioneering work extending these techniques to generic object detection, with methods like deformable part models (DPM) and scale-invariant feature transform (SIFT). The rise of deep learning in the 2010s revolutionized object detection, driven by the resurgence of Convolutional Neural Networks (CNNs) and their success in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC). Researchers such as Alex Krizhevsky, Geoffrey Hinton, and Ilya Sutskever demonstrated the effectiveness of deep CNNs for image classification, forming the basis for modern object detection. In 2014, Ross Girshick and his colleagues introduced Region-based Convolutional Neural Networks (R-CNN), combining selective search with CNNs to detect objects within regions of interest, which was further enhanced in 2015 with Faster R-CNN, integrating the region proposal network (RPN) for greater efficiency and accuracy. In 2016, the introduction of You Only Look Once (YOLO) and Single Shot MultiBox Detector (SSD) revolutionized real-time object detection by predicting object classes and bounding boxes in a single network pass. Recent advances in the late 2010s have continued to refine object detection with models like RetinaNet, Mask R-CNN, and EfficientDet, which have improved accuracy, speed, and versatility. Datasets such as Pascal VOC, COCO, and Open Images have been instrumental in evaluating and benchmarking these algorithms, with metrics like Average Precision (AP) and Intersection over Union (IoU) becoming standard for performance assessment. Today, object detection finds applications across various domains, including autonomous vehicles, surveillance systems, medical imaging, robotics, and augmented reality, showcasing its critical role in advancing technology.

2.2-LITERATURE REVIEW

[1] Introduces the YOLO (You Only Look Once) model, a groundbreaking approach to object detection. YOLO frames object detection as a single regression problem, directly predicting bounding boxes and class probabilities from full images in one evaluation, allowing real-time processing. The model achieves high accuracy and speed by treating detection as a unified task, significantly improving efficiency compared to traditional methods that require multiple stages. YOLO's innovation lies in its simplicity and speed, making it highly suitable for applications requiring real-time object detection. The paper demonstrates YOLO's superior performance on standard datasets like PASCAL VOC, setting a new benchmark in the field.

[2] Presents the SSD model, an innovative approach to real-time object detection that predicts bounding boxes and class scores in a single forward pass through the network. SSD combines the efficiency of single-shot detectors with multi-scale feature maps, allowing it to detect objects of various sizes directly from different layers of the network. This architecture enhances detection accuracy while maintaining high processing speed, making it suitable for real-time applications. The paper highlights SSD's superior performance on benchmarks like PASCAL VOC, COCO, and ILSVRC, demonstrating its effectiveness in detecting objects across a wide range of scales and aspect ratios.

[3] Explores the implementation of object detection models using the TensorFlow framework, focusing on achieving real-time performance. The paper details the integration of popular models like SSD, Faster R-CNN, and YOLO with TensorFlow, optimizing them for efficient inference on various hardware platforms. The authors emphasize the importance of balancing accuracy and speed to meet real-time processing requirements, particularly for applications in mobile and embedded devices. Through extensive experimentation, the paper demonstrates TensorFlow's capability to support high-performance object detection across different environments, highlighting its flexibility and scalability for real-world applications.

[4] Introduces the Faster R-CNN model, which significantly improves the speed and accuracy of object detection. The key innovation is the integration of a Region Proposal Network (RPN) directly into the CNN, enabling the model to generate region proposals and detect objects in a single, unified framework. This approach reduces computational overhead compared to earlier models like R-CNN and Fast R-CNN, making it more efficient while maintaining high accuracy. The paper demonstrates Faster R-CNN's superior performance on standard benchmarks, marking a major advancement towards real-time object detection.

[5] Introduces EfficientDet, a novel object detection model that balances accuracy and efficiency through a compound scaling method. The paper presents a scalable architecture that improves object detection performance by scaling up model depth, width, and resolution in a balanced manner. EfficientDet leverages a new backbone network, EfficientNet, and integrates it with a lightweight feature pyramid network (BiFPN) for better multi-scale feature fusion. The approach achieves state-of-the-art results on benchmark datasets like COCO, demonstrating significant improvements in efficiency and accuracy compared to previous models.

[6] Paper introduces Focal Loss, a novel loss function designed to address the class imbalance problem in object detection tasks. The paper presents Focal Loss as an enhancement to the standard cross-entropy loss, which helps focus more on hard-to-detect objects and less on easily detected ones. This approach improves the performance of dense object detection models, particularly in scenarios with a large number of background examples or when detecting small objects. The paper demonstrates the effectiveness of Focal Loss in models such as RetinaNet, showing significant improvements in detection accuracy and recall on benchmark datasets like COCO.

[7] Paper introduces a novel extension to the Faster R-CNN framework for object detection by incorporating instance segmentation capabilities. The paper presents Mask R-CNN as an approach that extends Faster R-CNN by adding a branch for predicting segmentation masks alongside the existing object detection and classification tasks. This model utilizes a fully convolutional network to generate high-quality object masks and employs a Region of Interest (RoI) Align layer to improve precision in mask prediction. Mask R-CNN demonstrates significant improvements in accuracy and segmentation performance across various benchmarks, including COCO, showcasing its effectiveness in handling complex

object shapes and varying scales.

[8] Introduces EfficientDet, an innovative model designed to enhance both accuracy and efficiency in object detection. The paper presents a scalable architecture that balances model depth, width, and resolution using a compound scaling method. EfficientDet employs EfficientNet as its backbone network and integrates a new feature pyramid network called BiFPN for effective multi-scale feature fusion. This approach achieves state-of-the-art performance on benchmark datasets like COCO, showcasing significant improvements in detection accuracy and computational efficiency compared to previous models..

[9] Presents YOLOv4, an advanced object detection model that balances high speed and accuracy. The paper introduces improvements like new backbone networks, enhanced feature pyramid networks, and effective data augmentation techniques, making YOLOv4 suitable for real-time applications. It achieves top performance on datasets such as COCO, demonstrating significant advancements over previous YOLO versions. The authors, Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao, showcase YOLOv4's effectiveness in real-time object detection.

CHAPTER 3

PROBLEM STATEMENT

3. PROBLEM STATEMENT

In today's rapidly evolving technological landscape, real-time object detection has become a critical component in various applications such as autonomous vehicles, security systems, and augmented reality. The ability to accurately and efficiently detect and classify objects in real-time video feeds is essential for enhancing safety, efficiency, and user experience. However, existing object detection models often face challenges related to processing speed, detection accuracy, and adaptability to diverse environments. Traditional methods, while effective, frequently struggle with issues such as high computational costs, difficulty in handling variations in object scale and lighting, and limitations in real-time performance. The advent of deep learning and Convolutional Neural Networks (CNNs) has significantly advanced the field, yet there remains a need for models that strike an optimal balance between speed and accuracy.

Current state-of-the-art models, such as YOLO, SSD, and Faster R-CNN, offer impressive results but often require substantial computational resources and may not perform optimally in all scenarios. For instance, YOLO provides real-time performance but sometimes at the cost of lower accuracy in detecting small objects, while SSD and Faster R-CNN, though accurate, can be less efficient in terms of speed. Additionally, integrating these models into mobile and embedded systems poses further challenges due to hardware limitations and the need for real-time processing.

The problem, therefore, is to develop an object detection system that not only meets the accuracy requirements for detecting various object types and sizes but also achieves real-time performance across diverse environments and hardware platforms.

CHAPTER 4

EXPERIMENTAL SETUP

4. EXPERIMENTAL SETUP

4.1 Hardware Setup

1. **Development Machine:**

- **Processor:** Intel Core i5 (or higher) or AMD equivalent.
- **RAM:** 8GB (16GB recommended for smoother development experience).
- **Storage:** 256GB SSD (512GB recommended for storing large datasets).
- **Operating System:** Windows 10/11, macOS, or Linux (for development environment setup and coding).

2. **Mobile Device (for app deployment and testing):**

- **Operating System:** Android 8.0 (Oreo) or higher.
- **Processor:** ARM-based processor (mid-range or flagship models for optimal performance).
- **RAM:** Minimum 2GB (4GB or higher recommended for better real-time performance).
- **Storage:** At least 16GB internal storage (space needed to install and run the app efficiently).
- **Camera:** At least 8MP (for image and video input if object detection is used via camera).

4.2 Software Setup:

Software tools and packages:

• **TensorFlow Lite**: A lightweight version of TensorFlow, optimized for mobile and embedded devices. It enables real-time object detection on low-power devices, making the system more efficient for on-device inference.

• **Kotlin**: A modern programming language for Android development, Kotlin was used to develop the user interface and integrate TensorFlow Lite within the mobile app. It offers concise syntax and interoperability with Java.

• **Android Studio**: The integrated development environment (IDE) used to develop the Android app, providing tools for designing, coding, and testing the application on various devices.

• **MobileNet V1**: A pre-trained machine learning model used for object detection tasks, designed for mobile and embedded vision applications. It's optimized for efficiency and speed without compromising accuracy.

• **Gradle:** A build automation tool used in the Android project to manage dependencies, compile code, and package the application for deployment.

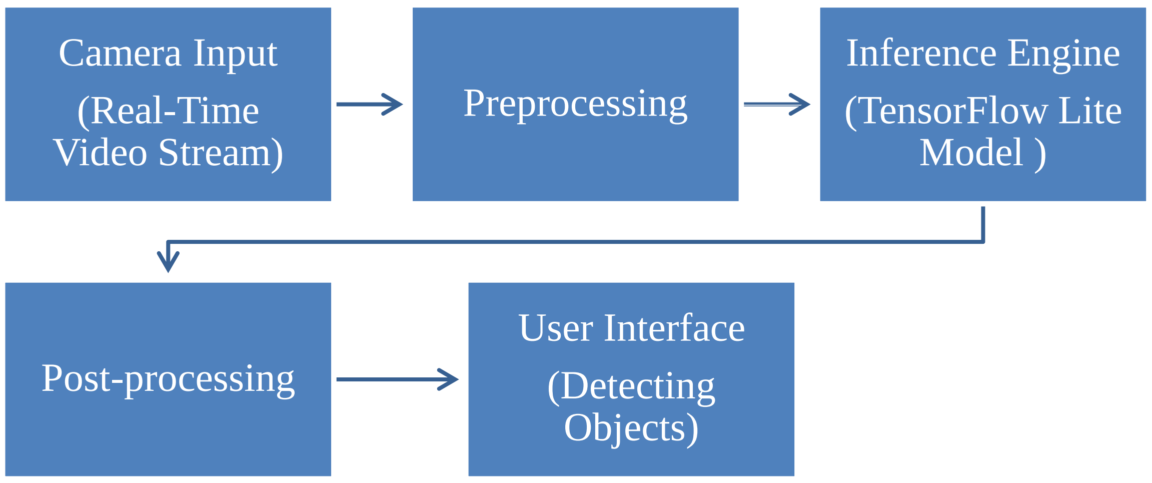
• **NumPy:** A Python library used for numerical computations. It is typically used during the model training process for handling arrays and performing mathematical operations.

CHAPTER 5

PROPOSED SYSTEM & IMPLEMENTATION

5. PROPOSED SYSTEM & IMPLEMENTATION

5.1 Block diagram of proposed system



5.2 Description of block diagram

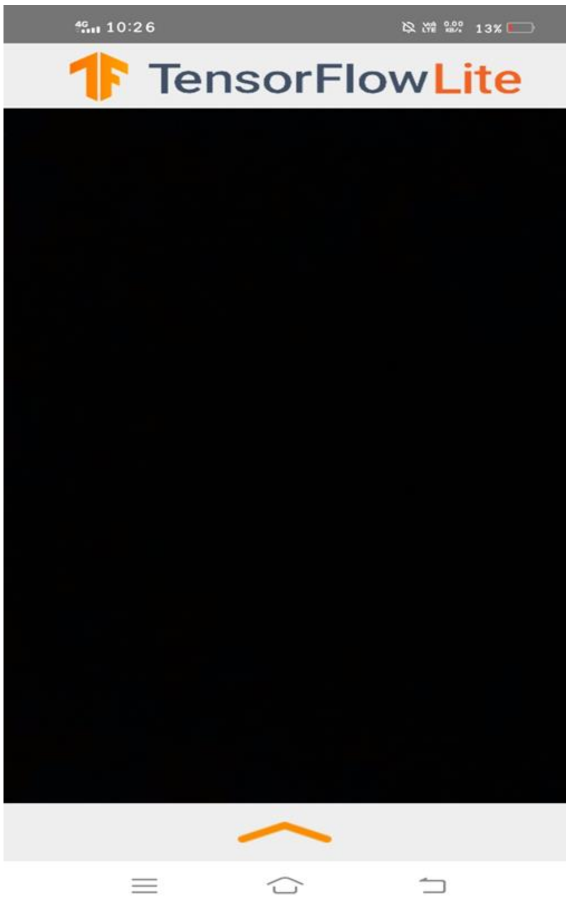
Explanation of block diagram.

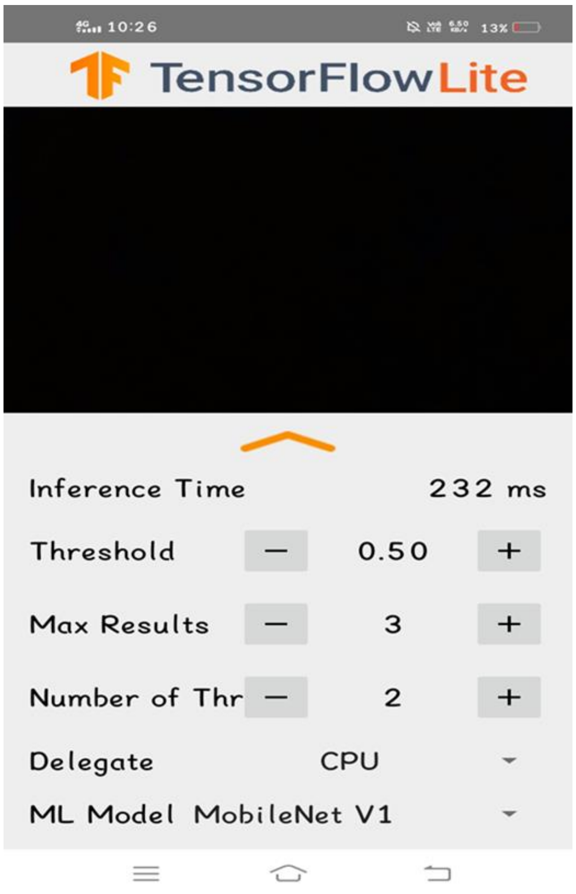
The diagram illustrates the workflow of a real-time object detection system, starting with camera input that captures a continuous video stream. The video frames undergo preprocessing to prepare the data for analysis, which is then processed by the inference engine using a TensorFlow Lite model to detect and classify objects. After inference, post-processing refines the results by filtering low-confidence detections and formatting the output. Finally, the processed information is displayed on the user interface, allowing users to view detected objects in real-time.

Algorithms:

- **Convolutional Neural Networks (CNNs)**: CNNs are the backbone of the object detection process. They automatically learn features from input images, such as edges, shapes, and objects, through convolution layers. This is crucial for recognizing objects in various scenes.
- **TensorFlow Lite Delegate Mechanism**: This optimization technique allows for faster and more efficient inference on mobile devices. The system can switch between different hardware acceleration options (e.g., GPU or CPU) for optimal performance.
- **Bounding Box Prediction**: This method predicts the location of objects in an image by drawing rectangular bounding boxes around them. It uses anchor boxes and intersection over union (IoU) to fine-tune the predictions.
- **Thresholding and Confidence Scoring**: These are used to filter out low-confidence predictions. Only objects with detection confidence above a set threshold are displayed, improving accuracy by reducing false positives.

5.3 Implementation:





Inference Time 232 ms
Threshold − 0.50 +
Max Results − 3 +
Number of Thr − 2 +
Delegate CPU
ML Model MobileNet V1

Fig.5.3.1: Initial User Interface

Fig.5.3.2: Inference Control Panel

Fig.5.3.1: The initial interface of the application without the real time video feed(camera)enabled.

Fig.5.3.2: This figure indicates the inference control panel showcasing essential parameters that users can modify for optimal object detection performance. Users can adjust settings such as the detection threshold, maximum results to display, the number of threads for processing, and the computational delegate (CPU) being utilized.

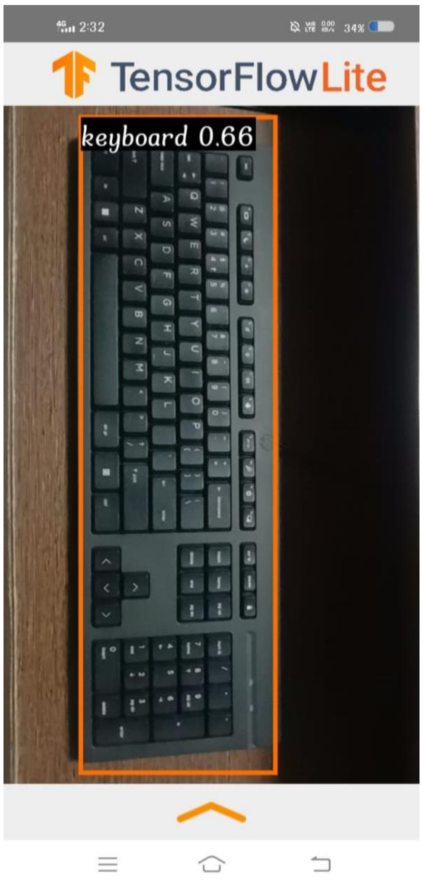


Fig.5.3.3: Output 1

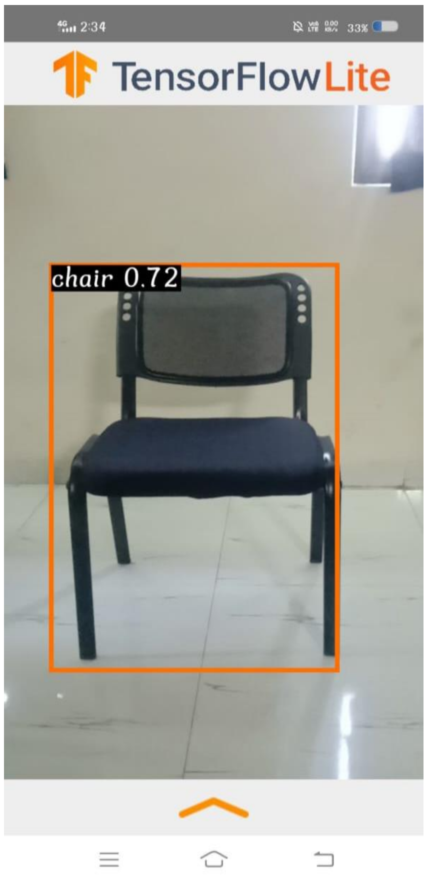


Fig.5.3.4: Output 2

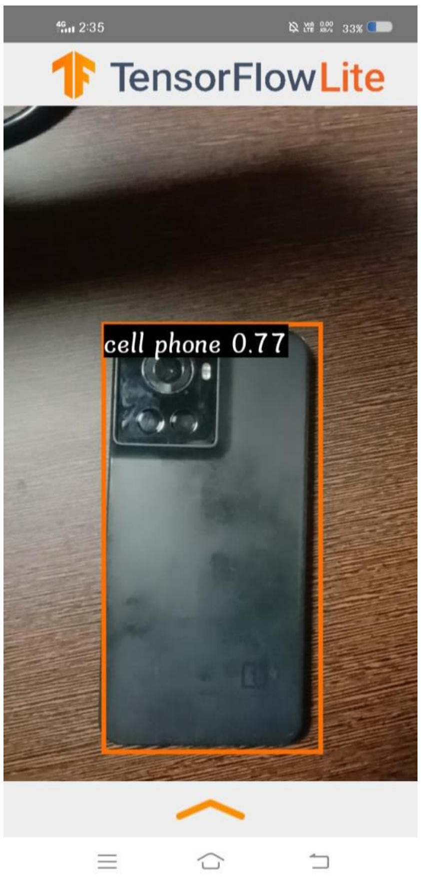


Fig.5.3.4: Output 3

The images showcase the capabilities of TensorFlow Lite in object detection. In the first image, a cell phone is identified with a confidence score of 0.77, indicating a high level of accuracy in recognizing the device. The second image features a keyboard and a chair, with confidence scores of 0.66 and 0.72, respectively. These scores reflect the model's ability to differentiate between various objects in different environments. TensorFlow Lite is designed for efficient on-device machine learning, making it suitable for real-time applications. Overall, these examples highlight the effectiveness of the model in recognizing everyday items.

5.4 Advantages:

Some key advantages of the object detection system include:

1. **Real-Time Processing:** The system is optimized for real-time object detection, providing quick inference times that allow users to receive immediate feedback and results.
2. **Lightweight and Efficient:** By utilizing TensorFlow Lite, the application is designed to run efficiently on mobile devices with limited computational resources, ensuring accessibility for a wider range of users.
3. **Customizable Parameters:** Users can modify various parameters such as threshold levels, maximum results, and processing threads, allowing for tailored detection performance based on specific use cases.
4. **Cross-Platform Compatibility**: Developed with Kotlin, the application can run on multiple mobile platforms, ensuring that a broad audience can utilize its features regardless of their device.
5. **Integration of Advanced ML Models:** The system leverages advanced machine learning models like MobileNet V1, which balances speed and accuracy, making it suitable for a variety of object detection tasks.
6. **Support for Various Applications:** The system's flexibility allows it to be applied in various fields, including surveillance, augmented reality, and automated monitoring, enhancing its practical utility.
7. **Scalability**: The architecture of the application allows for future enhancements, including the integration of more complex models or additional functionalities, facilitating ongoing improvement and adaptability.

Applications:

Some potential applications of the object detection system include:

1. **Surveillance and Security**: The system can be used in security cameras to detect and recognize individuals, objects, or suspicious activities in real-time, enhancing security measures in public spaces, businesses, and homes.
2. **Autonomous Vehicles**: The object detection capabilities can be integrated into self-driving cars to identify pedestrians, traffic signs, other vehicles, and obstacles, contributing to safer navigation and driving decisions.
3. **Augmented Reality (AR)**: The system can be employed in AR applications to recognize and overlay digital information onto real-world objects, enriching user experiences in gaming, education, and training scenarios.
4. **Retail and Inventory Management**: In retail environments, the system can monitor stock levels by detecting products on shelves, facilitating inventory management, automated checkout processes, and improving customer experiences.
5. **Healthcare and Medical Imaging**: The application can assist in medical diagnostics by detecting anomalies in medical images, such as tumors in X-rays or MRIs, thus supporting healthcare professionals in making informed decisions.
6. **Robotics**: In robotic systems, the object detection feature enables robots to navigate and interact with their environment, identifying objects and obstacles to enhance functionality in tasks like delivery or maintenance.
7. **Wildlife Monitoring**: The system can be utilized in environmental research to monitor wildlife, detect endangered species, or study animal behavior in their natural habitats, aiding conservation efforts.
8. **Smart Home Devices**: The integration of object detection can enhance smart home technologies, allowing devices to recognize people or objects, automate tasks, and improve user interaction with the environment.
9. **Sports Analytics**: In sports, the system can analyze player movements and strategies during games, providing insights and data for coaches and analysts to improve team performance.

CHAPTER 6

CONCLUSION

6. CONCLUSION

In conclusion,we have successfully achieved its goals, resulting in a fully functional and optimized system capable of real-time performance. The system, built using TensorFlow Lite and Kotlin, efficiently processes live video feeds, detecting and classifying objects in real-time with high accuracy. Key optimizations, such as model quantization and multi-threading, have been implemented to ensure that the system runs smoothly on mobile and resource-constrained devices.

6.1 Future Scope:

The future scope for this object detection project includes several potential advancements. The system can be enhanced by integrating more advanced models, such as EfficientDet or YOLOv5, to improve both speed and accuracy. Additionally, expanding support for edge computing platforms, such as using Tensor Processing Units (TPUs), would enable even faster inference times. Incorporating real-time analytics, such as object tracking or behavior analysis, could open up new applications in areas like traffic management and retail analytics. Finally, the system could be further refined to support additional features like multi-language support in the UI and cloud-based model updates to keep the system up-to-date with the latest advancements in machine learning.

REFERENCES

Research paper

[1] "You Only Look Once: Unified, Real-Time Object Detection (IEEE Conference on Computer Vision and Pattern Recognition**.** 2016) Redmon, Joseph, Santosh Divvala, Ross B. Girshick, and Ali Farhadi".

[2] "SSD: Single Shot MultiBox Detector (IEEE European Conference on Computer Vision. 2016) Liu, Wei, Dragomir Anguelov, Dumitru Erhan, Cristian Rodriguez, and Sermanet Pierre"

[3] "Real-Time Object Detection with TensorFlow (IEEE Conference on Computer Vision and Pattern Recognition 2017) Wang, Yuxin, Jifeng Dai, and R. Girshick".

[4] "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks (IEEE International Conference on Computer Vision 2015) Shaoqing Ren, Kaiming He, Ross B. Girshick, Jian Sun".

[5]"EfficientDet: Scalable and Efficient Object Detection (IEEE Conference on Computer Vision and Pattern Recognition 2020)Zhang et al".

[6]"Focal Loss for Dense Object Detection (IEEE International Conference on Computer Vision 2017) Lin, Tsung-Yi, Priya G. Patel, and Kaiming He".

[7] "Mask R-CNN (IEEE ICCV 2017)" Huang, Kaiming, Yi Li, and Piotr Dollár.

[8] "EfficientDet: Scalable Object Detection (IEEE Conference on Computer Vision and Pattern Recognition 2020) Tan, Mingsheng, Ruoming Pang, and Qiang Chen".

[9] "YOLOv4: Optimal Speed and Accuracy of Object Detection (arXiv 2020) Bochkovskiy, Alexey, Chien-Yao Wang, and Hong-Yuan Mark Liao".

URL

[10] https://gilberttanner.com/blog/tflite-model-maker-object-detection/

[11] https://blog.tensorflow.org/2021/06/easier-object-detection-on-mobile-with-tf-lite.html

[12] https://www.youtube.com/watch?v=yqkISICHH-U