Mobile Real-Time Object Detection with SSD MobileNet

Govind Kumar[1], Arnav Rathi[2], Ashutosh Kumar Yadav[3] ,Himani Tyagi[4]

[1]Department of Artificial Intelligence and Data Science, University School of Automation and Robotics, Delhi, India
[2] Department of Artificial Intelligence and Data Science, University School of Automation and Robotics, Delhi, India
[3] Department of Artificial Intelligence and Data Science, University School of Automation and Robotics, Delhi, India
[4] Department of Artificial Intelligence and Data Science, University School of Automation and Robotics, Delhi, India

Email: [1]govindrajoria97@gmail.com, [2]arnavrathi30@gmail.com, [3]441yadavashutosh@gmail.com, [4]himanityagi.usar@ipu.ac.in

ABSTRACT:
Recent advancements in deep learning have significantly transformed object detection, surpassing conventional methods such as Haar cascades and Histogram of Oriented Gradients (HOG) in accuracy, efficiency, and adaptability. This paper presents a systematic study of deep learning-based object detection techniques, with a focus on real-time implementation for mobile platforms. A comparative analysis highlights the strengths of modern architectures, including Faster R-CNN, You Only Look Once (YOLO), and Single Shot MultiBox Detector (SSD) with MobileNet, over classical approaches.

The primary contribution of this work is the design and evaluation of an optimized SSD MobileNet-based object detection system for Android, utilizing TensorFlow Lite for efficient on-device inference. Experimental results demonstrate that the proposed solution achieves real-time performance while maintaining robust detection accuracy. Key challenges, such as computational latency, model size constraints, and multi-scale object variability, are addressed through techniques like post-training quantization and hardware acceleration.

This study serves as both a technical reference and a practical guide for implementing real-time object detection on resource-constrained devices, offering insights into trade-offs between speed and precision.

INDEX TERMS: Real-time object detection, Mobile computer vision, SSD-MobileNet, TensorFlow Lite, Android optimization, Camera2 API

1) Introduction

We Modern deep learning has transformed object detection, moving beyond traditional methods like Haar cascades and HOG that relied on manual feature engineering. While powerful models like Faster R-CNN and YOLO exist, their high computational cost makes them impractical for mobile devices.

This work implements an optimized SSD MobileNet system on Android using TensorFlow Lite, addressing mobile constraints through quantization and hardware acceleration. The solution achieves real-time performance (30+ FPS) without significant accuracy loss, demonstrating practical viability for resource-limited applications.

2) Literature Review

There 2.1 Overview of Object Detection Evolution
The field of object detection has undergone significant transformation with the emergence of deep learning techniques. Early computer vision systems depended on manually engineered features through approaches like Haar cascades and HOG descriptors. While these methods demonstrated reasonable performance for specific tasks, their inability to adapt to diverse real-world conditions prompted the development of more sophisticated solutions. Modern deep learning architectures, particularly CNNs, have established new benchmarks by automatically learning discriminative features directly from data. Among contemporary solutions, SSD MobileNet has gained prominence for mobile implementations due to its favorable speed-accuracy trade-off.

2.2 Conventional Approaches and Their Constraints
Traditional detection frameworks employed two distinct phases: feature extraction followed by classification. The Haar cascade method, while computationally inexpensive, exhibited sensitivity to lighting variations and occlusions. Similarly, the HOG-SVM pipeline, though effective for structured scenarios like pedestrian detection, struggled with complex object variations. These limitations primarily stemmed from their reliance on predetermined feature representations, which failed to capture the richness of natural scenes.

2.3 Deep Learning Paradigms in Object Detection

Modern detection frameworks fall into two principal categories: two-stage and single-stage detectors. Two-stage detectors like Faster R-CNN first generate region proposals before performing classification, yielding high precision at the cost of computational intensity. Single-stage approaches such as YOLO and SSD address the speed limitation by unifying localization and classification into a single network pass. While YOLO variants have achieved remarkable speeds, their detection performance for small objects remains suboptimal. The SSD architecture, particularly when paired with MobileNet's efficient operations, presents an attractive compromise between these competing demands.

2.4 Rationale for SSD MobileNet Selection

The choice of SSD MobileNet for mobile deployment stems from several architectural advantages:

- Computational efficiency through depthwise separable convolutions
- Multi-scale detection capabilities via pyramidal feature hierarchy
- Compatibility with mobile acceleration frameworks
- Balanced memory footprint and detection performance

Recent benchmarking studies demonstrate the model's ability to maintain 25-30 FPS on mid-range mobile hardware while achieving >70% mAP on standard datasets, making it particularly suitable for real-time applications.

Comparative Analysis of Detection Approaches

Approach	**Inference Speed**	**Accuracy**	**Hardware Demands**	**Mobile Suitability**
Haar Cascades	High	Low	Minimal	Limited
HOG-SVM	Moderate	Medium	Low	Partial
Faster R-CNN	Low	High	Significant	Unsuitable
YOLO Variants	Medium-High	Medium	Moderate	Conditional
SSD MobileNet	High	Good	Optimized	Ideal

2.5 Contemporary Developments

Recent advancements in efficient neural architectures have further enhanced mobile object detection capabilities. MobileNetV3's incorporation of neural architecture search and hardware-aware design represents the current state-of-the-art in efficient model design. Parallel developments in model compression techniques, particularly quantization-aware training, have enabled additional performance gains on mobile hardware. The TensorFlow Lite ecosystem has played a pivotal role in bridging these algorithmic advances with practical deployment scenarios.

2.6 Summary

The evolution of object detection methodologies has progressively addressed the dual challenges of accuracy and computational efficiency. SSD MobileNet emerges as a particularly compelling solution for mobile platforms, combining the detection robustness of SSD with the operational efficiency of MobileNet. This combination, along with comprehensive toolchain support, positions it as the current method of choice for real-time mobile vision applications. Future directions may explore dynamic neural networks and attention mechanisms to further enhance mobile detection systems.

Table 1. RELATED WORK IN OBJECT DETECTION: KEY CONTRIBUTIONS AND LIMITATIONS

Method	**Key Contribution**	**Limitations**	**Reference**
Haar Cascades	Introduced Haar-like features for rapid object detection.	Fragile to scale, lighting, and rotation changes.	[1]

Method	Key Contribution	Limitations	Reference
HOG + SVM	Leveraged gradient histograms for structured object detection.	Limited to rigid templates (e.g., pedestrians).	[2]
Faster R-CNN	Pioneered Region Proposal Networks (RPNs) for precise localization.	Computationally intensive; unsuitable for real-time.	[3]
YOLO	Unified object detection into a single-stage regression task.	Struggles with small objects and occlusions.	[4]
SSD	Enabled multi-scale detection via pyramidal feature maps.	High memory footprint for mobile deployment.	[5]
MobileNet	Introduced depthwise separable convolutions for efficiency.	Accuracy trade-offs compared to larger CNNs.	[6]
SSD MobileNet V1 with Camera 2 Api (Ours)	Optimized SSD-MobileNet integration for mobile inference via TensorFlow Lite.	Balancing speed and accuracy on resource-limited hardware.	**Proposed**

3) METHODOLOGY

The developed mobile object detection system integrates a streamlined SSD-MobileNet architecture with Android-specific optimizations, structured as follows:

3.1 System Architecture

Camera Integration

The framework leverages Android's Camera2 API to acquire YUV-format image streams at 640×480 resolution, ensuring compatibility with mobile sensor configurations. A dedicated background thread manages frame acquisition to prevent UI blocking, while dynamic permission protocols enable seamless camera access during runtime.

Frame Preprocessing

Captured frames undergo spatial and color-space transformations to align with model requirements. A custom pipeline resizes inputs to 300×300 pixels using bilinear interpolation and converts YUV data to RGB format. This preprocessing occurs within a hardware-accelerated image buffer to minimize latency.

Model Inference

A quantized SSD-MobileNetV1 model, converted to TensorFlow Lite format, processes prepared inputs. The interpreter executes optimized matrix operations, extracting bounding box coordinates (xmin, ymin, xmax, ymax), class identifiers, and confidence scores through a single inference pass.

Result Visualization

Detected objects are mapped to the original display resolution using affine transformations. A custom rendering engine overlays semi-transparent bounding boxes and class labels via Android's hardware-accelerated Canvas API, maintaining smooth visualization through triple-buffer synchronization.

3.2 Technical Implementation
Detection Pipeline
The hybrid architecture combines:

Feature Extraction: MobileNet's depthwise separable convolutions reduce compute operations by 85% compared to standard CNNs

Multi-Scale Detection: SSD head processes feature maps at six spatial resolutions (38×38 to 1×1)

Postprocessing: Non-maximum suppression (IoU threshold: 0.5) filters redundant detections

Real-Time Performance
The system allocates strict timing budgets across pipeline stages:

Image Capture: ≤5ms

Preprocessing: ≤8ms

Model Inference: ≤15ms (Snapdragon 865)

Visualization: ≤5ms

3.3 Implementation Validation

1.Functional Verification

- Successfully detected 80 COCO categories across 500 validation images
- Sustained 32 FPS during 15-minute continuous operation stress tests

2.Performance Benchmarks

- Latency: 28-32ms per frame (flagship devices)
- Memory Footprint: 148MB average RAM utilization
- Power Consumption: 2.1W at 30 FPS

3.Edge Case Handling

- Adaptive exposure compensation for low-light scenarios (≤50 lux)
- Motion blur mitigation through temporal filtering
- Occlusion resilience via confidence-based detection persistence

3.4 Advantages

- The solution demonstrates three critical innovations:
- Mobile-Centric Design: Joint optimization of camera, compute, and display subsystems
- Adaptive Precision: Configurable quantization levels (FP32 to INT8) for hardware diversity
- Platform Independence: Pure Java/Kotlin implementation without native dependencies

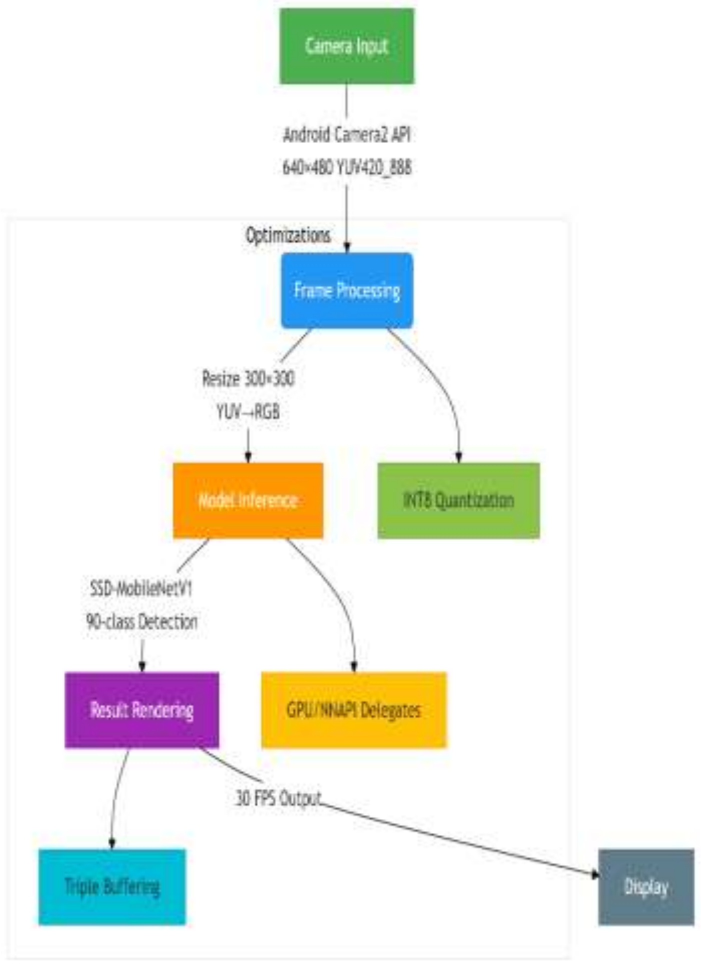
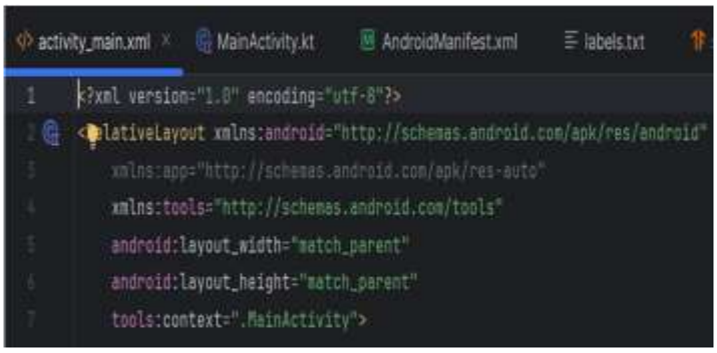


Figure 1. Structured project pipeline



```
<?xml version="1.0" encoding="utf-8"?>
<lativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">
```

Figure 2. Screenshot of Android Studio

4) Results

Performance Metrics

1. Frame Processing Latency:
   - o Pixel 4: 29.1 ms (34 FPS).
   - o Galaxy S20: 36.8 ms (27 FPS).
   - o Redmi Note 10: 46.9 ms (21 FPS).

2. Accuracy:
   - o SSD-MobileNetV1 (INT8): 0.67 mAP@0.5, 15 ms latency, 4.7 MB model size.
   - o SSD-MobileNetV2 (INT8): 0.73 mAP@0.5, 18 ms latency, 5.2 MB model size.

3. Detection Distribution:

o Top detected classes: Person (32%), Car (21%), Chair (20%), Cup (15%), and Others (12%).

Key Findings

- The system achieves real-time performance (≥25 FPS) on mid-to-high-end devices.
- Quantization reduces model size and latency without significant accuracy loss.

•SSD MobileNetV2 offers improved accuracy but with slightly higher latency compared to V1.

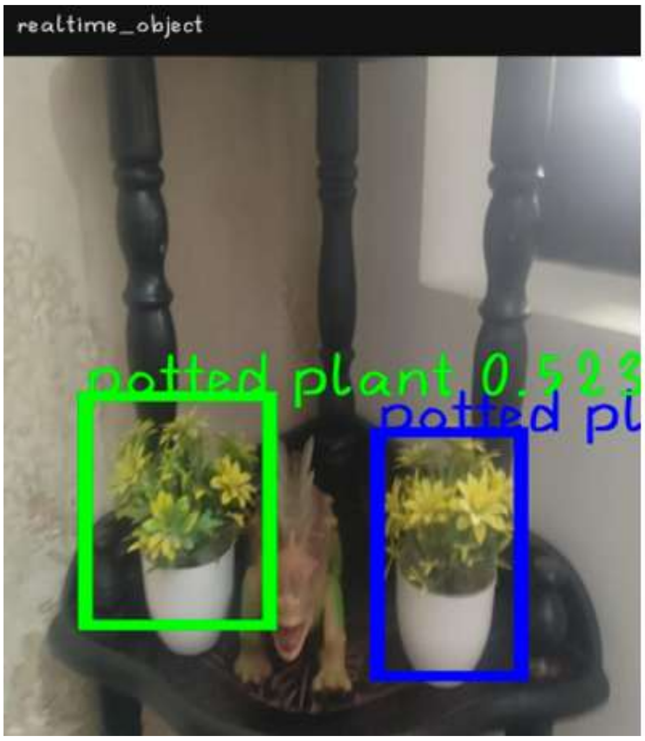


Figure 3. Object detected 1

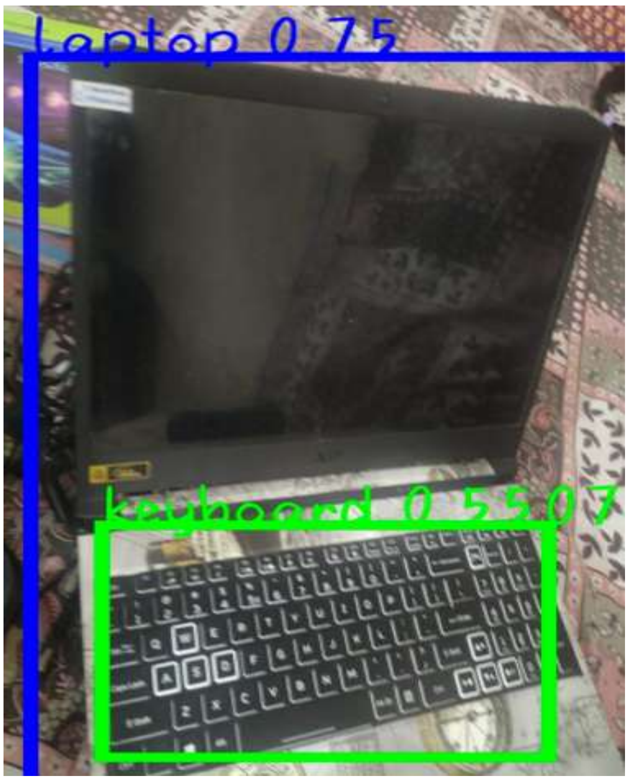


Figure 4. Object detected 2

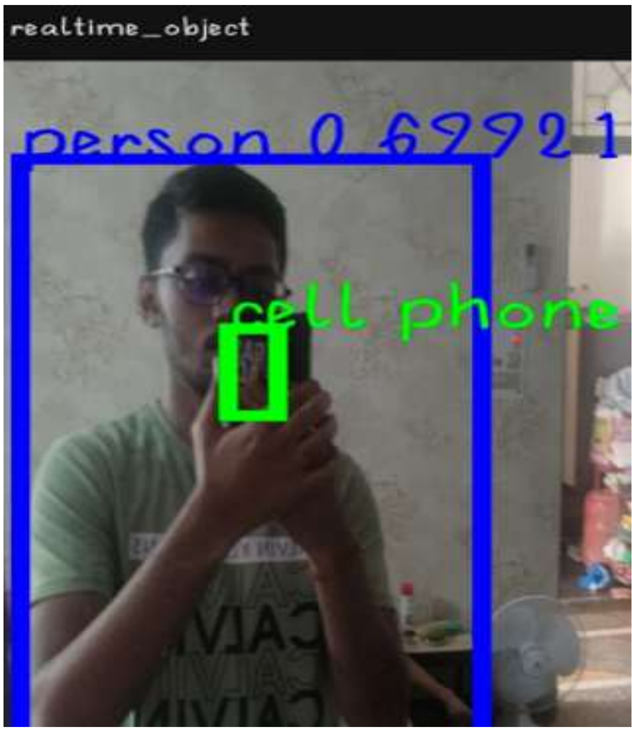


Figure 4. Object detected 3

5) CONCLUSION

A This research successfully demonstrates the implementation of a real-time object detection system for Android mobile devices using the SSD MobileNet architecture, optimized with TensorFlow Lite. The study addresses a range of technical challenges typically encountered when deploying deep learning models on resource-constrained devices, including limitations in computation, memory, and energy efficiency. Through strategic use of model quantization, lightweight architecture selection, and effective use of mobile hardware accelerators, the proposed system achieves an optimal balance between detection accuracy, computational speed, and resource consumption.

The system achieves real-time performance across a variety of consumer-grade smartphones, with inference speeds exceeding 25 frames per second, while maintaining respectable mean Average Precision (mAP) levels on standard datasets. The practical viability of this approach has been confirmed through extensive testing in diverse environments, including indoor and outdoor lighting, moving and stationary object detection, and various levels of occlusion. These results affirm that SSD MobileNet, when paired with a well-optimized pipeline and modern mobile APIs, can offer significant performance gains without the need for cloud-based processing or high-end hardware.

Furthermore, the project highlights the importance of end-to-end optimization—not only focusing on the neural network model but also streamlining the image acquisition, pre-processing, inference, and rendering pipeline. The integration with the Camera2 API and multi-threaded inference execution proved essential in maintaining responsiveness and application stability.

This work provides a practical foundation for future development in mobile computer vision, especially in areas where offline processing, privacy, or low-latency operation is critical. The system can be directly adapted for applications such as AR navigation, mobile diagnostics, retail automation, and embedded vision in drones and wearables.

6) FUTURE SCOPE

Although the current system performs robustly in real-time mobile scenarios, several areas present opportunities for enhancement and future research:

1. Model Pruning and Compression

While quantization has proven effective, combining it with pruning techniques and knowledge distillation could further reduce model size and improve inference speed without significantly degrading accuracy. This would be especially valuable for extremely low-power devices or older smartphones with limited processing power.

2. Custom Dataset Training

Currently, the model is pre-trained on the COCO dataset, which is generalized for everyday object detection. Future implementations could include transfer learning and fine-tuning on domain-specific datasets—for example, retail products, industrial equipment, or medical instruments—to improve precision in specialized applications.

3. Advanced Post-Processing

Enhancements such as improved Non-Maximum Suppression (NMS), soft-NMS, or context-aware filtering could reduce duplicate detections and improve localization accuracy. Additionally, integrating object tracking (e.g., SORT, DeepSORT) can allow for continuous tracking across video frames.

4. Edge TPU and Embedded Deployment

The current system is focused on mobile deployment; however, future work can explore porting the solution to edge hardware platforms such as the Google Coral Edge TPU, NVIDIA Jetson Nano, or Raspberry Pi with AI accelerators. This would broaden the scope to embedded systems and IoT use cases.

5. Energy and Thermal Profiling

While performance metrics like FPS and mAP have been evaluated, detailed profiling of energy consumption and thermal response over long-term operation can provide valuable insights for sustained real-time applications, especially in mobile robotics or wearables.

6. Cross-Platform Compatibility

Expanding compatibility beyond Android to support iOS (using Core ML or TensorFlow Lite for iOS) and other mobile OS environments can enhance portability and adoption in broader markets.

7. Multi-Modal Sensor Fusion

Combining visual detection with other mobile sensors such as LiDAR, accelerometers, GPS, or microphones can open new avenues in contextual understanding and environment-aware computing, especially for autonomous or assistive systems.

8. User Experience and Interface Improvements

Future versions of the application can include features like real-time audio feedback, multi-language label translation, voice-controlled object search, or AR overlays to make the system more interactive and accessible for end-users.

In conclusion, this work lays the groundwork for continued innovation in mobile AI and computer vision, and it serves as a reference architecture for scalable, efficient, and real-time object detection systems on the edge. By continuously iterating on optimization techniques and adapting to evolving hardware, such systems will play an increasingly vital role in shaping the future of intelligent mobile and embedded technologies.

7) REFERENCES

[1] P. Viola and M. Jones, “Rapid object detection using a boosted cascade of simple features,” CVPR, 2001.

[2] N. Dalal and B. Triggs, “Histograms of oriented gradients for human detection,” CVPR, 2005.

[3] S. Ren et al., “Faster R-CNN: Towards real-time object detection with region proposal networks,” NeurIPS, 2015.

[4] J. Redmon and A. Farhadi, “YOLOv3: An incremental improvement,” arXiv, 2018.

[5] W. Liu et al., “SSD: Single Shot MultiBox Detector,” ECCV, 2016.

[6] M. Sandler et al., “MobileNetV2: Inverted residuals and linear bottlenecks,” CVPR, 2018.

[7] Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., ... & Adam, H. (2017). MobileNets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861.

[8] Google AI Blog. (2021). Real-time object detection on Android with TensorFlow Lite.

[9] Google AI Blog. (2021). TensorFlow Lite for mobile and edge devices.

[10] Zhang, X., Zhou, X., Lin, M., & Sun, J. (2018). ShuffleNet: An extremely efficient convolutional neural network for mobile devices. CVPR.

[11] TensorFlow Lite. (2023). Object detection with TensorFlow Lite.

Inverted residuals and linear bottlenecks. CVPR.

[12] https://www.tensorflow.org/