Design and Implementation of Real-Time Object Detection Using Region-Based Convolutional Neural Network (R-CNN), TensorFlow, OpenCV and PyTorch

Omoyele Alex Mowemi
R.Eng, MCPN
Department of Computer and Industrial Production Engineering
Abiola Ajimobi Technical University
Ibadan, Oyo State, Nigeria
omoyele.mowemi@tech-u.edu.ng

Abdulrahman Ali Musa
Department of Computer Science
Federal School of Statistics
Ibadan, Nigeria
abdulone.alimusa@gmail.com

Hannah Temiloluwa Ogunseye
Department of Computer Science
Federal School of Statistics
Ibadan, Nigeria
anna.ogunseye@gmail.com

Ayomide Stephen Taiwo
Department of Computer Science
Federal School of Statistics
Ibadan, Nigeria
taiwoayomide899@gmail.com

***Abstract*—The exponential growth of big data and increasing demand for accurate object identification across various industries have created significant challenges in real-time detection and classification systems. This paper presents the design and implementation of a web-based application for accurate object detection and identification within images and real-time video streams using Region-based Convolutional Neural Networks (R-CNN). The system addresses critical limitations in security systems, autonomous vehicles, healthcare diagnostics, retail monitoring, and manufacturing quality control. The proposed solution employs PyTorch Faster R-CNN ResNet50 FPN V2 architecture, integrated with Streamlit for web deployment, OpenCV for image processing, and WebRTC for real-time video streaming. The implementation incorporates session state management, threading for concurrent processing, and caching mechanisms to optimize performance. Evaluation on diverse datasets demonstrates successful detection of over 80 object categories with confidence thresholds exceeding 50 percent, achieving processing speeds suitable for real-time applications. Results indicate reliable object identification in both static images and live video feeds, with particular effectiveness in examination settings and academic environments. The research contributes to advancing accessible computer vision applications, providing a foundation for enhanced object recognition systems deployable across industrial and academic contexts.**

***Index Terms*—object detection, convolutional neural networks, R-CNN, deep learning, computer vision, real-time processing, PyTorch, TensorFlow, OpenCV, WebRTC**

I. Introduction

Information serves as the fundamental driving force behind global evolution and human progress. As defined by contemporary information science, information pertains to interpretations that may be sensed or their abstractions [1]. The exponential increase in information volume presents significant challenges in efficient knowledge extraction and processing.

The advent of Big Data—characterized by structured, unstructured, and semi-structured data growing exponentially over time—has further complicated information management [2]. While industrial-scale tools such as Apache Spark, Apache Hadoop, and Tableau address large-scale data analysis, a critical gap exists in small-scale, real-time information examination systems.

Object recognition, also termed object identification, represents a critical application of computer systems in information gathering. The objective of object recognition is to determine the identity or category of an object in a visual scene from retinal or digital input [3]. This capability enables artificial intelligence implementations to recognize various entities from inputs including video and still camera images [4], with applications spanning video stabilization, advanced driver assistance systems, disease identification, and security surveillance.

A. Problem Statement

Contemporary industries face substantial challenges due to inadequate object identification capabilities. Security systems struggle with real-time threat detection, potentially compromising safety protocols. Autonomous vehicles encounter difficulties navigating complex environments due to insufficient object recognition, increasing accident risks. Healthcare diagnostics suffer from delayed or inaccurate analysis of medical imagery, directly impacting patient outcomes. Retail businesses face inventory management challenges and theft

prevention difficulties. Manufacturing processes experience inconsistent defect detection, resulting in quality degradation and increased production costs. Agricultural operations suffer from imprecise crop and pest identification, leading to reduced yields.

The human visual system, while remarkably capable, exhibits limitations in processing speed, consistency, and scalability when applied to industrial-scale object detection requirements. Manual detection processes are time-consuming, prone to errors, and lack robustness to variations in lighting, scale, and orientation.

B. Research Objectives

This research aims to design and implement an application capable of accurately detecting and identifying objects within images and videos, particularly suitable for examination settings and academic environments. Specific objectives include:

1) Design and implement a web-based application for object identification utilizing state-of-the-art deep learning architectures
2) Evaluate system functionality across diverse scenarios including static images and real-time video streams
3) Deploy the application for accessibility across institutional settings without requiring specialized hardware

C. Contributions

The primary contributions of this research include:

- **System Architecture**: Development of a modular, web-based object detection system integrating PyTorch, Streamlit, and WebRTC technologies
- **Real-time Processing**: Implementation of efficient real-time video processing with configurable performance optimization strategies
- **Accessibility**: Creation of a browser-based interface requiring no installation, enabling deployment across diverse institutional settings
- **Thread-safe Statistics**: Development of robust concurrent data management for real-time analytics
- **Practical Deployment**: Demonstration of cloud-deployable architecture suitable for resource-constrained environments

II. Related Work

A. Evolution of Object Detection

Object detection has undergone significant evolution from traditional computer vision techniques to modern deep learning approaches. Early methods relied on handcrafted features and machine learning classifiers.

1) Traditional Approaches: The Viola-Jones algorithm [5], introduced in 2001, utilized Haar cascades for face detection, achieving real-time performance but limited to frontal face detection. Histogram of Oriented Gradients (HOG) [6] captured edge structures effectively for pedestrian detection but suffered from computational complexity and sensitivity to occlusions. Scale-Invariant Feature Transform (SIFT) [7] provided robustness to scale and rotation variations but required computationally expensive feature matching.

2) Deep Learning Revolution: The introduction of LeNet by LeCun et al. [8] in 1998 established foundational CNN architecture for digit recognition. AlexNet [9], developed by Krizhevsky et al. in 2012, achieved breakthrough performance in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC), demonstrating the potential of deep CNNs for large-scale image classification with a top-5 error rate of 15.3 percent.

VGGNet [10], introduced by Simonyan and Zisserman in 2014, emphasized simplicity through uniform 3×3 convolutional filters with deeper architectures, improving accuracy while increasing computational requirements. ResNet [11], developed by He et al. in 2015, introduced residual connections to address vanishing gradient problems, enabling training of very deep networks with 50-152 layers and achieving state-of-the-art performance with 3.57 percent top-5 error on ImageNet.

B. Region-Based Approaches

1) R-CNN: Girshick et al. [12] introduced Region-based CNN in 2014, combining selective search for region proposals with CNN feature extraction. R-CNN demonstrated significant accuracy improvements but suffered from computational inefficiency, requiring approximately 50 seconds per image due to independent processing of approximately 2000 region proposals.

2) Fast R-CNN: Girshick [13] addressed R-CNN limitations in 2015 by introducing shared convolutional feature extraction and Region of Interest (RoI) pooling, enabling end-to-end training and reducing inference time by 213 times compared to R-CNN.

3) Faster R-CNN: Ren et al. [14] introduced the Region Proposal Network (RPN) in 2015, eliminating dependency on external proposal algorithms through integrated, learned region proposal generation. This architecture achieved real-time performance while maintaining high accuracy, forming the basis for numerous subsequent detection frameworks.

C. One-Stage Detectors

1) YOLO: Redmon et al. [15] reframed object detection as a regression problem in 2016, introducing You Only Look Once (YOLO). By processing images in a single pass through the network, YOLO achieved 40-155 frames per second, enabling real-time applications but with accuracy trade-offs for small objects.

2) SSD: Liu et al. [16] introduced Single Shot Multibox Detector in 2016, utilizing multi-scale feature maps for improved small object detection while maintaining real-time performance, balancing YOLO's speed with improved accuracy.

D. Technical Challenges

Recent literature identifies persistent challenges in object detection:

Small Object Detection: Detection of small objects remains challenging due to feature loss through successive convolutional and pooling layers, with state-of-the-art models

showing significantly lower average precision for small objects compared to large objects [17].

Occlusion Handling: Occlusion—where objects are partially or completely hidden—significantly reduces detection accuracy and efficiency [18]. Research continues on robust feature extraction methods resilient to partial visibility.

Real-time Processing: Balancing detection accuracy with computational efficiency remains critical for deployment in resource-constrained environments [19].

Data Dependency: Modern deep learning approaches require large, annotated datasets, with annotation quality directly impacting model performance [20].

E. Research Gap

While substantial progress has been made in object detection accuracy and speed, gaps remain in accessibility, deployment, real-time video processing, and practical integration. This research addresses these gaps through development of an accessible, web-based object detection system suitable for diverse institutional settings.

III. Methodology

A. System Design Overview

The proposed system follows a modular architecture separating concerns between user interface, machine learning pipeline, video processing, and data management. This design enables independent development, testing, and optimization of components while maintaining system cohesion.

The architecture adheres to several key principles: modularity with clear separation between ML processing, video handling, and UI components; scalability accommodating increased load through horizontal scaling; accessibility through web-based deployment requiring no client-side installation; performance optimization for resource-constrained environments; and maintainability through clean code structure with comprehensive documentation.

B. System Architecture

The system comprises five interconnected layers:

1) Frontend Layer: Implements a Streamlit-based responsive web interface providing intuitive access to detection functionality. The interface supports dual modes—batch image upload and live camera streaming—with real-time statistics visualization and interactive controls.

2) Machine Learning Pipeline: Utilizes PyTorch's pretrained Faster R-CNN ResNet50 FPN V2 model with COCO dataset weights. The pipeline implements model caching through Streamlit's cache resource decorator ensuring efficient loading across sessions, while torch no gradient context minimizes memory consumption during inference.

3) Video Processing Layer: Integrates WebRTC through streamlit-webrtc for browser-based video streaming. The implementation supports two processing modes: Standard Mode processes every frame for maximum accuracy, while Performance Mode processes every fifth frame for improved performance on resource-constrained systems.

4) Data Management Layer: Implements thread-safe statistics tracking using threading locks to ensure data integrity during concurrent operations from video processing threads.

5) Configuration Layer: Manages environment-specific settings, model caching strategies, and deployment configurations supporting local development, cloud deployment, and containerized environments.

C. Object Detection Pipeline

The detection pipeline follows a systematic workflow:

1) Image Preprocessing: Input images undergo transformation using model-specific preprocessing functions to ensure compatibility with the neural network architecture.

2) Inference: Processed images pass through the model with gradient computation disabled for efficiency, generating predictions including bounding box coordinates, class labels, and confidence scores.

3) Post-processing: Predictions undergo filtering based on confidence threshold (default: 50 percent), with bounding box coordinates, class labels, and confidence scores extracted for visualization and analysis.

4) Visualization: Detected objects are annotated with color-coded bounding boxes and confidence scores using PIL's ImageDraw module, providing intuitive visual feedback to users.

D. Real-Time Video Processing

Real-time processing implements frame-by-frame analysis with configurable optimization. Each video frame is converted to PIL image format, processed through the detection pipeline, and annotated with detection results before being returned to the video stream.

The system utilizes multiple STUN and TURN servers for robust connectivity across various network configurations, ensuring reliable operation even in challenging network environments including corporate firewalls and NAT traversal scenarios.

E. Deployment Architecture

The system supports multiple deployment scenarios:

Local Development: Utilizes Python virtual environments with dependency management through requirements.txt, local file system for caching, and development server on localhost port 8501.

Cloud Deployment: Streamlit Cloud integration provides automated dependency management, distributed caching mechanisms, and environment variable configuration for secure API management.

Container Deployment: Docker-based orchestration for scalable production environments enables consistent deployment across diverse infrastructure configurations.

IV. Implementation and Results

A. Implementation Details

1) Development Environment: The system was developed using Python 3.10 with the following key dependencies:

PyTorch 2.0.0 and TorchVision 0.15.0 for deep learning; Streamlit 1.28.0 for web interface; OpenCV and PIL for image processing; streamlit-webrtc 0.47.0 for video streaming; and NumPy 1.24.0 and Pandas 2.0.0 for data processing.

2) Hardware Specifications: Testing was conducted on standard computing hardware including Intel Core i5 processors with 8 GB RAM, demonstrating accessibility without requiring specialized GPU hardware for inference, though GPU acceleration is supported when available.

B. System Interface

The implemented system provides two primary operational modes:

1) Batch Image Processing Interface: Users upload single or multiple images through a drag-and-drop interface. The system displays original images alongside annotated versions with detected objects, bounding boxes, and confidence scores. Detection results are presented in tabular format showing object categories and confidence percentages.

2) Live Camera Interface: Real-time video streaming with frame-by-frame object detection provides immediate visual feedback through annotated video streams. Connection status indicators inform users of system state, while configurable processing modes balance accuracy and performance based on available resources.

C. Experimental Results

1) Detection Accuracy: Testing across diverse scenarios demonstrated overall precision greater than 85 percent for common object categories from the COCO dataset. The default 50 percent confidence threshold effectively filtered low-confidence detections. The system achieved successful simultaneous detection of multiple objects within single frames and reliable detection across 80+ object categories including persons, vehicles, animals, furniture, and electronics.

2) Performance Metrics: Table I presents comprehensive performance metrics comparing Standard Mode and Performance Mode across key operational parameters.

TABLE I
PERFORMANCE COMPARISON OF PROCESSING MODES

Metric	Standard	Performance
Inference Time	450 ms/frame	180 ms/frame
Memory Usage	< 2 GB	< 1.5 GB
Frame Rate	$\sim$2.2 FPS	$\sim$5.5 FPS
Detection Accuracy	87%	84%

3) Real-time Video Processing: The system successfully processed live video streams with acceptable latency for real-time applications. Performance mode provided smoother video experience on standard hardware while maintaining reliable detection accuracy. WebRTC integration ensured robust connectivity across network configurations.

D. Case Studies

1) Examination Monitoring: Deployment in academic examination settings demonstrated effective detection of unauthorized items (mobile phones, electronic devices, books) with real-time alerting capabilities. The system maintained greater than 90 percent detection accuracy for target objects under controlled lighting conditions.

2) Batch Image Analysis: Processing of 50 diverse images containing multiple objects showed consistent performance with average processing time of 2.1 seconds per image. Statistics aggregation provided valuable insights into object distribution and detection confidence patterns.

E. Comparative Analysis

Table II presents a comparative analysis between the proposed system and traditional solutions across key operational dimensions.

TABLE II
COMPARISON WITH TRADITIONAL SYSTEMS

Aspect	Proposed	Traditional
Accessibility	Web-based, no installation	Requires specialized software
Hardware	Standard computing	Often requires GPU
Deployment	Cloud-deployable	Local installation
Real-time	Yes, optimized	Limited
Interface	Intuitive web	Complex technical
Cost	Open-source	Often proprietary

F. Limitations

1) Technical Limitations: WebRTC-based camera access faces limitations on some cloud hosting platforms and may be restricted by corporate firewalls. Real-time video processing requires substantial resources; Performance mode addresses this through frame skipping. Like most detection systems, accuracy decreases for very small objects (less than 32×32 pixels). Live streaming requires stable internet connectivity.

2) Practical Constraints: Session-based statistics require manual persistence; automatic database integration would enhance usability. Currently limited to text-based logging; structured data export (PDF, CSV) would improve utility. System uses pre-trained models; custom model training interface would enhance domain-specific applications.

V. DISCUSSION

A. Key Contributions

This research successfully demonstrates the feasibility of deploying advanced object detection capabilities through accessible web-based interfaces. The integration of PyTorch's Faster R-CNN with Streamlit and WebRTC technologies provides a blueprint for democratizing computer vision applications.

The modular architecture separating concerns between ML processing, video handling, and user interface enables independent optimization and maintenance. Thread-safe statistics

management ensures data integrity during concurrent operations, while configurable processing modes accommodate diverse hardware capabilities.

By eliminating installation requirements and supporting standard computing hardware, the system removes traditional barriers to object detection technology adoption. Cloud deployment capability further enhances accessibility across institutional settings.

B. Applications and Impact

The system's design makes it suitable for diverse applications including educational settings for examination monitoring, attendance tracking, and classroom activity analysis; security and surveillance through centralized monitoring across distributed camera systems; retail and inventory management supporting product recognition tasks; and research and development providing a foundation for computer vision experimentation.

C. Future Research Directions

Several avenues for enhancement emerge from this work:

- Integration of additional detection architectures (YOLOv8, EfficientDet) would provide users with performance/accuracy trade-off options
- Development of a web-based interface for fine-tuning models on custom datasets
- Implementation of object tracking across frames, trajectory analysis, and behavioral pattern recognition
- PostgreSQL or MongoDB integration for automatic statistics persistence
- Development of native mobile applications for enhanced accessibility
- Implementation of federated learning capabilities for collaborative model improvement
- Optimization for edge computing devices (Raspberry Pi, NVIDIA Jetson)

D. Ethical Considerations

Deployment of object detection systems raises important ethical considerations. The system processes video streams without persistent storage, addressing privacy concerns. However, deployment in surveillance contexts requires careful consideration of consent and data protection regulations. Pretrained models may exhibit biases present in training datasets, requiring bias assessment for applications demanding fairness across demographic groups.

VI. Conclusion

This research successfully demonstrates the design and implementation of an accessible, web-based object detection system integrating state-of-the-art deep learning architectures with practical deployment considerations. The system achieves reliable object detection across 80+ categories with confidence thresholds exceeding 50 percent, while maintaining processing speeds suitable for real-time applications on standard computing hardware.

The modular architecture separating machine learning processing, video handling, user interface, and data management enables independent optimization while maintaining system cohesion. Integration of PyTorch's Faster R-CNN ResNet50 FPN V2 with Streamlit's web framework and WebRTC video streaming provides a blueprint for accessible computer vision application deployment.

Experimental results demonstrate the system's practical utility across diverse scenarios including batch image processing, real-time video analysis, and examination monitoring applications. While limitations exist regarding camera functionality on some cloud platforms and small object detection challenges, the system successfully addresses the identified research gap in accessible, web-based object detection tools.

The research contributes to advancing computer vision accessibility, demonstrating that advanced detection capabilities can be delivered through intuitive web interfaces without requiring specialized hardware or technical expertise. The open-source implementation provides a foundation for future enhancements including custom model training interfaces, advanced analytics capabilities, and mobile platform optimization.

Acknowledgment

The authors would like to thank Mallam Abdulrahman Musa Ali and Engr. Mowemi O.A. for their supervision and guidance throughout this research. We also acknowledge the Federal School of Statistics, Ibadan for providing the necessary resources and support.

References

[1] "Information," Wikipedia, Jun. 2024. [Online]. Available: https://en.wikipedia.org/wiki/Information

[2] Google Cloud, "Big data definition," Aug. 2023. [Online]. Available: https://cloud.google.com/learn/what-is-big-data

[3] M. C. Mozer, "Object recognition: Theories," in *International Encyclopedia of the Social and Behavioral Sciences*, 2001, pp. 10781-10785.

[4] TechTarget Contributor, "What is object recognition?" Feb. 2015. [Online]. Available: https://www.techtarget.com/whatis/definition/object-recognition

[5] D. M. Abdulhussien and L. J. Saud, "An evaluation study of face detection by Viola-Jones algorithm," *International Journal of Health Sciences*, vol. 6, pp. 4174-4182, 2022.

[6] "Histogram of oriented gradients," Wikipedia, 2024. [Online]. Available: https://en.wikipedia.org/wiki/Histogram_of_oriented_gradients

[7] D. Shut, "Introduction to SIFT (scale invariant feature transform)," Medium, 2019. [Online]. Available: https://medium.com/@deepanshut041/introduction-to-sift

[8] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278-2324, 1998.

[9] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84-90, 2012.

[10] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[11] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2016, pp. 770-778.

[12] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2014, pp. 580-587.

[13] R. Girshick, “Fast R-CNN,” in *Proc. IEEE Int. Conf. Computer Vision*, 2015, pp. 1440-1448.
[14] S. Ren, K. He, R. Girshick, and J. Sun, “Faster R-CNN: Towards real-time object detection with region proposal networks,” *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, pp. 1137-1149, 2017.
[15] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You only look once: Unified, real-time object detection,” in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2016, pp. 779-788.
[16] W. Liu et al., “SSD: Single shot multibox detector,” in *Proc. European Conf. Computer Vision*, 2016, pp. 21-37.
[17] Z. Li, Q. Guo, B. Sun, D. Cao, Y. Li, and X. Shuai, “Small object detection methods in complex background: An overview,” *Int. J. Pattern Recognition and Artificial Intelligence*, vol. 37, no. 2, 2023.
[18] M. Yazdi and T. Bouwmans, “New trends on moving object detection in video images captured by a moving camera: A survey,” *Computer Science Review*, vol. 28, pp. 157-177, 2018.
[19] A. Amjoud and M. Amrouch, “Object detection using deep learning, CNNs and vision transformers: A review,” *IEEE Access*, vol. 11, pp. 35479-35516, 2023.
[20] J. Ren and Y. Wang, “Overview of object detection algorithms using convolutional neural networks,” *J. Computer and Communications*, vol. 10, no. 1, pp. 115-132, 2022.