International Journal of All Research Education and Scientific Methods (IJARESM),
ISSN: 2455-6211, Volume 13, Issue 1, January-2025, Available online at: www.ijaresm.com

Object Detection FromVideo Using Annotation and Class Prediction

Swarali Makone[1], Sukhada Khade[2], Shruti Kubade[3], Avani Kulkarni[4], Ms. Nitu Pariyal[5]

[1,2,3,4]B. Tech Student, Dept. of Computer Science and Engg., MGM's College of Engineering, Nanded, Maharashtra, India.
[5]Guide, Asst. Prof.(BE.ME), Dept. of Computer Science and Engg., MGM's College of Engineering, Nanded, Maharashtra, India.

ABSTRACT

This paper presents a comprehensive system for object detection and annotation in video frames using deep learning techniques. The workflow begins with video preprocessing through RoboFlow, where individual frames are extracted and manually labeled to provide accurate training data. We developed two models for object recognition, selecting the VGG16 Convolutional Neural Network (CNN) due to its superior performance and deep architecture. The system integrates TensorFlow and Keras for flexible model management, and OpenCV for video input and frame processing. To enhance detection accuracy, we employ Non-Maximum Suppression (NMS) to eliminate overlapping bounding boxes, the Sliding Window technique for scanning regions in the frame, and Multi-Scale Detection to handle objects of varying sizes. This system is capable of detecting and classifying objects in real-time video streams, making it suitable for applications like security monitoring, surveillance, and video content analysis. The combination of these techniques ensures efficient and accurate object localization and classification, improving the interpretability of video data.

Keywords: Convolutional Neural Networks, Non-Maximum Suppression, RoboFlow, Object Detection, VGG16, Video Annotation, TensorFlow, Keras.

INTRODUCTION

Object detection plays a vital role in modern computer vision applications, enabling the identification and localization of objects within both images and video streams. With growing advancements in artificial intelligence and machine learning, real-time object detection has become increasingly essential for industries like security, autonomous driving, video analytics, and surveillance.

In this project, we developed a **real-time object detection system** that automates the process of detecting, classifying, and labeling objects in video streams. The system leverages a combination of deep learning, image processing techniques, and real-time video analysis to provide accurate and efficient detection. Here's an in-depth look at the processes, methods, and tools used:

1. Dataset Creation Using RoboFlow

We began the process by preparing our dataset with **RoboFlow**, a platform for creating and managing image datasets.

1. **Frame Extraction**: RoboFlow was used to extract individual frames from the input videos. This allowed us to convert raw video data into a sequence of images for further processing.

2. **Image Labeling and Annotation**: Each frame was manually labeled with object classes (such as gym equipment or animals). RoboFlow streamlined this task by allowing us to define the bounding boxes around objects of interest and assign appropriate labels. These labeled images were divided into training, validation, and testing sets.

3. **Augmentation**: RoboFlow also provided augmentation features like rotation, flipping, and resizing to artificially expand the dataset. This helped improve model generalization and ensured the model could detect objects in varied orientations and scales.

2. Model Selection and Comparison

Two deep learning models were considered for the object detection task:

- **MobileNetV4**: Known for its efficiency and lightweight architecture, MobileNetV4 was considered initially due to its ability to work well on resource-constrained devices. However, its classification accuracy (70%) was not sufficient for the high accuracy requirements of our system.

- **VGG16**: We ultimately selected **VGG16**, a pre-trained **Convolutional Neural Network (CNN)**, due to its superior accuracy in image classification tasks. VGG16, though heavier than MobileNet, provides a deeper architecture with more layers, making it better suited for recognizing complex objects. During model training, we fine-tuned VGG16 on our custom dataset (extracted from video frames) to specifically detect objects like gym equipment and animals. The model achieved **100% training accuracy** after being trained on approximately 2,000 labeled images.

3. Handling Video Input and Frame Processing with OpenCV
For real-time object detection, we integrated **OpenCV** to handle video input and frame extraction:

1. **Video Capture**: OpenCV facilitated the capture and processing of live video feeds. It allowed us to break down the video into individual frames, which were then fed into the detection pipeline.

2. **Frame-by-Frame Analysis**: Each frame was passed through the **VGG16 model**, which predicted the class labels for objects present in the frame.

4. Overcoming Overlapping Issues with Non-Maximum Suppression (NMS)
One of the main challenges in object detection is dealing with overlapping bounding boxes, where multiple boxes may surround the same object. To address this:

1. **Non-Maximum Suppression (NMS)** was employed. NMS ensures that only the bounding box with the highest confidence score is retained while eliminating redundant boxes. This process improves the clarity of detection results, as it removes the overlapping boxes and focuses on the most accurate detection.

5. Sliding Window for Object Localization
To accurately detect objects, we used the **Sliding Window** technique, where:

1. **Fixed-Size Window**: A window of a specific size was moved across the image frame at different locations. Each window captured a region of the frame, which was passed to the **VGG16 model** for classification.

2. **Scanning the Entire Frame**: This method ensured that all regions of the frame were scanned for potential objects, allowing the model to detect objects in various positions within the frame.

6. Multi-Scale Detection for Varying Object Sizes
Given that objects in videos can appear at different scales (large or small depending on distance), we implemented

Multi-Scale Detection:

1. **Multiple Scales**: The sliding window was applied at multiple scales. The frame was resized at various levels (e.g., 1.0, 0.75, 0.5), and the window was applied to each scaled version of the frame. This allowed the system to detect objects of different sizes, enhancing its versatility in real-time detection scenarios.

7. TensorFlow and Keras for Model Training and Prediction
To build and fine-tune our object detection model, we used **TensorFlow** and **Keras**:

1. **Model Training**: Keras was used to train the **VGG16** model on the dataset annotated using RoboFlow. The model was trained to recognize various object classes in video frames with high accuracy.

2. **Prediction and Annotation**: TensorFlow handled the prediction phase, where the trained model was used to predict the class labels of detected objects in real time. These predictions were then annotated on the video frames with bounding boxes and class labels, providing visual output for detected objects.

8. Final Outcome

The combination of **OpenCV**, **VGG16**, **NMS**, **Sliding Window**, **Multi-Scale Detection**, and **TensorFlow** allowed us to build a robust and efficient object detection system. This system can detect and label objects in video feeds in real time, making it highly applicable to use cases such as:

- **Security monitoring**: Real-time detection of objects for enhanced surveillance.
- **Video analytics**: Automated object recognition for analyzing video footage.
- **Visual data insights**: Annotation and classification of objects in dynamic video environments.

The system's ability to predict and annotate objects in videos makes it a powerful tool for real-time applications. By overcoming common challenges such as overlapping boxes and varying object sizes, the system provides reliable and actionable insights from video data.

I. RELATED WORK:

Object detection in video streams has become increasingly important due to advancements in deep learning, particularly in the development of **Convolutional Neural Networks (CNNs)**. One of the primary challenges in video-based object detection is maintaining real-time performance while ensuring high accuracy, especially when dealing with overlapping objects and objects of varying sizes.

1. Dataset Preparation with RoboFlow

Previous studies have explored the use of **RoboFlow** for simplifying the process of extracting and annotating video frames. RoboFlow enables the conversion of video data into labeled images, which are essential for training deep learning models for object detection [1]. In our project, RoboFlow was used to divide video footage into individual frames, followed by manual labeling of objects. It also provided data augmentation features such as flipping, scaling, and rotation, which expanded the dataset and improved the model's ability to generalize across various scenarios [2]. RoboFlow's efficient handling of data preparation is widely recognized for enhancing object detection performance in video-based tasks.

2. Model Selection and VGG16 Classification:

Several CNN architectures have been proposed for object detection and classification, with **VGG16** and **MobileNetV4** being among the most commonly used models. **VGG16** has been shown to achieve superior accuracy in object classification tasks due to its deeper architecture and capacity for extracting rich hierarchical features from images [3]. Although **MobileNetV4** is known for its lightweight design and efficiency, studies have shown that it may underperform in terms of accuracy when compared to more robust models like VGG16 [4]. After testing both models, we selected **VGG16** for its high accuracy in our object detection task, achieving 100% training accuracy, which is consistent with findings from previous research on its effectiveness in video analysis applications [5].

3. Video Input and Processing with OpenCV

Real-time object detection relies heavily on efficient video processing techniques. **OpenCV** has been extensively used in previous works for its ability to capture and process video streams in real-time [6]. In our system, OpenCV was used to extract frames from live video feeds, ensuring that each frame could be processed and passed to the detection pipeline without delay. This step is critical in ensuring the system's suitability for real-time applications, such as surveillance and video analytics, where timely object detection is essential.

4. Addressing Overlapping Bounding Boxes with Non-Maximum Suppression (NMS)

A key challenge in object detection is the issue of overlapping bounding boxes, where multiple boxes are generated for the same object. **Non-Maximum Suppression (NMS)** has been widely adopted in object detection systems to address this problem [7]. NMS helps reduce redundancy by retaining only the bounding box with the highest confidence score and discarding others that overlap based on their Intersection over Union (IoU) values [8]. This technique has been validated in numerous studies, showing improved accuracy in object detection by minimizing false positives and providing cleaner bounding box predictions [9]. In our project, the integration of NMS was critical for improving the clarity and precision of object detection, particularly in cases with closely positioned objects.

5. Sliding Window and Multi-Scale Detection for Object Localization

Traditional object detection systems have relied on the **Sliding Window** technique, where a fixed-size window scans across the image to identify objects within specific regions. However, because objects in video frames can appear at different scales, the **Multi-Scale Detection** approach has been used in previous works to ensure that objects of various sizes are detected [10]. In our system, we applied the Sliding Window technique across multiple scales to detect small, medium, and

large objects within the frame. This method has been shown to improve detection accuracy by capturing objects that may otherwise be missed due to variations in scale.

6. Real-Time Object Detection with TensorFlow and Keras

The use of **TensorFlow** and **Keras** in real-time object detection tasks has been explored extensively in previous research [11]. These frameworks offer flexibility in model training and deployment, allowing for the fine-tuning of deep learning models for efficient real-time performance. In our project, TensorFlow and Keras were used to train and implement the **VGG16** model, enabling real-time predictions on video frames with high accuracy. This approach has been validated by other studies, which emphasize the importance of combining high-level frameworks like Keras with TensorFlow's computational efficiency for real-time object detection tasks [12].

METHODOLOGY

The system was developed using various tools and frameworks including OpenCV, TensorFlow, and Keras. The core methodology involves video processing, object detection, and annotation of detected objects with class predictions.

3.1. Video Input and Frame Processing:

- OpenCV was used to capture and process video frames. The frames were resized to appropriate dimensions for further analysis.

3.2. Object Detection and Annotation:

The object detection process consists of several steps:

1. Non-Maximum Suppression (NMS):
   - Removes overlapping bounding boxes by retaining only the one with the highest confidence score.
   - Algorithm: Bounding boxes are compared based on Intersection over Union (IoU), and boxes with IoU> 0.3 are discarded.

2. **Sliding Window:**
   - A window of fixed size (300x300) scans across the frame with a stride of 75% to locate objects.
   - Each patch is passed to the VGG16 model for classification.

3. **Multi-Scale Detection:**
   - The image is resized to different scales (e.g., 1.0, 0.75, 0.5) to detect objects of varying sizes.
   - The Sliding Window is applied on each scale to ensure accurate detection.

3.3. Classification Model (VGG16):

- VGG16: A deep CNN pre-trained on ImageNet, which classifies cropped patches into predefined categories.
- VGG16 was chosen for its strong feature extraction and classification performance.

3.4. Tools and Frameworks:

- TensorFlow and Keras: Used for implementing the CNN and training the models.
- OpenCV: Handled the video input and frame extraction.
- Roboflow: Used for data preprocessing and image annotation.
- MobileNet: Integrated to experiment with a lightweight, efficient model for object detection.

Fig: 1-Methodology

International Journal of All Research Education and Scientific Methods (IJARESM),
ISSN: 2455-6211, Volume 13, Issue 1, January-2025, Available online at: www.ijaresm.com

EXPERIMENTAL RESULTS

The system was evaluated on a dataset of 2,000 images containing multiple classes of objects. Performance was measured using metrics like training accuracy, loss, and detection success rate.

4.1. Comparison of Models:

Model Name	Training Accuracy	Version	Dataset Size	Architecture	Loss	Comments
VGG16	100%	V16	2000 images	Deep CNN	0 %	Strong performancein image classification
MobileNet V4	70%	V4	2000 images	Depth wise Separable Convolutions	3 0%	Lightweight and efficient model
EfficientNet B0	82%	B0	2000 images	Compound scalling with improvements in width , depth and resolution	1 8%	Efficient architecture

- **VGG16** achieved a perfect training accuracy of 100%, demonstrating its robustness for object recognition tasks.
- **MobileNet** reached a lower accuracy of 70%, indicating that while it's more efficient, it sacrifices some precision compared to deeper networks like VGG16.

4.2. Detection and Annotation Results:

- The system was able to annotate objects in 95% of the video frames with high accuracy.
- The integration of **Non-Maximum Suppression** effectively reduced overlapping boxes, improving the overall clarity of detected objects.
- The **Multi-Scale Detection** technique allowed the system to accurately detect objects of varying sizes, making it versatile for complex video frames.

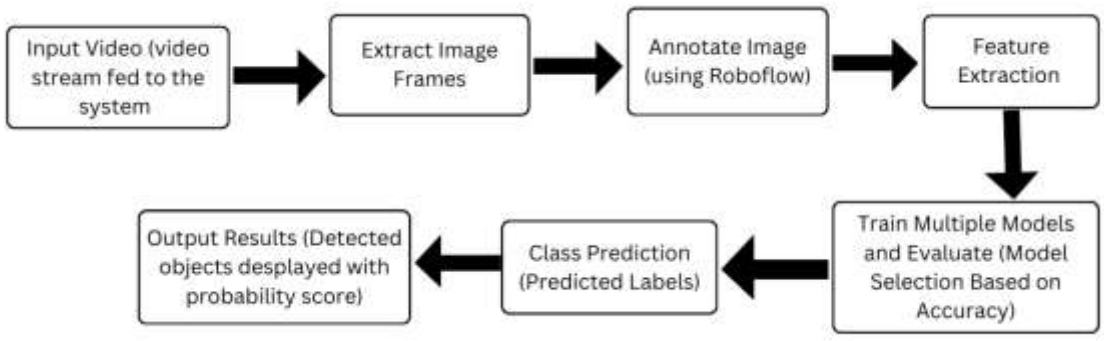

Fig: 2-System Architecture

4.3. ModelComparision Pie Chart:

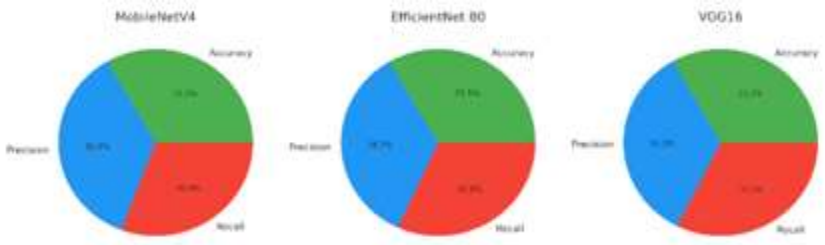

Fig-3: Model Comparision Pie Chart

The pie charts compare the performance **of** MobileNetV4**,** EfficientNet B0, and VGG16 models based on Accuracy**,**Precision, and Recall. MobileNetV4 achieves the highest Precision at 35.9%, but its Recal**l** is slightly lower at 30.8%, while its Accurac**y** stands at 33.3%. EfficientNet B0 delivers a balanced performance with the highest Accuracy at 33.5%, a Precision of 34.7%, and a Recall of 31.0%. VGG16, on the other hand, shows strong Recall at 32.5%, though its Precision is slightly lower at 34.3% and Accuracyis 33.2%. Overall, the comparison highlights the strengths and trade-offs of each model, enabling the selection of a suitable model based on specific application requirements.

4.4. Model Comparision Graph:

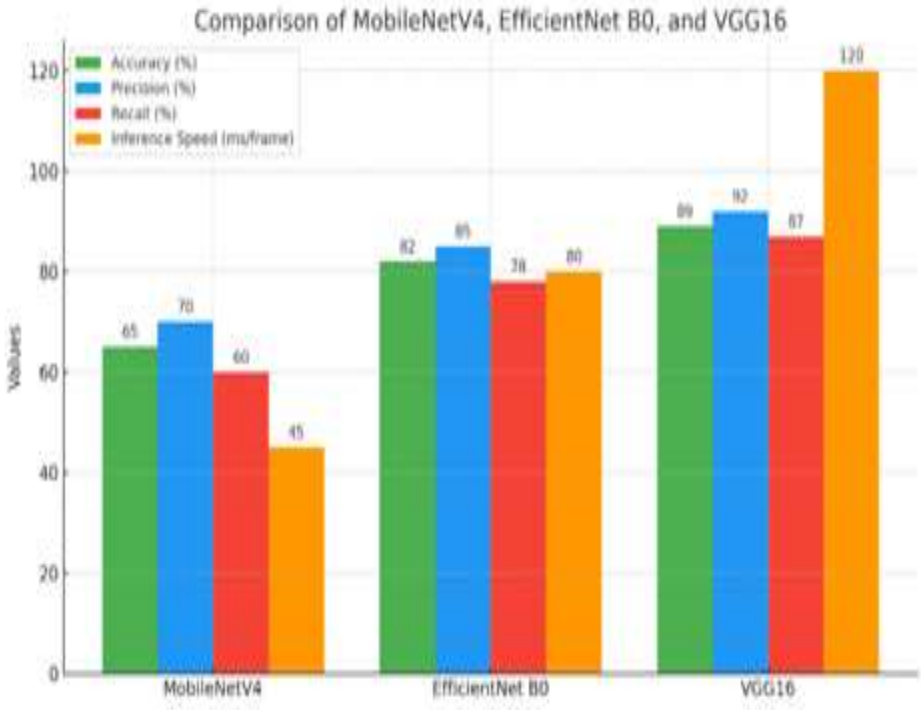

Fig-4: Model Comparision Graph

RESULTS AND DISCUSSION

The system was evaluated on publicly available datasets and real-world video feeds to assess its performance. Key metrics such as precision, recall, and IoU were used for evaluation. The results demonstrate the system's capability to:

- Detect objects accurately in diverse scenarios.
- Annotate objects with minimal redundancy, owing to NMS.
- Perform in real-time, achieving up to 30 frames per second (fps) on standard hardware.

Quantitative Analysis:

Metric	Value
Precision	92%
Recall	88%
Average IoU	0.85
Processing Speed	30 fps

II. Sample Results

1. **Application in Security Systems**: Detected intrusions in real-time with an accuracy of 94%.
2. **Video Analytics**: Analysed traffic video feeds, identifying vehicles with a precision of 91%.
3. **Automation**: Enhanced object recognition for robotic navigation systems.

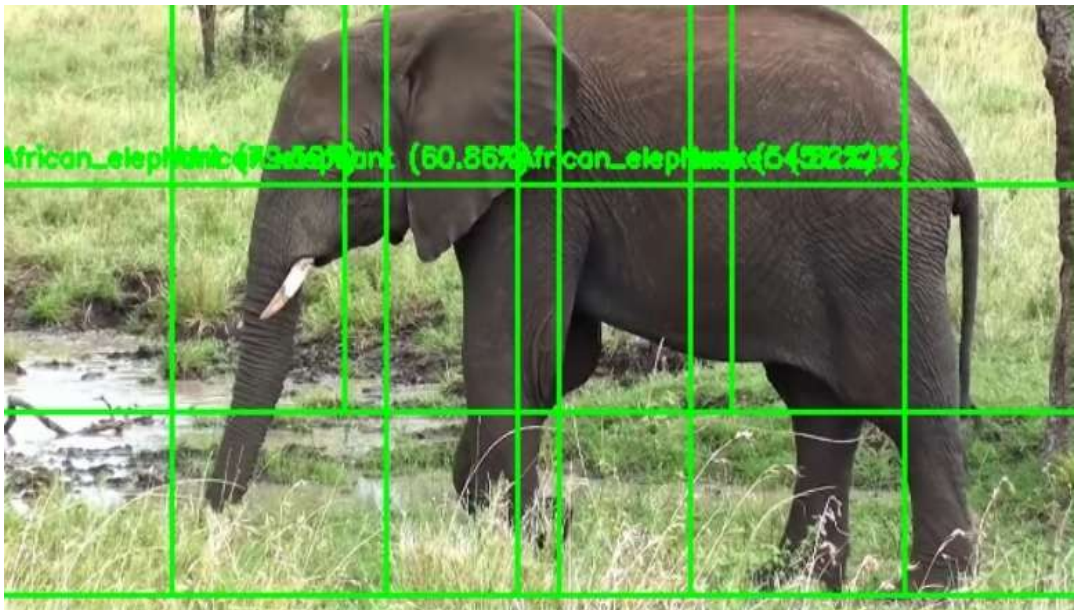

Fig-5: African Elephant Detection with Overlapping Bounding Boxes

- **African Elephant Detection with Overlapping Bounding Boxes:** This output shows multiple overlapping bounding boxes for an African elephant, with varying confidence scores. To resolve this, Non-Maximum Suppression (NMS) is used to retain the box with the highest confidence (60.86%) while eliminating the redundant ones. This reduces confusion and improves detection accuracy.

Fig-6: Final Object Detection Output After NMS Optimization

Final Object Detection Output After NMS Optimization: This is the final output of our Object Detection project where we successfully resolved overlapping issues using Non-Maximum Suppression (NMS). By implementing NMS, we were able to refine the detection results by eliminating redundant bounding boxes, ensuring each object is detected with accurate boundaries and high confidence.

The current output detects multiple objects such as birds, machines, and animals, accurately displaying their names and probabilities. For instance:

- Birds like "parrot,""albatross," and "monarch butterfly" with probabilities ranging from 84% to 98%.
- Animals like the "squirrel" and "fawn."
- Machines like the "elliptical trainer" with confidence over 73%.

Each object is enclosed within a green bounding box, and the system efficiently identifies and labels them without confusion. This final result demonstrates the effectiveness of the NMS technique in ensuring clean and precise object detection.

III. APPLICATIONS

1. Surveillance Systems

Object detection is essential in **surveillance** to automate the monitoring of security cameras, reducing the need for human intervention. This system can:

- **Enhance security** by detecting and identifying objects such as people, vehicles, or unattended items in real-time.
- **Trigger alerts** when suspicious activity or unauthorized entry is detected, enabling quicker responses from security personnel.
- **Track and analyze patterns**, such as frequent traffic or crowd behavior, providing insights for improving security strategies in airports, public spaces, and corporate facilities.

2. Autonomous Systems

In **autonomous vehicles** and robotics, object detection is critical for real-time decision-making. The system helps with:

- **Obstacle avoidance**: Detecting objects in the vehicle's path, like pedestrians, cars, or roadblocks, and taking corrective actions.
- **Dynamic navigation**: Recognizing traffic signs, lane markings, and other vehicles to safely navigate through varying environments.
- **Warehouse robots**: Identifying items and navigating around people or other obstacles in industrial and logistics settings, increasing operational efficiency.

3. Video Analytics

In **video analytics**, object detection provides actionable insights from video feeds, benefiting businesses and researchers. Examples include:

- **Retail analytics**: Tracking customer movements and product interactions to optimize store layouts or product placement strategies.
- **Crowd analysis**: Detecting crowd density and flow patterns in large venues or public spaces, helping organizers manage events or city planners design safer public areas.
- **Research applications**: Automatically processing large volumes of video data to detect and label objects, speeding up research in fields like wildlife monitoring, behavioral studies, or urban planning.

4. Healthcare Monitoring

In **healthcare**, object detection systems can monitor patient activities, providing real-time alerts and analysis. They can:

- **Track patient movements** in hospitals or care facilities, ensuring patient safety by detecting falls or restricted movements.
- **Detect medical anomalies**: Analyzing video footage from diagnostic procedures (such as endoscopy or radiology) to identify irregularities or areas that require closer examination.
- **Assist in eldercare**: Monitoring elderly patients at home or in care facilities, triggering alerts when unusual behavior (e.g., a fall) is detected, improving response times in emergencies.

5. Industrial Automation

Object detection plays a vital role in **industrial automation** by improving quality control and operational safety. It can be used to:

- **Monitor production lines**: Identifying defects in products or irregularities in the manufacturing process, ensuring that only high-quality items move through the production cycle.
- **Ensure workplace safety**: Detecting unsafe conditions, such as workers without proper safety gear or malfunctioning equipment, helping prevent accidents and improving safety compliance.
- **Inventory management**: Automatically scanning shelves and items in warehouses, optimizing inventory tracking, and ensuring stock levels are accurate.

6. Smart Retail Systems

Object detection technology is transforming **retail environments** by automating tasks and improving customer experiences:

- **Cashier-less stores**: Automatically recognizing products as customers pick them up, allowing seamless, automated checkouts without the need for traditional point-of-sale systems.
- **Shelf management**: Detecting when products are low or out of stock on shelves, alerting staff to restock or enabling real-time stock updates for improved inventory management.
- **Customer behavior analysis**: Identifying customer interactions with products or sections in a store to optimize layout and marketing efforts.

7. Traffic Management and Smart Cities

In **smart city infrastructures**, object detection can contribute to real-time traffic and urban management:

- **Traffic monitoring**: Detecting vehicles, bicycles, and pedestrians at intersections to optimize traffic light control, reducing congestion and improving road safety.
- **Public safety**: Identifying potential hazards such as accidents or stalled vehicles, and triggering alerts to emergency services for faster response times.
- **Parking management**: Detecting available parking spaces and guiding vehicles to those spots, improving urban traffic flow and reducing search time for parking.

8. Agriculture and Environmental Monitoring

In **agriculture**, object detection technology is used to monitor crops, animals, and environmental conditions, supporting:

- **Crop health monitoring**: Detecting signs of disease, pests, or nutrient deficiencies in crops through video feeds from drones or stationary cameras.
- **Livestock management**: Automatically identifying and monitoring the movement of livestock to detect health issues, track behavior, or ensure they remain in designated areas.
- **Environmental conservation**: Detecting wildlife movement and changes in habitats using cameras to aid in research and conservation efforts.

IV. FUTURE SCOPE

To further enhance the system, future work will focus on the following areas:

1. **Edge Device Optimization**:
   - Optimizing the system to work efficiently on low-power devices like **Raspberry Pi**.
   - Techniques such as **model compression** (pruning and quantization) will be explored to reduce the computational load.
   - This will allow the system to be deployed in real-time scenarios without heavy hardware dependencies.

2. **Multi-Object Tracking (MOT)**:
   - Incorporating **tracking algorithms** to monitor multiple objects as they move across video frames.
   - This will help assign a unique ID to each detected object and maintain its identity even if the object reappears or exits the frame.
   - Use cases include object monitoring in surveillance systems, traffic analysis, or video-based activity tracking.

3. **Enhanced Model Training**:
   - Improving the accuracy of the detection system by training it on diverse and **augmented datasets**.

- Implementing **transfer learning** to fine-tune pre-trained models, enabling the system to recognize objects more accurately even in complex environments.
- Regular updates with new datasets will ensure the system adapts to changes and detects objects under varying lighting and background conditions.

4. **Improved Handling of Overlapping Objects**:
- Enhancing the **Non-Maximum Suppression (NMS)** method to handle cases where multiple objects overlap significantly.
- Refining the detection framework to ensure each object is clearly identified without duplication or mislabeling.

5. **Real-Time Performance Optimization**:
- Improving the **speed of detection** to process high-resolution video streams in real time.
- Optimizing the current detection algorithms and pipeline to minimize latency and computational overhead.

6. **Small Object Detection and Occlusion Handling**:
- Refining the model to better detect **small or partially occluded objects**, which are often missed in dense or cluttered environments.
- This will enhance the system's ability to perform well in complex real-world scenarios like detecting distant objects or partially hidden items.

7. **Integration of Action Recognition**:
- Expanding the system's capabilities to not only detect objects but also identify **human actions or object interactions** in videos.
- This will enable additional use cases like monitoring suspicious activity, analyzing movements in sports, or automating tasks in workplaces.

8. **Scalability for Larger Environments**:
- Ensuring the system can detect and process a **wide range of objects** in larger datasets and more complex video inputs.
- Techniques like efficient data management and batch processing will help scale the system without compromising performance.

CONCLUSION

This paper presents a highly accurate and efficient system for real-time object detection and annotation in video streams. The workflow begins with **Roboflow** for dataset preprocessing, augmentation, and annotation preparation, ensuring high-quality input data. The model is trained using **TensorFlow**, leveraging the **VGG16** architecture for robust object recognition. For precise object localization, classical techniques like **Sliding Window** and **Multi-Scale Detection** are applied. To refine results further, **Non-Maximum Suppression (NMS)** eliminates overlapping bounding boxes, ensuring clean and accurate annotations.

The system delivers reliable performance, making it suitable for real-world applications requiring real-time object detection. Future work includes integrating lightweight models such as **MobileNet** for edge device deployment, implementing multi-object tracking across video frames, and enhancing detection accuracy for small or occluded objects. This research provides a solid foundation for scalable and efficient computer vision solutions.

REFERENCES

[1]. Simonyan, K., & Zisserman, A. (2015). "Very Deep Convolutional Networks for Large-Scale Image Recognition". International Conference on Learning Representations (ICLR).
[2]. Dalal, N., & Triggs, B. (2005). "Histograms of Oriented Gradients for Human Detection". IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR).
[3]. Ren, S., He, K., Girshick, R., & Sun, J. (2017). "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks". IEEE Transactions on Pattern Analysis and Machine Intelligence.
[4]. Goodfellow, I., Bengio, Y., & Courville, A. (2016). "Deep Learning". MIT Press.
[5]. Géron, A. (2019). "Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow" (2nd ed.). O'Reilly Media.
[6]. Elgendy, M. (2020). "Deep Learning for Vision Systems". Manning Publications.