REAL TIME OBJECT DETECTION USING DNN

A PROJECT REPORT

TABLE OF CONTENTS

CHAPTER NUMBER	TITLE	PAGE NUMBER
1	INTRODUCTION	5
2	COMPANY PROFILE	5
3	ABSTRACT	5
4	OBJECT RECOGNITION	6
5	IMPACT OF OBJECT RECOGNITION	7
6	IMPLEMENTING OBJECT RECOGNITION	7
6.1	PRE-TRAINED MODEL	8
6.2	TRANSFER LEARNING	9

6.3	BUILDING FROM SCRATCH	10
7	DNN - DEEP NEURAL NETWORK	10
8	WHY MOBILENET SSD	11
9	BASIC SYNTAX	12
10	BLOCK DIAGRAM	15
11	CODE DESCRIPTION	15
12	OUTPUT	18
13	APPLICATIONS	18
14	FUTURE SCOPE	19
15	CONCLUSION	19

1. COMPANY PROFILE

8Queens Software Technologies Private Limited is a technology-focused firm with a brilliant team of engineers and designers who offer unique and user-friendly corporate software solutions. We offer custom software development, web and mobile application, digital marketing, maintenance, digital marketing, testing and support services. Our dedication to excellence guarantees that we give the greatest degree of client satisfaction while also establishing long-term partnerships built on trust and transparency.

2. INTRODUCTION

Artificial Intelligence, or AI, has the potential to drastically change many aspects of the future world. Object recognition is one of the key areas where AI is having a big influence. Here's a quick rundown of how artificial intelligence (AI) will be used in the future, with a focus on object recognition and the implications that will follow:

AI for Object Identification:

Deep Learning Frameworks: Artificial Intelligence (AI), particularly deep learning models, has shown impressive results in object identification tests. To identify and categorize objects in photos and videos, Convolutional Neural Networks (CNNs) and other deep learning architectures are trained on extensive datasets.

Pre-trained Models: In object recognition, using pre-trained models minimizes the requirement for in-depth training on particular datasets. These models are versatile and can be applied to a variety of activities by using transfer learning to fine-tune them for particular tasks or domains.

Real-time Processing: By utilizing hardware acceleration and optimization algorithms, AI enables real-time object recognition. This is critical for low latency applications like augmented reality, driverless cars, and video surveillance.

Multi-class Recognition: Artificial Intelligence makes it possible to identify items from various classes or categories. When a system must concurrently detect and distinguish between multiple objects, this capacity comes in handy.

3. ABSTRACT

Real-time object recognition is a critical aspect of computer vision applications, with numerous practical applications ranging from surveillance systems to augmented reality. This paper presents an approach to achieve real-time object recognition using a pre-trained model with Deep Neural Networks (DNN). Leveraging the capabilities of pre-trained models significantly reduces the computational cost and training time. The proposed system utilizes

a state-of-the-art DNN model that has been pre-trained on a large dataset, capturing diverse visual features.

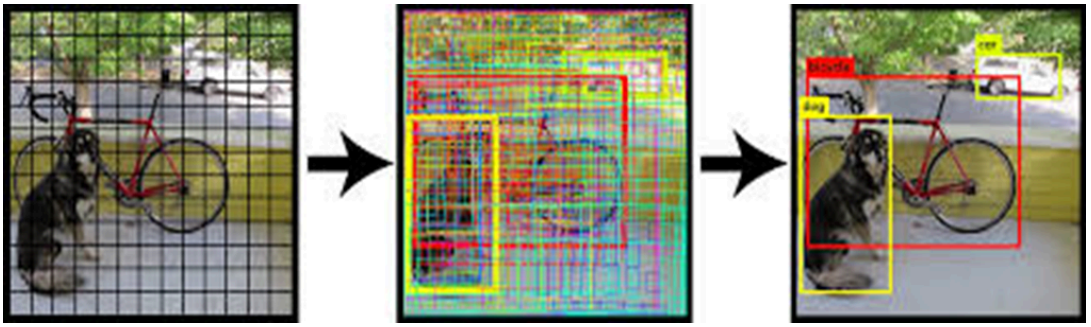
The methodology involves loading the pre-trained model and deploying it in a real-time environment to recognize objects in streaming input frames. By integrating the model with a real-time video processing pipeline, the system achieves low-latency object recognition, making it suitable for applications where timely decision-making is crucial. The DNN model's ability to generalize across different object categories enables the system to detect a wide range of objects with high accuracy.

Experimental results demonstrate the effectiveness of the proposed approach in terms of both accuracy and real-time performance. The system's robustness is evaluated under various scenarios, including varying lighting conditions and occlusions. The findings highlight the potential for deploying pre-trained DNN models in real-world applications, showcasing their adaptability and efficiency in addressing complex object recognition tasks. The presented framework serves as a foundation for developing intelligent systems capable of real-time object recognition in diverse and dynamic environments.

4. OBJECT RECOGNITION

Object recognition is a computer vision technique for identifying objects in images or videos. Object recognition is a key output of deep learning and machine learning algorithms. When humans look at a photograph or watch a video, we can readily spot people, objects, scenes, and visual details.

The field of artificial intelligence (AI) that deals with robots' and other AI implementations' capacity to identify different objects and entities is called object recognition. Robots and AI systems can recognise and distinguish items from inputs such as still and video camera images thanks to object recognition. 3D models, component identification, edge detection, and appearance analysis from various perspectives are some of the methods used for object identification. At the intersections of robotics, machine vision, neural networks, and artificial intelligence lies object recognition. Among the businesses involved are Google and Microsoft, whose Kinect system and Google's autonomous vehicle both rely on object recognition.



5. IMPACT OF AI IN OBJECT RECOGNITION

For contemporary algorithms, detecting an item is a rather simple process. Furthermore, machine learning has sufficient powers to handle it. But in terms of object recognition and labeling, these systems require more sophisticated technologies. The next generation of machine learning, known as deep learning or neural network models, is employed by the developers in these situations.

Typically, Python, an OpenCV object recognition library, and the TensorFlow object recognition API are used to construct such applications. A pre-developed set of features for face recognition, voice recognition, speech recognition, multiple object recognition with visual attention, handwriting analysis, and selective search for object recognition is included in this library.

AI's Effect on Object Recognition

Enhanced Accuracy: AI-powered object recognition systems frequently outperform conventional computer vision techniques in terms of accuracy. This improves the accuracy and dependability of apps that use object recognition.

Efficiency and Automation: AI increases efficiency and decreases manual involvement by automating the item detection and classification process. This is especially useful in situations when processing huge amounts of visual input quickly is required.

Enhanced Security: AI-driven object recognition helps to make surveillance systems more secure. It can improve general safety precautions by instantly identifying and notifying authorities of any potential risks or anomalies.

Creative Applications: The development of AI object recognition opens the door to creative applications like augmented reality, which combines digital and real-world data to create a seamless experience for consumers.

6. IMPLEMENTING OBJECT RECOGNITION

Object recognition, a crucial task in computer vision, can be approached through various methodologies. One effective strategy is to leverage a pre-trained model, allowing us to benefit from the knowledge gained on a vast dataset for a generic object recognition task. In this approach, a model, such as VGG, ResNet, or MobileNet, is selected based on its success in general object recognition.

Transfer learning becomes instrumental in tailoring the pre-trained model to a specific object recognition task. This involves fine-tuning the model's parameters on a smaller dataset that is pertinent to the target task. The original classification layers are often removed, and custom layers are added to adapt the model to the unique classes or categories relevant to the application at hand. The key distinction lies in the utilization of

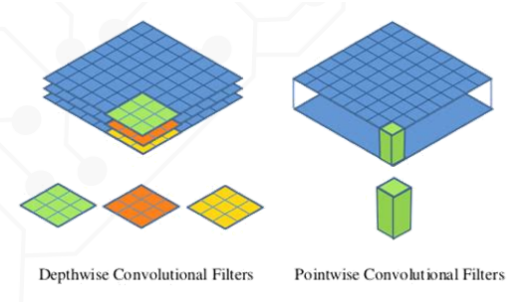
the learned features (weights) from this pre-trained model on a new, smaller dataset. By freezing the weights of the pre-trained model, we preserve the general features it has acquired. Additional custom layers are then added to the model to facilitate learning task-specific features. The model is then trained on the new dataset, focusing on refining its capabilities for the specific object recognition requirements.

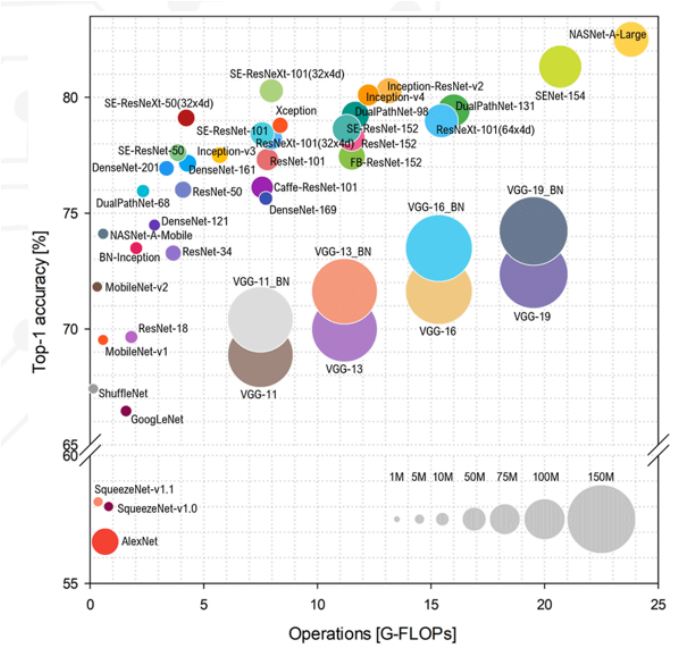
For those seeking a more tailored and customized solution, building an object recognition model from scratch is a viable option. In this approach, the neural network architecture is designed from the ground up, allowing for complete customization. The model is initialized with random parameters, and training commences on a dataset specifically curated for the target object recognition task.

In our object recognition model implementation, we employ a holistic approach that integrates three key strategies. Building upon this foundation, we seamlessly transition into transfer learning, fine-tuning the pre-trained model's parameters to align with our class weights and enhance its adaptability to our specific target classes. Simultaneously, we incorporate a methodology that involves starting with a pre-trained model on a large dataset and strategically leveraging its learned features on our new, smaller dataset through transfer learning. Additionally, we embrace the flexibility of building certain components from scratch, designing a bespoke neural network architecture for our object recognition task. This comprehensive approach ensures the model is not only accurate and optimized for our defined classes but also flexible enough to cater to diverse and nuanced object recognition scenarios.

6.1. Pre- Trained Model

Here we are using the MobileNetSSD model which uses the caffe model for object detection. We have a variety of models to perform object detection in real time but the reason to prefer Mobilenet is because of its convolution methods. The MobileNet model is based on depth wise separable convolutions which are a form of factorized convolutions. These factorize a standard convolution into a depthwise convolution and a 1×1 convolution called a pointwise convolution. For MobileNets, the depthwise convolution applies a single filter to each input channel. The pointwise convolution then applies a 1×1 convolution to combine the outputs of the depthwise convolution. A standard convolution both filters and combines inputs into a new set of outputs in one step. The depthwise separable convolution splits this into two layers – a separate layer for filtering and a separate layer for combining. This factorization has the effect of drastically reducing computation and model size. The SSD architecture is a single convolution network that learns to predict bounding box locations and classify these locations in one pass. Hence, SSD can be trained end-to-end.





6.2. Transfer learning

Transfer learning is a powerful technique in machine learning, particularly in the context of deep neural networks, where a model trained on one task is leveraged to perform a different but related task. In the realm of object detection, transfer learning has proven to be immensely effective, allowing the application of pre-trained models to new datasets or tasks with limited labeled data.

Fine-Tuning for Custom Object Detection:

Assume you have a pre-trained model on a general object detection dataset, like COCO. For a specific application, such as detecting custom objects in a retail environment, you can fine-tune the pre-trained model on a smaller dataset containing images of your specific objects. This process enhances the model's ability to detect those custom objects in real-time.

Adapting Models to Specific Environments:

In scenarios like autonomous vehicles or surveillance systems, transfer learning allows the adaptation of pre-trained models to specific environments. For instance, a model initially trained on urban scenes can be fine-tuned to recognize objects in rural landscapes, enhancing its real-time object detection capabilities in diverse settings.

Domain Adaptation for Security Applications:

In security applications, a model trained on one type of security camera feed can be adapted to another camera system through transfer learning. This ensures that the model remains effective in real-time object detection across different camera setups and perspectives.

6.3. Building from scratch

In object recognition model development, the approach of building from scratch involves the meticulous creation of a neural network architecture tailored to the specific requirements of the task. This process begins with the design of the model architecture, ensuring its suitability for object recognition. The model parameters are then initialized randomly, and extensive training on the target dataset follows. Notably, this method typically demands a larger dataset compared to transfer learning. An optional fine-tuning step may be incorporated based on the model's performance on validation data. The inherent benefits of building from scratch include unparalleled customization of the model architecture to suit the unique characteristics of the task at hand. Furthermore, this approach facilitates task-specific learning, allowing the model to acquire features that are specifically relevant to the defined object recognition task, providing a tailored and precise solution.

7. DNN - Deep Neural Network

Deep Neural Networks (DNN) stand as formidable structures in the field of artificial intelligence and machine learning. These systems, inspired by the intricate connectivity of the human brain's neural networks, hold the capacity to understand intricate patterns and make complex decisions. DNNs, characterized by layers of interconnected nodes, have become pivotal in various applications, with a notable presence in the code presented for real-time object detection.

DNN's Role in Object Recognition:

The code exemplifies the utilization of a DNN, specifically the MobileNet SSD architecture, to perform real-time object detection. MobileNet SSD, chosen for its lightweight design, showcases the prowess of deep neural networks in the domain of computer vision.

In the realm of object recognition, DNNs leverage their autonomous learning capabilities to discern complex patterns from raw data. Unlike traditional computer vision

techniques reliant on manually crafted features, DNNs autonomously learn these features through exposure to extensive datasets.

The code is crafted using Python, incorporating OpenCV and TensorFlow to construct a real-time object detection system. The MobileNet SSD model, loaded with pre-trained weights, processes each frame of the camera feed. Essential steps include resizing frames, creating input blobs, forwarding data through the DNN, and drawing bounding boxes around identified objects.

Advantages of DNN in Object Recognition

Feature Autonomy*:*

DNNs demonstrate a remarkable ability to autonomously learn hierarchical features, eliminating the need for meticulous manual feature engineering. This adaptability enhances performance across diverse datasets and scenarios.

Precision Enhancement:

The inherent capability of DNNs to grasp intricate patterns contributes to elevated accuracy in object recognition tasks. This is particularly evident in scenarios where objects exhibit variations in scale, orientation, and lighting conditions.

Versatility:

DNNs provide a high level of flexibility, allowing the integration of diverse architectures and models tailored to specific use cases. Their versatility makes DNNs indispensable tools for a wide spectrum of applications extending beyond object recognition.

In essence, the presented code utilizing the MobileNet SSD architecture is a tangible manifestation of the effectiveness of DNNs in real-time object detection. As technological strides continue, DNNs are poised to play a pivotal role in shaping the landscape of artificial intelligence and computer vision applications.

8. WHY MOBILENET SSD

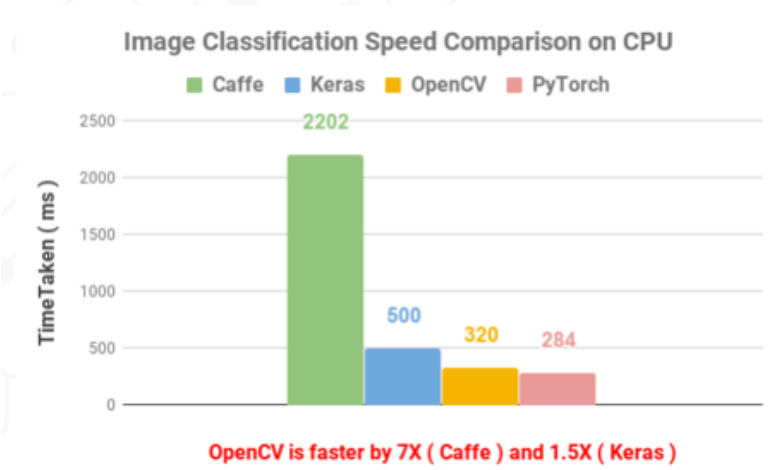
The adoption of MobileNet SSD for real-time object detection in the provided code is rooted in its lightweight architecture, a crucial attribute that enhances its efficiency in resource-constrained environments. This streamlined design facilitates swift inference, making it an optimal choice for applications requiring real-time responsiveness, such as those deployed on mobile phones and edge devices.

One of the key strengths of MobileNet SSD lies in its ability to strike a delicate balance between accuracy and speed. While it may not match the precision of larger and more complex models, its performance is often more than sufficient for various practical

scenarios. This balance is particularly advantageous in real-time applications, where swift processing of visual data is paramount.

The versatility of MobileNet SSD is another contributing factor to its selection. Its ease of fine-tuning allows developers to adapt the model to diverse object recognition tasks without compromising overall performance. This adaptability adds a layer of flexibility, making MobileNet SSD a versatile solution for a range of real-world applications.

The rationale behind choosing MobileNet SSD revolves around its lightweight design, balanced trade-off between accuracy and speed, adaptability, compatibility with mobile and edge devices, and efficient feature learning capabilities. Together, these characteristics position MobileNet SSD as a well-suited solution for the demanding requirements of real-time object detection.



9. BASIC SYNTAX

- **Loading Image from Disk to DNN**
    1. cv2.dnn.blobFromImage
    2. cv2.dnn.blobFromImages

➢ The cv2.dnn.blobFromImage and cv2.dnn.blobFromImages functions in the OpenCV library are used to preprocess images for deep neural network (DNN) inference. These functions are commonly employed when working with pre-trained deep learning models, especially in object detection, classification, and other computer vision tasks.

- cv2.dnn.blobFromImage:
    - This function is used to create a 4-dimensional blob from a single input image. The blob is a standardized format that is suitable for input to a deep neural network. The function takes the following parameters:
    - image: The input image that needs to be preprocessed.
    - scalefactor: Multiplier to scale the pixel values. It is usually used to

normalize the pixel values by dividing them by a certain factor (e.g., 255.0) to bring them into the range [0, 1].
        - size: The spatial size of the output blob (width, height).
        - mean: Mean subtraction values for each channel. These values are subtracted from the corresponding channels of the image.
        - swapRB: Flag to indicate whether to swap the Red and Blue channels. This is often set to True as many pre-trained models expect the input in BGR format.
        - crop: Specifies whether to crop the image after resizing.
        - The function returns the blob that can be fed into a deep neural network for inference.
    - cv2.dnn.blobFromImages:
        - Similar to blobFromImage, this function creates a 4-dimensional blob, but it is designed to handle multiple images simultaneously. It takes a list of images as input and generates a blob. The parameters are similar to blobFromImage, but there are slight differences:
        - images: A list of input images.
        - scalefactor, size, mean, swapRB, crop: Similar to blobFromImage.
        - The function returns a blob that corresponds to the batch of images provided.

- **Import Model from various Framework**
    1. cv2.dnn.createCaffeImporter
    2. cv2.dnn.createTensorFlowImporter
    3. cv2.dnn.createTorchImporter
    4. cv2.dnn.readNetFromCaffe
    5. cv2.dnn.readNetFromTensorFlow
    6. cv2.dnn.readNetFromTorch
    7. cv2.dnn.readhTorchBlob

➢ The provided functions from the OpenCV cv2.dnn module are used for working with various deep learning frameworks and loading pre-trained models into the OpenCV framework.

- cv2.dnn.createCaffeImporter: Creates an importer for models trained using the Caffe framework, allowing users to load and utilize these models within OpenCV.
- cv2.dnn.createTensorFlowImporter: Similar to the Caffe importer, this function creates an importer for models trained using the TensorFlow framework, enabling seamless integration with OpenCV.
- cv2.dnn.createTorchImporter: Creates an importer for models trained with the Torch framework, facilitating the use of Torch-based models in OpenCV applications.
- cv2.dnn.readNetFromCaffe: Reads a pre-trained model from a file in Caffe's model definition and weights format, initializing a network for

further use in OpenCV.

- cv2.dnn.readNetFromTensorFlow: Reads a pre-trained model from a TensorFlow model file, allowing for integration of TensorFlow models within OpenCV workflows.
- cv2.dnn.readNetFromTorch: Reads a pre-trained model from a Torch script, providing compatibility for Torch-based models in OpenCV applications.
- cv2.dnn.readhTorchBlob: Reads a pre-trained model from a Torch blob file, enabling the incorporation of Torch models in OpenCV workflows.

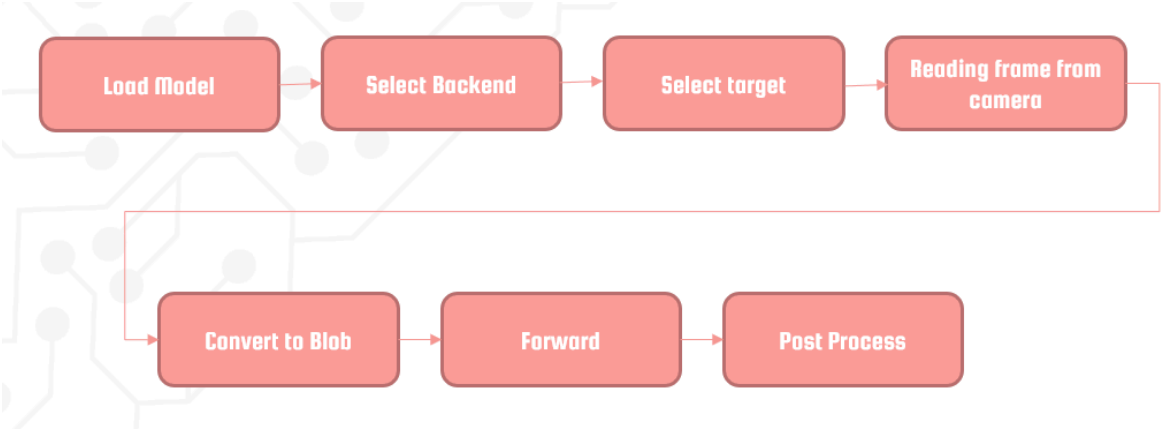
- **Blob Image**
    1. MeanSubtractedNormalizedImage=cv2.dnn.blobFromImage(resizedImage,scalingFactor, Spatial Size, Mean Subtraction Values)
    2. blob = cv2.dnn.blobFromImage(imResizeBlob,0.007843, (300, 300), 127.5)

➢ These lines of code are using the cv2.dnn.blobFromImage function in OpenCV to preprocess images for deep neural network (DNN) inference, typically in the context of object detection or classification. Here's an explanation of each line:

- MeanSubtractedNormalizedImage = cv2.dnn.blobFromImage(resizedImage, scalingFactor, Spatial Size, Mean Subtraction Values):
    - resizedImage: The input image that needs to be preprocessed.
    - scalingFactor: A multiplier to scale the pixel values. It is commonly used for normalization, e.g., dividing pixel values by 255.0 to bring them into the range [0, 1].
    - Spatial Size: The spatial size of the output blob (width, height).
    - Mean Subtraction Values: Mean subtraction values for each channel. These values are subtracted from the corresponding channels of the image.
    - The function returns a blob representing the preprocessed image, suitable for input to a deep neural network.

- blob = cv2.dnn.blobFromImage(imResizeBlob, 0.007843, (300, 300), 127.5):
    - imResizeBlob: The input image that needs to be preprocessed.
    - 0.007843: A scaling factor for normalization.
    - (300, 300): The spatial size of the output blob (width, height).
    - 127.5: Mean subtraction values for each channel. In this case, it is used to shift the pixel values after normalization.
    - The function returns a blob representing the preprocessed image, suitable for input to a deep neural network.

10. BLOCK DIAGRAM



11. CODE

```
import numpy as np

import imutils

import time

import cv2
```

The script starts by importing NumPy for efficient numerical operations, imutils for convenient image processing functions, time for handling time-related operations, and OpenCV (cv2) for computer vision functionalities.

```
prototxt = "MobileNetSSD_deploy.prototxt.txt"

model = "MobileNetSSD_deploy.caffemodel"

confThresh = 0.2

CLASSES = ["background", "aeroplane", "bicycle", "bird", "boat",

    "bottle", "bus", "car", "cat", "chair", "cow", "diningtable",

    "dog", "horse", "motorbike", "person", "pottedplant", "sheep",

    "sofa", "train", "tvmonitor"]

COLORS = np.random.uniform(0, 255, size=(len(CLASSES), 3))

print("Loading model...")

net = cv2.dnn.readNetFromCaffe(prototxt, model)
```

```
print("Model Loaded")
```

This segment initializes the script by defining the file paths for the MobileNet SSD model and its corresponding prototxt file. Configuration parameters such as the confidence threshold (confThresh) and object classes (CLASSES) are set. Random colors are generated for each class to enhance visualization. The script then loads the MobileNet SSD model using OpenCV's dnn module.

```
print("Starting Camera Feed...")

vs = cv2.VideoCapture(0)

time.sleep(2.0)
```

The script initializes the video capture using OpenCV (cv2.The default camera (index 0) is selected, and a brief delay (time.sleep(2.0)) allows the camera to stabilizera to stabilize before entering the real-time object detection loop.

```
while True:

    _, frame = vs.read()

    frame = imutils.resize(frame, width=500)

    (h, w) = frame.shape[:2]

    imResizeBlob = cv2.resize(frame, (300, 300))

    blob = cv2.dnn.blobFromImage(imResizeBlob, 0.007843, (300, 300), 127.5)

    net.setInput(blob)

    detections = net.forward()

    detShape = detections.shape[2]
```

This part of the code initiates a loop for real-time object detection. Each video frame is read, resized for efficient processing, and a blob is generated for input to the MobileNet SSD model. The model processes the blob, resulting in object detections.

```
    for i in np.arange(0, detShape):

        confidence = detections[0, 0, i, 2]

        if confidence > confThresh:

            idx = int(detections[0, 0, i, 1])
```

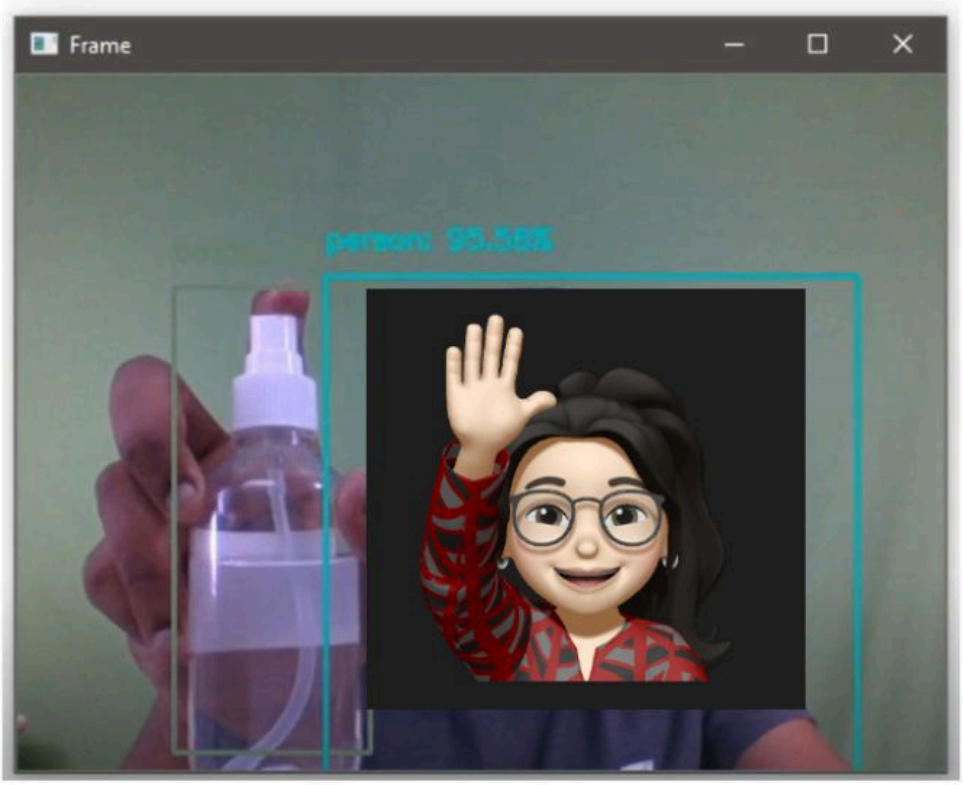
```
box = detections[0, 0, i, 3:7] * np.array([w, h, w, h])
(startX, startY, endX, endY) = box.astype("int")
label = "{}: {:.2f}%".format(CLASSES[idx], confidence * 100)
cv2.rectangle(frame, (startX, startY), (endX, endY), COLORS[idx], 2)
if startY - 15 > 15:
    y = startY - 15
else:
    y = startY + 15
cv2.putText(frame, label, (startX, y),
    cv2.FONT_HERSHEY_SIMPLEX, 0.5, COLORS[idx], 2)
```

Within the loop, this section processes each object detection, filtering based on confidence scores. Bounding boxes are drawn around detected objects, and class labels with confidence percentages are displayed on the frame.

```
    cv2.imshow("Frame", frame)
    key = cv2.waitKey(1)
    if key == 27:
        break
vs.release()
cv2.destroyAllWindows()
```

This part of the code introduces user interaction, allowing the user to exit the real-time display by pressing the 'Esc' key. Upon script completion, the video capture resource is released, and OpenCV windows are closed.

12. OUTPUT



13. APPLICATIONS

1. ***Surveillance and Security Systems:***

The script is valuable for real-time surveillance, enabling security systems to automatically detect and track objects in live camera feeds. This application enhances security measures by identifying potential threats or unusual activities promptly.

2. ***Smart Cities and Traffic Management:***

In smart city applications, the script can monitor traffic in real-time. It identifies and analyzes objects such as vehicles and pedestrians, contributing to efficient traffic management and optimization of transportation systems.

3. ***Retail Analytics:***

Retail stores can leverage the script for real-time monitoring of customer behavior and product interactions. Object detection assists in tracking customer movement, analyzing foot traffic patterns, and evaluating the popularity of products.

4. ***Human-Computer Interaction (HCI):***

The script supports gesture recognition and object interaction in HCI applications.

It can be integrated into gaming, virtual reality, and other interactive systems, enhancing user engagement through real-time object detection.

5. ***Autonomous Vehicles:***

Real-time object detection is crucial for the development of autonomous vehicles. The script identifies and tracks objects on the road, aiding the vehicle's perception system and contributing to safe navigation in dynamic environments.

14. FUTURE SCOPE

The future scope of the real-time object detection script is promising, with potential advancements and applications in diverse areas. Future exploration could involve integrating advanced deep learning models to enhance object detection accuracy and robustness. Continued research in computer vision may lead to adopting state-of-the-art architectures, enabling the script to handle a broader range of scenarios.

Additionally, there's potential to extend the script by incorporating multi-modal inputs, combining visual information with sensors like depth sensors or LiDAR. This enhancement would significantly improve the script's performance in challenging lighting conditions and scenarios involving occluded objects.

Future iterations of the script might focus on real-time tracking and behavior analysis of detected objects. This expansion could find applications in anomaly detection, triggering alerts or automated responses in security and surveillance systems when deviations from expected object behavior are observed.

Moreover, the script's adaptability to edge computing platforms offers an exciting avenue for exploration. Optimizations for efficient inference and lower power consumption can enhance accessibility, making the script applicable in edge computing scenarios and contributing to the deployment of intelligent systems in various edge environments. In specific industries, such as healthcare, the script can evolve to address unique needs, assisting in patient monitoring or surgical procedures. Collaborations with domain experts and user feedback could inform refinements tailored to specific use cases and industry requirements.

15. CONCLUSION

In conclusion, the real-time object detection script, seamlessly integrated with the MobileNet SSD model, stands as a beacon for transformative applications within computer vision. Its continuous evolution through the integration of advanced deep learning models ensures perpetual refinement and elevates the precision of object detection, positioning the script at the forefront of technological progress.

Looking ahead, the script envisions expanding its capabilities through the integration of multi-modal inputs, promising substantial improvements in performance and aligning with the evolving demands of dynamic environments. Beyond enhanced accuracy, the strategic trajectory emphasizes real-time tracking and behavioral analysis of detected objects, particularly valuable in anomaly detection for security and surveillance systems. The script's adaptability to edge computing platforms emerges as a promising avenue, facilitating deployment in resource-constrained devices and underscoring its relevance in diverse edge environments. As it charts its course into the future, collaborative efforts with domain experts and user feedback will be instrumental in shaping its evolution, ensuring precision, and relevance in addressing nuanced use cases and industry requirements.

In essence, the real-time object detection script holds the promise of leaving an indelible mark on the landscape of intelligent systems, dynamically shaping the future of real-time visual processing across diverse sectors.