Deeploy CV Project

Assignment: 2

Name: Mayank Agrawal

Roll Number: 230639

ID : 213

Branch: CE

Date of Submission: 28/12/2024

The source code and solutions for this assignment can be found at: GitHub Repository Link

Question 1: Object Detection Using YOLOv5 and SSD Models

Insights

This question provides object detection using YOLOv5 and SSD models. YOLOv5 demonstrated impressive speed and precision, making it ideal for dynamic, real-time applications. Its ability to handle input directly without extensive preprocessing simplified the implementation process. In contrast, SSD required input transformation but exhibited robust detection accuracy, particularly when coupled with confidence filtering to eliminate low-confidence detections.

The visualization process, involving bounding boxes and class labels in different colors for the two models, highlighted their respective detection strengths. This comparison underlined YOLOv5's efficiency for real-time scenarios and SSD's effectiveness for tasks prioritizing accuracy over speed. Overall, the implementation showcased the trade-offs between speed and accuracy in object detection models.

Differences Noticed

In side-by-side comparison:

- YOLO provided better detection for multiple and overlapping objects in the frame.
- SSD showed its strengths in detecting larger objects with precise bounding boxes.
- Some objects were not detected by either model, possibly due to training dataset limitations.

Model Details

YOLOv5: YOLOv5 (You Only Look Once) is a fast and efficient object detection model. It detects objects in a single step, making it ideal for real-time applications like security

cameras or self-driving cars. It's simple to use and works quickly without needing a lot of extra steps, which is why it's often chosen for tasks where speed is important.

SSD Model: The SSD (Single Shot MultiBox Detector) model is great at finding objects of different sizes. It takes a bit more time than YOLOv5 because it checks for objects at multiple scales to ensure better accuracy. This makes SSD a good choice for situations where detecting small or large objects clearly is more important than speed.

Output Image

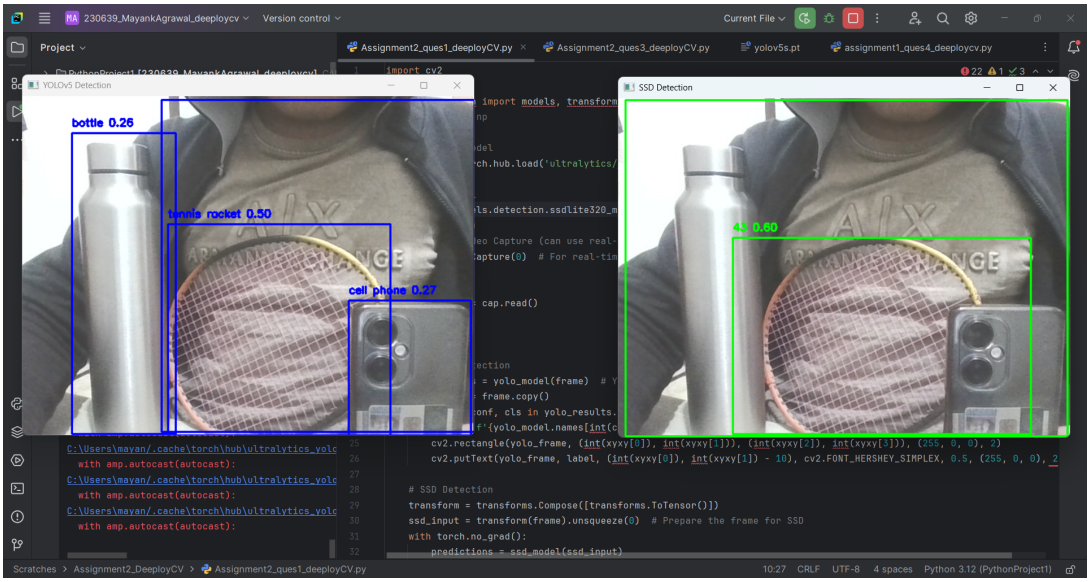

Figure 1: Output Image of Object Detection

Video demonstration: Google Drive Link

Question 3: Flag Detection Using SSD

Insights

Model Loading and Initialization: The SSD model was successfully loaded using OpenCV's dnn module, ensuring compatibility with the Caffe framework. Proper file path validation is critical to avoid runtime errors, as seen in the `load_ssd_model` function, which checks the existence of the prototxt and model files before loading.

Versatile Input Handling: The program supports both local image paths and URLs, enhancing flexibility. When handling URLs, it uses the `requests` library to fetch images, converting them to a NumPy array for compatibility with OpenCV functions. This demonstrates an effective way to handle multiple input sources in a single workflow.

Confidence Thresholding: By filtering detections based on a confidence threshold (default set at 0.5), the implementation ensures only reliable predictions are considered. This reduces noise and improves detection accuracy, particularly important in applications like flag classification.

Class-Specific Logic: The program incorporates specific logic to classify detected objects as flags. It currently supports flags like "Indonesia" and "Poland," demonstrating

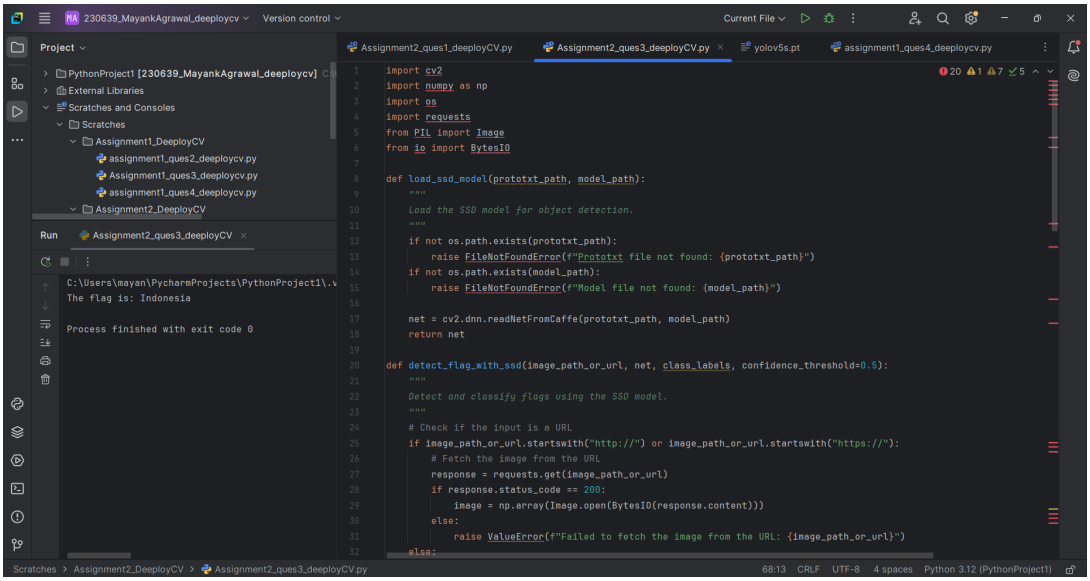

Figure 2: Output Image of Flag Detection

how custom object detection can be tailored for niche tasks. The use of `class_labels` allows for easy expansion by adding more categories.

Output Visualization: The bounding boxes and confidence scores drawn on the image provide clear visualization of detected objects. This makes it easy to interpret results and validate model performance during testing.

Real-World Use Case

The detection and differentiation between similar-looking flags (e.g., Indonesia and Poland) highlight the need for precise training and classification in object detection models. This approach could be extended for broader applications, such as identifying flags in sports events or diplomatic settings.

Potential Improvements

- Fine-tuning the confidence threshold for optimal accuracy.
- Expanding the `class_labels` dictionary to include more flag categories.
- Enhancing image preprocessing to handle diverse lighting conditions or orientations.

Output Image