Real-Time Object Detection and Tracking Using Custom AI Model (First Check-In)

Overview

For this first check-in of my final project in the Machine Vision class, I have implemented the initial stages of a real-time object detection and tracking system using a custom-trained Convolutional Neural Network (CNN). The goal is to recognize and track a specific object through a live webcam feed using Python, OpenCV, and TensorFlow. The project emphasizes training a lightweight, custom neural network rather than using large pre-trained models like YOLO or SSD, giving more control over the dataset and architecture.

Goals Accomplished (First Check-In Deliverables)

1. Dataset Collection

- Captured a total of 261 images using a custom Python script (capture_frames.py) that interfaces with the webcam.
- The target object (e.g., an AirPods case) was photographed under various lighting conditions and angles to support robust training.

2. Environment and Project Setup

- Created a fully structured Python project with folders for scripts, dataset, and model.
- Set up a virtual environment and installed all required packages including OpenCV, TensorFlow, NumPy, and scikit-learn.

3. Annotation Tool Setup (CVAT)

- Explored multiple annotation tools including LabelImg and Makesense.ai. Ultimately, CVAT was chosen for its professional labeling interface and export support for YOLO format.
- All images were successfully uploaded to CVAT and the annotation interface has been verified. Annotation will be carried out post-check-in.

4. Model Architecture and Training Pipeline

- Designed and implemented a custom CNN in TensorFlow to predict bounding box coordinates from a single object image.

- Wrote the train.py script to load YOLO-style annotation files, process image-label pairs, and train the detection model.
- Created main.py to perform live webcam detection and visualize the bounding box overlaid on the frame.

Deliverables

For this first check-in, I have completed the following:

- A collection of 261 labeled images (raw, unannotated)
- Fully functional capture_frames.py, train.py, and main.py scripts
- Configured and tested CVAT for annotation
- Established the neural network pipeline for training and real-time use

---

Challenges Faced

- Annotation Tool Limitations: LabelImg was unreliable on my system, so I had to explore alternatives and settle on CVAT, which added some setup time.
- Dataset Diversity: Capturing images with sufficient variability in object position and lighting took multiple iterations.
- Label Format Conversions: Preparing for training with YOLO-style labels required writing additional code to parse annotations into usable bounding box formats.

Next Steps

- Complete annotation of at least 200 images using CVAT, exporting in YOLO format.
- Train the CNN model using the annotated dataset and evaluate its accuracy.
- Integrate a real-time tracking module into main.py using OpenCV's tracking algorithms (e.g., KCF or MOSSE).
- Optimize the detection speed by minimizing detection frequency and resizing input frames.

Conclusion

At this stage, the foundational components of the object detection system are in place. The image dataset is complete, the training pipeline is built, and the real-time webcam interface is working. The next phase will focus on training the model with labeled data and enabling reliable real-time tracking. This milestone represents approximately 30% completion of the overall project.