OBJECT DETECTION FOR HOME SECURITY

CSE573- Computer Vision and Image Processing

Name: Venkata Mani Sai Praneeth Chidella UBIT:50560459 UBName: vchidell

Name: Reddy Kamal Teja Thota UBIT:50560696 UBName: reddykam

Application:

The goal of this is to create a Home Security System that work in real-time, identifying and categorizing objects like people, pets, packages, or any unusual items through security cameras. The system will continuously monitor the surroundings and send alerts in case of suspicious activity. This approach will be beneficial for homeowners, apartment residents, and businesses looking for an automated way to enhance security.

State of Art:

Existing security systems using motion or simple object detection often trigger false alarms from shadows, leaves, or pets. We enhance the approaches by comparing the performance of YOLOv8, YOLOv10 and YOLOv12 to find the most accurate and efficient model for real-time object detection, distinguishing between critical and non-critical such as difference between person and a small animal object.

Similar projects: (1) "YOLO-based Real-Time Object Detection for Surveillance" (IEEE) – Uses YOLO for security monitoring. (2) GitHub:ultralytics/yolov8 – Offers pre-trained YOLOv8 models for object detection.

Inputs and Outputs:

- Inputs: Live video feed from a laptop webcam or security cameras or any recorded footage.
- Intermediate Outputs: Frames with bounding boxes, detected object labels.
- Final Outputs: Labeled objects in video feed, logs of detected objects, and optional alert notifications.

Required Data:

- Pre-trained Model Data: This project will use the COCO dataset, which contains labeled images of common objects like people, animals, and vehicles for training and testing. we will be using an existing data set (coco dataset)
- data acquisition: Using pre-trained YOLO models on the COCO dataset. if needed we will collect real-world webcam footage for model fine-tuning for better results.
- Scale of Data: COCO dataset contains over 118,000 training images across 80 object categories.

Coding Resource Requirements:

We are going to usePython as the primary programming language, utilizing libraries such as OpenCV, PyTorch, YOLOv8, YOLOv10, YOLOv12 and NumPy and we are going to use TensorFlow/PyTorch framework. I am leveraging pre-trained YOLO models from the Ultralytics GitHub repository while customizing detection thresholds, bounding box filtering, and alert mechanisms to enhance performance. Additionally, I am independently contributing to the implementation of real-time video feed processing using OpenCV, developing an alert mechanism for notifications via email or SMS, and optimizing inference speed to ensure efficient real-time execution.

Computational Resource and Effort Requirements:

The project will require a capable development machine with a GPU for efficient training and real-time inference. Edge computing solutions like Raspberry Pi may be used for deployment, potentially enhanced with a TPU for improved performance. Training on large datasets like COCO can take a considerable amount of time, while real-time object detection will be much faster on a GPU compared to a CPU.

The project is expected to require approximately **2-3 days of GPU time** for model training and fine-tuning. In terms of effort, the estimated workload per person is around **90-120 hours**, covering tasks such as data preprocessing, model customization, testing, and performance optimization to ensure real-time detection capabilities.

Evaluation:

The success of the project will be defined by achieving accurate real-time detection of humans, animals, and packages with minimal false positives and false negatives. The system should maintain an FPS of 30+ for real-time performance. Evaluation will be based on metrics like mAP (Mean Average Precision) for overall accuracy, precision & recall assessing detection correctness, and FPS to ensure real-time capability. Testing will cover both simple scenarios with clear visibility and challenging cases such as low-light conditions, occluded objects, and small objects in the background. Among the three models used, we will select the best-performing one based on these evaluation metrics.

Project Expectations:

we are excited in using YOLO models for accurate and fast object detection in security systems. This will help improve the effectiveness of real-world security applications by making them more reliable and responsive. This project has the potential to enhance various security applications, making them smarter and more capable of handling complex scenarios.

we hope to gain a deep understanding of real-time object detection, explore optimization techniques for low-latency processing, and learn how to develop effective real-world security solutions. Additionally, we aim to improve our skills in integrating machine learning models into practical applications. This project will provide valuable insights into building efficient, scalable systems that can handle complex tasks in real-time environments.

Additional Guidelines:

The analysis will also discuss limitations such as poor lighting or clutter in the environment, but during this evaluation with each YOLO model, the execution time will be checked and optimized for real-time detection. The present results will be used in suggesting such potential improvements as fine-tuning and advanced data augmentation. Moreover, the project will not only focus on retraining a classification head but will also perform a comparative analysis of YOLO models at a broad level using both custom and pre-trained data.

Progress Report 1

Progress to Date:

Since the initial project proposal, we have refined our approach based on instructor feedback. Our work has focused on the **comparative analysis of YOLO models (v8, v10, v12)** for home security applications. Key developments include:

- **Model Setup & Testing**: Implemented and tested YOLO models using a live webcam feed.
- **Performance Metrics Evaluation**: Captured mAP, FPS, Precision, and Recall for real-time detection.
- **Data Collection & Preprocessing**: Used **COCO dataset** along with real-world webcam footage.
- **Implementation Improvements**: Optimized detection thresholds, bounding box filtering, and alert mechanisms.

Changes to the Original Proposal:

Following instructor feedback, we made the following adjustments:

1. **Expanded Model Scope**: Initially focused on YOLOv8 but extended the study to YOLOv10 and YOLOv12.
2. **Enhanced Data Evaluation**: Introduced a broader dataset to test models under **challenging conditions** (low-light, occlusions, small objects).
3. **Performance Benchmarks**: Included additional evaluation metrics to **ensure real-time feasibility** in a home security setting.

Preprocessing Steps:

- Step 1 - Data Acquisition:
  Dataset Used: COCO dataset for pre-trained models.

- **Step 2 - Data Cleaning**
  Removing Unwanted Frames:
  Filter out blurry, low-quality, or redundant frames.
  Remove frames with little to no relevant objects.

- **Step 3 - Image Processing for Model Readiness**
  Resizing Images:
  Resize frames to 640×640 or the required input size for YOLO models.

Maintain aspect ratio to prevent distortion.
Normalization:
Convert pixel values to a range of [0,1] or [-1,1] to ensure consistent model inputs.

Planned Tasks:

- **Fine-Tuning & Hyperparameter Optimization**: Adjust detection parameters for better real-time accuracy.
- **Edge Deployment Considerations**: Investigate Raspberry Pi/NVIDIA Jetson for model deployment.
- **Alert System Development**: Integrate email/SMS notifications for detected security events.
- **Finalizing Model Selection**: Identify the best-performing YOLO model based on efficiency and accuracy.
- **Report & Presentation Preparation**: Document findings and create visual results for submission.

Schedule Through Project Completion:

Task	Timeline	Status
Data collection and preprocessing	**Up to Feb 25**	**completed**
Model selection and Training	**Ongoing**	**In progress**
Advanced Model bench marking	**March**	**In progress**
Fine - Tuning the model	**March**	**Awaiting**
Final Testing and deployment	**April**	**Awaiting**
Final Release	**April 24**	**Awaiting**

Key Accomplishments:

We have successfully completed several milestones in the project:

1. **Implementation of Multiple YOLO Models**
    a. Integrated **YOLOv8, YOLOv10, and YOLOv12** using the **PyTorch framework**.
    b. Established a **uniform testing pipeline** for fair comparison.
2. **Comparative Performance Evaluation**
    a. Collected **detection metrics** and compared inference speeds across models.
    b. Identified strengths and weaknesses of each YOLO version.
3. Improved **frame processing speeds** to enhance real-time execution.

Progress report 2:

1. **Model Selection & Implementation**
    a. Completed evaluation of YOLOv8, YOLOv10, and YOLOv12 for home security applications.
    b. Set up and tested models using live webcam feeds.
    c. Captured performance metrics including mAP, FPS, Precision, and Recall.
2. **Training & Testing**
    a. Used the COCO dataset for data collection.
    b. Preprocessed data by removing unwanted frames, resizing, and normalization.
    c. Trained the models, ensuring optimal performance in real-time detection.
    d. Conducted benchmarking to compare model performance.
3. **Performance Optimization**
    a. Fine-tuned detection thresholds and bounding box filtering.
    b. Enhanced alert mechanisms for better response time.
    c. Optimized frame processing speeds to improve real-time execution.

Task	Timeline	Status
Data collection and preprocessing	**Up to Feb 25**	**completed**
Model selection and Training	**April 2nd**	**completed**
Advanced Model bench marking	**ongoing**	**In progress**
Fine - Tuning the model	**March**	**Awaiting**
Final Testing and deployment	**April**	**Awaiting**
Final Release	**April 24**	**Awaiting**

Final Remarks:

The project has successfully completed model selection, training, and initial testing. The next phase involves fine-tuning and deployment to determine the best model for real-time home security monitoring.