Object Detection in Monitoring Systems

Taraka Maruthi Paruchuru
tparuchu@depaul.edu

Nov 17, 2024

Abstract

This project focuses on enhancing object detection for surveillance systems by optimizing the YOLOv11 model. While the pre-trained YOLO model effectively detects common objects like people, bicycles, and cars, it struggled with detecting specialized items such as weapons. To address this, we annotated a dataset with weapon images using Roboflow, Trained and fine-tuned the YOLO model on this custom dataset. Our approach significantly improved detection performance, achieving higher accuracy, precision, and recall for weapon detection. Our fine-tuned YOLOv11 model achieved a mAP of 89.5%, significantly improving from the pre-trained model's 41.5% . The results demonstrate the importance of model optimization using custom dataset for training and parameter tuning and for specialized tasks in real-time object detection applications, particularly in security and surveillance contexts.

1 Introduction

The primary objective is to develop a system capable of accurately identifying and tracking objects in real-time, providing valuable data for security surveillance. Existing systems lack precision in detecting small or specialized objects like weapons, which poses challenges for real-time security monitoring. One significant challenge is the variability in object appearances, shapes, and sizes, which can complicate detection and classification tasks. Additionally, environmental factors such as lighting conditions, occlusions, and weather can affect the performance of detection algorithms. So we aim for improving detection accuracy and testing of a surveillance systems real-time feasibility

2 Related Work

The field of object detection has seen significant advancements over the past decade, primarily driven by developments in deep learning. A key paper in this area is "Object detection using YOLO: challenges, architectural successors,

datasets and applications," which reviews the evolution of object detection algorithms. It classifies these algorithms into two categories: two-stage detectors, such as Fast R-CNN, and single-stage detectors, like YOLO (You Only Look Once) Another important study, "Application of Deep Learning for Object Detection," discusses the role of deep learning techniques, particularly convolutional neural networks (CNNs), in enhancing object detection performance. This paper highlights the significant improvements in visual recognition tasks, such as image classification, localization, and detection, driven by deep learning frameworks My project will build upon these existing works by focusing specifically on object monitoring within the context of real-time detection. While previous studies have explored general object detection using HAAR Cascade, HOG SVM and Pre-trained YOLO Model, I aim to tailor the YOLO architecture to improve object detection accuracy and efficiency in surveillance monitoring scenarios.

3 The Methodology

This section explains the approach used to customize the YOLOv11 model for object detection in surveillance systems. The methodology includes dataset preparation, model customization, data preprocessing, training, and evaluation.

Dataset Preparation The dataset was built by sourcing images of weapons and other relevant objects (e.g., police, civilians, attackers, Weapons) from publicly available datasets and surveillance footage. The dataset included annotated images with bounding boxes for four classes:

- Weapon
- Attacker
- Civilian
- Police

Annotations Annotation was performed using Roboflow, a robust tool for creating and managing labeled datasets. Bounding boxes were drawn around objects of interest. Class labels were assigned to each object. Data Augmentation techniques were used to generate images under varying conditions. **Data Split** The dataset was divided into training, validation, and test sets Percentage of images used for Train, Test and Valdation. 70% Training 20% Validation 10% Test **Augmentation** Addressed issues of critical real time images under varying conditions and class imbalance (fewer attacker images) by applying data augmentation techniques such as flipping, rotation, and brightness adjustments to diversify the dataset.

YOLOv11 Model Customization The YOLOv11 model was initialized with pre-trained weights from the COCO dataset, which includes 80 common object classes. These weights provided a strong foundation for general object detection. **Fine-tuning** To adapt the model to the custom dataset, the final

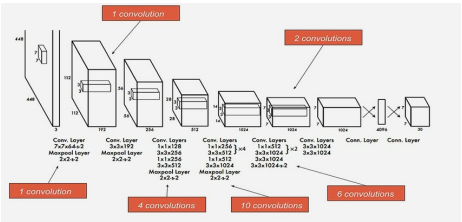

Figure 1: Yolo Architechture

layers were fine-tuned to detect the new object classes specific to our Project Specific Surveillance needs so Adjusted the model to recognize four new classes. Re-trained the detection head on the custom dataset while freezing earlier layers to retain general feature extraction capabilities.

Environment Training was performed on a machine with the following specifications:

1. GPU: Apple M1 Pro
2. Framework: PyTorch torch-2.5.1
3. Library: Ultralytics YOLOv11(8.3.32)
4. Epochs: 50
5. Optimizer: Adam
6. Loss Function :The total loss consisted of three components Localization Loss,Confidence Loss,Classification Loss.
7. Evaluation Metrics: Mean Average Precision (mAP),Precision,Recall,Speed.

4 Preliminary/Background

YOLO architecture is similar to GoogleNet. As illustrated below, it has 24 convolutional layers, four max-pooling layers, and two fully connected layers.

The YOLO architecture typically consists of the following components: Backbone: A convolutional neural network (CNN) that extracts features from the input image. Popular backbones include Darknet53, EfficientNet, and ResNet. Neck: Processes the feature maps from the backbone. Often involves feature pyramid networks (FPNs) to combine features from different layers, improving detection performance at multiple scales. Head: Predicts bounding boxes and class probabilities for each cell in the grid. Typically uses a series of convolutional layers and fully connected layers The architecture works as follows by intially Resizing the input image into 448x448 before going through the convolutional network. Next a 1x1 convolution is first applied to reduce the number of channels, followed by a 3x3 convolution to generate a cuboidal output. The

activation function under the hood is ReLU, except for the final layer, which uses a linear activation function. Some additional techniques, such as batch normalization and dropout, regularize the model and prevent it from overfitting. YOLO often uses anchor boxes to predict bounding boxes of different sizes and aspect ratios. Non-Maximum Suppression (NMS) technique used to eliminate redundant detections and improve accuracy.

The pre-trained YOLO model performed well for detecting general objects like people and vehicles but struggled with detecting specialized objects, such as weapons, which were not part of the pre-trained model's dataset. To address this limitation, we annotated a dataset containing weapons using Roboflow and retrained the model on this custom dataset.

The assumption made in the methodology was that a sufficiently large and diverse dataset of weapon images would improve the model's ability to detect these objects accurately. Fine-tuning the YOLO model on this dataset would result in better detection performance for security and surveillance applications

The below metrics to evaluate the models object detection performance.

Precision - The proportion of true positive detections among all predicted positive detections.

Recall - The proportion of true positive detections among all actual positive objects in the dataset.

mAP (mean Average Precision) - A performance metric that calculates the average precision across all classes, considering different IoU thresholds.

5 Numerical Experimental Evaluation

The experiments were conducted to evaluate the performance of the customized Roboflow, YOLO pre-trained model on real-time surveillance video feeds.

Metric	**YOLO Pretrained V11 Object Detection (Fast)**	**YOLO (Customized)**
Model Type	YOLO Detection pt (Fast)	YOLOv11 best trained
Checkpoint	COCOn	Custom Dataset
mAP	41.5%	89.5%
Precision	62.9%	90.7%
Recall	34.1%	88.2%

Table 1: Object Detection Performance Metrics for YOLO PT and YOLO Customized

6 Conclusion

In this project, we found that the pre-trained YOLO model performs well for detecting common objects like people, bicycles, and cars but struggled with detecting specific items like weapons. To overcome this, we customized the model by annotating images with weapon data and fine-tuning its parameters.

Figure 2: Customized YOLOv11 Inference

This process significantly improved the model's performance, enabling it to better detect weapons. Our results highlight the importance of adapting pre-trained models to specific tasks for enhanced detection accuracy, especially in security and surveillance applications. Future work could focus on expanding the object categories and further improving robustness.