Artificial Neural Network

Md. Hasnain Ali
Student Id: 2010976153

June 2025

Problem 1

Capture a video without any human face using a smart phone and detect object using Ultralytics provided YOLOv8, YOLOv11 and YOLOv12 models. Compare their object detection capability, speed.

In this experiment, we evaluate and compare the performance of three object detection models from the YOLO (You Only Look Once) family—YOLOv8, YOLOv11, and YOLOv12—on a real-time surveillance video captured in an office environment. The comparison focuses on two key metrics: detection speed (frames per second, FPS) and object detection capability (average objects detected per frame).

Setup

A smartphone-captured video was processed using the Ultralytics framework for each of the three models. The models analyzed each frame of the video, drawing bounding boxes and labeling detected objects. We saved frame 1000 from each model as a visual sample and computed the average FPS and average number of detected objects per frame across the video.

Results and Discussion

YOLOv8

Speed: 12.28 FPS.
Detection: Detected an average of 12.06 objects per frame. In frame 100, it detected 3 "tv" objects with confidence scores of 0.75, 0.65, and 0.80. However, it missed other visible objects such as "chair" and "book".

YOLOv11

Speed: 18.32 FPS (highest among all).
Detection: Detected an average of 13.19 objects per frame. In frame 100, it identified 3 "tv" (0.73, 0.64, 0.83), 2 "chair" (0.28, 0.26), and 1 "book" (0.57). It demonstrated both high speed and robust detection accuracy.

Figure 1: Object detection output from YOLOv8 on frame 100

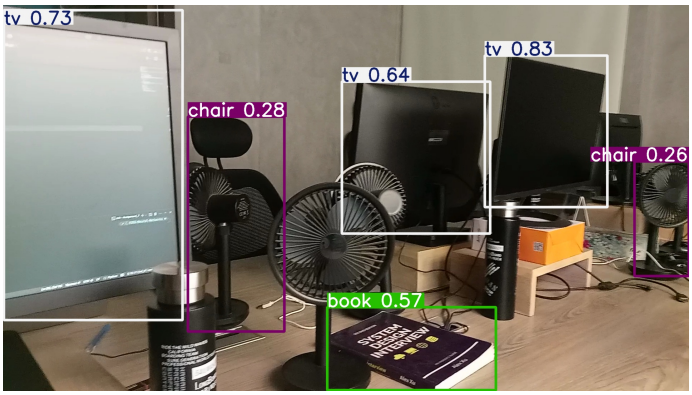

Figure 2: Object detection output from YOLOv11 on frame 100

YOLOv12

Speed: 6.63 FPS (slowest).
Detection: Detected the highest average objects per frame (13.39). In frame 100, it detected 3 "tv" (0.78, 0.49, 0.80), 2 "chair" (0.35, 0.38), and 1 "book" (0.58). However, the low FPS suggests a high computational load.

Results

This comparative study highlights the trade-offs between detection speed and object recognition accuracy among YOLOv8, YOLOv11, and YOLOv12:

- **Best Speed:** YOLOv11 with 18.32 FPS.
- **Most Objects Detected:** YOLOv12 with 13.39 average objects per frame, though at a much slower FPS.

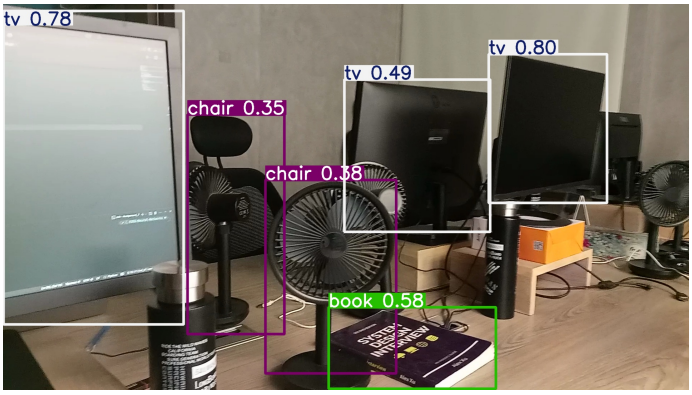

Figure 3: Object detection output from YOLOv12 on frame 100

- **Balanced Performance:** YOLOv11, offering both high speed and robust object detection, appears most suitable for real-time applications.

Given the computational efficiency and detection accuracy of YOLOv11, it is the most promising model for deployment in real-time surveillance systems, aligning with the needs of intelligent security monitoring applications.

Problem 2

Fine-tune any Ultralytics YOLOv8 model to prepare it as a face detector using WIDER FACE training set by splitting it into training set and validation set.

Selected Model

For this task, we selected the **YOLOv8n (Nano)** model from the Ultralytics YOLOv8 family due to its lightweight architecture and fast inference capabilities. Despite its compact size, YOLOv8n offers competitive performance, making it ideal for real-time face detection scenarios on edge devices or constrained hardware environments.

Dataset Description

We utilized the **WIDER FACE** dataset, a widely adopted benchmark for face detection that presents significant variability in scale, pose, occlusion, and expression. The dataset has train, test and validation split:

- **Training Set:** 12,880 images
- **Validation Set:** 3,226 images
- **Test Set:** 16,097 images (test has no labels)

Each image is annotated with bounding boxes indicating the location of human faces. The annotations were converted to YOLO format to align with the training requirements of the Ultralytics framework.

Data Preparation

- Converted WIDER FACE annotations from MATLAB format to YOLO-compatible format (i.e., `class_id, x_center, y_center, width, height` normalized to image size).
- I have taken the training set and split them into 90% train and 10% validation set:
  - **Training Set:** 11592 images
  - **Validation Set:** 1288 images
- Organized the dataset directory in the structure required by Ultralytics: `images/train`, `images/val`, `labels/train`, and `labels/val`.

Training Configuration

The fine-tuning was carried out using the Ultralytics YOLOv8 training interface with the following hyperparameters:

- **Model:** yolov8n.pt (pre-trained weights)
- **Epochs:** 30
- **Image Size:** 640×640 pixels
- **Batch Size:** 16
- **Frozen Layers:** First 13 layers frozen to retain base feature extraction
- **Number of Classes:** 1 (Face)
- **Optimizer:** SGD (default in Ultralytics framework)
- **Learning Rate Schedule:** Cosine decay

Problem 3

Compare fine-tuned YOLOv8 based face detector with face detector available in the following GitHub directory using the validation set of WIDER FACE dataset.

The performance of various YOLOv8-based face detection models was evaluated on the WIDER FACE validation set, categorized into Easy, Medium, and Hard subsets. The following table presents the Average Precision (AP) scores for each model:

Table 1: Comparison of YOLOv8 variants on WIDERFACE validation set (our experiment)

Model	**Easy Val. AP**	**Medium Val. AP**	**Hard Val. AP**
YOLOv8 Large	0.9728	0.9496	0.7902
YOLOv8 Medium	0.9736	0.9467	0.7651
YOLOv8 Nano	0.8812	0.8251	0.6023
YOLOv8 Nano (Ours)	**0.9110**	**0.8618**	**0.6097**

Analysis

The results in Table 1 highlight the performance of different YOLOv8 variants on the WIDER FACE validation set. The fine-tuned YOLOv8 Nano model (denoted as "Ours") demonstrates noticeable improvements over the baseline YOLOv8 Nano model from the GitHub repository [**?**], particularly in the *Easy* and *Medium* subsets.

Problem 4

Build a face detector having the architecture of YOLOv1, train it using WIDERFACE WIDER FACE training set by splitting it into training set and validation set.

This report presents the implementation of a face detector based on the YOLOv1 architecture trained on the WIDER FACE dataset. The model divides input images into grids and predicts bounding boxes with confidence scores and class probabilities. We describe model architecture, training procedure.

Model Architecture

The YOLOv1 architecture consists of multiple convolutional layers for feature extraction, followed by fully connected layers for detection. The network outputs a tensor of shape $7 \times 7 \times (5 + C)$, where C is the number of classes (in this case, 1 for face detection), and 5 corresponds to bounding box parameters and confidence.

The main layers include:

- Convolutional layers with kernels of sizes 7x7, 3x3, and 1x1 for multi-scale feature extraction.
- Max-pooling layers to reduce spatial dimensions.
- Dense layers with dropout to learn complex features and prevent overfitting.

Training Procedure

The model was trained using the Adam optimizer with a learning rate of 0.001. The loss function combined bounding box coordinate regression and confidence score prediction using Mean Squared Error (MSE). Training was conducted over 20 epochs with a batch size of 8. The dataset was shuffled and batched to ensure stable training.