BT22CSD009 Krishna Khandelwal

BT22CSD011 Kyatham Manideep

BT22CSD012 Johad Rasul

BT22CSD013 Sohan Meshram

BT22CSD014 Anurag Khobragade

BT22CSD015 Aryan Chandoliya

BT22CSD017 Brij Patel

CVDL Assignment

Group 2

Dr. Jagdish chakole

2025

1. **Objective**

The objective is to train Faster R-CNN and YOLOv5 on the same dataset under equivalent conditions to ensure a fair comparison. The aim is to optimize hyperparameters for both models to maximize performance. It is essential to evaluate detection accuracy using metrics such as mAP, IoU, precision, and recall. In addition, measuring inference speed through FPS and latency is a critical component of the comparison. The study will analyze performance variations across different object sizes and classes. It also seeks to assess the computational efficiency regarding training time and resource utilization. The robustness of each model will be tested under challenging scenarios like occlusion and cluttered backgrounds. An in-depth error analysis will identify common failure modes for both models. The research will compare the trade-offs between high detection accuracy and rapid inference speed. Ultimately, this work will provide practical insights into which model is better suited for specific real-world applications.

1. **Introduction**

Object detection plays a pivotal role in computer vision, with applications spanning autonomous driving, surveillance, and robotics. In this report, we examine two leading object detection models—YOLOv5 and Faster R-CNN—each representing distinct methodological approaches to the task. By leveraging the Pascal VOC 2012 dataset, we conduct a detailed comparison of their architectures, performance metrics, and practical deployment scenarios. This analysis aims to highlight the strengths and weaknesses of both models, providing insights into their suitability for various real-world applications.

1. **Dataset Overview**

The Pascal VOC 2012 dataset is used for evaluation, containing:

* **Number of objects**: 3,524
* **Number of images**: 1,500
* **No NaN values or duplicates detected**
* **Dataset split**:
  + Train: 11,927 images
  + Validation: 3425 images
  + Test: 1713 images
* **Number of classes**: 20
  + Examples: Aeroplane, Bicycle, Bird, Boat, Car, Person, etc.

1. **Architectural Differences**

### **YOLOv5 (You Only Look Once v5)**

* **Architecture & Design:**YOLOv5 is a single-stage detector designed for efficiency and speed. It processes the entire image in one pass, predicting bounding boxes and class probabilities directly without a separate region proposal stage.
* **Backbone – CSPDarknet53:**The network uses CSPDarknet53 as its backbone. This architecture leverages cross-stage partial networks (CSP) to reduce computation while maintaining strong feature representation, making it lightweight and efficient.
* **Detection Head & Anchor-based Detection:**YOLOv5 uses an anchor-based mechanism for object detection. Multiple anchor boxes are predefined at different scales and aspect ratios. During inference, the network directly predicts bounding boxes and class probabilities for each anchor in one forward pass, streamlining the detection process.
* **Feature Extraction & PANet Integration:**For effective multi-scale feature fusion, YOLOv5 employs a PANet (Path Aggregation Network). This enhances the information flow from lower to higher layers, improving the detector’s ability to capture objects of various sizes and ensuring fine details are preserved in deeper layers.
* **Speed & Real-time Optimization:**The model is highly optimized for real-time applications. Its single-stage architecture, along with an efficient backbone and feature fusion strategy, allows for rapid inference, making it suitable for tasks requiring high frame rates and low latency.
* **Inference Process:**During inference, YOLOv5 processes the input image in a single pass. It generates predictions by applying learned filters across the image, thereby directly outputting both bounding box coordinates and class probabilities in one seamless operation. This direct approach minimizes computational overhead and maximizes speed.

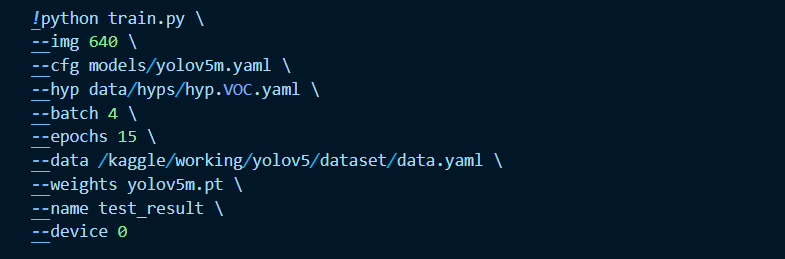
### **Faster R-CNN (Region-based Convolutional Neural Network)**

* **Architecture & Design:**Faster R-CNN is a two-stage detector, which means it separates the object detection process into two distinct phases. The first stage proposes potential regions of interest (ROIs) using a Region Proposal Network (RPN), and the second stage refines these proposals by classifying them and adjusting their bounding boxes.
* **Backbone – ResNet/VGG with FPN:**The network typically utilizes robust convolutional backbones like ResNet or VGG. These are often enhanced with a Feature Pyramid Network (FPN) that builds multi-scale feature maps, providing strong representations at various scales and helping the model detect objects with significant size variations.
* **Region Proposal Network (RPN):**The RPN is a key component in Faster R-CNN. It slides over the convolutional feature map and generates multiple region proposals per spatial location. Each proposal is assigned an objectness score, indicating the likelihood of containing an object, and initial bounding box coordinates. This step reduces the number of regions that need to be processed in the subsequent stage.
* **Detection Head – Classification and Regression:**After the RPN, the proposed regions are cropped from the feature map using a technique like RoI pooling. These cropped regions are then passed through a series of fully connected layers. This detection head is responsible for fine-tuning the bounding boxes and assigning class labels, effectively refining the initial proposals from the RPN.
* **Speed Considerations:**The two-stage processing inherent in Faster R-CNN tends to be slower than single-stage detectors like YOLOv5. This is due to the additional computation required for generating region proposals and the subsequent refinement process, making it less ideal for real-time applications but often more precise in detection.
* **Inference Process:**In the first stage, the network generates a set of candidate regions with the RPN, which are then refined by the detection head in the second stage. This staged approach allows for more detailed and accurate object localization and classification, though at the expense of increased computational complexity compared to single-pass methods.

1. **Training Details**

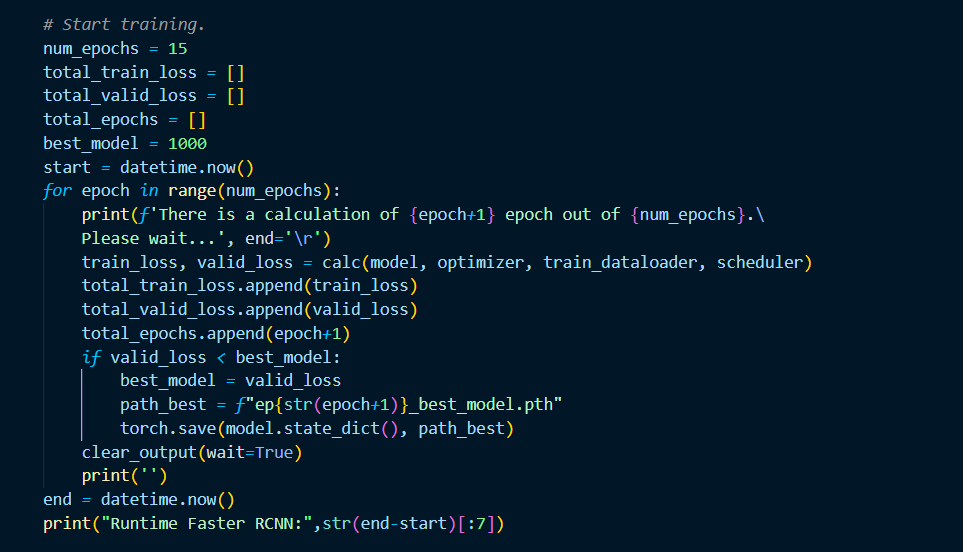
**YOLOv5 Training**

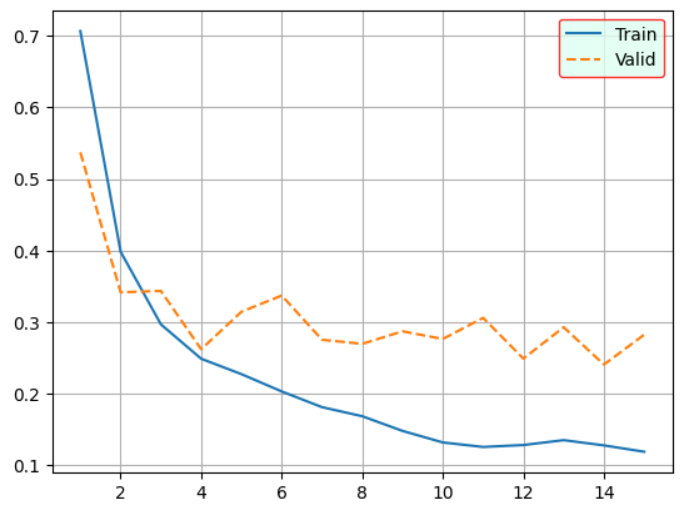
* **Epochs**: 15
* **Image size**: 640
* **Batch size**: 32
* **Pre-trained weights**: yolov5m.pt
* **Optimizer**: Adaptive Momentum (Adam) or Stochastic Gradient Descent (SGD)
* **Training Command**:



**Faster R-CNN Training**

* **Epochs**: 5
* **Optimizer**: Adam or SGD
* **Learning Rate Scheduler**: StepLR or CosineAnnealingLR
* **Training Code**:





1. **Performance Comparison**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Model** | **mAP@0.5** | **mAP@0.5:0.95** | **Precision** | **Recall** |
| YOLOv5 | 0.2071 | 0.1249 | 0.81033 | 0.15092 |
| Faster R-CNN |  |  |  |  |

**Original Image:**

****

**Faster rcnn Output:**

****

**Yolo Output :**

****

****

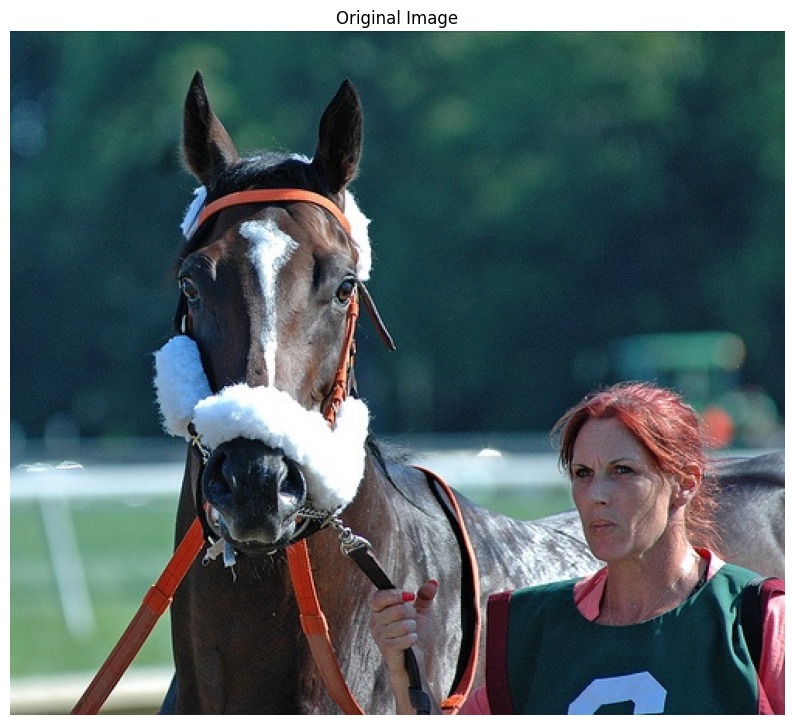
**faster rcnn Output**

****

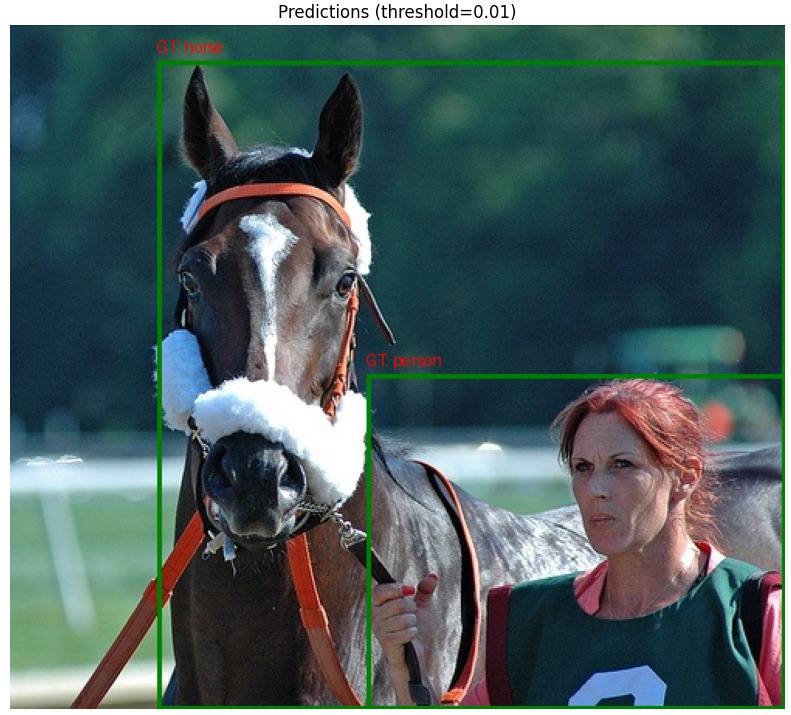
**Yolo Output**

****

**Original Image**

****

**Faster rcnn**

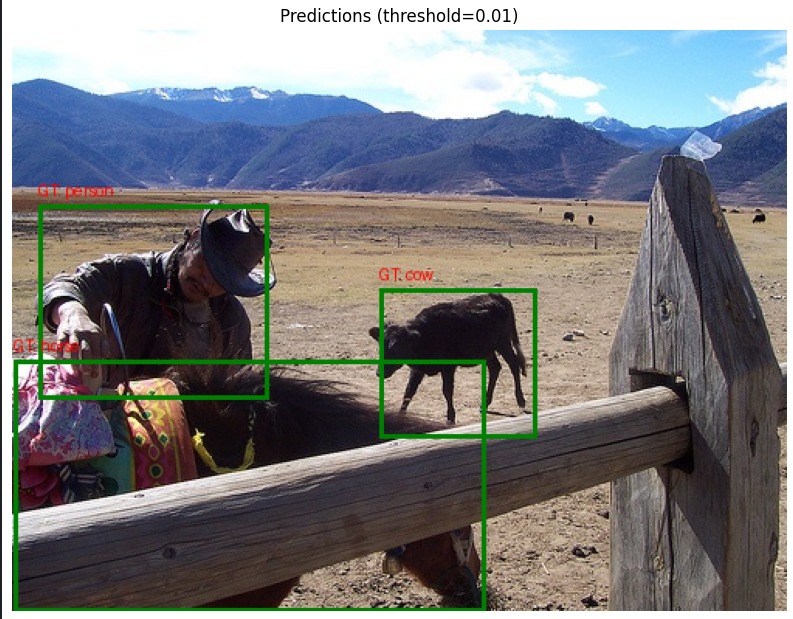


**Output of Yolo**

****

**Original Image**

****

**Output of rcnn  
**

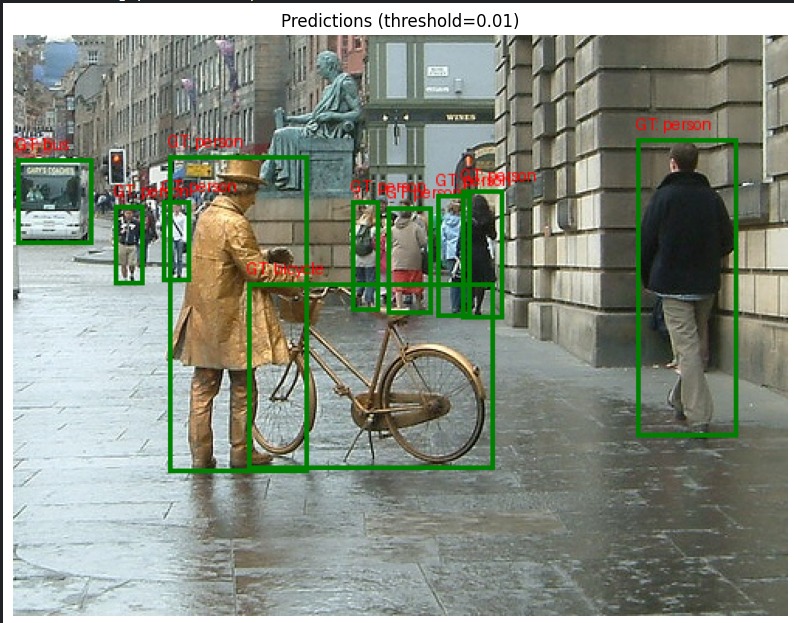
**Output of yolo**

****

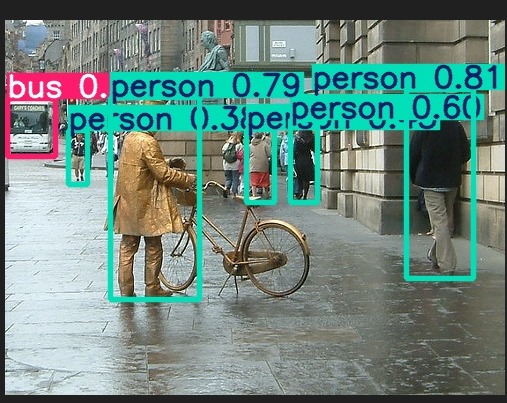
**Original Image**

****

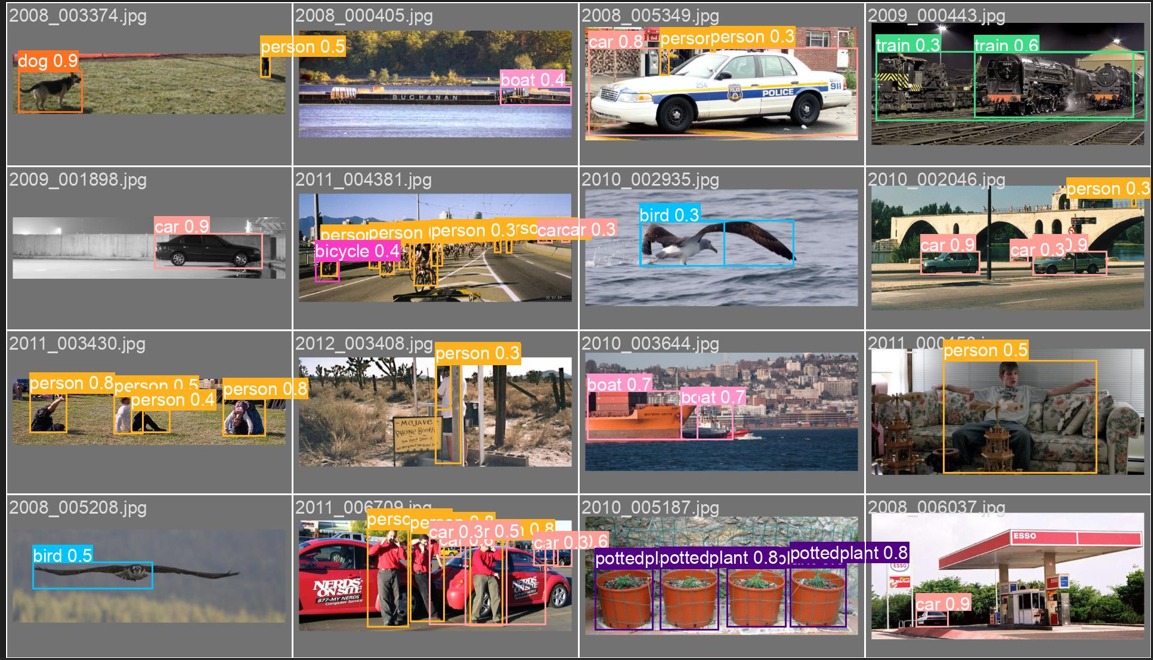
**Faster rcnn**

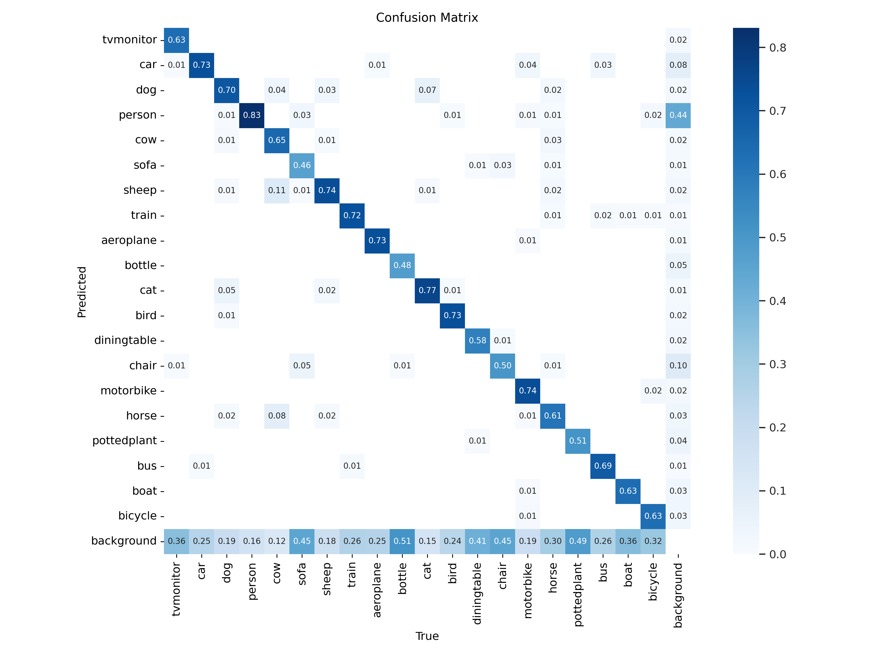
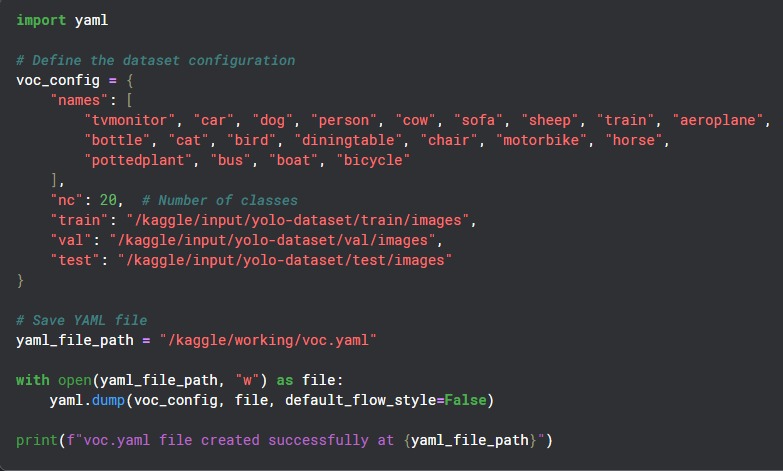
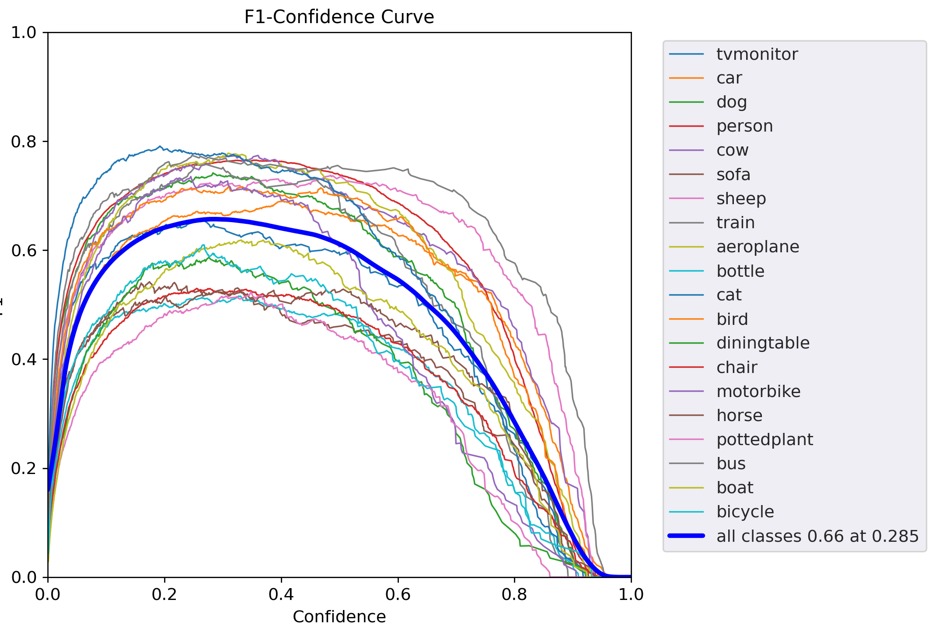
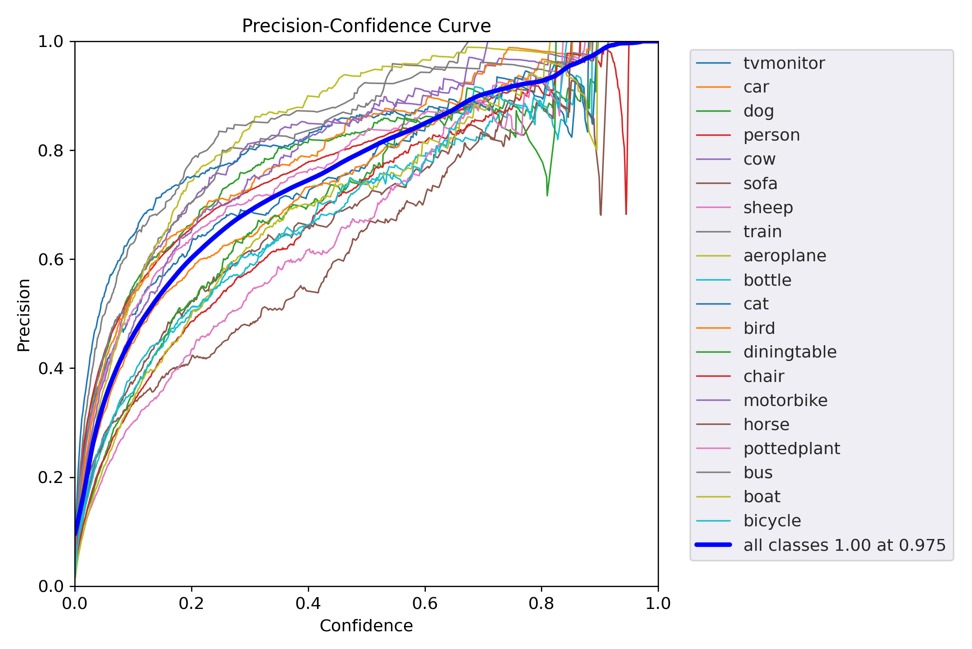
****

**Yolo Output:**

****

**YOLO OUTPUT:**

****

****

****

| **Metric** | **Value** | **Explanation** |
| --- | --- | --- |
| **mAP (metrics/precision)** | **0.67 (at epoch 15)** | **Indicates high overall detection performance on the validation set.** |
| **Inference Time for yolo** | **6.4ms** |  |
| **Inference Time**  **for rcnn** | **59.7ms** | **Yolo is faster than rcnn** |
| **Preprocess time** | **0.4 ms** | **Pre-processing time in object detection refers to the time taken to prepare the input image before feeding it into the deep learning model. It includes resizing, normalization, data augmentation, and other preprocessing steps** |

1. **Practical Considerations**

**When to Use YOLOv5**

* **Real-time applications** (e.g., autonomous driving, real-time surveillance)
* **Fast inference speeds** required
* **Lower computational cost**

**When to Use Faster R-CNN**

* **High-accuracy tasks** where detection quality matters more than speed
* **Medical imaging, aerial photography**, and applications where false positives are costly
* **Handles small objects better** than YOLOv5 due to refined proposals

1. **Conclusion**

This report has compared YOLOv5 and Faster R-CNN using the Pascal VOC 2012 dataset, highlighting key architectural differences and performance trade-offs. YOLOv5, with its single-stage design, is optimized for speed, making it suitable for real-time applications, while Faster R-CNN, using a two-stage approach, typically achieves higher accuracy at the expense of slower inference times. The findings indicate that the choice between these models should be guided by the specific use case and available computational resources. Future work should involve training Faster R-CNN on the same dataset to capture the missing metrics and conduct a more detailed error analysis, which could further refine the comparison and inform potential improvements in both models.