Assignment 07 : YOLO

UTA027 : Artificial Intelligence

Raghav B. Venkataramaiyer

Mar '25

A07 : Object Detection with YOLO 5-May/12-May

1 Introduction

1.1 Background

Object detection is a computer vision technique that identifies and locates objects within images or videos. It goes beyond image classification by drawing bounding boxes around detected objects, enabling machines to "see" and understand their environment. This is crucial for applications like autonomous driving, surveillance, robotics, and medical imaging, where identifying object locations is essential for decision-making and automation.

The YOLO (You Only Look Once) family is a popular series of object detection models known for their speed and efficiency. YOLOv8 is a cutting-edge version that offers state-of-the-art performance.

YOLO models are particularly well-suited for this task because they process the entire image in a single pass through the neural network. This design choice makes them significantly faster than other object detection methods, enabling real-time performance. Despite their speed, YOLO models, especially YOLOv8, achieve high accuracy, making them ideal for applications requiring both speed and precision.

1.2 Learning Objectives

1. Understand the YOLO architecture.
2. Set up a development environment for YOLO.

3. Prepare a dataset for YOLO training.
4. Train a pre-trained YOLO model on a custom dataset.
5. Evaluate the performance of a trained YOLO model.
6. Perform object detection on images/videos using the trained model.

1.3 Dataset

Use the Pascal VOC Dataset which is available off the shelf with both PyTorch and TensorFlow ecosystem.

The PASCAL VOC dataset is a standard dataset for object detection tasks. It defines a specific set of object classes that models are trained to detect. The classes in the PASCAL VOC dataset are:

Person `person`

Animal `bird`, `cat`, `cow`, `dog`, `horse`, `sheep`

Vehicle `aeroplane`, `bicycle`, `boat`, `bus`, `car`, `motorbike`, `train`

Indoor `bottle`, `chair`, `dining table`, `potted plant`, `sofa`, `tv/monitor`

Dataset preparation and annotation are crucial for the PASCAL VOC dataset's effectiveness in training robust object detection models. High-quality annotations, including accurate bounding boxes and class labels, enable models to learn precise object locations and categories. Consistent annotation across the dataset ensures that models generalize well to unseen images.

Proper dataset preparation, such as splitting the data into training, validation, and test sets, is essential for evaluating model performance and preventing over-fitting.

1.4 Software and Libraries

1. Python 3.x
2. PyTorch or TensorFlow (specify your framework of choice).
3. `torchvision` (if using PyTorch) or `tensorflow-datasets` (if using TensorFlow).
4. `ultralytics` (for YOLOv5 or v8).
5. Recommended: Using a GPU for training.

2 Tasks

Starter Code: [IPython Notebook on Gist]

2.1 Task 1: Training the YOLO Model

1. Train the YOLO Model on the VOC Dataset.
2. Pay special attention to training configuration, documented here, or here, and also listed under the title "Ref `model.train` options," in the starter code.
3. Modify/ Update it to achieve better performance.

2.2 Task 2: Model Evaluation

1. Evaluate the model under the following metrics:
   - Precision
   - Recall
   - Intersection over Union (IoU)
   - Mean Average Precision (mAP)

   The candidates are encouraged to use library functions *e.g.* PyTorch Eval Library, instead of rolling off their own implementations.
2. Analyse the results and discuss the model's strengths and weaknesses.
3. Explain the confusion matrix.

2.3 Task 3: Inference and Visualization

1. Capture 10-20 images from "your daily life" to qualitatively assess the performance of your model.
2. Visualise the results.
3. Analyse the results.

2.4 Tips

1. You are encouraged to train the model in multiple runs successively improving upon it by parts, instead of a single pass.

2. Remember to save the model in a persistent storage device, lest your endeavours be in vain.
3. Consider to resume training from an earlier checkpoint. Remember that your runtime might be limited with free cloud compute! *E.g.* on Google Colab you may monitor the estimated available runtime under "View Resources."

3 Submission

- Code files (Python scripts or Jupiter Notebooks)
- A report (in PDF format) that includes:
  - Introduction
  - Methodology (explaining the data preparation, training process, and evaluation methods)
  - Results (including tables and figures showing the evaluation metrics)
  - Discussion (analysing the results, discussing challenges, and suggesting future improvements)
- Trained model weights.

4 Additional Tips for Students

- Provide links to relevant documentation and tutorials that have been of help.
- You are encouraged to use version control (Git).
- Use a consistent coding style.
- Comment the code clearly.