YOLO Object Detection Using Deep Neural Networks

1. Introduction

Object detection is a key problem in computer vision that involves identifying objects present in an image and locating them with bounding boxes. With the rise of deep learning, convolutional neural networks (CNNs) have significantly improved object detection accuracy and speed.

YOLO (You Only Look Once) is a state-of-the-art object detection algorithm that performs detection as a single regression problem, making it extremely fast and suitable for real-time applications. This project focuses on implementing a YOLO-based object detection system using deep neural networks.

---

2. Problem Statement

Traditional object detection techniques are computationally expensive and slow, making them unsuitable for real-time applications. The objective of this project is to design and implement an efficient object detection system that can accurately detect multiple objects in images with high speed using the YOLO algorithm.

---

3. Objectives

The main objectives of this project are:

- To understand the working of YOLO object detection.
- To train a YOLO model on a standard dataset.
- To evaluate the model using suitable performance metrics.
- To deploy the trained model for inference on new images.

---

4. Dataset Description

For this project, the **COCO128 dataset** was used.

- COCO128 is a subset of the COCO dataset.
- It contains **128 images** with annotated objects.
- The dataset includes multiple object categories such as person, car, dog, bottle, etc.
- Labels are provided in YOLO format.

This dataset is suitable for quick training and experimentation.

---

5. YOLO Architecture

YOLO is a single-stage object detector that predicts bounding boxes and class probabilities directly from full images in one forward pass.

Key Features of YOLO:

- Single neural network for detection
- High inference speed
- End-to-end training
- Real-time object detection capability

A pre-trained YOLO model was used and fine-tuned on the COCO128 dataset using transfer learning.

---

6. Methodology

6.1 Environment Setup

The project was implemented using Python with the following libraries:

- PyTorch
- Ultralytics YOLO
- OpenCV
- NumPy

The required dependencies were installed using a `requirements.txt` file.

6.2 Data Preprocessing

- Images were resized to **416 × 416** pixels.
- Pixel values were normalized.
- Data augmentation techniques such as flipping and scaling were applied automatically during training.

6.3 Model Training

- A pre-trained YOLO model was used.
- Transfer learning was applied to adapt the model to the dataset.
- Training was performed for **50 epochs**.
- The optimizer and learning rate were automatically selected.
- GPU acceleration was used for faster training.

During training, loss values such as box loss, classification loss, and distribution focal loss (DFL) were monitored.

6.4 Model Evaluation

The trained model was evaluated using:

- **Mean Average Precision (mAP)**
- **Precision**
- **Recall**

Validation was performed on a separate validation set to measure generalization performance.

6.5 Inference

After training, the best model weights (`best.pt`) were saved automatically.
The trained model was used to perform object detection on new images using an inference script.

---

7. Results

- The YOLO model successfully detected multiple objects in test images.
- Bounding boxes and class labels were accurately predicted.
- The model demonstrated fast inference speed suitable for real-time applications.
- Training and validation losses decreased steadily over epochs.

The trained model file was saved as `best.pt`.

---

8. Deployment

The trained YOLO model was deployed using a Python-based inference pipeline.
An API-based deployment approach using FastAPI was implemented, allowing users to upload images and receive detection results in JSON format.

This deployment approach enables easy integration with web or cloud-based applications.

9. Conclusion

In this project, a YOLO-based object detection system was successfully implemented using deep neural networks. The model was trained using transfer learning on the COCO128 dataset and achieved accurate and fast object detection. YOLO proves to be an efficient solution for real-time object detection tasks.

10. Future Scope

- Training the model on a larger custom dataset.
- Deploying the model on edge devices such as Jetson Nano.
- Optimizing the model using TensorRT or ONNX.
- Extending the system for video-based object detection.

11. References

1. Redmon, J. et al., *You Only Look Once: Unified, Real-Time Object Detection*
2. COCO Dataset
3. PyTorch Documentation
4. Ultralytics YOLO Documentation