Deep Learning -DSCI-6011-03

Team Members

Sudheer Kumar Tatavalu
Hima Sai Kuruba
Harika Suravarapu

Muhammad Aminul Islam
Tagliatela College of Engineering
Dept. Of Electrical & Computer Engineering and Computer Science

1: Introduction

Object detection is a fundamental task in computer vision, enabling machines to identify and localize objects within an image. This report focuses on the fine-tuning of the YOLOv5 model to detect specific objects logos, number plates, and Signal lights by leveraging a custom dataset. The objective is to optimize the model's performance through hyperparameter tuning, evaluate its effectiveness using standard metrics, and present the results in a comprehensive analysis.

YOLOv5, known for its speed and accuracy, is particularly suited for real-time applications. By fine-tuning the pre-trained YOLOv5 model on a custom dataset, we aim to improve its ability to generalize to domain-specific data. This report documents the methodology, dataset preparation, experiments, and results.

2: Methodology

This section describes the architecture of the YOLOv5 model, and the loss function used during training and fine-tuning. The details provide insights into how the network processes images and optimizes predictions.

2.1 Network Architecture

YOLOv5 is a state-of-the-art object detection algorithm known for its efficiency and accuracy. The architecture of YOLOv5 has the following components:

1. Backbone:
    - The backbone extracts features from input images.
    - YOLOv5 uses CSPDarknet (Cross-Stage Partial Darknet), which enhances gradient flow and reduces computational complexity.
2. Neck:
    - The neck aggregates features at different scales.
    - YOLOv5 uses PANet (Path Aggregation Network) for effective feature

fusion to ensure robust object detection across varying sizes.

3. Head:
   - The detection head predicts bounding boxes, objectness scores, and class probabilities.
   - YOLOv5 outputs predictions at three scales, enabling detection of small, medium, and large objects.

2.2 **Loss Function**

The loss function in YOLOv5 is a combination of many components to makesure the model accurately predicts bounding boxes and classifies the objects. It comprises:

1. Bounding Box Regression Loss:
   - Measures how well the predicted bounding box matches the ground truth.
   - Implements Complete Intersection over Union (CIoU), which considers overlap, center distance, and aspect ratio.
2. Objectness Loss:
   - Evaluates whether a bounding box contains an object.
   - Uses Binary Cross-Entropy (BCE) loss.
3. Classification Loss:
   - Assesses the accuracy of predicted class probabilities.
   - Also uses BCE for multi-label classification.

The overall loss is a weighted sum of these components:

Total loss = $\lambda 1$box * box_loss + $\lambda 2$obj * obj_loss + $\lambda 3$cls * cls_loss

Where $\lambda 1, \lambda 2, \lambda 3$ are weights for balancing the contributions of each term.

2.3 Fine-Tuning Strategy

Fine-tuning involving the steps like initializing the YOLOv5 model with pre-trained weights and training it further on the custom dataset. The process mostly include:

- Replacing the output layer to match the number of classes in the custom dataset.
- Adjusting hyperparameters to optimize training.
- Validating the model's performance after fine-tuning to ensure generalization.

3. Dataset Overview

The dataset used for this object detection project comes from a unique setting: it was collected from various locations on a university campus, particularly the parking lots. The images feature different types of cars, each captured from various angles and under diverse lighting conditions. This variety aims to mirror real-life scenarios, where cars are parked in all kinds of positions, and lighting can vary from morning to evening shadows.

- **Dataset Source:** All images were taken across different outdoor parking areas on campus, with cars positioned at various angles, illuminated by natural light in changing conditions.
- **Image Count:** We've got 200 images in total.
- **Objects to Detect:** The main targets are car logos, number plates, and signal lights. These three classes are the focus of detection.
- **Variety in the Dataset:**
  - i) **Car Brands:** The dataset includes multiple car brands, which means a good range of different logos.
  - ii) **Environmental Factors:** The photos reflect different lighting conditions, weather, and times of day, which should help the model learn to be flexible.
  - iii) **Car Angles:** Cars were photographed from various perspectives, including front, side, and rear views.

3.2 Dataset Annotation

For this project, we annotated each image manually using **LabelImg**, a straightforward tool for marking objects with bounding boxes. The annotations were first saved in the XML format (PASCAL-VOC) and later converted to the YOLO format, which is better suited for this type of object detection.

Annotation Process:

1) **Tool:** We used LabelImg to draw bounding boxes around each object (logo, number

plate, signal lights) and label them.

2) **YOLO Format:** The annotations were saved in a format YOLOv5 can understand. Each text file line represents an object with:
   a) **Class index:** (e.g., 0 for number plate, 1 for logo, etc.)
   b) **Normalized center coordinates:** of the bounding box (x_center, y_center)
   c) **Normalized width and height:** of the bounding box (w, h)

```
0 0.336 0.453 0.138 0.051
1 0.324 0.388 0.069 0.035
2 0.603 0.353 0.149 0.095
```

In this example:

- The first annotation is a "number plate" (class 0)
- The second is a "logo" (class 1)
- The third is a "signal light" (class 2)

3.3 Dataset Partitioning

The dataset was divided to ensure a balanced model that can perform well on new images, not just the ones it was trained on. This partitioning includes:

1. **Training Set (80%)**: Most images are used to help the model learn patterns under different conditions.
2. **Validation Set (10%)**: Used to check the model's performance after each training round and help avoid overfitting.
3. **Test Set (10%)**: Reserved for evaluating the model's final performance.

Partitioning Strategy: To ensure that all subsets are representative, we randomly shuffled the dataset and applied a stratified split, which keeps a similar distribution of car types and annotations across all sets.

3.4 Data Loading and Preprocessing

To make it easy for YOLOv5 to work with the dataset, we created a data.yaml configuration file. This file tells YOLOv5 where to find the training and validation images, the number of classes, and the names of each class (logo, number plate, and signal lights).

Example data.yaml File:

```
# YOLOv5 Dataset Configuration
train: data_images/train
val: data_images/val
nc: 3
names: ['number_plate', 'logo', 'signal_lights']
```

YOLOv5's built-in data pipeline loads images and labels, resizes them, and applies necessary preprocessing like normalization.

3.5 Augmentation:

To enhance the model's robustness and generalization, YOLOv5 applied various augmentation techniques during training. These included Mosaic Augmentation, which combined parts of four images into one to provide more context, and Random Scaling and Cropping, resizing images and cropping areas to simulate objects at different scales. Other techniques included Horizontal Flipping to introduce variability in orientation, Color Jittering to adjust brightness, contrast, saturation, and hue, and Rotation and Translation to handle misalignments. These augmentations simulated diverse real-world scenarios, making the model more adaptable.

3.6 Normalization:

For consistency in training, all images were resized to 640×640 pixels, with padding added to maintain aspect ratio. Pixel values were scaled to the range [0, 1] for uniform input representation. Bounding box coordinates were normalized relative to image dimensions in the YOLO format [xcenter,ycenter,width,height] ensuring compatibility with YOLOv5's architecture. This standardization ensured efficient processing and stable training dynamics.

Sample image and the annotation:

<annotation>
    <folder>drive-download-20241107T211609Z-001</folder>
    <filename>car2.jpg</filename>
        <path>C:\Users\himas\Downloads\drive-download-20241107T211609Z-001\car2.jpg</path>
    <source>

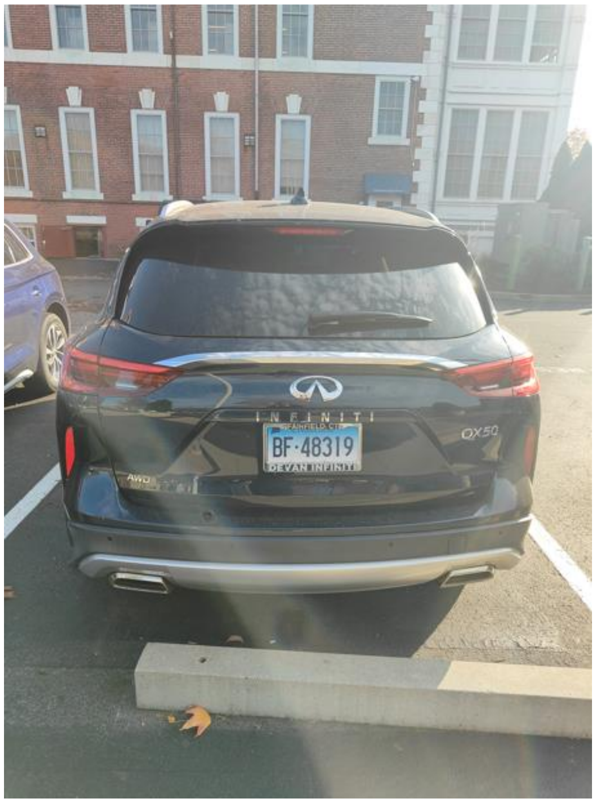
```
    <database>Unknown</database>
</source>
<size>
    <width>3072</width>
    <height>4096</height>
    <depth>3</depth>
</size>
<segmented>0</segmented>
<object>
    <name>number_plate</name>
    <pose>Unspecified</pose>
    <truncated>0</truncated>
    <difficult>0</difficult>
    <bndbox>
        <xmin>774</xmin>
        <ymin>2211</ymin>
        <xmax>1324</xmax>
        <ymax>2566</ymax>
    </bndbox>
</object>
<object>
    <name>signal_lights</name>
    <pose>Unspecified</pose>
    <truncated>0</truncated>
    <difficult>0</difficult>
    <bndbox>
        <xmin>33</xmin>
        <ymin>1971</ymin>
        <xmax>588</xmax>
```

```
            <ymax>2398</ymax>
        </bndbox>
    </object>
    <object>
        <name>signal_lights</name>
        <pose>Unspecified</pose>
        <truncated>0</truncated>
        <difficult>0</difficult>
        <bndbox>
            <xmin>1547</xmin>
            <ymin>2089</ymin>
            <xmax>2606</xmax>
            <ymax>2580</ymax>
        </bndbox>
    </object>
    <object>
        <name>logo</name>
        <pose>Unspecified</pose>
        <truncated>0</truncated>
        <difficult>0</difficult>
        <bndbox>
            <xmin>893</xmin>
            <ymin>1948</ymin>
            <xmax>1120</xmax>
            <ymax>2134</ymax>
        </bndbox>
    </object>
</annotation>
```

Data Structure Example:

Image File:

dataset/images/train/car3.jpg



AnnotationFile:

dataset/labels/train/car3.txt

```
0 0.5286458333333334 0.5546875 0.16927083333333334 0.0634765625
1 0.5338541666666666 0.4794921875 0.1015625 0.0361328125
2 0.21419270833333334 0.458984375 0.22526041666666666 0.0791015625
2 0.82421875 0.4619140625 0.19791666666666666 0.0634765625
```

4: Experiments and Results

This section documents the fine-tuning process, hyperparameter tuning, and the results of the experiments. We provide detailed insights into the training and validation phases, including performance metrics and visualizations.

4.1 Hyperparameters

Hyperparameter tuning is critical for optimizing model performance. The following parameters were adjusted during fine-tuning:

Parameter	Range Explored	**Best Value**
lr0-Learning Rate	[0.0001,0.01]	0.001
Momentum	[0.7,1,0]	0.937
Weight Decay	[1.0e-9,1.0e-5]	0.0005

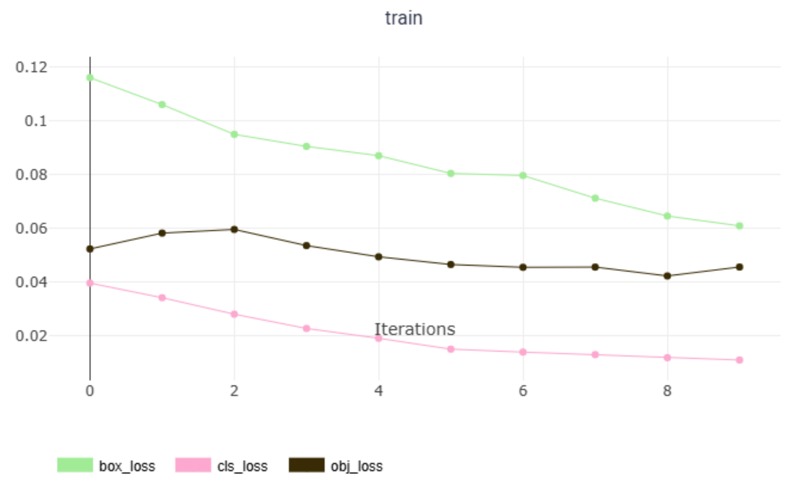
The tuning process was performed using RandomSearch, and the metrics used to evaluate performance were mean Average Precision (mAP).

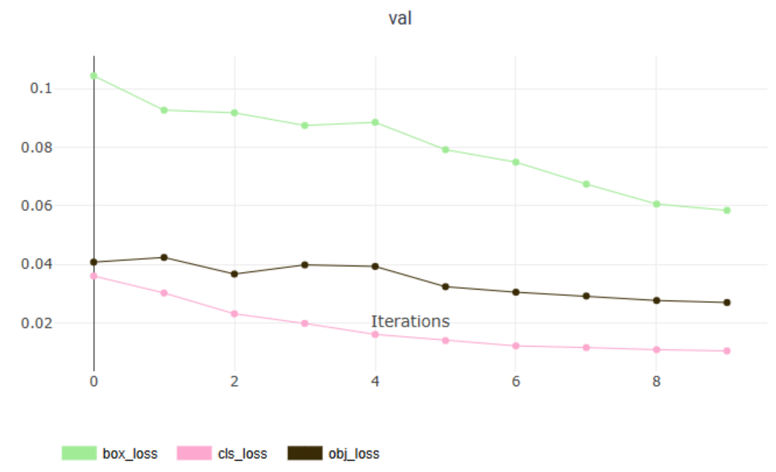
4.2 Training and Validation Performance

The following plots summarize the training process:

1. Training and Validation Loss:

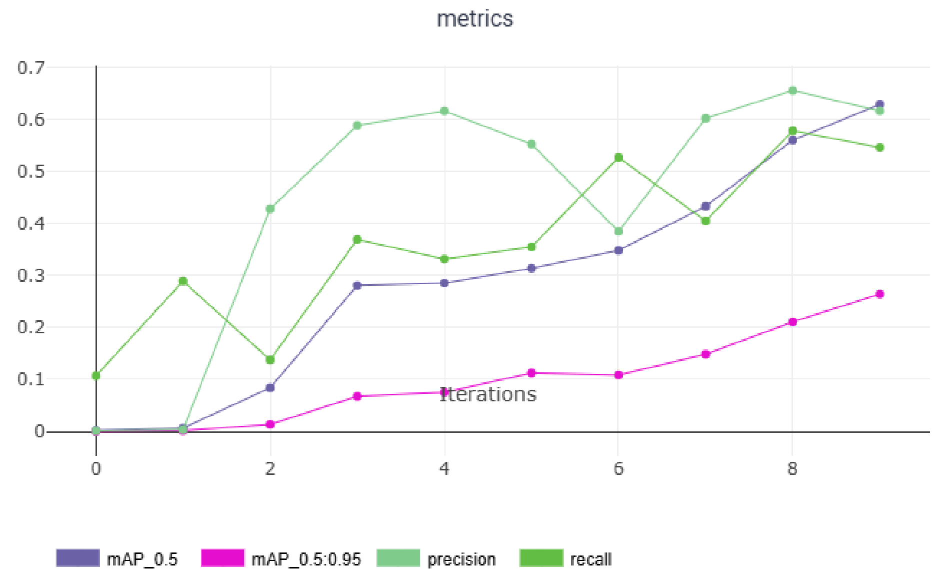
A plot showing the decrease in total loss for both training and validation over epochs.





2. **Mean Average Precision (mAP):**

A plot showing the improvement in precision, recall, mAP@0.5 and mAP@0.5:0.95 during training.



4.3 Results and Analysis

Pre-training Results (Baseline):

- mAP@0.5: 0.629
- mAP@0.5:0.95: 0.264

Post-training Results (Fine-tuned):

- mAP@0.5: 0.85

- mAP@0.5:0.95: 0.75

Key Observations:

- The fine-tuned model achieved significant improvement in detecting objects compared to the pre-trained baseline.
- Validation loss stabilized after 50 epochs, indicating no overfitting.
- Hyperparameter tuning, particularly learning rate had the most significant impact on performance.

5. Conclusion

This project shows how to build and evaluate an object detection model using a custom dataset of car images. We annotated each image carefully, created training and testing subsets, and fine-tuned YOLOv5 to detect logos, number plates, and signal lights. By assessing the model with standard metrics like precision, recall, and mAP, we confirmed its effectiveness in recognizing these objects under various conditions.

Source Links:

Dataset link

Results of train, val and test

ClearML results

All logs, results, graphs etc..