Systems Engineering, Submission 2

OBJECT TRACKING -SUPERVISED COMPUTER VISION

Systems Engineering and Engineering Management
South Westphalia University of Applied Sciences, Soest

Group 20

-

Wesley Nyakeriga
30439461

-

Karanam Sai Teja
30446182

-

Chao-Kuo Yueh
30428218

2024-01-16

Tahasanul Ibrahim, M.Sc.
Prof. Dr.-Ing. Andreas Schwung

Contents

Abstract

Object tracking is becoming more crucial in everyday applications, particularly with the rising interest in AI applications like self-driving vehicles. This second report explores object tracking methods, focusing on R-CNN and YOLO methodologies. Our analysis examines object detection techniques, outlining their strengths and weaknesses. By Using YOLOv5 and ResNet18, we conduct rigorous testing and comparison with the assistance of GitHub repositories. Our evaluation is based on the analysis of key parameters such as training loss, validation loss, and mAP. By analyzing these parameters we shall determine which algorithm excels in this assignment and why.

Additionally, this report utilizes the Pascal-voc-2012 custom dataset for training, testing, and validating our algorithms. This report aims to show the results of the performance of YOLOv5 and ResNet18 in object-tracking applications. The subsequent sections will delve into the methodology, results, and discussion, offering a scientific and accessible exploration of our findings.

Keywords

Object tracking, Computer vision, RCNN, ResNet18, YOLOv5, Comparative analysis, Custom dataset, Pascal-voc-2012 , Algorithms.

1. Methodology

1.1. Methodology and Data Pre-processing Overview

Data Set and Classes

For this project, we used the Pascal-voc-2012 dataset, which had 20 classes. These objects namely were: Aeroplane, Bicycle, Bird, Boat, Bottle, Bus, Car, Cat, Chair, Cow, Dining Table, Dog, Horse, Motorbike, Person, Potted Plant, Sheep, Sofa, Train, TV/Monitor [1].

Data Preparation

The Pascal-voc-2012 dataset used in this report was organized into training, testing, and validation sets by running Python code for loading and sorting data. additionally, We ensured compatibility of the dataset with YOLOv5 and ResNet-18 input requirements for effective testing by installing the required environments, libraries, and dependencies.

Framework Implementation

This assessment was done in YOLOv5 and ResNet-18 frameworks. We accessed them from their respective websites and GitHub repositories. Essentially, YOLOv5 is a version of the YOLO architecture, while ResNet-18 is a deep convolutional neural network design that was developed to overcome the difficulties associated with training very deep neural networks. ResNet's core innovation is the use of residual blocks, which feature skip connections or shortcuts that allow the network to learn residual functions [2].

Model Configuration

The environment used for running our codes was customized so that we could implement the YOLOv5 and ResNet-18 to suit our specific object detection task by installing the required packages and dependencies.

Prerequisites

To ensure proper running of the code used, ensured PyTorch and related dependencies were installed as prerequisites for both YOLOv5 and ResNet-18.
ResNet-18 is a deep convolutional neural network design made up of 18 deep layers that was developed to overcome the difficulties associated with training very deep neural networks. ResNet's core innovation is the use of residual blocks, which feature skip connections or shortcuts that allow the network to learn residual functions [2].

Training:

For training purposes, YOLOv5 training scripts were deployed on the prepared dataset with adjusted hyperparameters like learning rate, batch size, and epochs. The ResNet-18 algorithm was also trained in the same way by running it on Python. Further information on this will be found in the code submitted together with this report.

1.2. The Training Process

1.2.1. Training Process for YOLOv5 (You Only Look Once):

We first ensured the environment for running YOLOv5 was set up correctly as shown below.

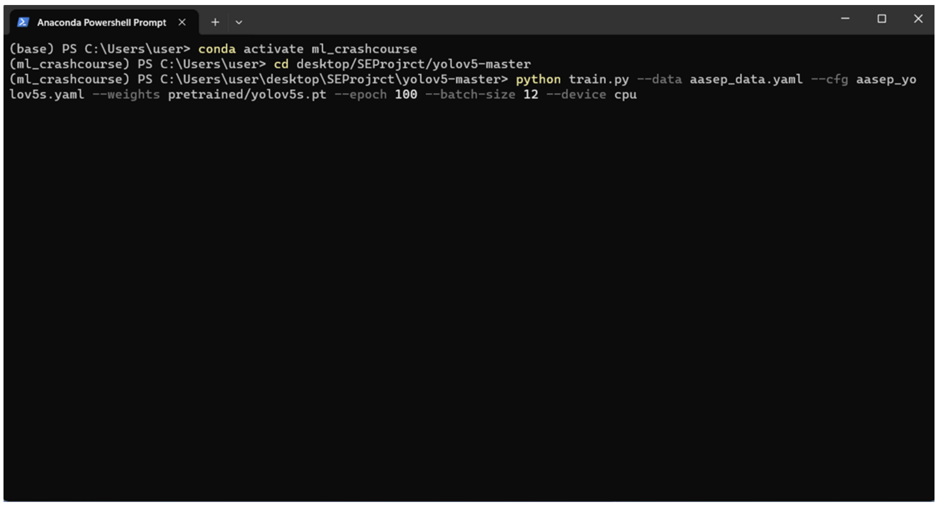
- Activate visual environment:
  - `conda activate ml-crashcourse`
- Install required packages:
  - `conda install -c conda-forge opencv`
  - `pip install -r requirements.txt`
  - `pip install pyqt5`
  - `pip install labelme`
- Change directory:
  - `cd desktop/SEProjrct/yolov5-master`
- Start training

```
python train.py --data aasep_data.yaml --cfg aasep_yolov5s.yaml
--weights pretrained/yolov5s.pt--epoch 100 --batch-size 12 --device cpu\\
```

- Resume Training

```
python train.py --data aasep_data.yaml --cfg aasep_yolov5s.yaml
    --weights runs/train/exp/weights/last.pt --epoch 100 --batch-size 12
    --device CPU
```

Figure 1.1 below shows the initializing steps and environment setup.



```
(base) PS C:\Users\user> conda activate ml_crashcourse
(ml_crashcourse) PS C:\Users\user> cd desktop/SEProjrct/yolov5-master
(ml_crashcourse) PS C:\Users\user\desktop\SEProjrct\yolov5-master> python train.py --data aasep_data.yaml --cfg aasep_yo
lov5s.yaml --weights pretrained/yolov5s.pt --epoch 100 --batch-size 12 --device cpu
```

Figure 1.1.: YOLO Powershell

1.2.2. Training Process for RCNN (Region-based Convolutional Neural Network)

Like YOLO, we began the process by installing the required packages and associated dependencies.

Steps for Training, Testing, and Validation

- Import the required libraries and associated packages and dependencies
- Data transform
- Data loader for training and validation dataset
- Load pre-trained resnet18 model
- Load loss functions and optimizer
- Training and validation loop
- Calculate the training and validation losses
- Save the results
- Save the model

Steps for Testing

- Import the required libraries
- Data transform
- Data loader for the testing dataset
- Load the saved pre-trained model
- Load the pre-trained Resnet18 model
- Testing loop
- Calculate the overall mAP and class-wise AP
- Save the results

The results obtained from ResNet-18 will be discussed in the next chapter.

2. Results

2.1. Evaluation Metrics

In this report, the evaluation metrics required were:

- Training Loss
- Validation Loss
- Mean Average Precision score

2.1.1. YOLOv5 Results

The following results were obtained from YOLOv5 (Full results can be found in the Excel files submitted with this report).

- Training Loss = 0.01421129
- Validation Loss = 0.0296319
- Mean Average Precision = 0.60509

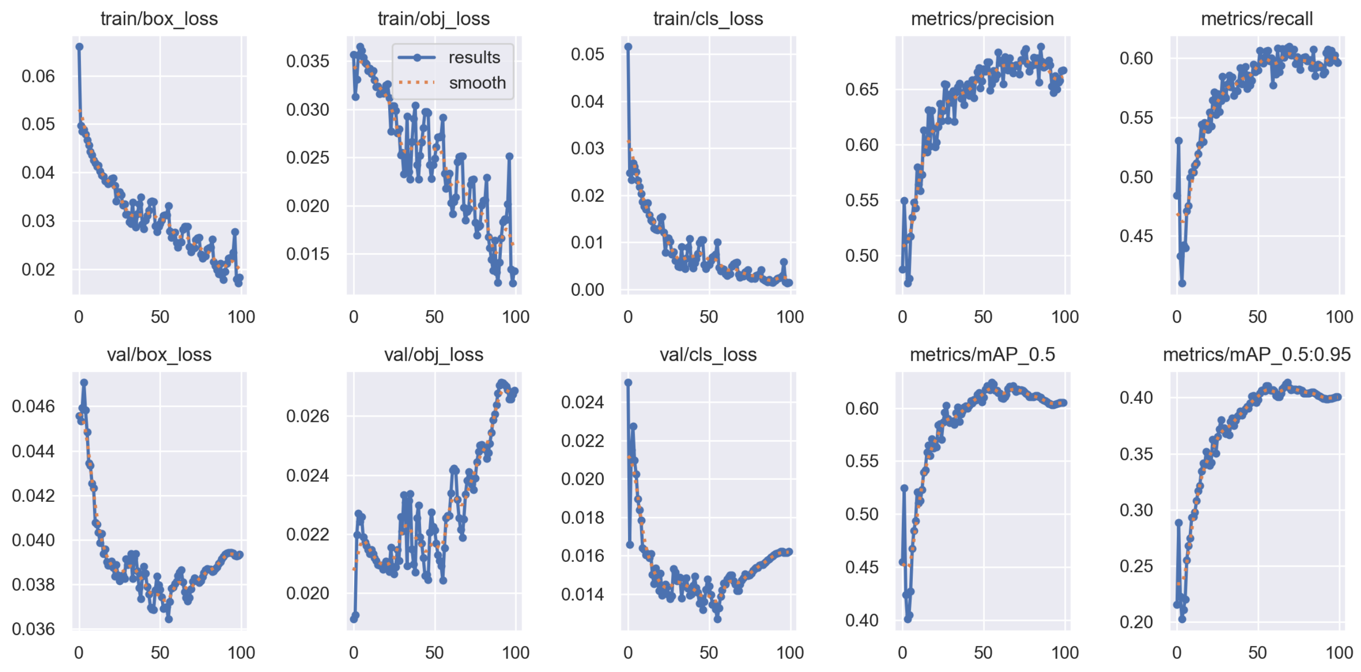


Figure 2.1.: Results

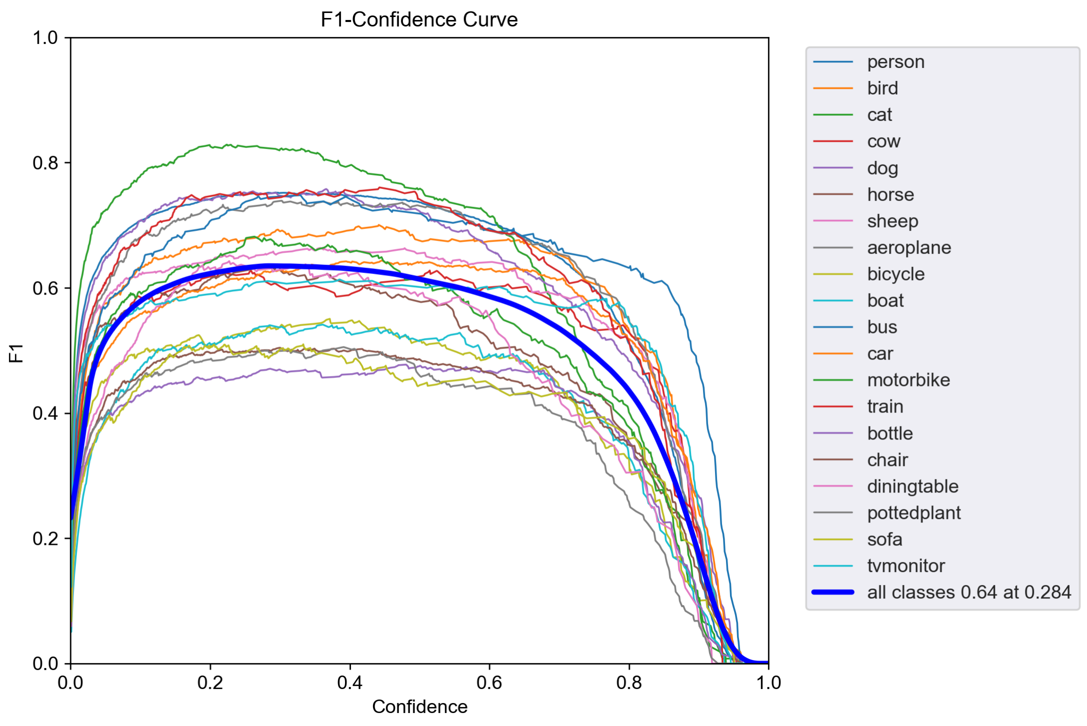


Figure 2.2.: F1 Confidence Curve

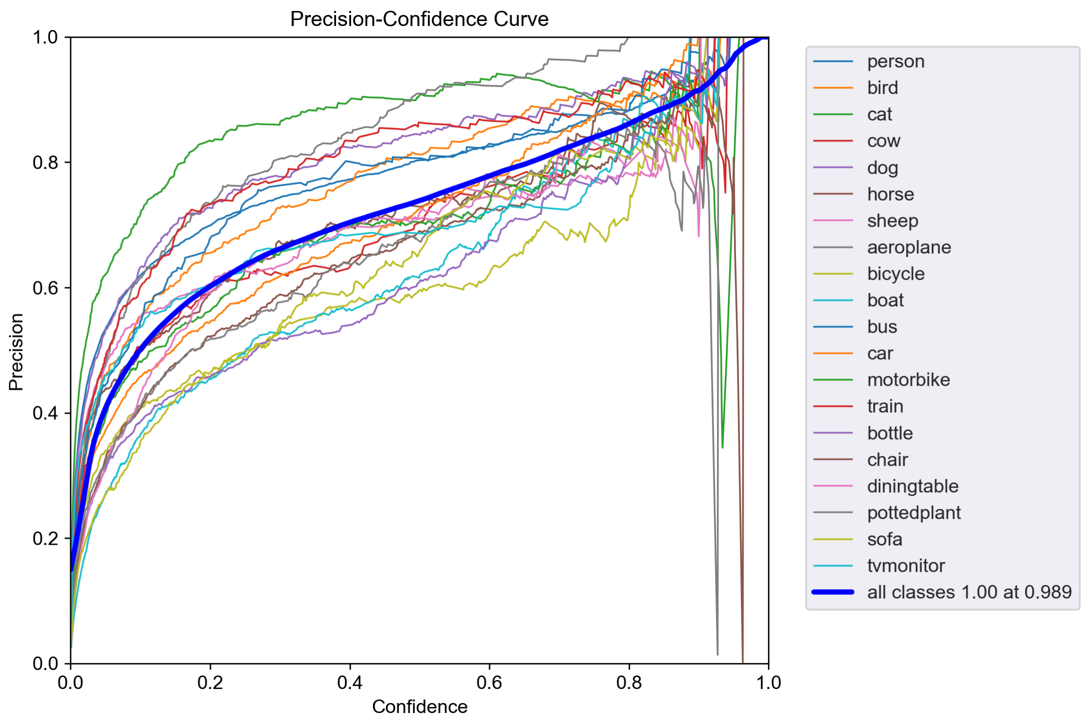


Figure 2.3.: P Curve

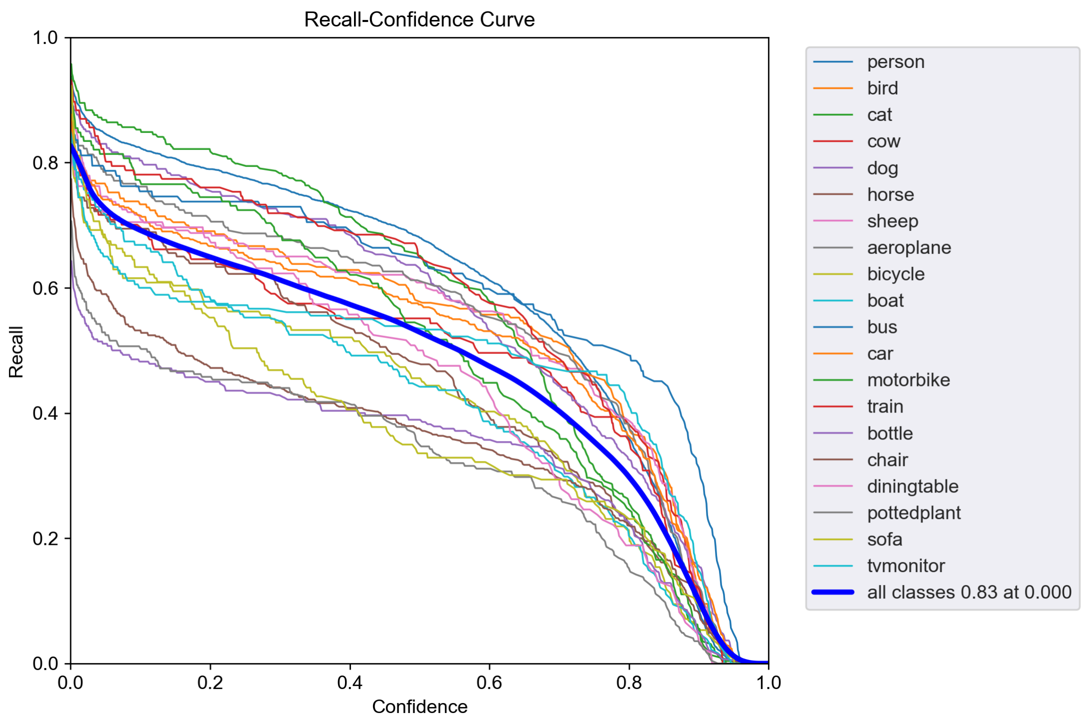


Figure 2.4.: Recall Confidence Curve

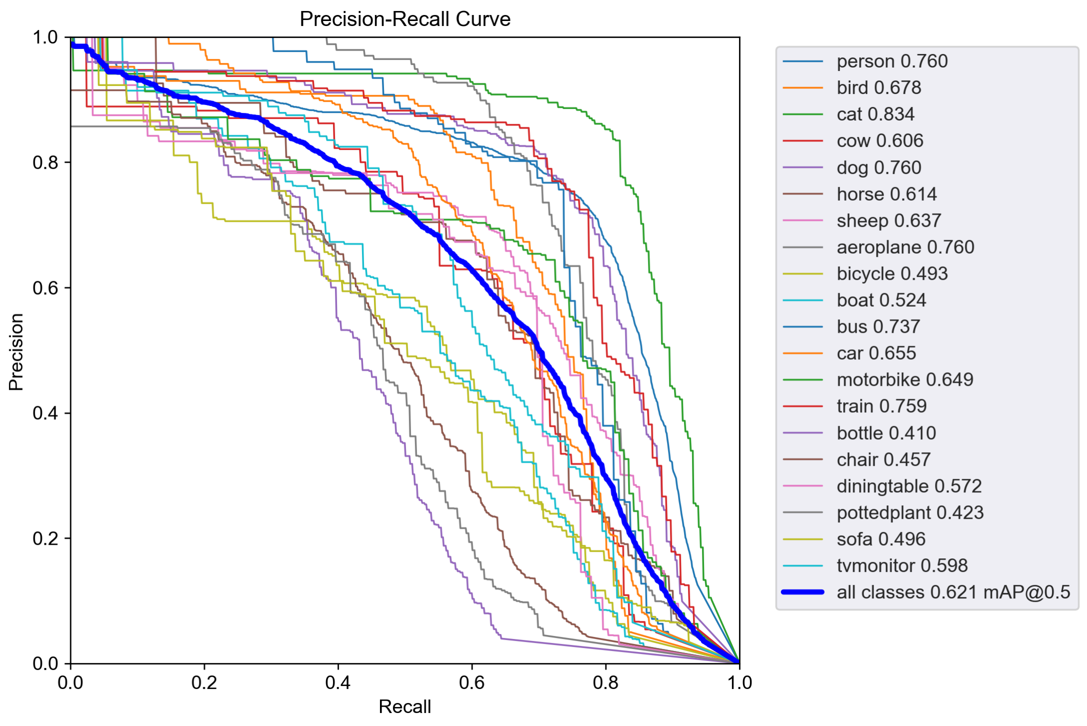


Figure 2.5.: Precision-Recall Curve

2.1.2. ResNet-18 Results

The following results were obtained for RCNN

- Training Loss = 0.776610172
- Validation Loss = 0.80282933
- Mean Average Precision = 0.543209053

Figure 2.6 below shows the training and validation losses.

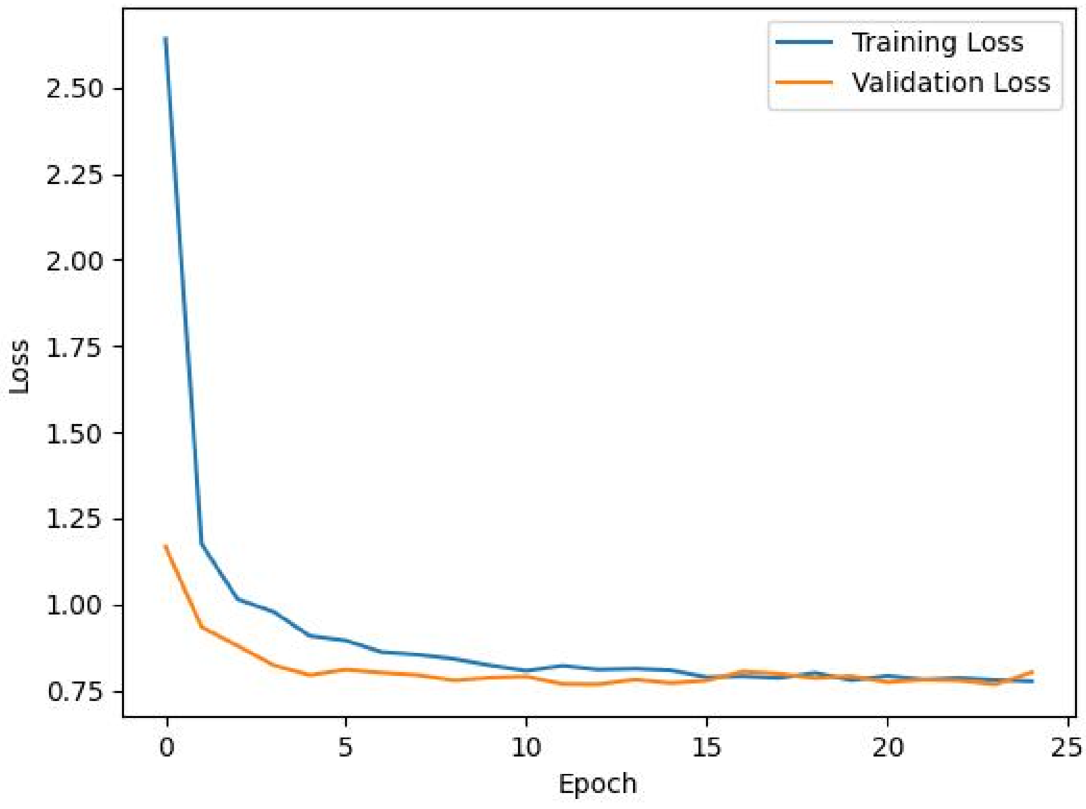


Figure 2.6.: RCNN Loss Curve

3. Discussion

3.1. Key Metrics for YOLO Evaluation

Tables 3.1 and 3.2 below show the environment and configuration that we used to run YOLOv5.

Environment	
CPU	i5-1335U
miniconda3	
python	3.10.13
spyder	5.5.0
opencv	4.8.0

Table 3.1.: Environment Details

Configuration	**YOLO v5**
data	aasep_data.yaml
cfg	aasep_yolov5s.yaml
weights	yolov5s.pt
epoch	100
batch-size	12
device	cpu
workers	4

Table 3.2.: YOLO v5 Configuration

TABLE 3.3 shows all results obtained from YOLOv5

Metric	**Value**
train/box_loss	0.018333
train/obj_loss	0.013219
train/cls_loss	0.0015127
val/box_loss	0.039344
val/obj_loss	0.026854
val/cls_loss	0.016214
mAP_0.5	0.60509
mAP_0.5:0.95	0.40054

Table 3.3.: mAP, Training and Validation Metrics

3.1.1. Mean Average Precision (mAP) Score

The mAP score is used to evaluate the performance of object detection models. It combines precision and recall across multiple levels of confidence, thereby enabling assessment of the ability of the object detection model to accurately identify and locate objects in an image.

Calculation
The calculation of the mAP score is done automatically by YOLOv5. In detail, however, the calculation of mAP involves computing the average precision (AP) for each class and then taking the mean across all classes.

Relevance
The mAP score is relevant in object detection models because it considers the trade-off between precision and recall, offering a more nuanced evaluation than a single-point metric. It is particularly useful in scenarios where certain classes might be rare or of varying importance [3].

3.1.2. Training Loss

The training loss parameter is a measure of how well the model is performing during the training phase. It represents the difference between the predicted output and the true value.

Calculation

Typically, it involves metrics like cross-entropy loss for classification tasks or mean squared error for regression tasks.

Assuming individual losses, The overall loss function is depicted by [4]:

$$\begin{gathered}\text{Classes Loss (BCE Loss)} : L_{\text{classes}} \\ \text{Objectness Loss (BCE Loss)} : L_{\text{objectness}} \\ \text{Location Loss (CIoU Loss)} : L_{\text{location}}\end{gathered}$$

The total loss (L_{total}) is computed as the sum of these losses:

$$L_{\text{total}} = \lambda_1 \cdot L_{\text{classes}} + \lambda_2 \cdot L_{\text{objectness}} + \lambda_3 \cdot L_{\text{location}} \tag{3.1}$$

Therefore, the total training losses obtained were 0.01421129.

Relevance

Observing the training loss is important for the model's parameter optimization during training. A decreasing training loss indicates that the model is learning and adjusting to the training data.

3.1.3. Validation Loss

Validation loss measures the model's performance on a separate dataset that is not used during training. It serves as an estimate of the ability of the model to generalize to new, unseen data.

Calculation

Similar to training loss, its calculation follows the same process as seen in the formula [3.1].

The validation loss obtained was: 0.0296319

Relevance

Validation loss is crucial for preventing overfitting. If the training loss decreases but the validation loss increases, this shows that the model is overfitting the data and that it may not generalize well [3,4].

3.2. Key Metrics for RCNN Evaluation

Tables 3.4 and 3.5 below show the results obtained per epoch and class respectively

Table 3.4.: Training and Validation Losses

Epoch	Training Loss	Validation Loss
1	2.6410537	1.167305925
2	1.175052692	0.934351409
3	1.014844534	0.879542831
4	0.977900933	0.823457541
5	0.908448129	0.794713194
6	0.894944499	0.810571384
7	0.861210456	0.801512764
8	0.853793373	0.794818415
9	0.841563194	0.779135199
10	0.822983533	0.787320466
11	0.80721825	0.790467582
12	0.82187084	0.768977606
13	0.810676003	0.767956502
14	0.81338332	0.781883169
15	0.808749002	0.771523156
16	0.788667391	0.778815053
17	0.791217859	0.805197848
18	0.786118104	0.798398387
19	0.800956862	0.785616312
20	0.779542951	0.791867628
21	0.792379712	0.773877502
22	0.782141528	0.781547915
23	0.786460992	0.778731276
24	0.780805218	0.767958236
25	0.776610172	0.80282933

Class	**AP**
aeroplane	0.834613476
bicycle	0.486796298
bird	0.787787751
boat	0.533640871
bottle	0.353034563
bus	0.723909911
car	0.533482374
cat	0.80891843
chair	0.209254197
cow	0.495732455
diningtable	0.242049502
dog	0.673713452
horse	0.379471911
motorbike	0.430153999
person	0.785437875
pottedplant	0.341046042
sheep	0.712177929
sofa	0.293427202
train	0.728000137
tvmonitor	0.511532681
mAP	**0.543209053**

Table 3.5.: Class-wise Average Precision (AP) Values

3.3. Comparative Analysis

We evaluated the performance of the models based on mAP, training loss and validation loss parameters. The goal was to determine which model outperformed the other and why. The results are shown in Table 3.6 below.

Metric	YOLO	RCNN
mAP	0.60509	0.543209053
Training Loss	0.01421129	0.776610172
Validation Loss	0.0296319	0.80282933

Table 3.6.: Comparison of Metrics between YOLO and RCNN

mAP (Mean Average Precision):

YOLO has a higher mAP score as compared to RCNN, indicating that it performs better in terms of precision across different classes.

Training Loss:

YOLOv5 also showed significantly lower training loss compared to RCNN, suggesting that YOLO's training process is more efficient and the model is learning the training data better than ResNet-18.

Validation Loss:

Similar to the training loss, YOLO also has a lower validation loss, indicating better generalization performance on unseen data compared to RCNN [5].

Summary:

In summary, in this assignment, YOLO outperforms RCNN in terms of mAP and training/validation loss. Lower training and validation losses generally suggest that the model is learning the underlying patterns in the data more effectively.

3.4. Conclusion

In conclusion, this study investigated the methodologies and performance evaluation of object tracking using YOLO and RCNN , with a focus on the YOLOv5 and ResNet-18 frameworks. This analysis was conducted on the Pascal-voc-2012 custom dataset, encompassing a variety of object classes.

During our analysis, we analyzed three key parameters, namely training loss, validation loss, and the mAP score. in our results, YOLOv5 showcased superior performance as compared to RCNN, as evidenced by a higher mAP and lower training/validation losses. As a result of our findings, the conclusion was that YOLOv5 outperformed ResNet-18 in this task. However, it's important to consider other factors such as the number of epochs trained, batch size, and other hyperparameters involved. These different factors may have an overall impact on the results.

Bibliography

[1] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, “The Pascal Visual Object Classes (VOC) Challenge,” *International Journal of Computer Vision*, vol. 88, no. 2, pp. 303–338, Sep. 2009, doi: https://doi.org/10.1007/s11263-009-0275-4.

[2] ScienceDirect, “Residual Neural Network - an overview — ScienceDirect Topics,” www.sciencedirect.com, Jan. 14, 2024. Available: https://www.sciencedirect.com/topics/computer-science/residual-neural-network.

[3] B. Ghojogh and M. Crowley, “The Theory Behind Overfitting, Cross Validation, Regularization, Bagging, and Boosting: Tutorial,” *arXiv:1905.12787 [cs, stat]*, May 2019. Available: https://arxiv.org/abs/1905.12787.

[4] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, “The Pascal Visual Object Classes (VOC) Challenge,” *International Journal of Computer Vision*, vol. 88, no. 2, pp. 303–338, Sep. 2009, doi: https://doi.org/10.1007/s11263-009-0275-4.

[5] Y. Bai et al., “How Important is the Train-Validation Split in Meta-Learning?,” *proceedings.mlr.press*, Jul. 01, 2021. Available: http://proceedings.mlr.press/v139/bai21a.html.

List of Figures

List of Tables

A. Appendix

A.1. Abbreviations

R-CNN - Region-based Convolutional Neural Network
YOLO - You Only Look Once
SVM -support vector machine