Alexandria university

MLA summer training, 2023

Report On

---

8 class object detection using YOLO algorithm

Mahmoud Mohamed Fathallah

Mohamed Ahmed Elgreatly

Mahmoud Mohamed Ramzy

Project Supervisors:TA.Abdallah Dawood

Abstract

We have a dataset of 8 vehicles object detection (Motorcycle – car – bus - LCV(Light Motor Vehicle) – Truck – Tractor - Multi-Axle), our goal is to use YOLOV1 architecture to detect the 8 vehicles correctly. The dataset name is Vehicle Detection 8 Classes | Object Detection on kaggle. We choose YOLOV1 architecture as our base model because of its simplicity to build it from scratch based on the paper provided in the reference section, we also use loss function provided in the paper and mAP(mean average precision) as the way to measure accuracy of the model. The pretrained models we use are YOLOV8 in week 2 of working and YOLOV5 in week 3 of working. All results will be in the last section Results and comparisons .

Contents

1 Week 1 work **4**
1.1 Data exploration 4
1.2 Baseline model 5
1.3 Loss function 5

2 Week 2 work **6**
2.1 Baseline model training 6
2.2 Mean Average Precision 7
2.3 YOLOV8 pretrained model 7

3 Week 3 work **8**
3.1 Base model Editing 8
3.2 YOLOV5 pretrained model 10

4 Week 4 work **11**

5 Results and comparisons **12**
5.1 Base Model Final Result YOLOV1 12
5.2 Final Result Of YOLOV5 13
5.3 Final Result Of YOLOV8 14

1 Week 1 work

1.1 Data exploration

In first week we started exploring the data and understanding the data set more, there were two folders images and labels which had 8218 images with their labels and a text file that contained the 8 classes. The label files had the bounding boxes represented by five numbers first the class number then the (x,y) coordinates of the center of the box and then the width and height of the box. Then we created a data frame to get more information about our data, the following figure represents the number of occurrences of different classes in the data and we can see that the class car has the most number of occurrences and the class tractor had the smallest number

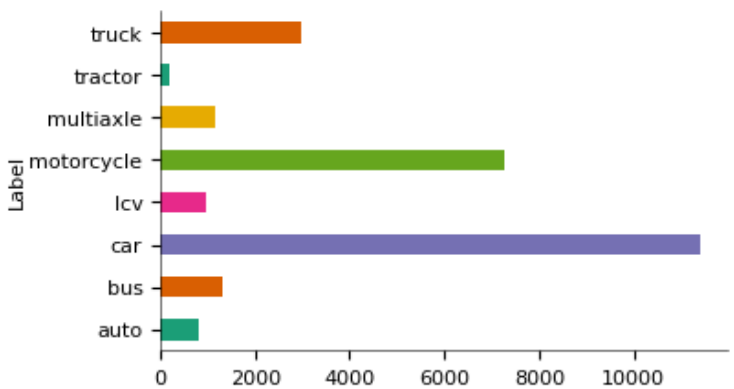

Figure 1: different classes and number of objects

To explore the data even further we visualised the bounding boxes for some images to see random examples and check their validity and correctness and here is an example:

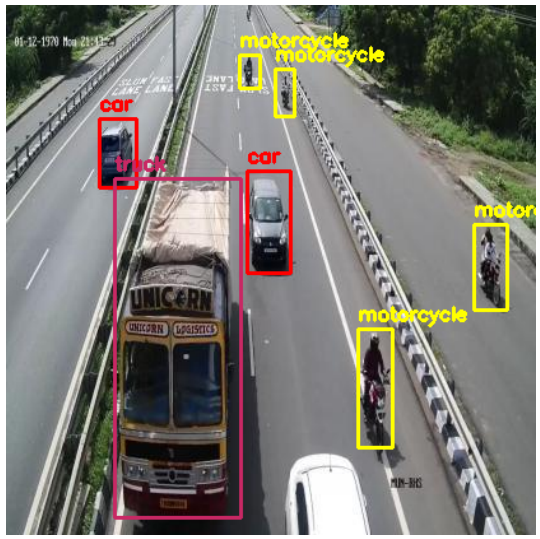

Figure 2: example of bounding box visualization

1.2 Baseline model

After the data exploration we then began working on our YOLO v1 model from the paper cited in the references the model has 24 convolutional layers followed by 2 fully connected layers the only difference in our implementation is that our output layer outputs shape (7, 7, 13) as we have only 8 classes. in our implementation we used tensorflow library as we used it in the deeplearning specialization courses. The following figure shows the architecture of the model.

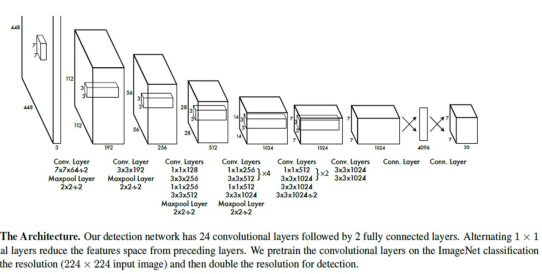

Figure 3: The Architecture. Our detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating 1×1 convolutional layers reduce the features space from preceding layers. We pretrain the convolutional layers on the ImageNet classification task at half the resolution (224×224 input image) and then double the resolution for detection.

Figure 3: the architecture of our model from the paper

1.3 Loss function

Loss function as illustrated in the paper is a sum of 4 losses : coordinate loss, object loss, no object loss, and class loss.
We first get each loss then we added them and multiply with the lambda corresponding to the loss to make it stable, by increasing lambda coordinate and make it 5, and

decreasing lambda no object and make it 0.5.
Problems that Faced us : When we try to get the root of width and height only from the whole tensor we cannot do it directly. First we tried to convert it into np then make our calculations and then return it back into tensor, and this worked well in the test, but unfortunately in the model run it didn't work so we tried another idea. We get a sub-tensor that contains only width and height by"tf.gather" then we take the square root and at last we update the whole tensor by the values of width and height by "tf.tensor_scatter_nd_update"

loss function:

$$\begin{aligned}
&\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right] \\
&+ \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right] \\
&+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2 \\
&+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2 \\
&+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2 \qquad (3)
\end{aligned}$$

Figure 4: the loss function of our model from the paper

After we finished data pre-processing by resizing the images to (448,448) and converting the labels from files to the appropriate input shape to the model which is (7,7,13), we then started training our model, but we faced a problem when we first tried to train our model because the images data didn't fit in memory.

2 Week 2 work

2.1 Baseline model training

In this week we first tried to find a good way to load our data to train the model. We tried using our custom data loader but when we trained the model it took more than 3 hours for one epoch which was a really long time, so we looped over the data after partitioning it into 20 parts which took about 15 minutes per epoch. Then we started training our model for 8 epochs and the curve for the loss during first 8 epochs is in the following figure.

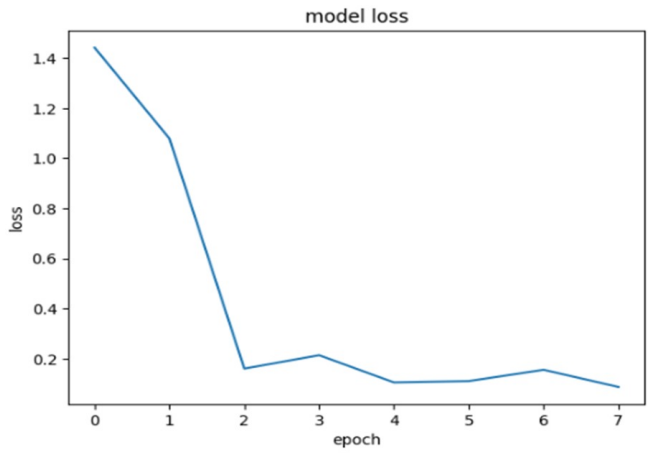

Figure 5: graph of loss for first 8 iterations

2.2 Mean Average Precision

Also in this week we implemented mean average precision metric from scratch as we couldn't find a ready made function to do it, there were some functions but they didn't work with our model, there were some problems with the implementation and the function kept outputting values very close to zero. We also made our non-max suppression function to get our final predictions

2.3 YOLOV8 pretrained model

In this week we tried Ultralytics YOLOv8 model which was pretrained on the coco dataset. We trained the model on our data for about 30 iterations in week 2 but we trained it for 85 iterations in week 4 and here are some examples of the predictions of the model.

Figure 6: some predictions examples

3 Week 3 work

3.1 Base model Editing

In this week we solved some problems related to calculations of MAP metric to run correctly on data. We also run around 75 epoch through dataset.There is an example of 15 iteration during training of model.

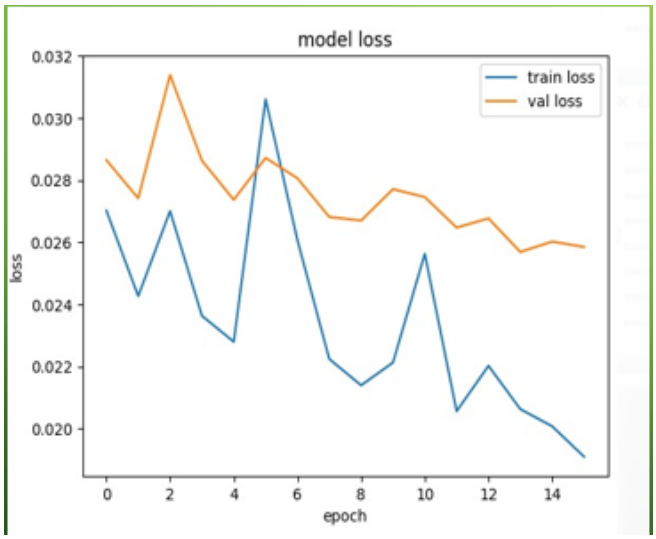

Figure 7: example of 15 iteration during training of model

In non-max suppression there were also another problem in the data entered that we could solve. Finally, print AP for each class and the total mAp and loss for all test data.In the following 2 figures there is an example of predicted images of data by our model and their ground truth images.

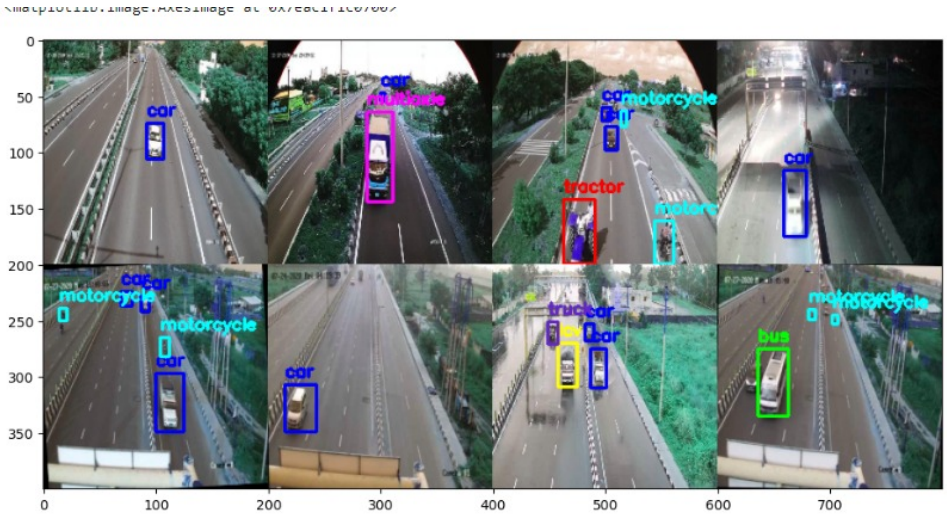

Figure 8: example of ground truth images of data

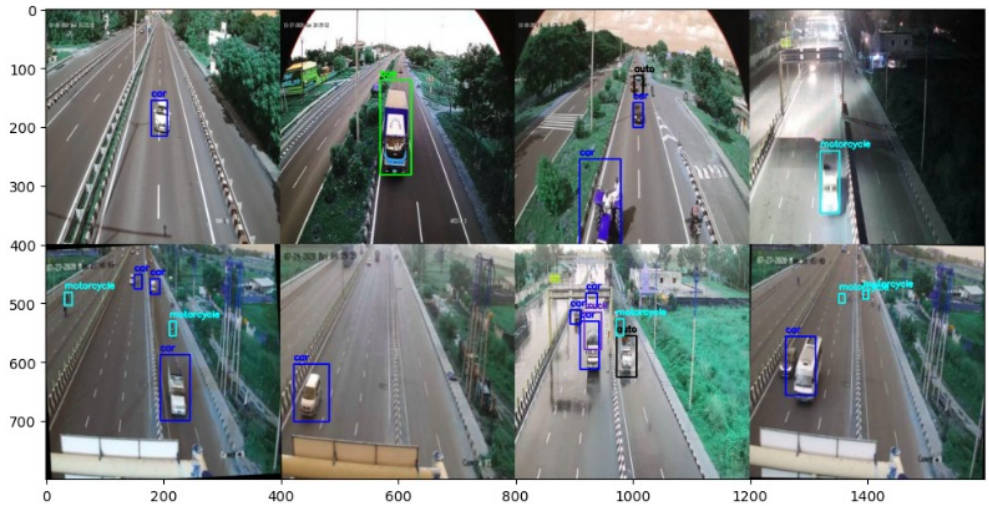

Figure 9: example of predicted images by our model

Final result of this week base model, but there is an issue that has been solved in week 4.

(Label Name)	(AP)	(total MAP)
Auto	10.65 %	—-
Bus	10.66 %	—-
Car	31.06 %	—-
LCV	10.259 %	—-
motorcycle	0.97 %	—-
Multiaxle	25.73 %	—-
Tractor	14.73 %	—-
Truck	22.21 %	—-
Total MAP		16.88 %

Total loss of test data = 0.022 .

3.2 YOLOV5 pretrained model

We made yolo v5 by cloning github repository then inserted our content"classes, images to train on, images for verification" and then train it for 100 iteration on images which

are split 80 - 20 for train - val and here are some sample predictions from validation data.

Figure 10: some sample predictions

4 Week 4 work

In this week we just push data through non-max supression before calculating mAP metric and run more epochs to increase mAP of predicted data but there is an overfit between test and validation loss. We also run YOLOV8 around 85 epoch to make comparison with YOLOV5 easy. And we also try to increase mAP metric for our base model YOLOV1.

5 Results and comparisons

5.1 Base Model Final Result YOLOV1

(Label Name)	(AP)	(total MAP)
Auto	19.7 %	—-
Bus	17.04 %	—-
Car	33.49 %	—-
LCV	20.18 %	—-
motorcycle	16.42 %	—-
Multiaxle	30.88 %	—-
Tractor	42.18 %	—-
Truck	24.88 %	—-
Total MAP		25.60 %

example of AP for each class and total mAP

Total loss of test data = 0.01866 .

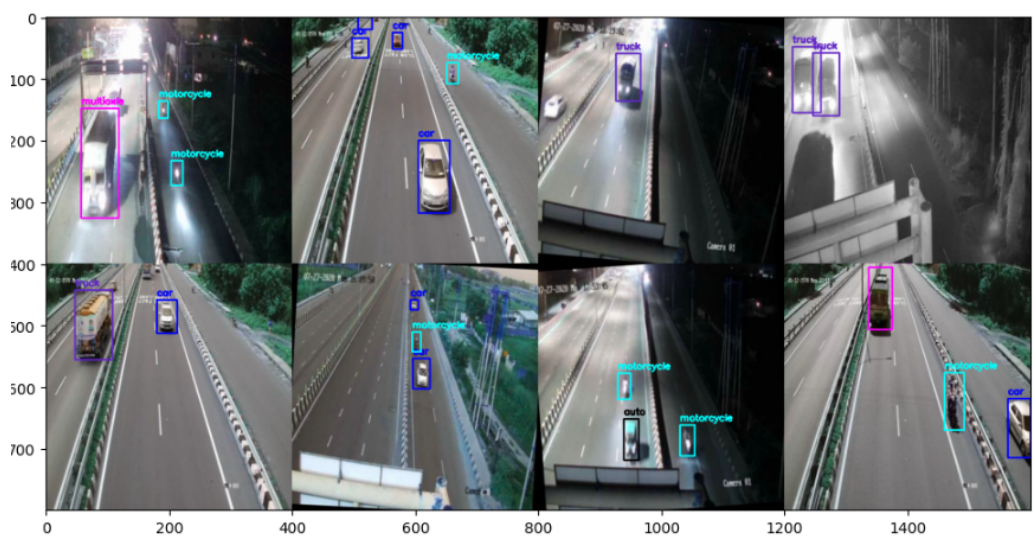

Figure 11: example of ground truth images of data

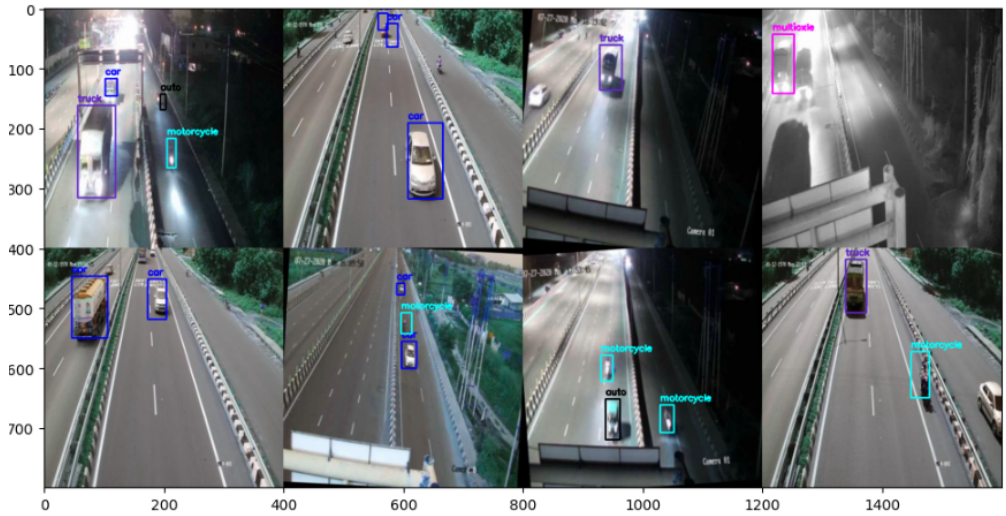

Figure 12: example of predicted images by our model

5.2 Final Result Of YOLOV5

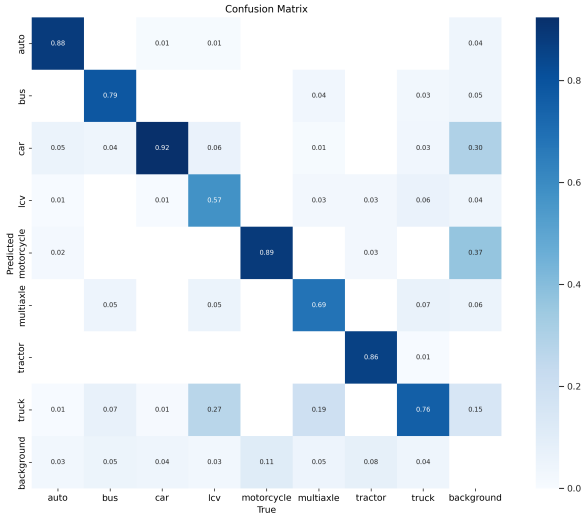

Figure 13: confusion matrix

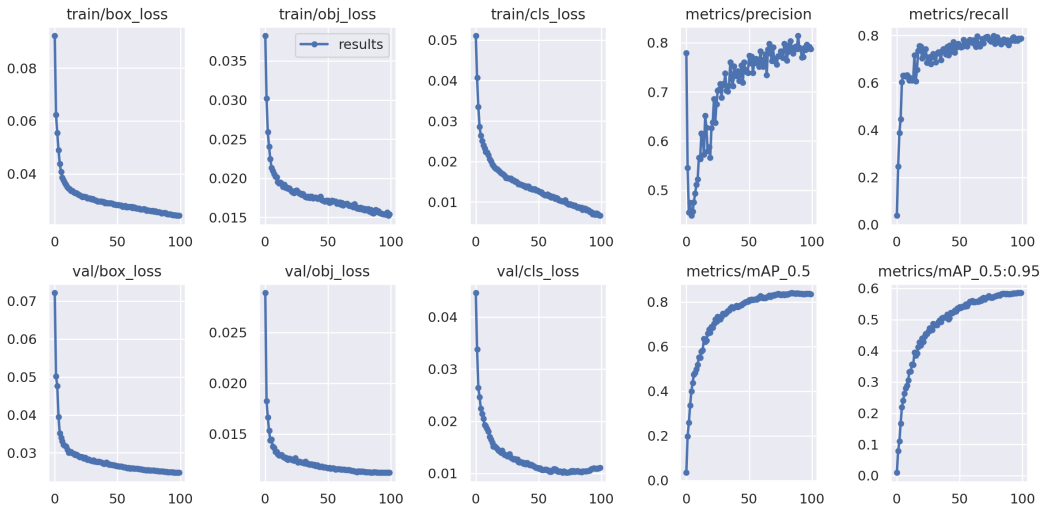

Figure 14: results graphs

5.3 Final Result Of YOLOV8

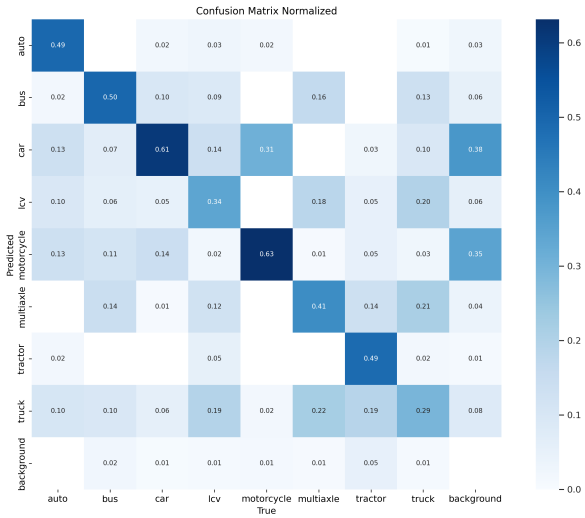

Figure 15: confusion matrix

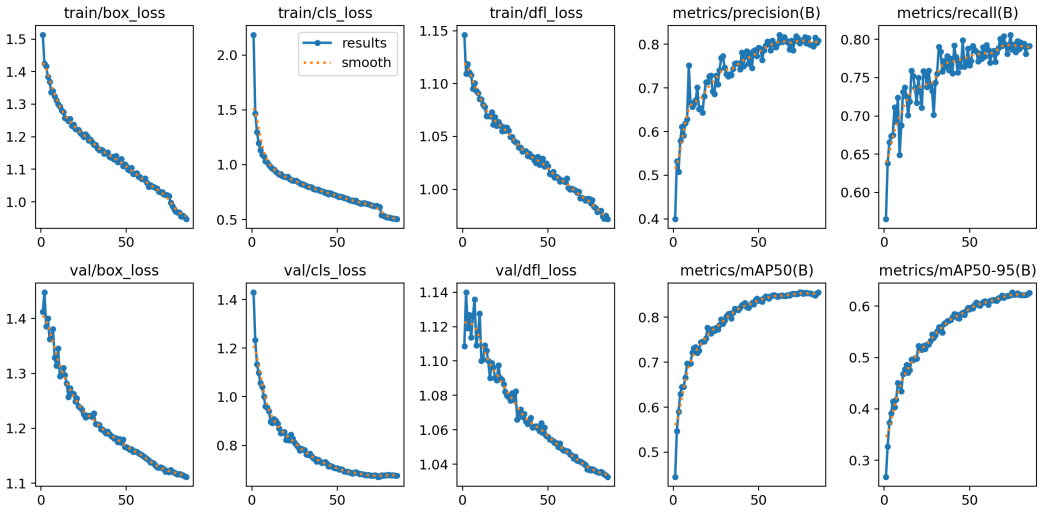

Figure 16: results graphs

References

[1] Paper Name. "You Only Look Once: Unified, Real-Time Object Detection",
Available online: https://arxiv.org/abs/1506.02640

[2] Data Name. "Vehicle Detection 8 Classes Object Detection",
Available online:
https://www.kaggle.com/datasets/sakshamjn/vehicle-detection-8-classes-object-detec

[3] Explanation video. " Explain YOLO V1",
Available online: https://www.youtube.com/watchv=zgbPj4lSc58

[4] Explanation video. "MAP Based on this idea in githup link and youtube link",
Available online: https://www.youtube.com/watchv=FppOzcDvaDI&t=496s

[5] Explanation githup repo. "MAP Based on this idea in githup link ",
Available online:
https://github.com/aladdinpersson/Machine-Learning-Collection/blob/
master/ML/Pytorch/object_detection/YOLO/utils.py

[6] Explanation video. " helpful playlist",
Available online:
https://www.youtube.com/playlistlist=PLlH6o4fAIji76vBRv54WPQr0MHgEiipL7