Yolo Object Detection on PASCAL VOL

I. Introduction:

Yolo is aimed to detect objects of an images with you only look once method. In compared to other object detection models, YOLO has better training efficiency in terms of the speed and the accuracy with the you look at one method.

II. Implementation:

The methodology of YOLO loss function is mainly focusing on penalizing the deviations between the predicted bounding box and the corresponding ground truth box. The loss function can be separated into 4 main loss functions from the formula: .

1. Regression Loss:
   - The loss function compares the predicted bounding boxes and the ground truth boxes of each image in order to improve localization accuracy.
   - The loss function calculates the x, y, w, h of the pred_boxes and target_boxes.
   - size (-1, 4) and size (-1, 4): boxes x, y, w, h
   - MSE loss is used for the loss function.
2. Contain Confidence Loss:
   - The MSE loss penalize the differences between predict bounding box and ground truth box confidence score in order to have the bounding box that match the true presence of objects in the bounding boxes having the significant overlap with the ground truth(IOU).
   - Contain object bounding box confidence scores are set to 1, and no object bounding box is set to 0.
   - box_pred_conf (x, y, w, h) -> (-1, 1)
   - box_target_conf -> (-1, 1)
3. No Object Confidence Loss:
   - The no object loss function focuses on penalizing instances where the model incorrectly predicts the presence of an object in bounding boxes that do not actually contain objects. By doing so, it ensures that the model does not produce high confidence scores for regions where there are no objects. This helps prevent false detections and contributes to an overall improvement in the accuracy of object detection.
   - By multiply the $\lambda noobj = 0.5$, the output confidence score would be lower.
   - No_object_mask mask the predicted boxes with object absence.
   - MSE loss compute the loss of predicted bounding boxes without object and the ground truth confidence of no object cells.
4. Class Prediction Loss
   - The class prediction loss function aims to enhance the precision of class predictions for each bounding box. It achieves this by penalizing deviations in the model's predicted

probabilities across different classes for each bounding box. The goal is to ensure that the model assigns accurate class probabilities to objects within the predicted bounding boxes, contributing to an overall improvement in the model's ability to correctly classify objects.

- The MSE loss helps the model assign high probabilities to the correct class and low probability to incorrect class of each bounding box containing an object.

5. Total Loss:
   - As the formula tells, the total loss is the sum of above loss functions with the penalize $\lambda noobj = 0.5$ and $\lambda coord = 5$.
   - Total loss = class_prediction_loss + λnoobj * no_object_loss + λcoord * regression_loss + contain_object_loss

III. Discussion on Training

First of all, the adjustment of the shape of each loss function parameters need a lot of time to understand why and how those variables should be adjust☺ The implementation was mostly following the formula of yolo loss function, but it certainly needs a comprehensive understanding of the tensor shape and how to use the bounding boxes x, y, w, h, class, and confidence score in order to fit into the formula desired tensor size or scalar.

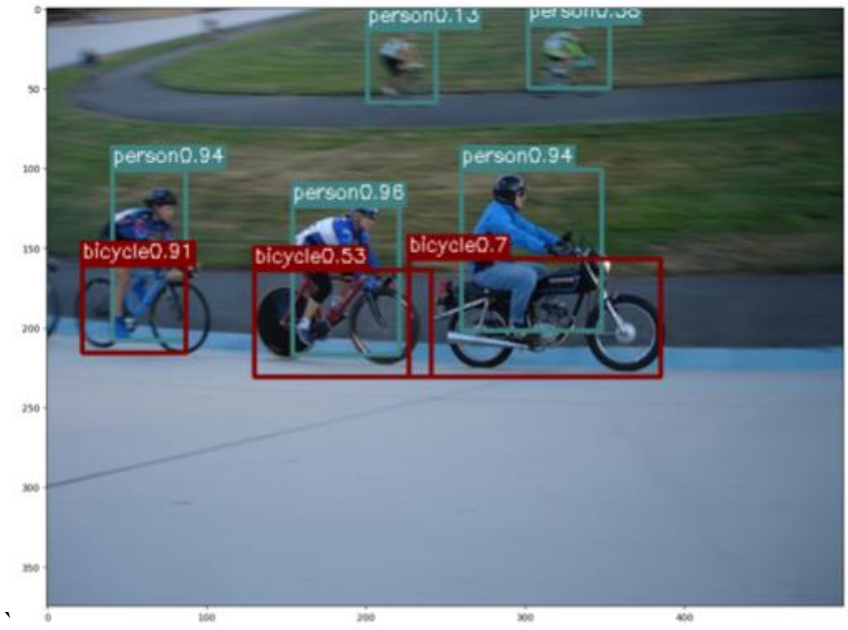
Secondly, my model's mAP per 5 epochs are like below:

5: 0.17
10: 0.34
15: 0.39
20: 0.43
25: 0.47
30: 0.47
35: 0.50
40: 0.51
45: 0.513
50: 0.512

It is quite different compared to the given mAP, the reason might be due to the differences between updated training set. The essay also mentioned the adjustment of learning rate by each epoch, which the train detector functions has also set as the similar proportion from the essay (epoch 1~25, learning rate =1e-3 / epoch 26~40 learning rate=1e-4 / epoch 41 ~ 50 learning rate =1e-5)
There are differences between each class ap, which could indicate that the ability of the model prediction on those classes. Overall, the sum mAP of final shows that the model is under the expectation.

Example predictions:

As the prediction image shows, the AP of clear and non covered object could have a better prediction, while those ambiguous riders, covered bikes has lower performance. In addition, the loss function indeed lower the wrong falsely detection of non object cells.



IV. Results Analysis

The validation mean Average Precision is 0.512, and mAP of 4950 test images is 0.40589 as the Kaggle result shows. The test images seem to have lower accuracy then the validation, which could indicate that the improvement of model generalization is yet to be done.

V. Video On YOLO v1

The video used the cv2 module to separate the 4’20 music video to approximately 5000 images. Deeply look into the bounding boxes that gives the result, the bounding boxes are showing a bit inaccurate in terms of the correct location and the correct class. In the scene of many people dancing in a party, the bounding boxes seems to locate some “non people” places, I think the reason might be that the background is quite dark and the separated images might have too many people moving at the same time leading to inaccurate in localization. Overall, the images or the scene with static objects and person seem to have better bounding box localization and more correct class prediction. A snippet of video:

VI. DenseNet as Pretrain Model

The first try:

The replacement of Resnet50 to Densenet121 gives the mAP of epoch 50 - 0.519, while the highest mAP of epoch 45 was 0.529. The starting points of 5 epochs mAP is 0.31 which is quite better than resnet50. However, the Kaggle result shows 0.38, which is lower than the resnet50 pretrained model performance.

The concept of DenseNet using feed-forward methods to improve the cons of Resnet having too many parameters, DenseNet is said to use 1/3 of parameters with response to the higher efficiency in model training. The process of the training detector use DenseNet121, although it said to use higher RAM when training, the overall efficiency in parameters using seems to be lower than the resnet50 training.

Nonetheless, the test data result seems to be much lower than the Resnet50. The potential reason might be due to the overfitting problem in terms of feed-forward methodology, which lead to the generalization problem. Color space or more data augmentation methods can be used for reducing overfitting problem if possible. Higher training epoch could also be a possible method to improve performance.

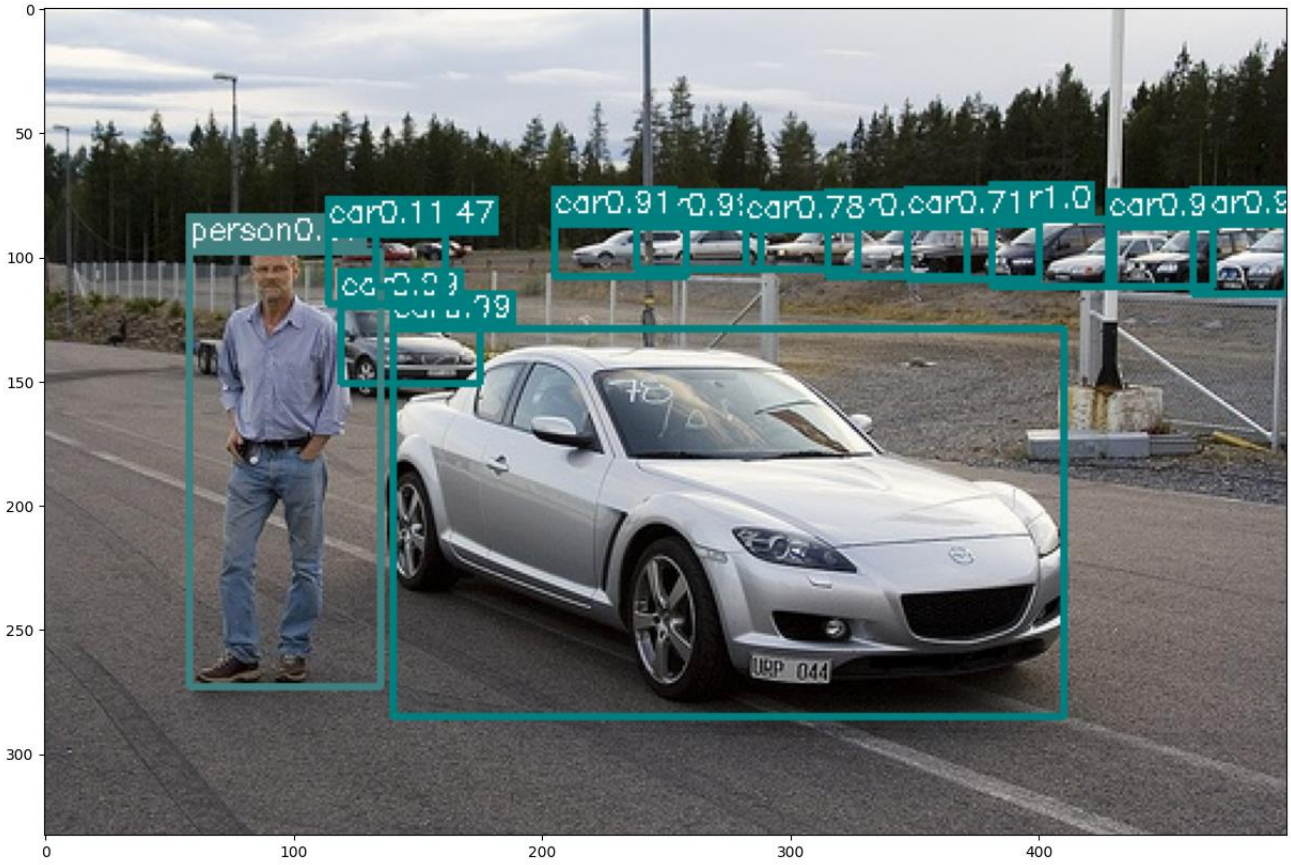
(densnet_yolo.py)

Overall, the DenseNet121 shows the higher efficiency in training rather than higher accuracy.

5: 0.31

10: 0.42

15: 0.46
20: 0.48
25: 0.45
30: 0.43
35: 0.51
40: 0.51
45: 0.52
50: 0.526
Kaggle: 0.38
Below is the predict example image of Yolo v1 using DenseNet121 as pretrained model:



The second try:
I've check the overall performance comparison of Resnet50 and Densent121 and found out that Resnet50 seems to have higher accuracy than densnet121. However, due to the limitation of both Colab RAM (tried to use Densenet161 and Resnet152 but found the RAM not enough problem), I changed to increase the number of epochs and the descent of learning rate (60, 1e-4 -> 75, 1e-5 -> 90, 1e-6). The 4 loss number seems to lower due to the significant change in the learning rate, and did help the improvement of mAP score from 0.49 to 0.54. The Kaggle submission gives the result of 0.4002 with the epoch 80 detector predictions, which is much better than training to epoch 50 with 0.38 by using Densenet121.

I've also tried to train to 100 epochs, however, the performance does not go higher or better. The limitation f the decrease of loss is constrained, even though increasing the epochs and adjust the learning rate gradually. Therefore, I think that I should choose other net with both efficiency and high accuracy model in order to improve the performance.

VII. Conclusion

In conclude, the implementation of yolo loss is a crucial method for the YOLO v1 model. However, the low mAP tells that both the loss function and the pre-trained model could be adjust in several ways for example number of epochs, learning rate adjustment (both don't contribute too much to the better performance), or the choice of pretrained model, or moreover, use the YOLO v2 or ever better YOLO model to have the better performance on object detection.