Deep Learning HW4

曾以諾
Institute of Data Science
National Cheng Kung University
Tainan, R.O.C.
re6111032@gs.ncku.edu.tw

***Abstract*—This report is an description document for HW4. And the GitHub link is as follows**

GitHub Link:
https://github.com/butterfly2012010/DeepLearning_HW4

I. Introduction

Key Task

1) Design a network architecture that can produce both semantic segmentation and object detection results. Please note that the output should not be panoptic or instance segmentation results. If you are unsure of the difference, please consult with the teaching assistant.
2) Your network must be able to learn object detection and semantic segmentation capabilities from two different datasets respectively. Therefore, you will need to modify the dataloader to read and train from these two datasets separately. Be aware that during training for object detection, the weights learned for semantic segmentation might be forgotten. To tackle this, you could consider alternating learning (switching tasks every iteration), or first mastering one task before moving on to the next, using a relatively lower learning rate to avoid catastrophic forgetting, a known issue in deep learning.

Hint

1) To address the first task, you could consider starting with an existing instance or panoptic segmentation network, and removing the instance or panoptic head to meet the requirements. This approach would necessitate writing your own segmentation/detection loss. Alternatively, you could choose any state-of-the-art (SOTA) object detection network and add a new segmentation head (or even a neck) to it. This would only require the addition of a segmentation loss.
2) For training with different datasets, the simplest approach would be to use the same code but with two dataloaders, and alternate between them as needed. Each training session would start by reading the weights from the last checkpoint. Another tip is that once the training has yielded some results, you can freeze the parameters of the earlier layers (or set a very low learning rate) to prevent the model from forgetting previously learned weights.

Incidentally, there is an extra bonus in this homework requirement, and I do the network pruning with keras API.

Model compression: Apply one of the model compression techniques like quantization, pruning, and matrix factorization. Evaluate the performance and speed (run-time) for the compressed and naive models.

II. Implementation

A. Model Architecture

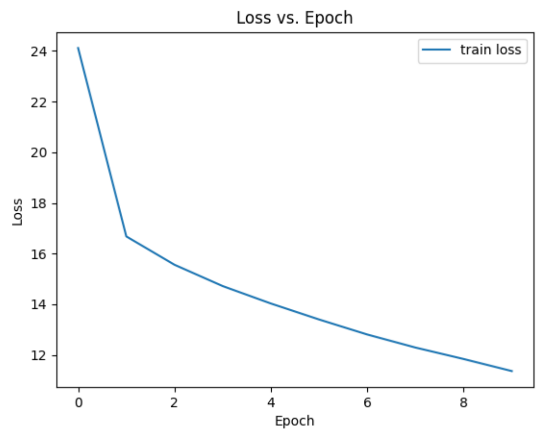
I use YOLOv3 framework as the mainly architecture of object detection task. For the semantic segmentation task, I refer to the slides in Ch.10 and use ChatGPT to help me do segmentation head.

The object detection architecture is in the README.md since the table is too long and I have not been able to put them into IEEE form, so please checkout the README.md on GitHub repository.

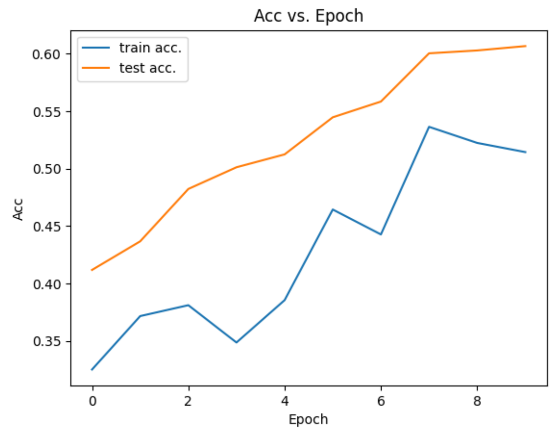
Also, The semantic segmentation architecture is in the README.md.

B. Metric Plot

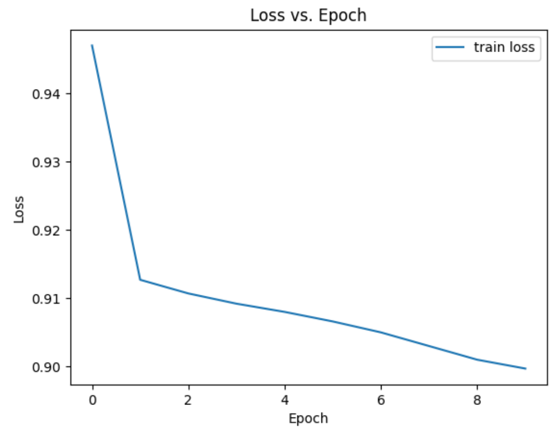
- Loss for Object Detection



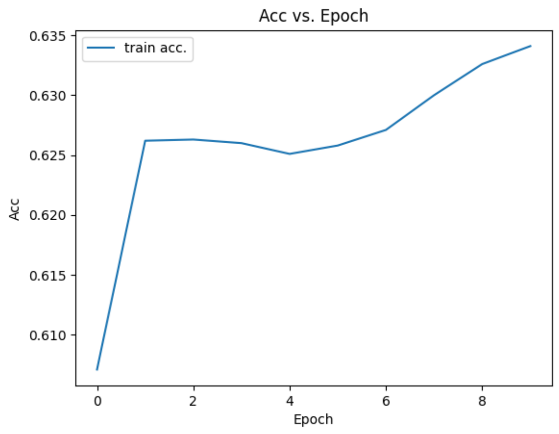
- Accuracy for Object Detection



- Loss for Semantic Segmentatioon



- Accuracy for Semantic Segmentatioon



III. References

References

[1] aladdinpersson-Machine-Learning-Collection
[2] Object__Detection__YOLOV3
[3] ChatGPT