HOMEWORK 1

16824 VISUAL LEARNING AND RECOGNITION (FALL 2023)
https://piazza.com/class/llo9w21ejlp2f

RELEASED: Mon, 18th Sep 2023
DUE: Mon, 2nd Oct 2023
Instructor: Jun-Yan Zhu
TAs: Zhipeng Bao, Wen-Hsuan Chu, Yeojin Jung, Rutika Moharir

START HERE: Instructions

- **Collaboration policy:** All are encouraged to work together BUT you must do your own work (code and write up). If you work with someone, please include their name in your write-up and cite any code that has been discussed. If we find highly identical write-ups or code or lack of proper accreditation of collaborators, we will take action according to strict university policies. See the Academic Integrity Section detailed in the initial lecture for more information.
- **Late Submission Policy:** There are a **total of 5** late days across all homework submissions. Submissions that use additional late days will incur a 10% penalty per late day.
- **Submitting your work:**
  - We will be using Gradescope (https://gradescope.com/) to submit the Problem Sets. Please use the provided template only. You do **not** need any additional packages and using them is **strongly discouraged**. Submissions must be written in LaTeX. All submissions not adhering to the template will not be graded and receive a zero.
  - **Deliverables:** Please submit all the `.py` files. Add all relevant plots and text answers in the boxes provided in this file. To include plots you can simply modify the already provided latex code. Submit the compiled `.pdf` report as well.

NOTE: Partial points will be given for implementing parts of the homework even if you don't get the mentioned accuracy as long as you include partial results in this pdf.

1 PASCAL multi-label classification (20 points)

In this question, we will try to recognize objects in natural images from the PASCAL VOC dataset using a simple CNN.

- **Setup:** Run the command `bash download_dataset.sh` to download the train and test splits. The images will be downloaded in `data/VOCdevkit/VOC2007/JPEGImages` and the corresponding annotations are in `data/VOCdevkit/VOC2007/Annotations`. `voc_dataset.py` contains code for loading the data. Fill in the method `preload_anno` to preload annotations from XML files. Inside `__getitem__` add random augmentations to the image before returning it using [TORCHVISION.TRANSFORMS]. There are lots of options and experimentation is encouraged. Implement a suitable loss function inside `trainer.py` (you can pick one from here). Also, define the correct dimension in `simple_cnn.py`.
- **Question:** The file `train_q1.py` launches the training. Please choose the correct hyperparameters in lines 13-19. You should get a mAP of around 22 within 5 epochs.
- **Deliverables:**
  - The code should log values to a Tensorboard. Compare the `Loss/Train` and `mAP` curves of the model with and without data augmentations in the boxes below. You should include the two curves in a single plot for each metric.

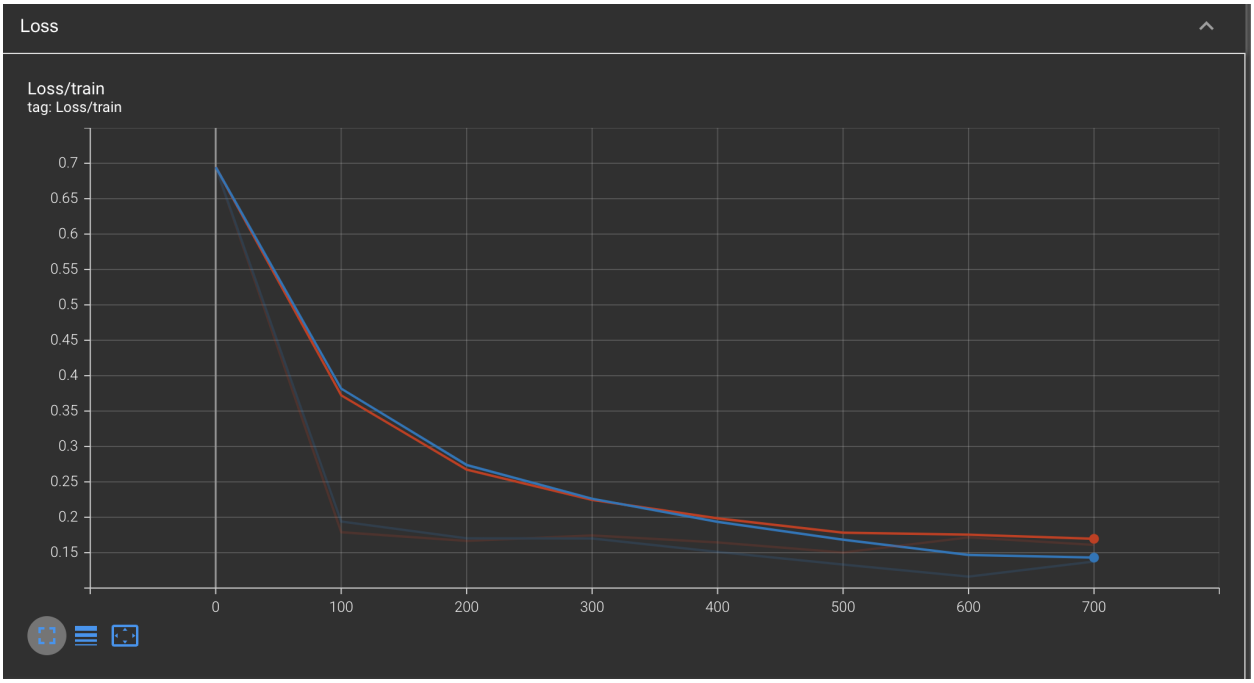

Figure 1.1: `Loss/Train` with and without data augmentations.

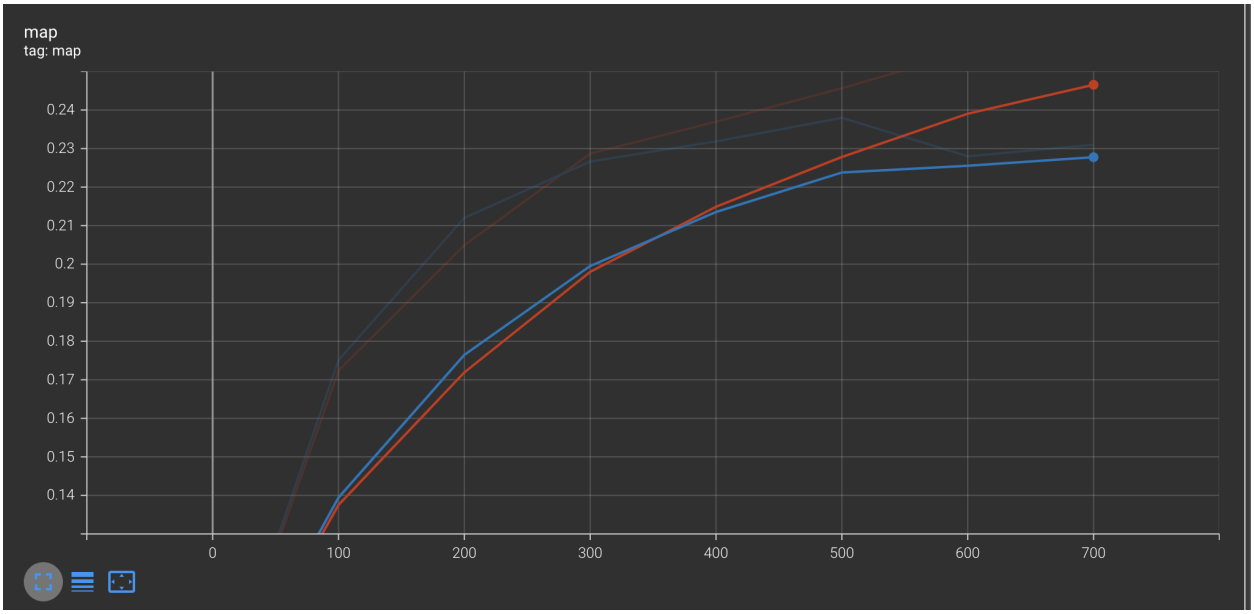

Figure 1.2: `mAP` with and without data augmentations.

– Report the `Loss/Train`, `mAP` and `learning_rate` curves of your best model logged to Tensorboard in the boxes below.

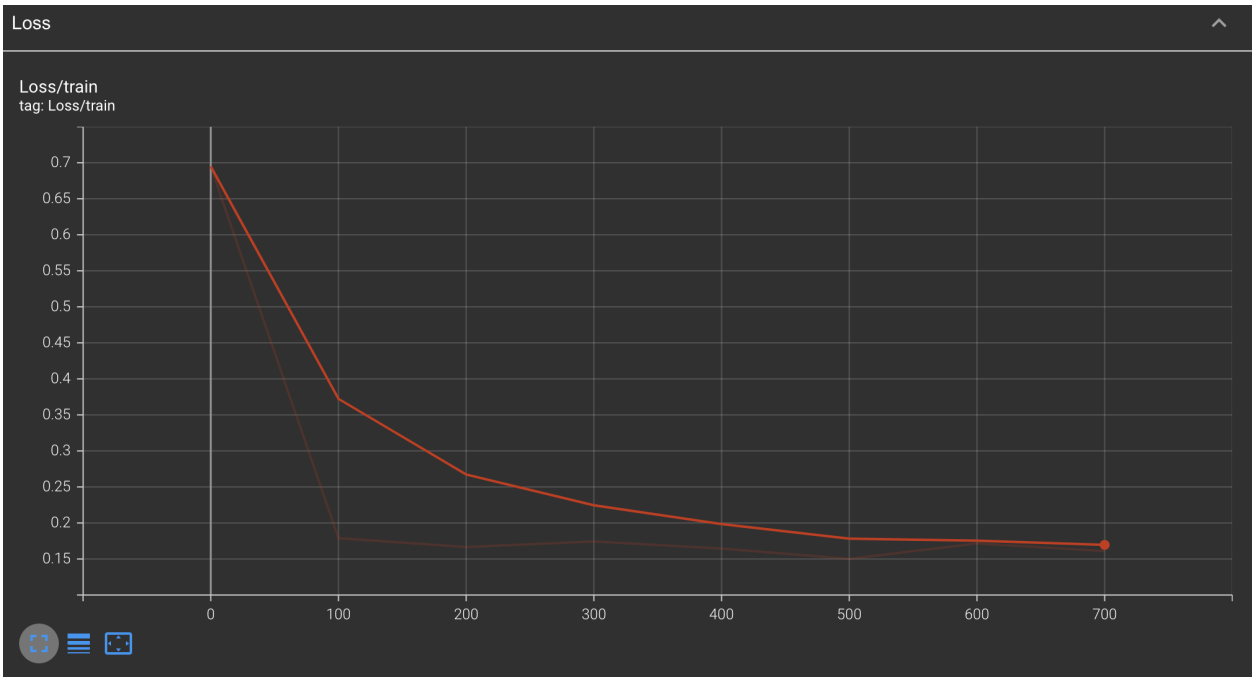

Figure 1.3: `Loss/Train` for simple CNN

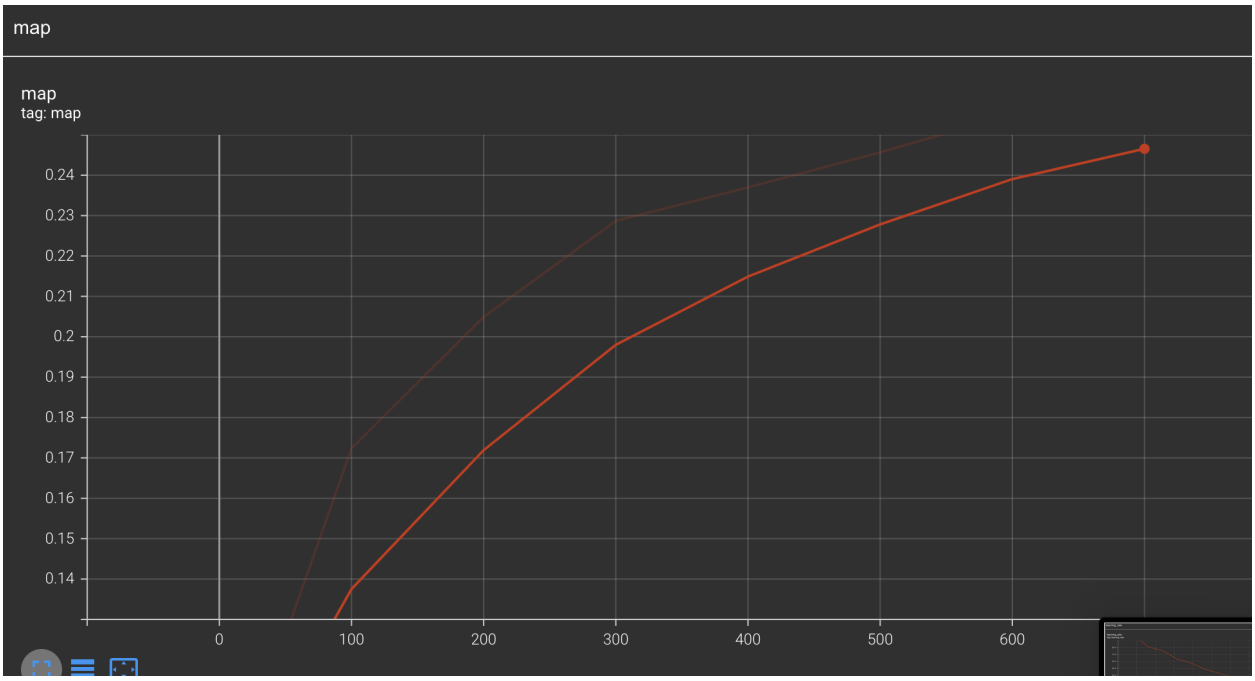

Figure 1.4: `mAP` for simple CNN

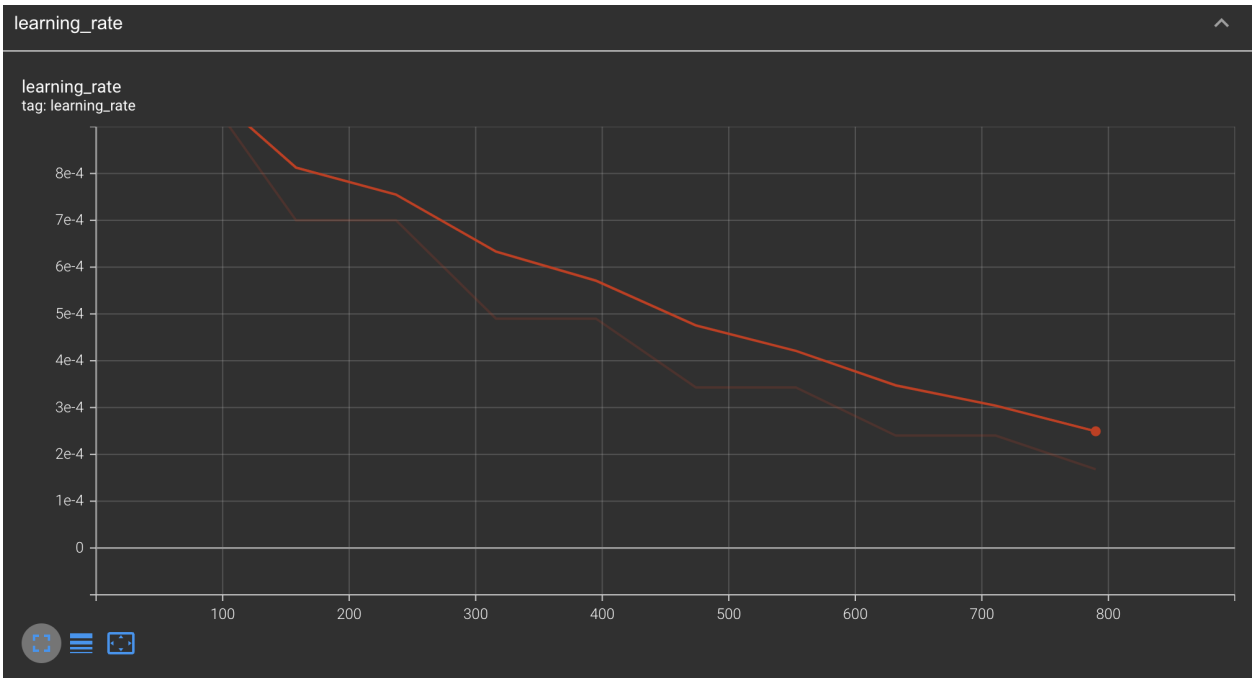

Figure 1.5: `learning_rate` for simple CNN

– Describe the hyperparameters you experimented with and the effects they had on the train and test metrics.

Solution:
My experiments included changing different augmentations, loss functions and changing the learning rate and its scheduler to a suitable value. The red line was the one which was trained with augmentations and the blue line was the one which was trained without augmentations. The orange line shows my best model which was finalized after all my experiments. I noticed that including a lot of augmentations made my training suffer so I ended up using a few augmentations for my model training with augmentations, these included Random Horizontal Flip and not vertical flip since they were giving me results which were not favourable. I also ended up using Gaussian blur and Color Jitter which marginally helped my solution to generalize. As for the learning rate, based on my past experience, I saw that Adam optimizer is very good for low starting learning rates of 1e-4. I used the Step-LR and changed the gamma to 0.8 and step size to 2 since I wanted my learning rate to be modified slightly every 2 steps. As for the losses, I tried with cross entropy loss but due to some errors, I notice MSE loss also worked in this case but I reverted back to Cross Entropy to give me minor improvements in mAP and reducing the loss.

2 Even deeper! Resnet18 for PASCAL classification (20 pts)

Hopefully, we get much better accuracy with the deeper models! Since 2012, much deeper architectures have been proposed. ResNet is one of the popular ones.

- **Setup:** Write a network module for the Resnet-18 architecture (refer to the original paper) inside `train_q2.py`. You can use Resnet-18 available in `torchvision.models` for this section. Use ImageNet pre-trained weights for all layers except the last one.
- **Question:** The file `train_q2.py` launches the training. Tune hyperparameters to get mAP around 0.8 in 50 epochs.
- **Deliverables:** Paste plots for the following in the box below.
  - Include curves of training loss, test MAP, learning rate, and histogram of gradients from Tensorboard for `layer1.1.conv1.weight` and `layer4.0.bn2.bias`.

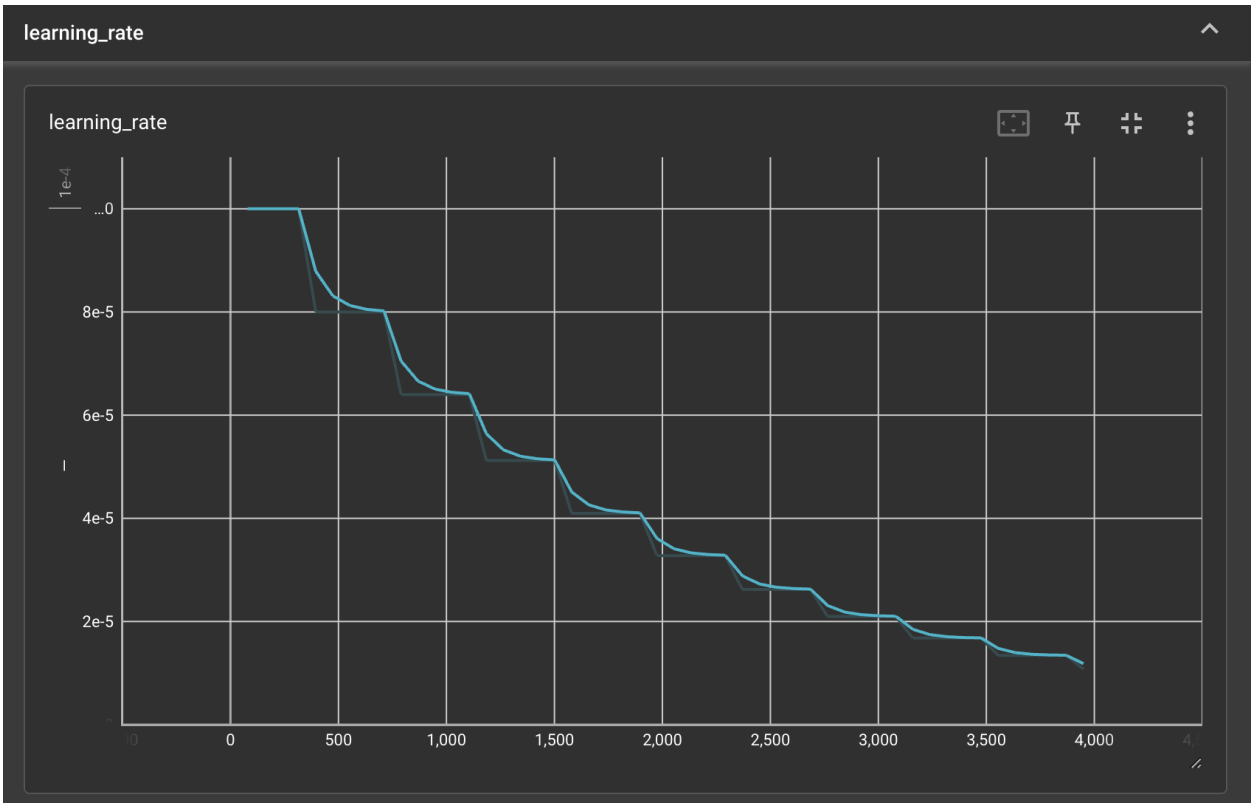

Figure 2.1: `learning_rate` for ResNet

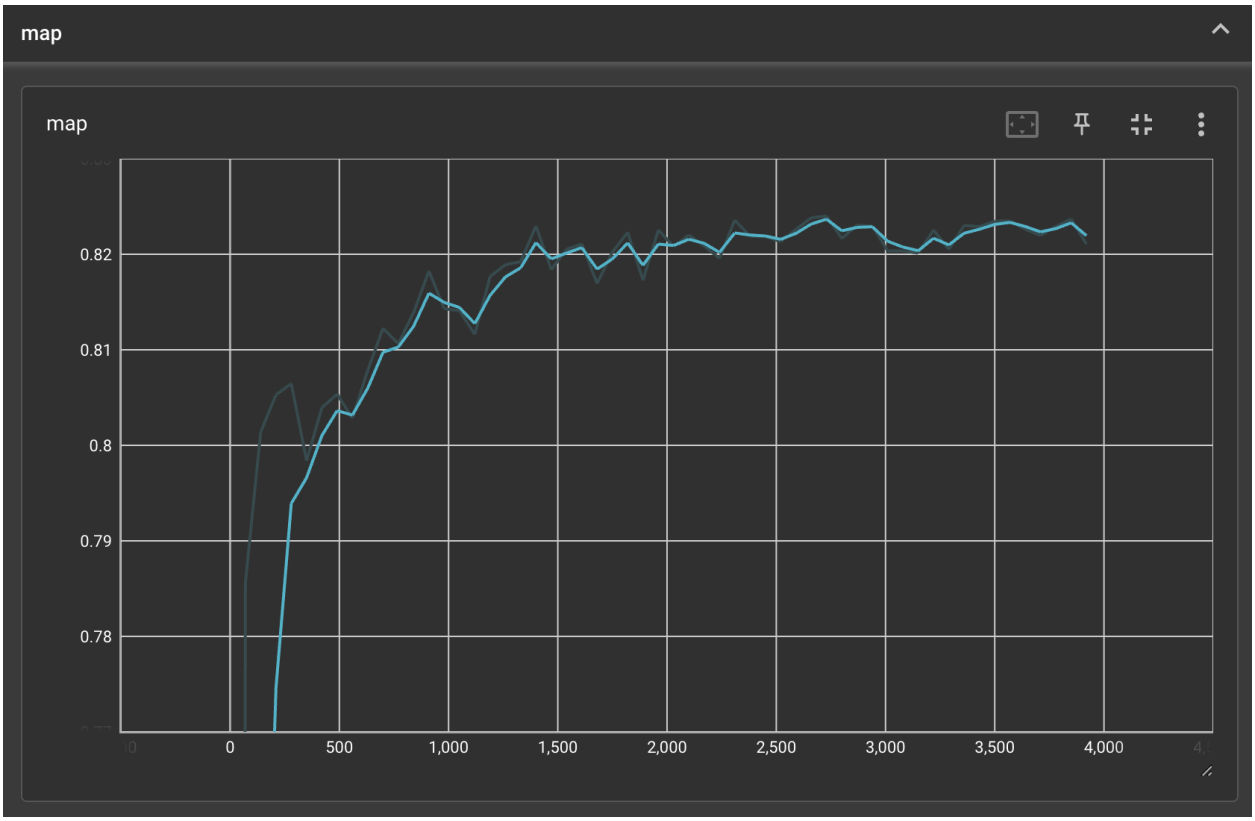

Figure 2.2: mAP for ResNet

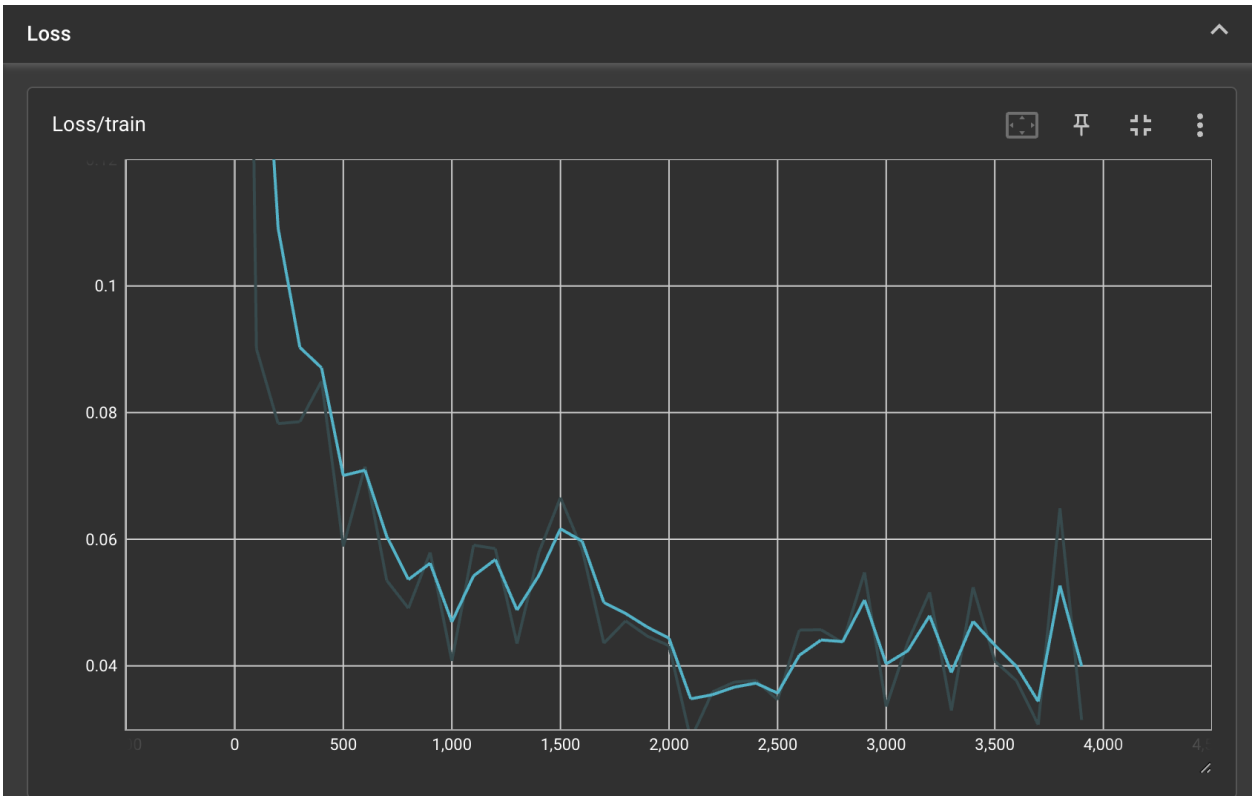

Figure 2.3: `Loss/Train` for ResNet

– How does the test mAP and training loss change over time? Why do you think this is happening?

Solution:
The graphs in light blue are the metrics that I logged for the ResNet network. The test mAP increases over time and likewise the loss/train decreases over time which is a good sign that the network is being fine-trained. I noticed that as the loss decreases, the mAP also increases since they are related to each other and the better our model fits, the mAP should also be better. I did realize that the mAP started to plateau over time which indicated the model was reaching the saturation point of training in 50 epochs. The graphs show an exaggerated change but the values of loss and mAP nearly stayed the same after about 2000 iterations. I feel that there is a chance that we can reduce the losses more and that might lead to better performance but that depends on further experimentation on a test set so as to see if the model is generalizing or not.

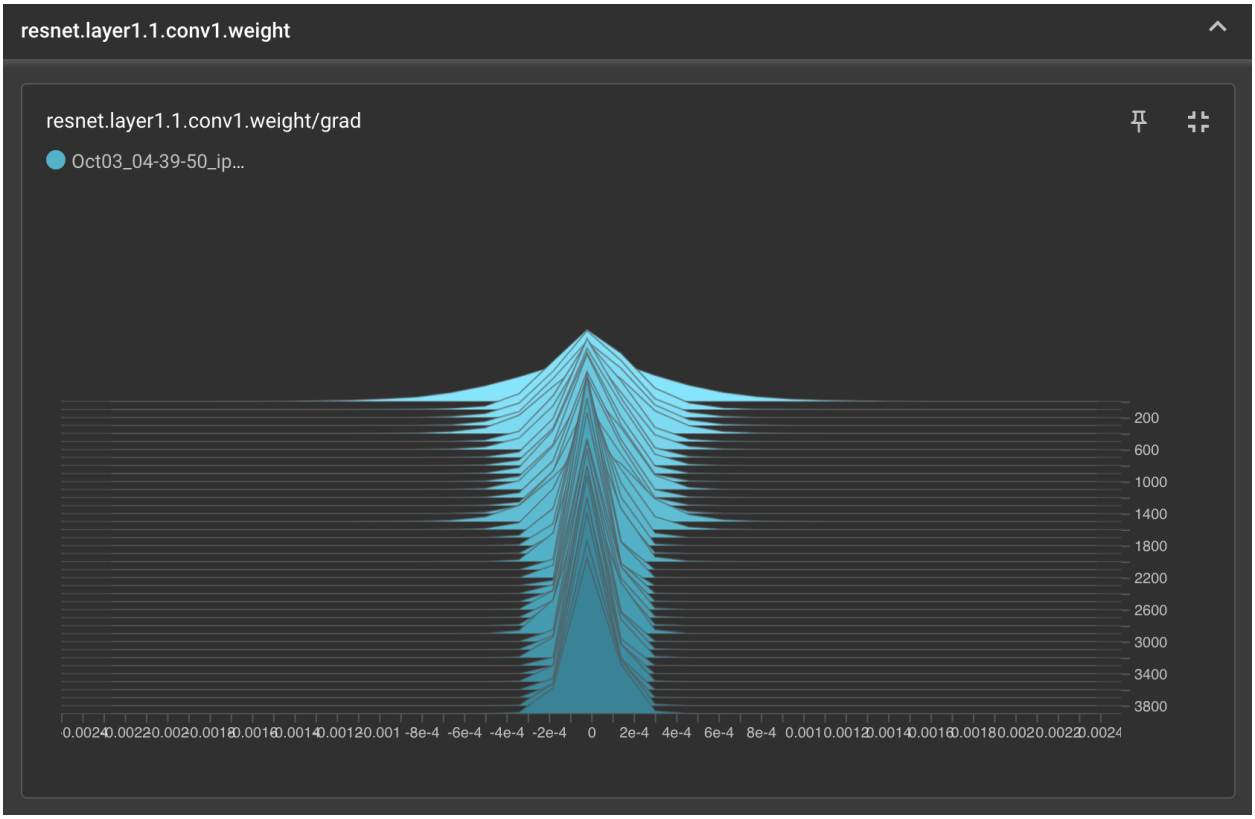

Figure 2.4: Histogram of `Conv1` layer

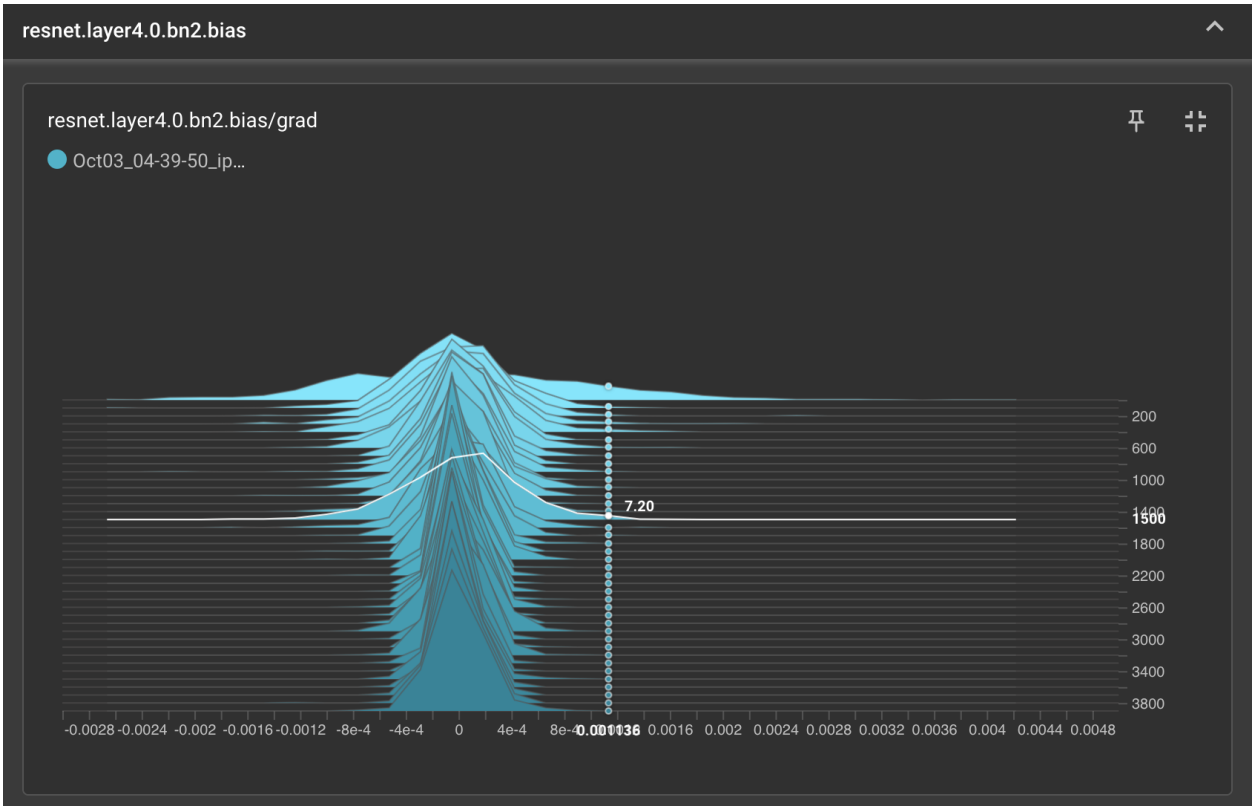

Figure 2.5: Histogram of `BN4` layer

– Compare the two histogram plots. How do they change over time? Why do you think this is happening?

> **Solution:**
> In the two histogram plots of the Conv1 and the BN4 layer, we notice that the distribution gets narrower as out iterations increase. Both have very similar plots with different variance in both of them. A batch norm layer is used to address the internal covariance shift and to stabilize the gradients. We see over time the gradients are being stabilized as the number of iterations increase as seen from the narrowing of the histogram and that is expected to happen since when training, the model adapts to the mean and standard deviation of the dataset.

– We can also visualize how the feature representations specialize for different classes. Take 1000 random images from the test set of PASCAL, and extract ImageNet (finetuned) features from those images. Compute a 2D t-SNE (use sklearn) projection of the features, and plot them with each feature color-coded by the GT class of the corresponding image. If multiple objects are active in that image, compute the color as the "mean" color of the different classes active in that image. Add a legend explaining the mapping from color to object class.

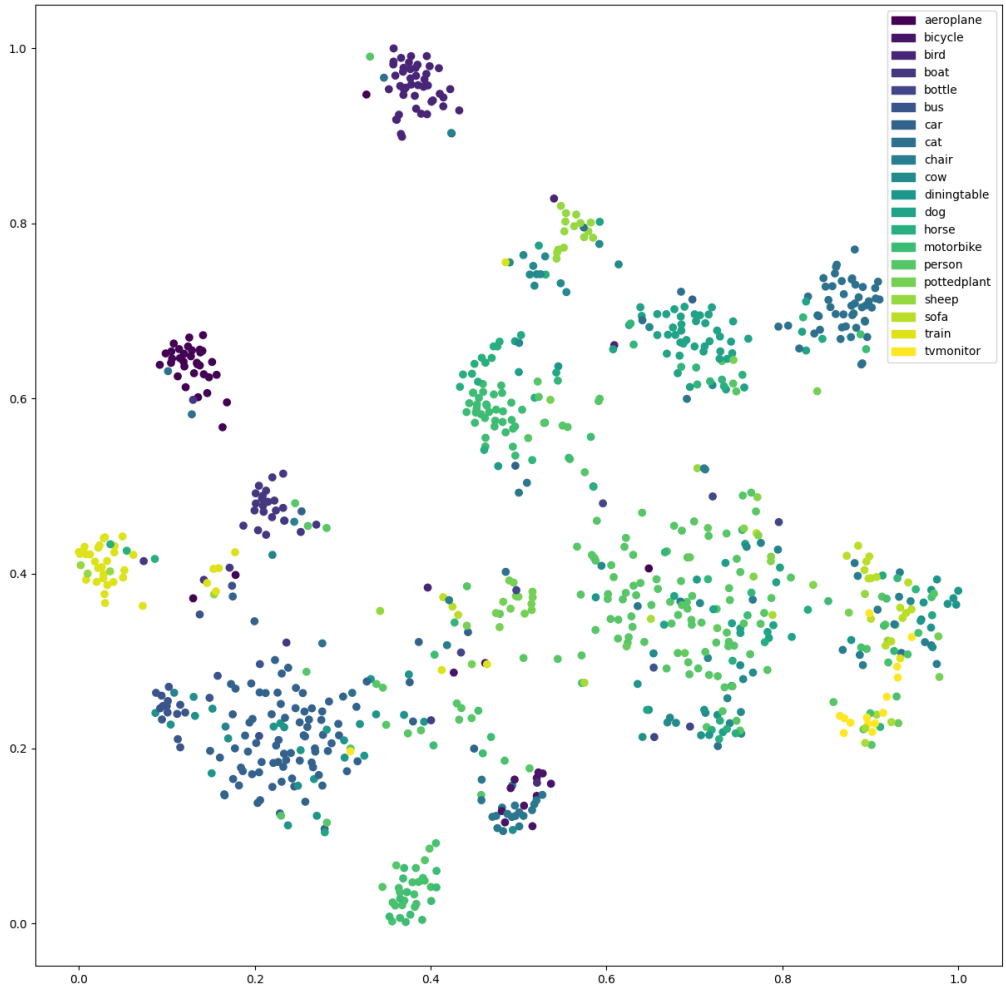

Figure 2.6: t-SNE

– Briefly describe what you observe in the t-SNE plot. Does this match your expectations?

Solution:
t-SNE is an algorithm that is used to visualize high dimension data in lower dimensions and that is what can be seen in the plot. We see that our model is able to predict individual classes well and we notice some amount of separation in different classes. By using the colormap to visualize this, we see that the classes close to each other in index have a similar color but the similar objects tend to form clusters which can be easily classified.

3 Supervised Object Detection: FCOS (60 points)

In this problem, we'll be implementing supervised Fully Convolutional One-stage Object Detection (FCOS).

- **Setup**. This question will require you to implement several functions in `detection_utils.py` and `one_stage_detector.py` in the `detection` folder. Instructions for what code you need to write are in the `README` in the `detection` folder of the assignment.

  We have also provided a testing suite in `test_one_stage_detector.py`. First, run the test suite and ensure that all the tests are either skipped or passed. Make sure that the Tensorboard visualization works by running 'python3 train.py –visualize_gt'; this should upload some examples of the training data with bounding boxes to Tensorboard. Make sure everything is set up properly before moving on.

  Then, run the following to install the `mAP` computation software we will be using.

  ```
  cd <path/to/hw/>/detection
  pip install wget
  rm -rf mAP
  git clone https://github.com/Cartucho/mAP.git
  rm -rf mAP/input/*
  ```

  Next, open `detection/one_stage_detector.py`. At the top of the file are detailed instructions for where and what code you need to write. Follow all the instructions for implementation.

- **Deliverables.**
  - It's always a good idea to check if your model can overfit on a small subset of the data, otherwise there may be some major bugs in the code. Train your FCOS model on a small subset of the training data and make sure the model can overfit. Include the loss curve from over-fitting below.

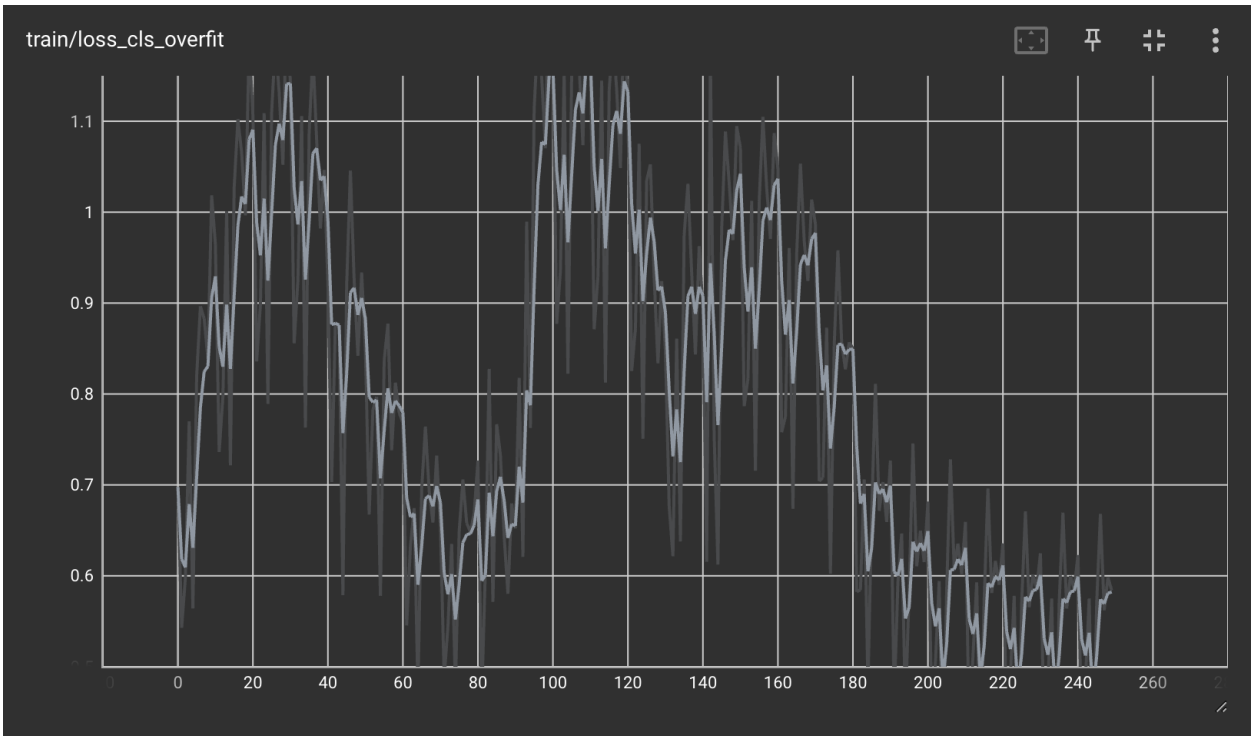

Figure 3.1: Overfitting Train/Loss cls Curve

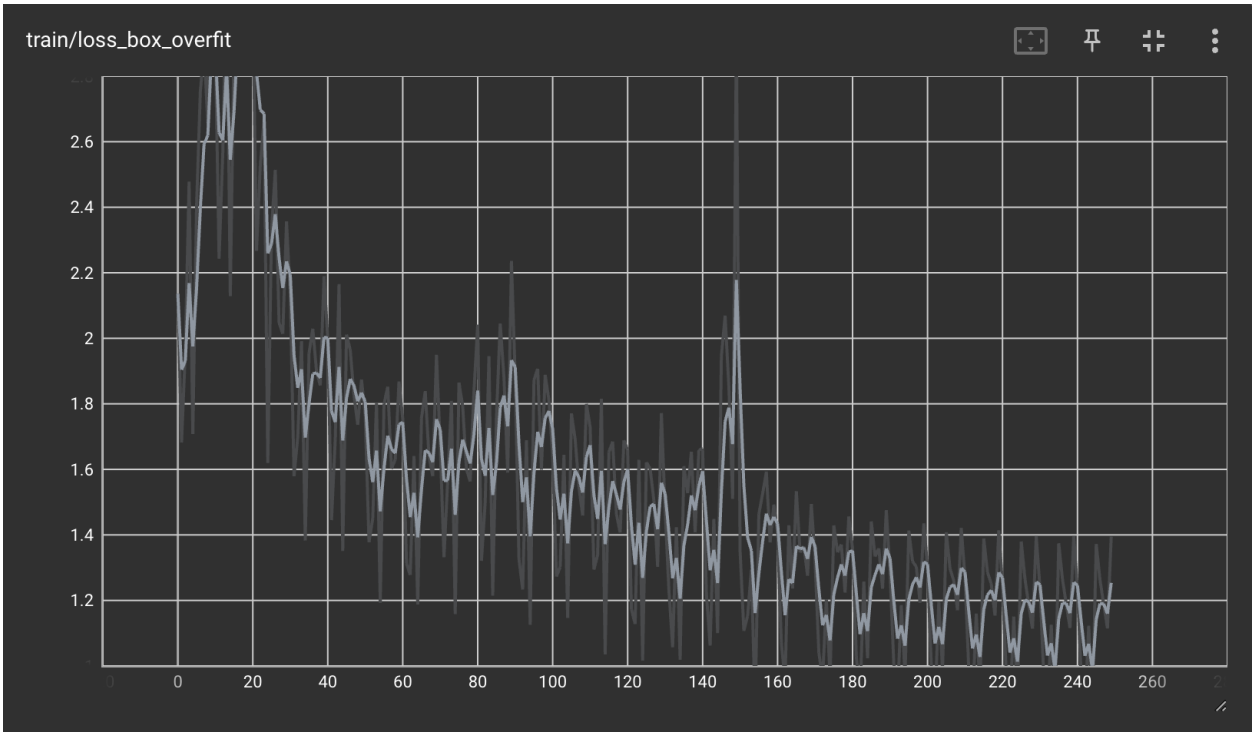

Figure 3.2: Overfitting Train/Loss box Curve

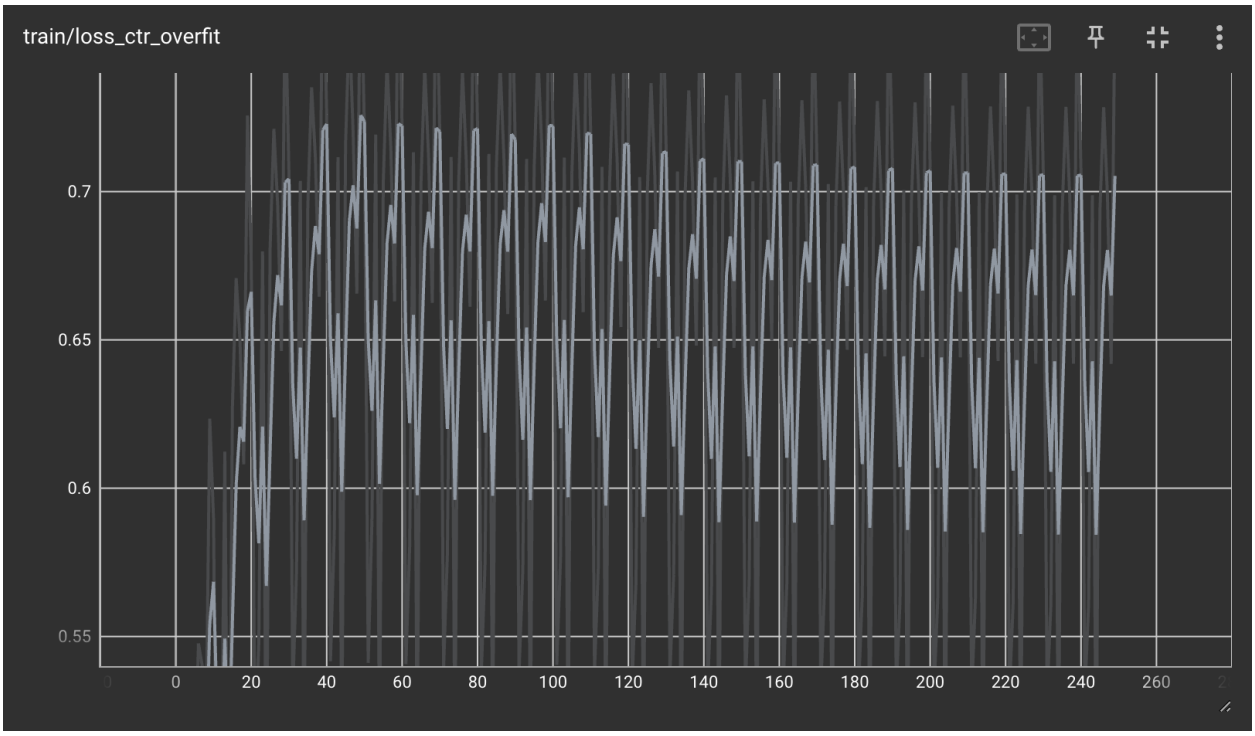

Figure 3.3: Overfitting Train/Loss ctr Curve

– Next, train FCOS on the full training set and include the loss curve below.

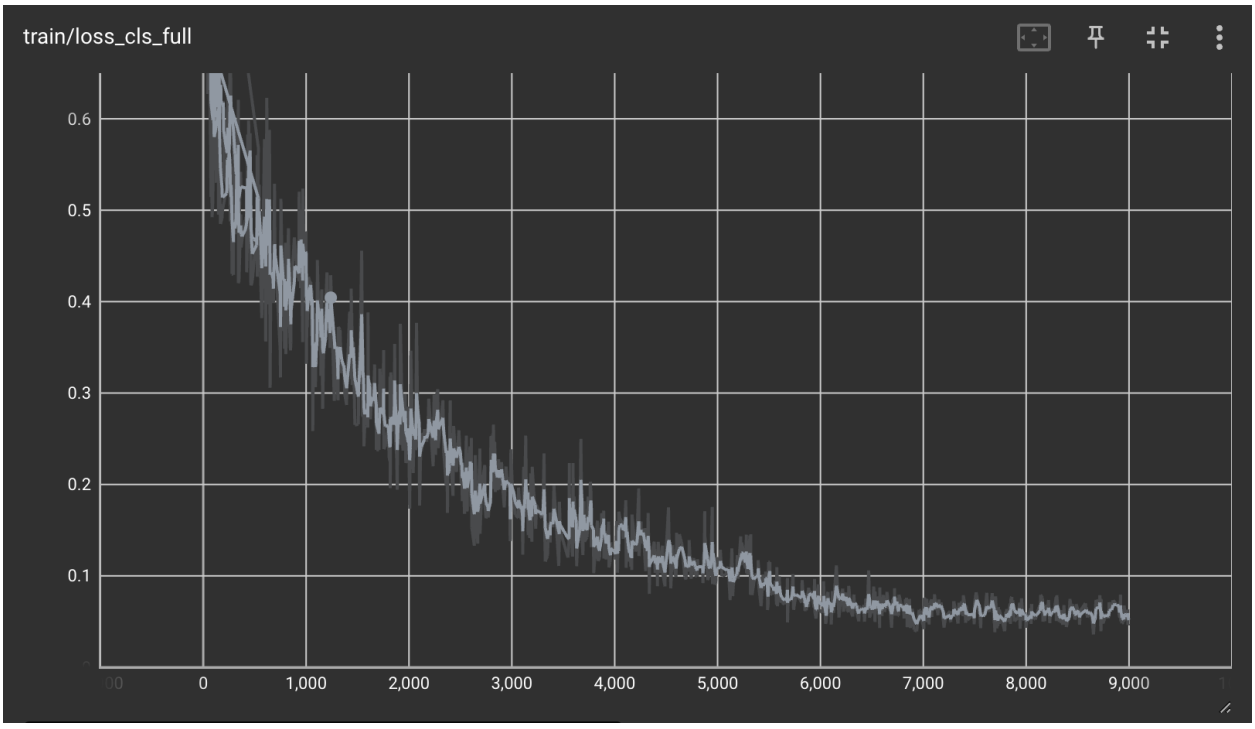

Figure 3.4: Full Training/Loss cls Curve

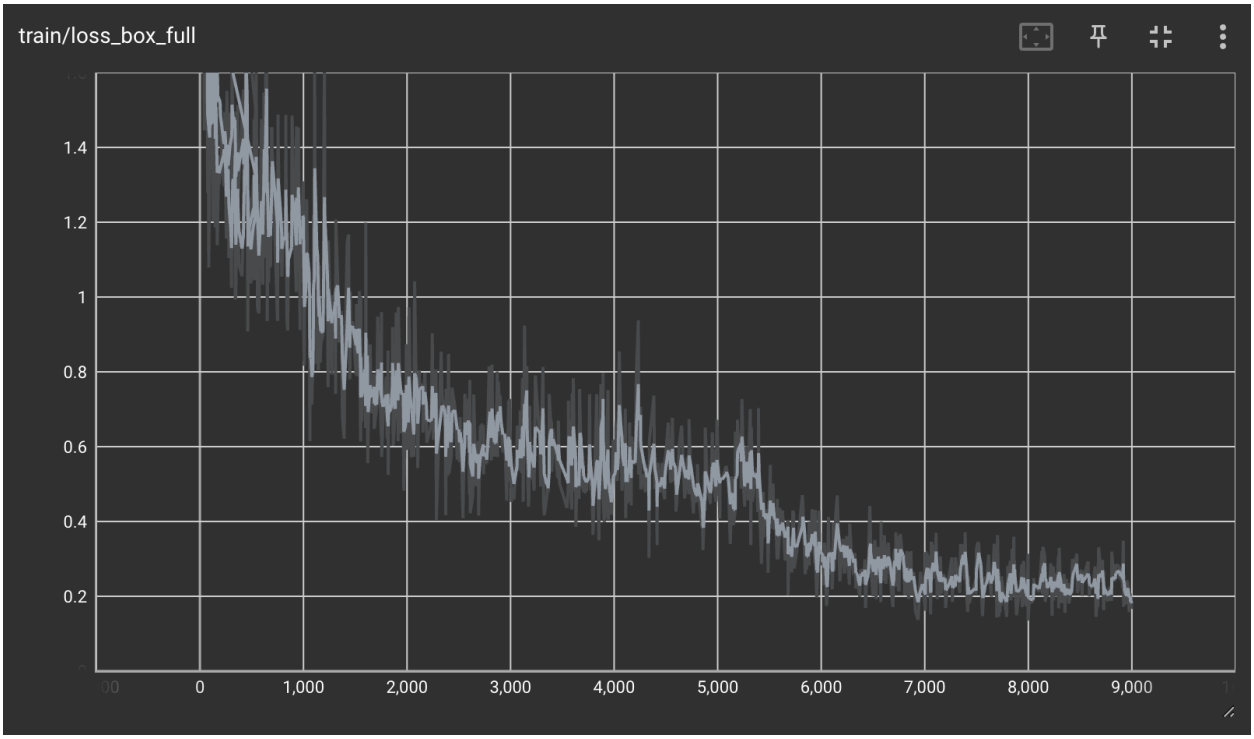

Figure 3.5: Full Training/Loss box Curve

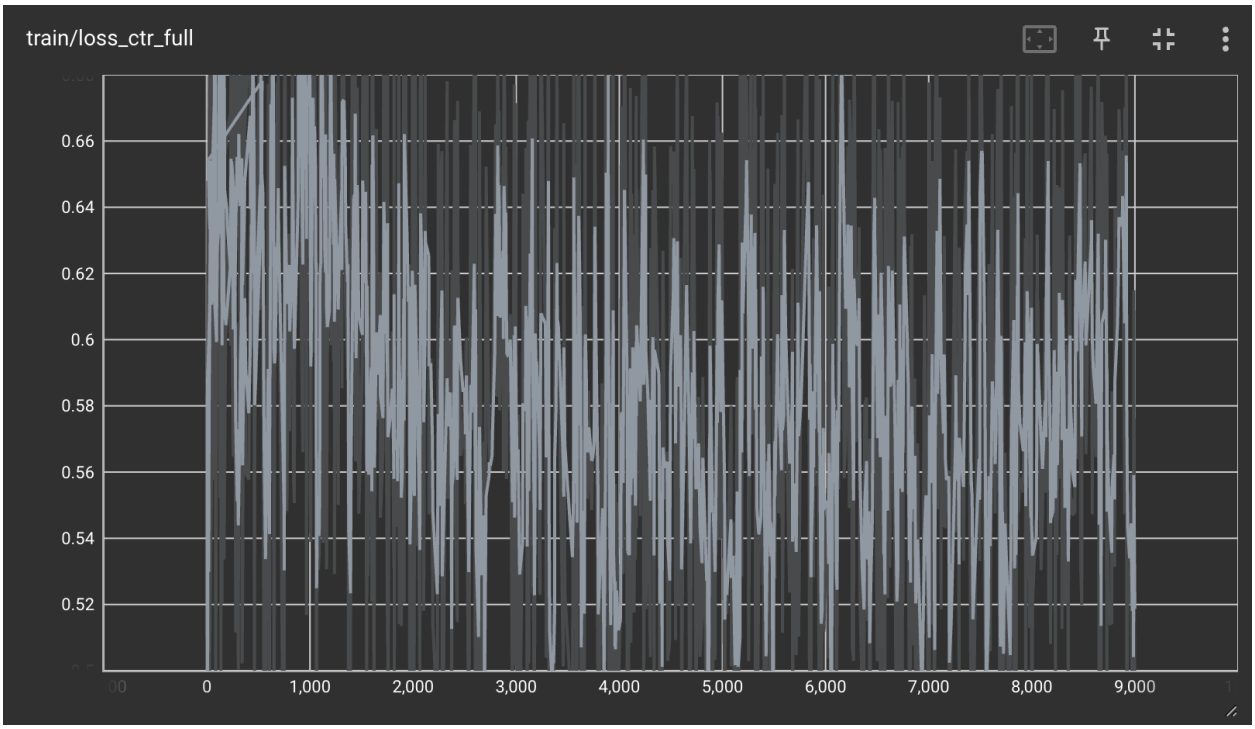

Figure 3.6: Full Training/Loss ctr Curve

– Include the plot of the model's class-wise and average mAP. If everything is correct, your im-

plementation should reach a mAP of at least 20.

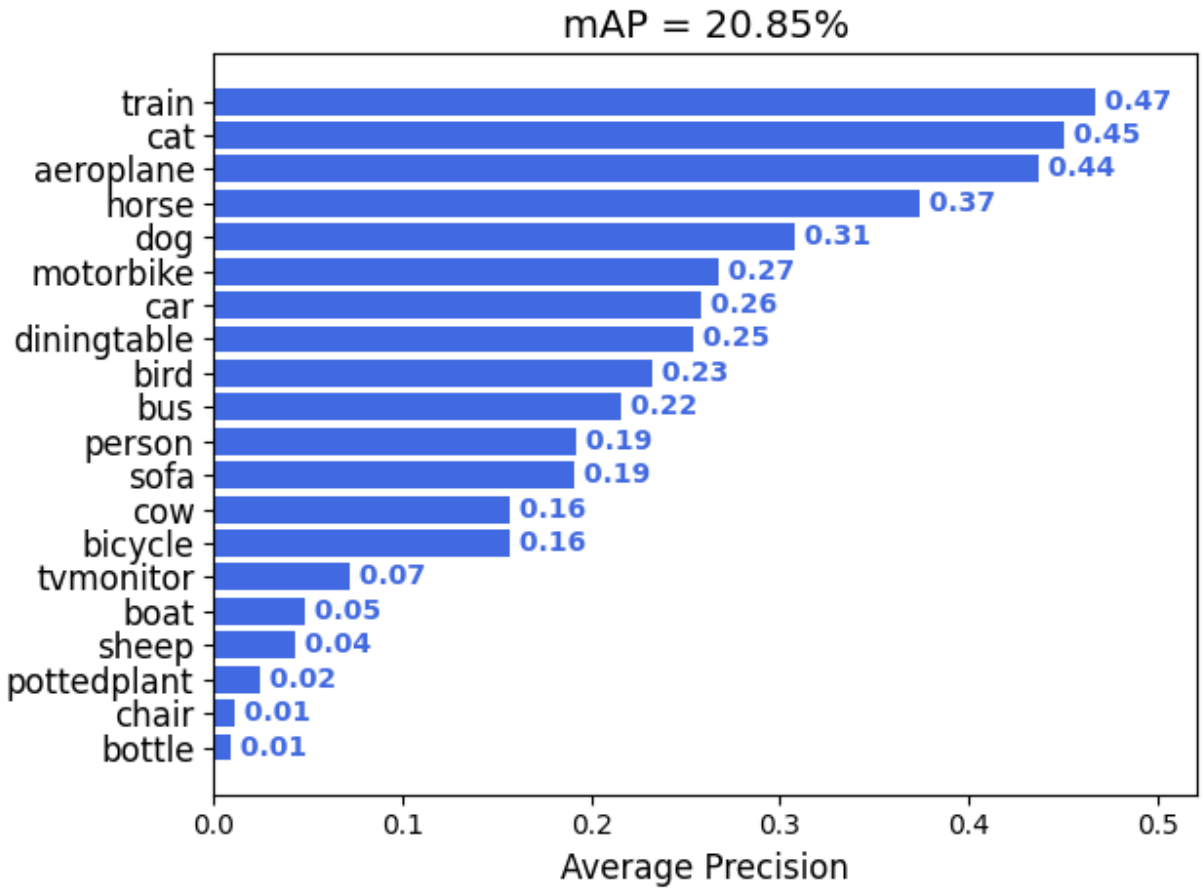

Figure 3.7: mAP plots

– Paste a screenshot of the Tensorboard visualizations of your model inference results from running inference with the `--test_inference` flag on.

Figure 3.8: Tensorboard Inference Visualization

– What can you conclude from the above visualizations? When does the model succeed or fail? How can you improve the results for the failure cases?

Solution:
I ended up taking my previous checkpoint and re-training it again for my inference. The training/loss curves are as expected as they reduce over time which indicates the model is training. I also notice that the ctr loss curve is very jittery. We see that large objects like train and aeroplanes having a high Average Precision while smaller objects like bottles do not have a great value for it. Also cats and dogs also have a decently high AP. This can be attributed to the fact that the Pascal VOC dataset contains a lot of images with these classes. From the inference, we do see the model does a decent job of creating the bounding boxes for the images and is able to identify multiple classes in an image too. I think the model fails when a lot of objects are in the same image and that makes it slightly difficult for the detector to detect the classes. It also might fail in the case where there are occlusions which prevent the detector from detecting the class. It also fails when the objects are small which is indicated by the AP of bottle class to be very low. For improving the results of a failure case, we can use more data that contains special cases and train it for a longer time with more optimized solutions. We can also use data augmentation techniques for training to make the model more generalizable. It would also make sense to play around with the thresholds to get bounding boxes as desired.

Collaboration Survey Please answer the following:

1. Did you receive any help whatsoever from anyone in solving this assignment?

   ◯ [✓] Yes

   ◯ No

   - If you answered 'Yes', give full details:
   - (e.g. "Jane Doe explained to me what is asked in Question 3.4")

   Thomas Klein helped me in understanding FPN network and its implementation. He helped me understand the shapes that had to be passed in the FCOS network.
   Sumesh KS helped me in the colormap for legend plotting on t-SNE plot and helped me optimize my code for training which took 9 hours to train if loops were used.
   Boga Balaji helped me in debugging my inference code since I was getting 0.0 mAP.

2. Did you give any help whatsoever to anyone in solving this assignment?

   ◯ [✓] Yes

   ◯ No

   - If you answered 'Yes', give full details:
   - (e.g. "I pointed Joe Smith to section 2.3 since he didn't know how to proceed with Question 2")

   I helped Rithwik Jayanth in P2 and understanding P3.
   I also discussed with Sumesh KS on how to implement the P3 and why he was getting a shape error.
   I also helped Mukul Ganwal in implementing Resnet fc layer.

3. Note that copying code or writeup even from a collaborator or anywhere on the internet violates the Academic Integrity Code of Conduct.