HOMEWORK 1

16824 VISUAL LEARNING AND RECOGNITION (SPRING 2023)
https://piazza.com/class/lcy4ow5l5xp2fl

RELEASED: Mon, 6th Feb 2023
DUE: Wed, 22nd Feb 2023
Instructor: Deepak Pathak
TAs: Ananye Agarwal, Rohan Choudhury, Murtaza Dalal, Russell Mendonca

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 7** late days across all homework submissions. Submissions more than 7 days after the deadline will receive a 0.
- **Submitting your work:**
  - We will be using Gradescope (https://gradescope.com/) to submit the Problem Sets. Please use the provided template only. 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. You should report the `Loss/Train`, `map` and `learning_rate` curves logged to tensorboard in the box below.

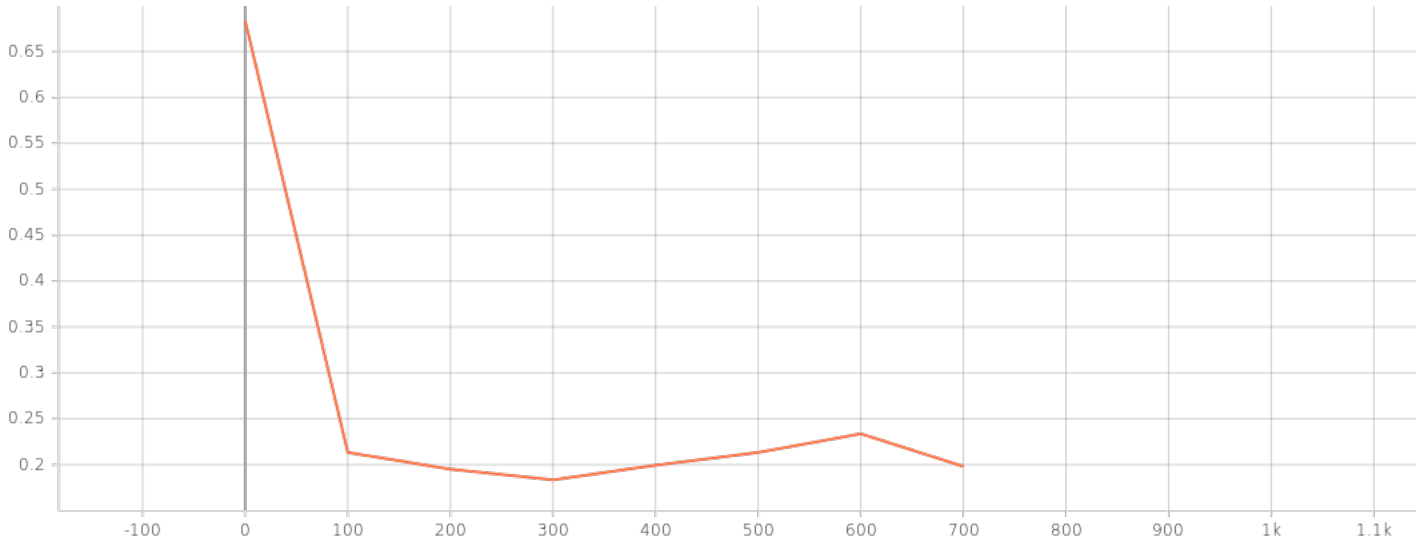

Figure 1.1: `Loss/Train` for simple CNN

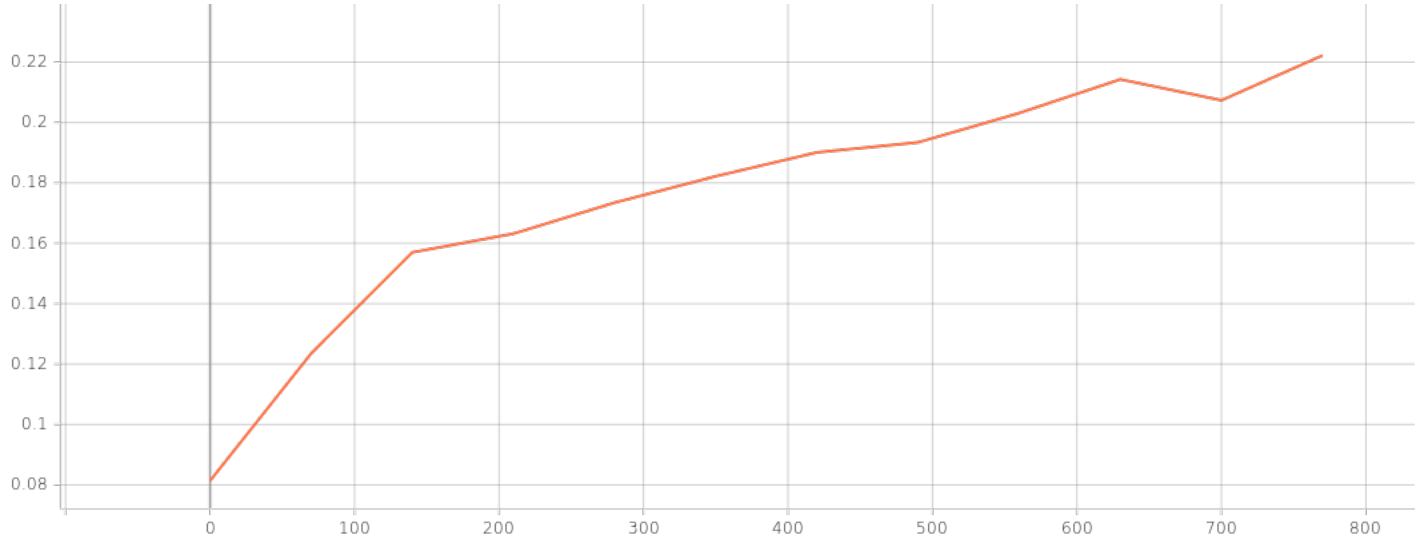

Figure 1.2: `map` for simple CNN

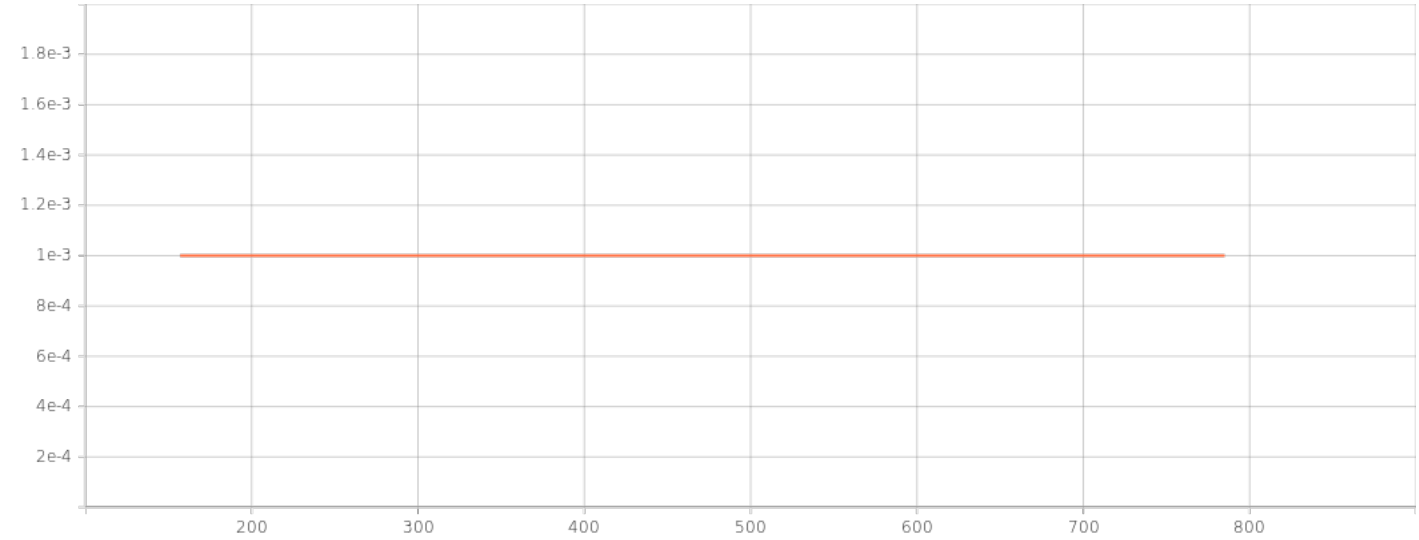

Figure 1.3: `learning_rate` for simple CNN

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

Hopefully we all got much better accuracy with the deeper model! 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 pretrained 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 historgram of gradients from tensorboard for `layer1.1.conv1.weight` and `layer4.0.bn2.bias`.
  - 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.

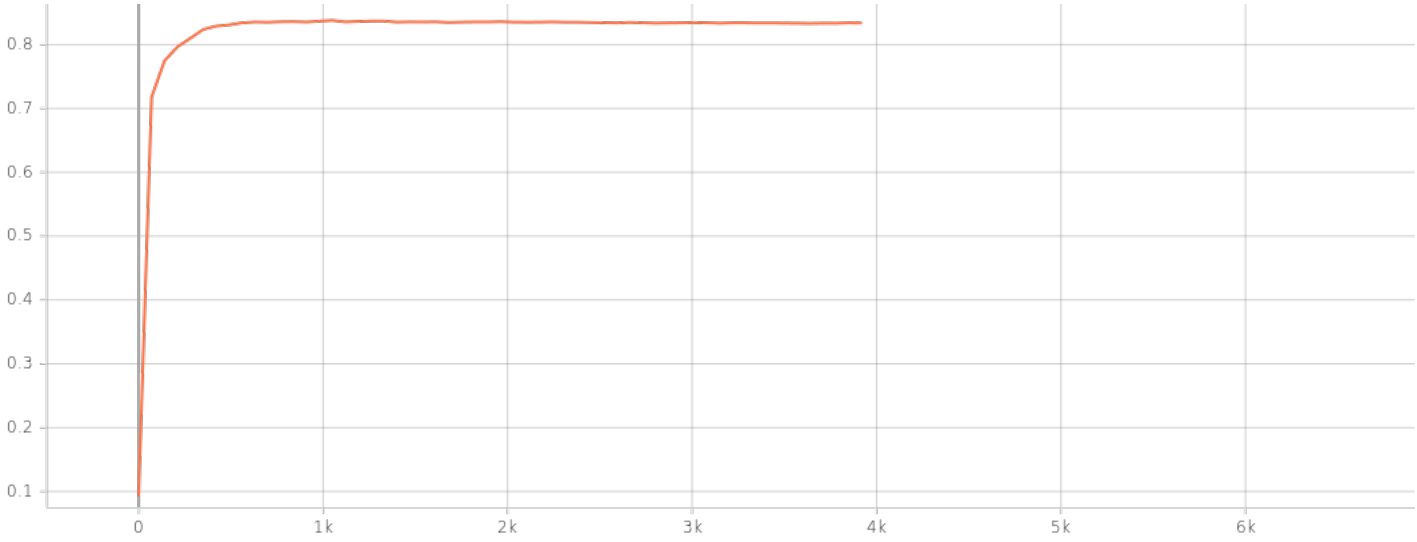

Figure 2.1: mAP

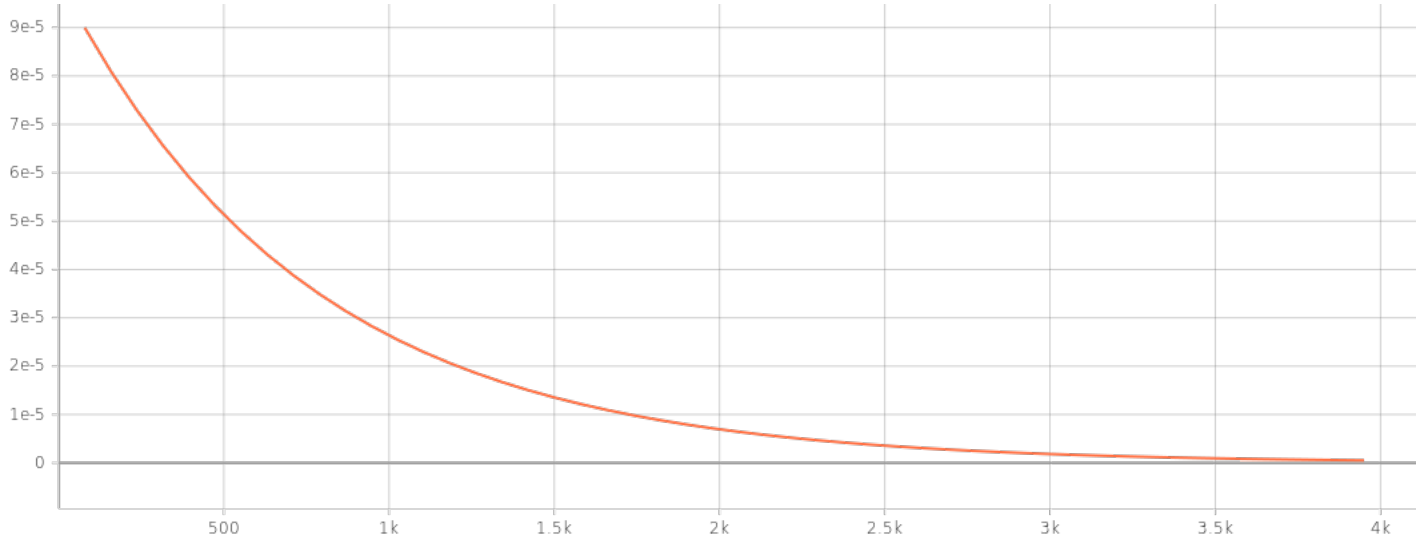

Figure 2.2: learning_rate

Figure 2.3: Training Loss

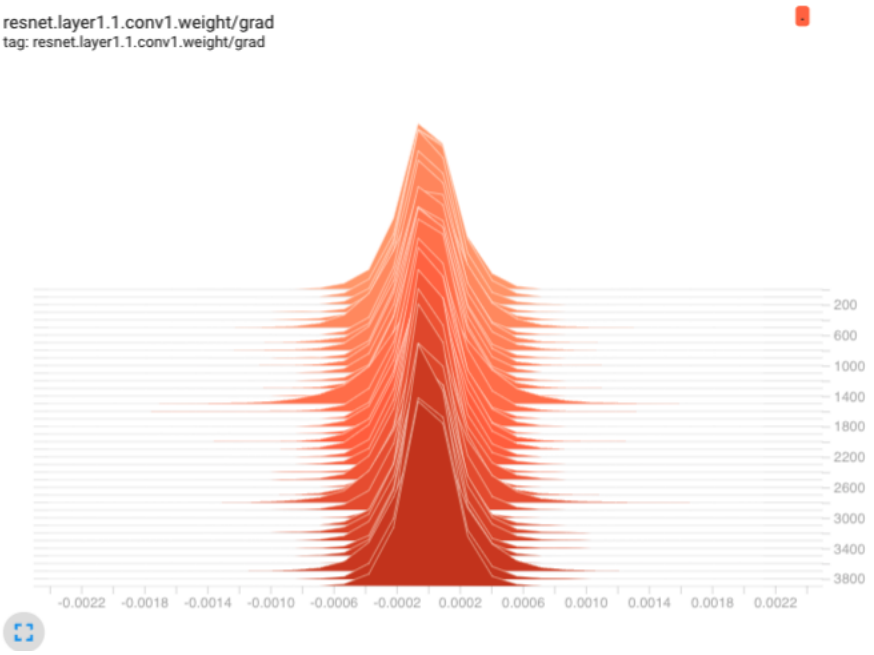

Figure 2.4: histogram_conv1

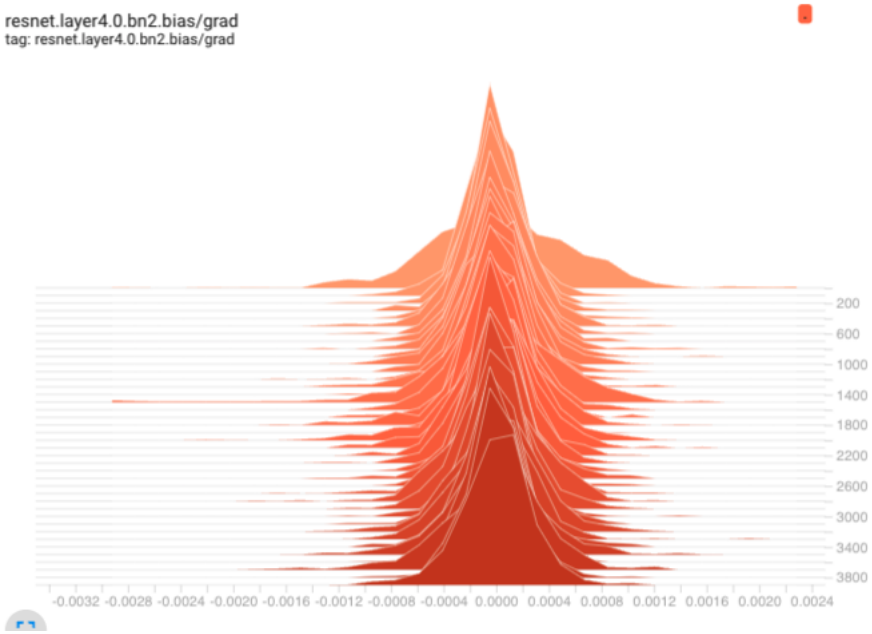

Figure 2.5: histogram_bn

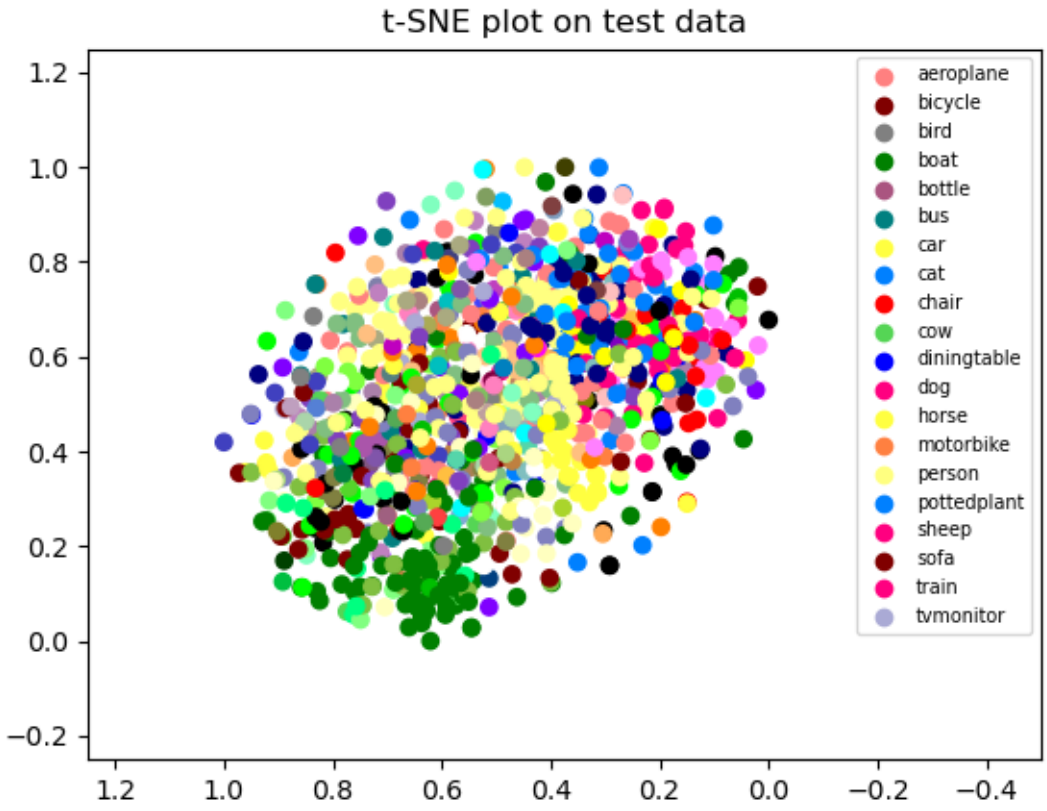

Figure 2.6: t-SNE

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/detection_utils.py` and `detection/one_stage_detector.py`. 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 boxe 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.** Below, you will need to provide:
  - The loss curve from over-fitting a small model to the training set
  - The loss curve from training your full model
  - A screenshot of the your model results on TensorBoard from running model inference.
  - The final mAP plot.

1. Paste your plot of the loss curve from training your FCOS model on a small subset of the training data.

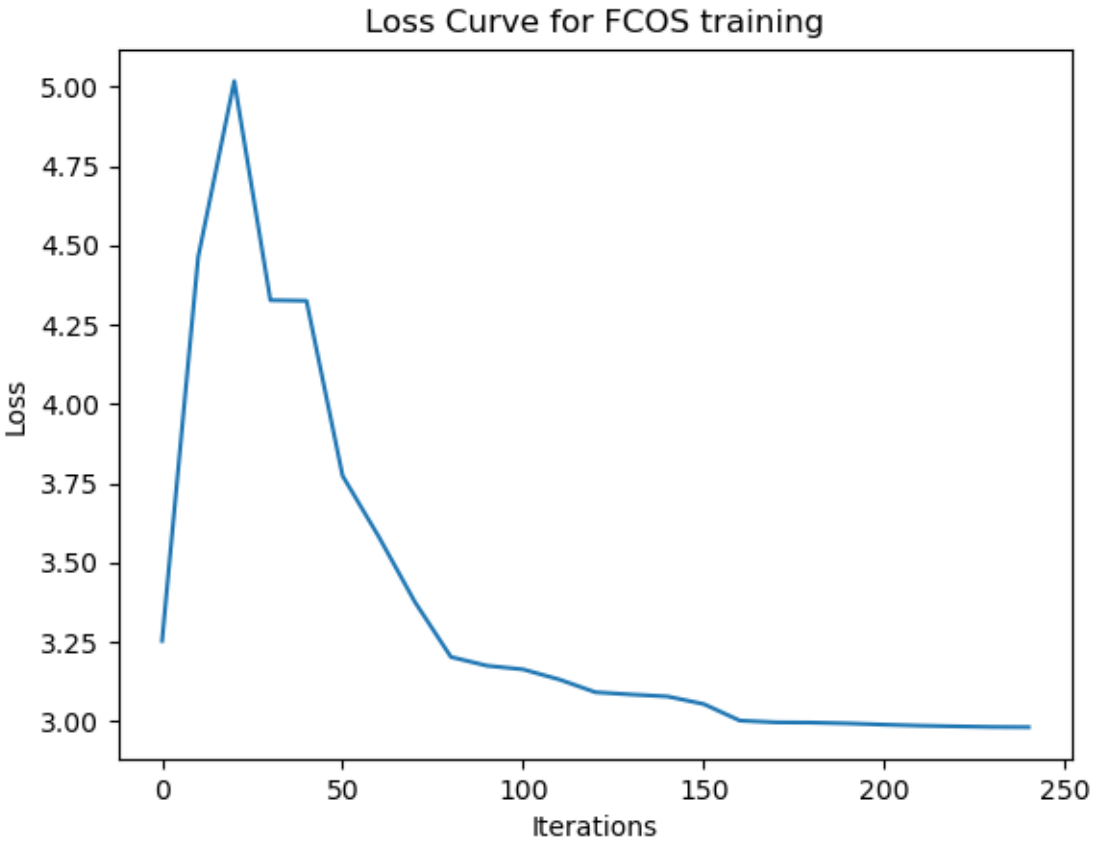

Figure 3.1: Overfit Training Curve

2. Paste your plot of the loss curve from training your FCOS model on the entire training set.

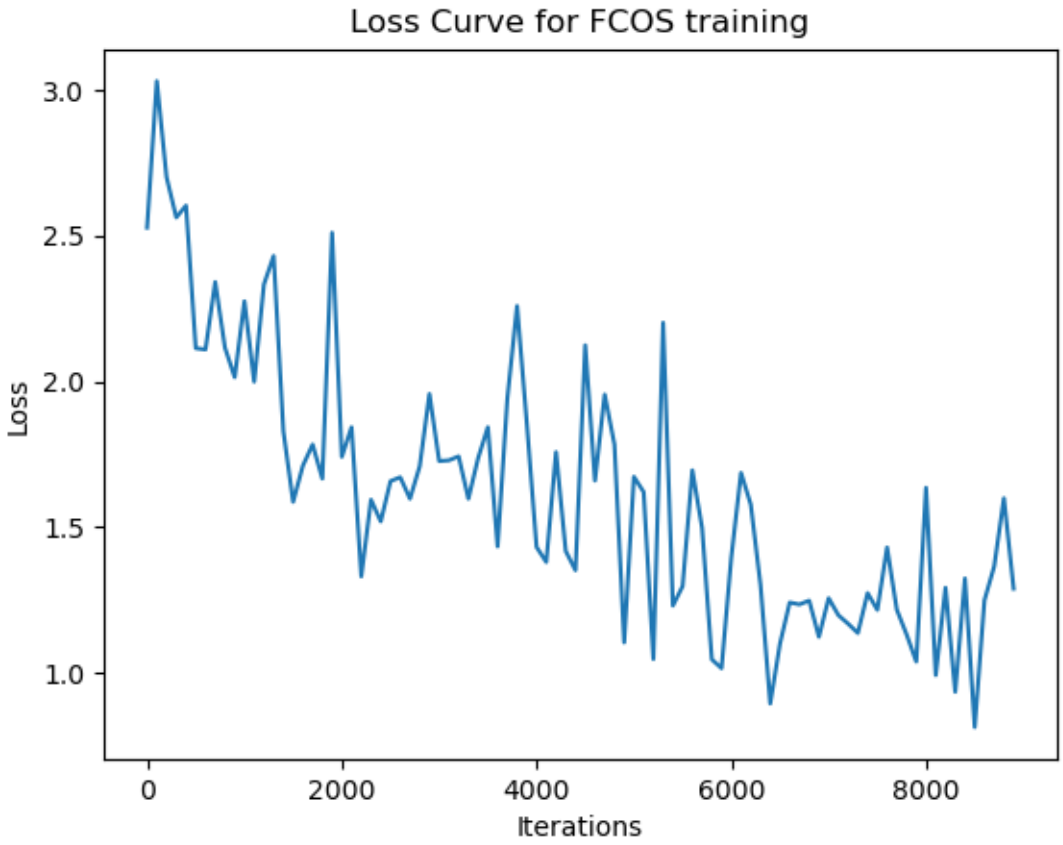

Figure 3.2: Full Training Curve

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

Figure 3.3: Tensorboard Inference Results

4. Paste the plot of the model's classwise and final mAP. If everything is correct, your implementation should reach at least 20 mAP.

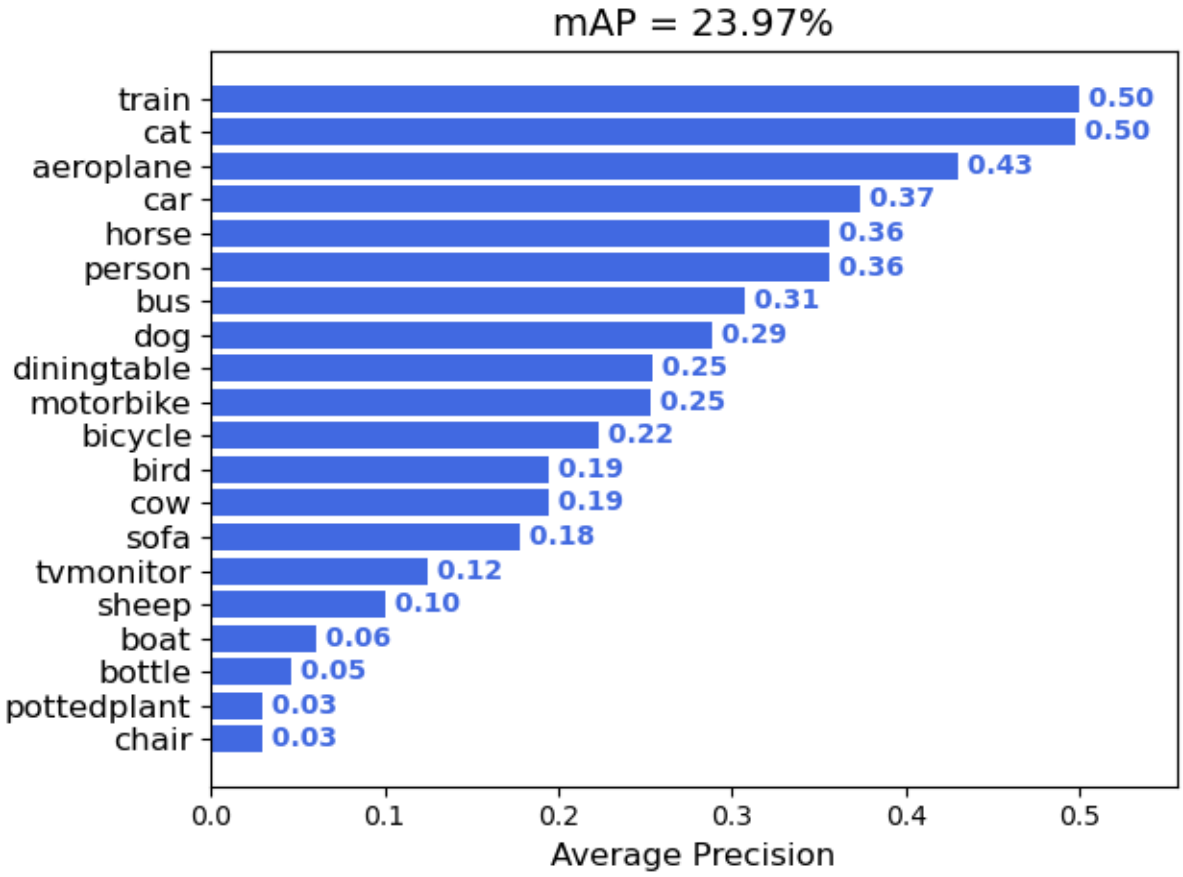

Figure 3.4: Final mAP.

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")

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")

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