# 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.

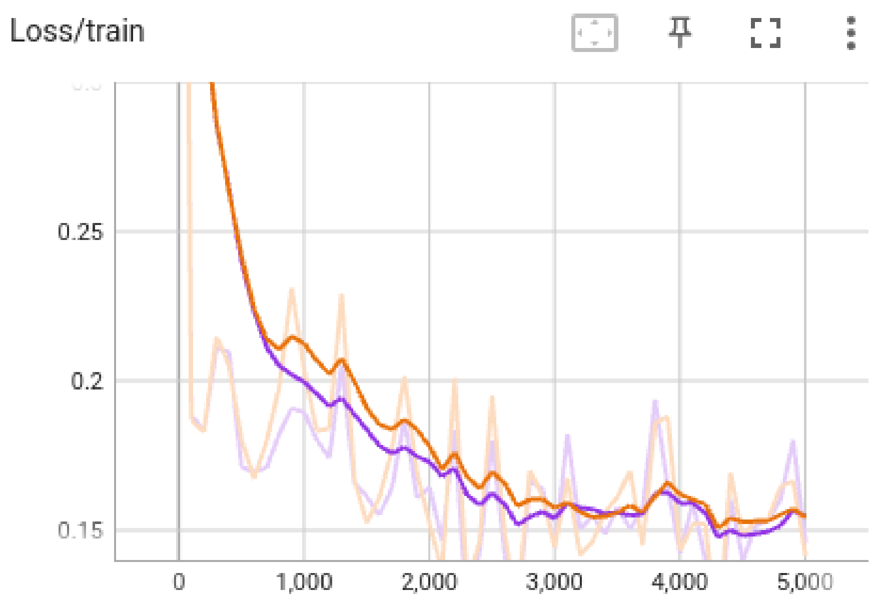


Figure 1.1: `Loss/Train` with (orange) and without (violet) data augmentations.

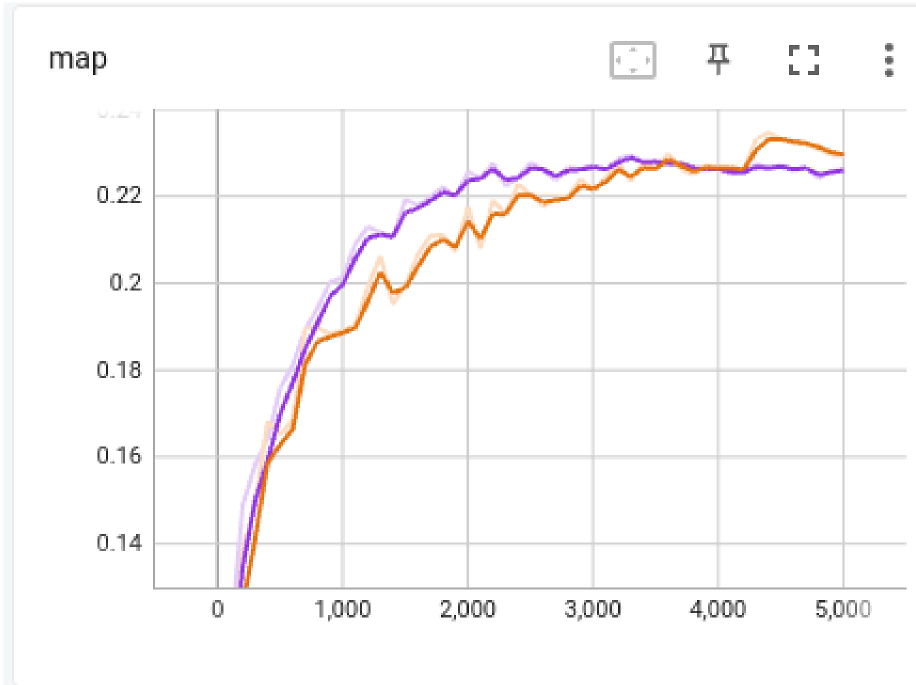


Figure 1.2: `mAP` with (orange) and without (violet) data augmentations.

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

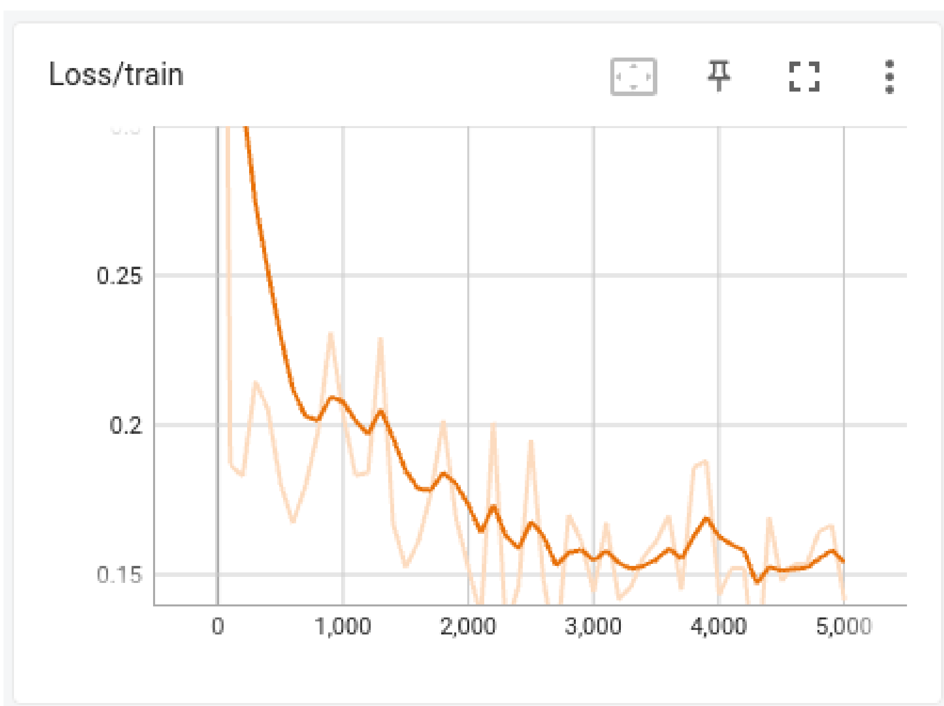


Figure 1.3: `Loss/Train` for simple CNN

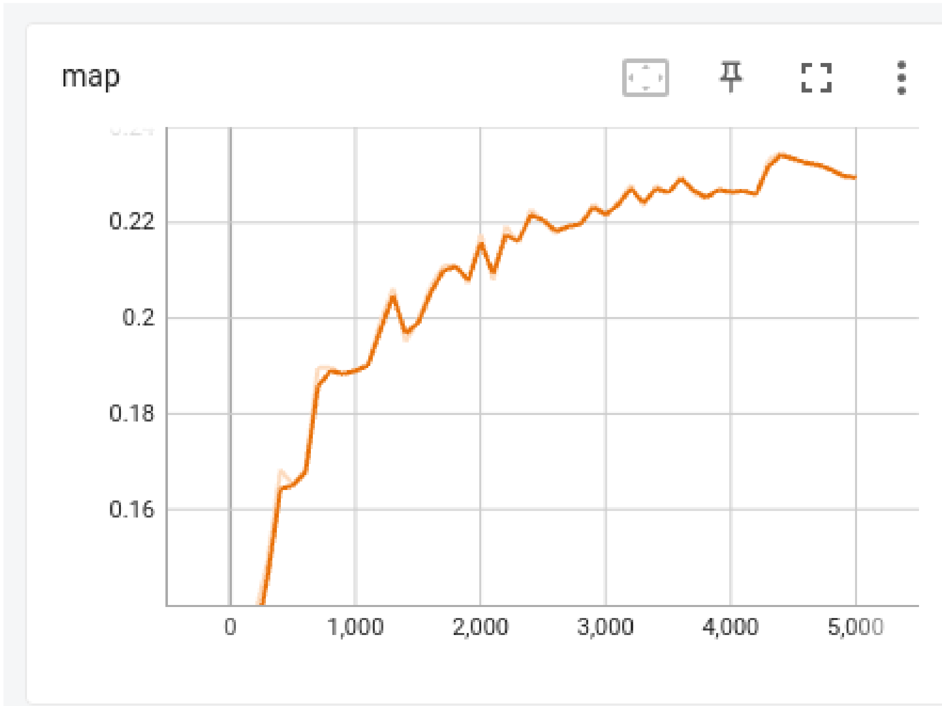


Figure 1.4: `mAP` for simple CNN

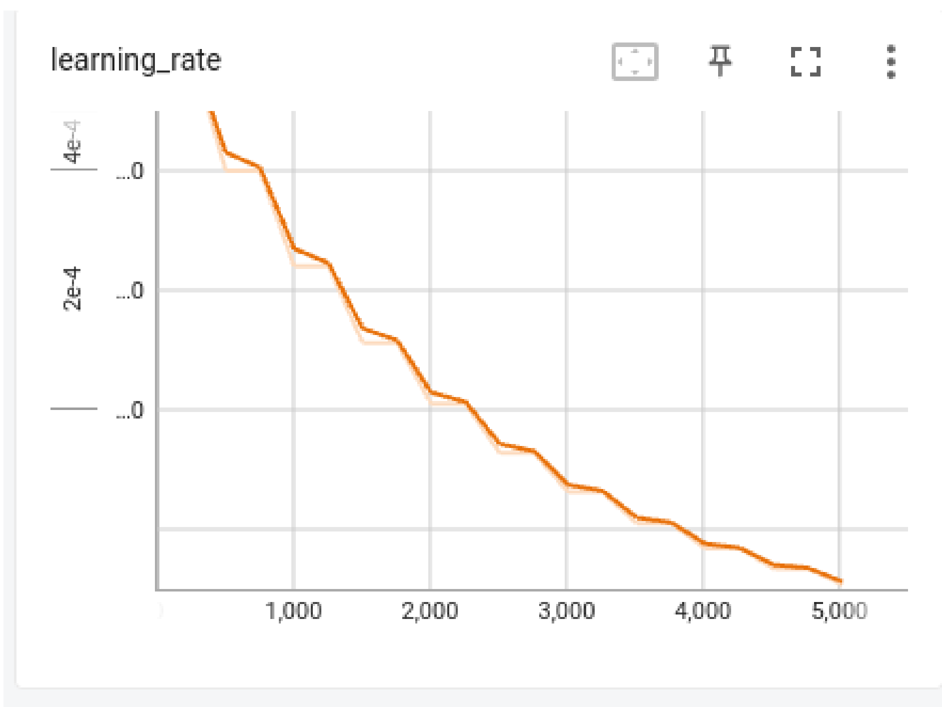


Figure 1.5: `learning_rate` for simple CNN

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

**Solution:**
$args.batch\_size = 20$
$args.device = cuda$
$args.epochs = 20$
$args.gamma = 0.8$
$args.inp\_size = 64$
$args.log\_every = 100$
$args.lr = 0.0005$
$args.save\_at\_end = False$
$args.save\_freq = -1$
$args.step\_size = 2$
$args.test\_batch\_size = 1000$
$args.val\_every = 100$

## 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`.

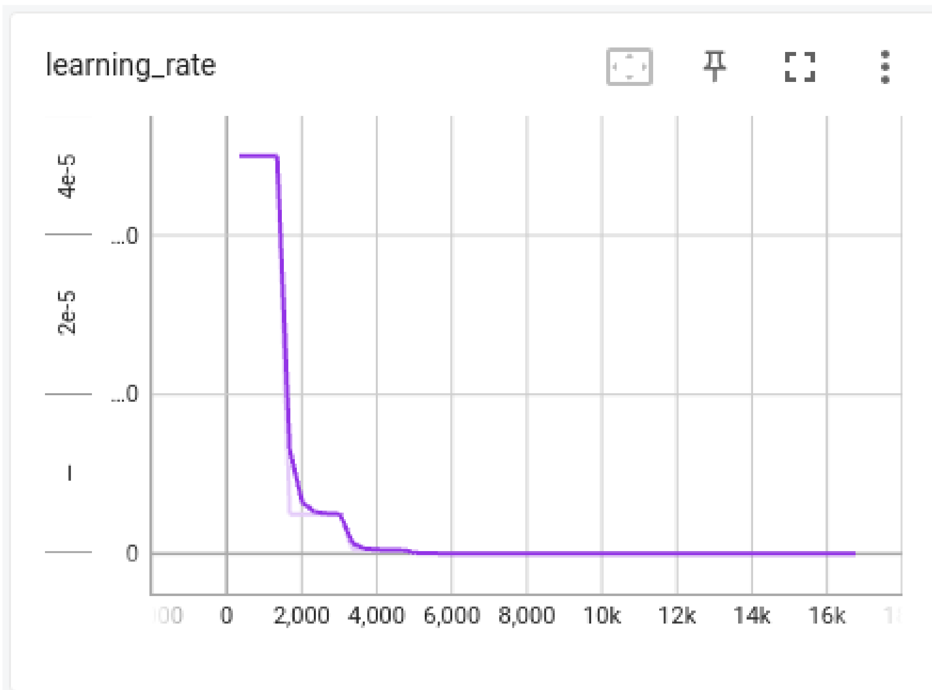


Figure 2.1: `learning_rate` for ResNet

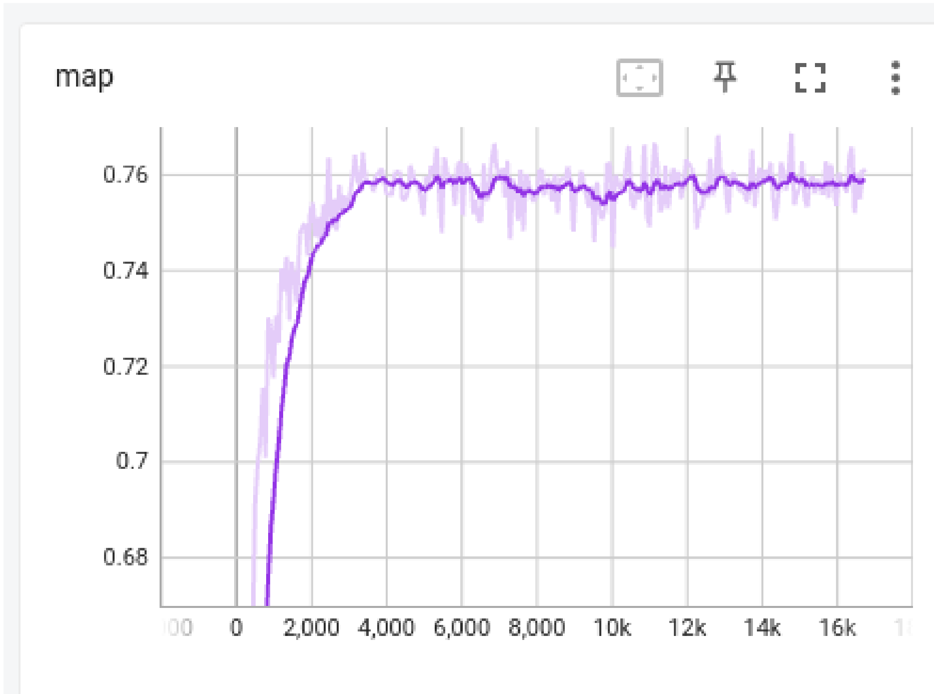


Figure 2.2: `mAP` for ResNet

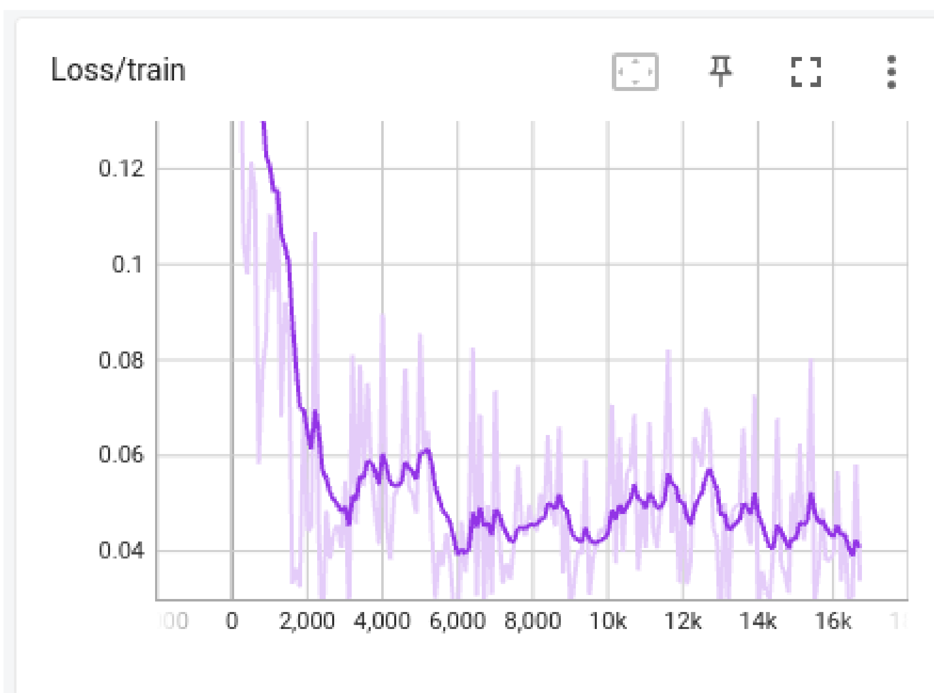


Figure 2.3: `Loss/Train` for ResNet

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

**Solution:**
The loss decreased dramatically at the beginning of the training, same as for mAP, which increased significantly but as iteration goes on, the net seems to have a hard time finding a good descent direction, and some overfitting happens, as the mAP only slowly increased.

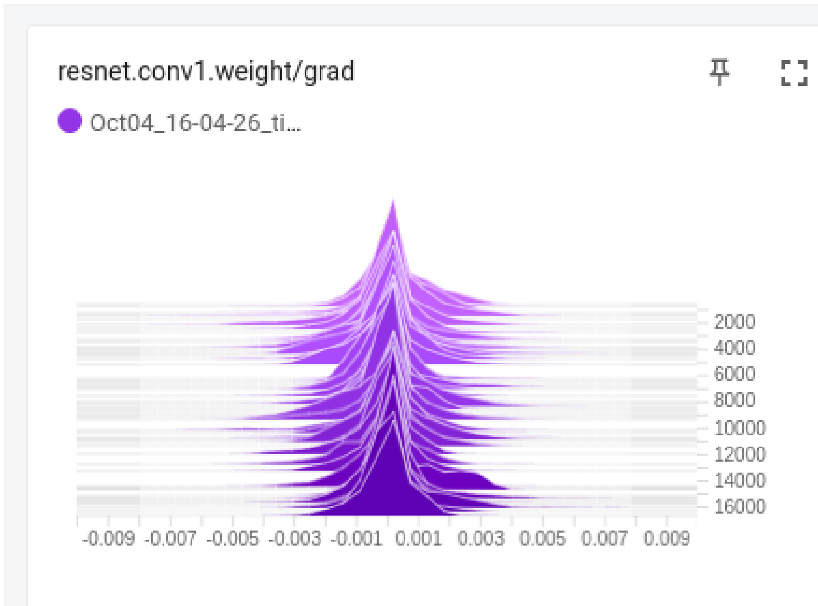


Figure 2.4: Histogram of `Conv1` layer

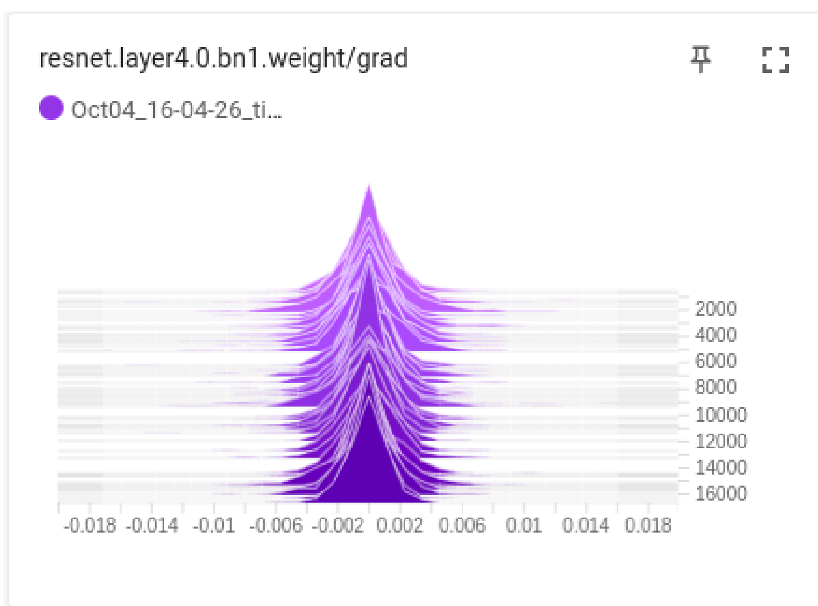


Figure 2.5: Histogram of `BN4` layer

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

**Solution:**
The Conv1 layer has a wider distribution than the layer of BN4. training continues, the weights in the Conv1 layer adjust to capture useful features, causing the histogram to shift and become more focused around specific values.

– 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.

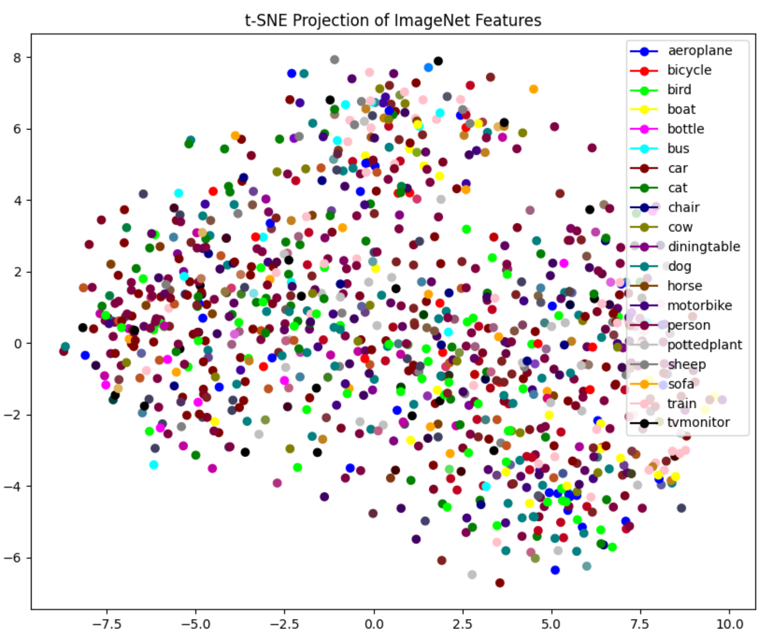


Figure 2.6: t-SNE

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

**Solution:**
Overall the t-SNE plot does not show any visible clustering, which is not my expectation. I think the main issue is the hyperparameter tuning of the t-SNE algorithm, as I tried many combinations but none of them seems to work. Another factor might be the relatively low mAP from the model trained, as the final mAP only reached about 0.76, smaller than 0.8 from the homework description.

# 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.

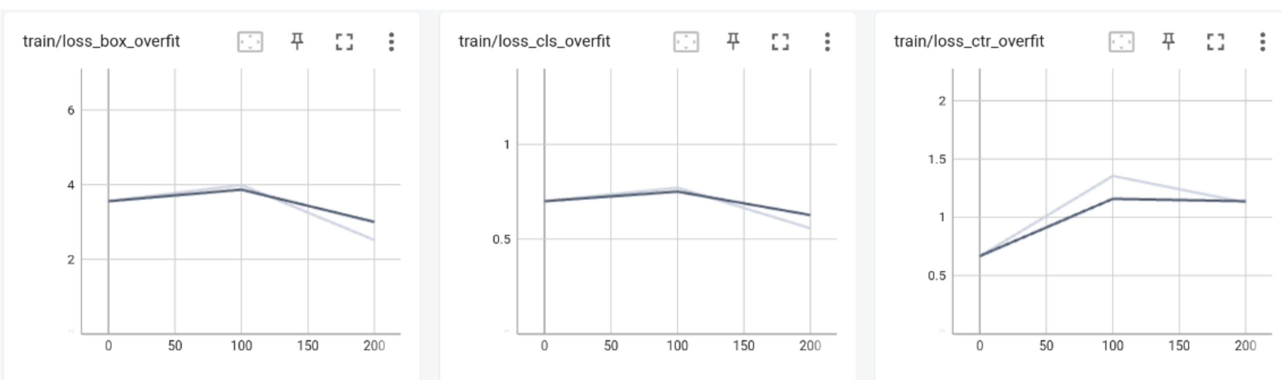


Figure 3.1: Overfitting Training Curve (eval step size changed to 100 for tensorboard writer)

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

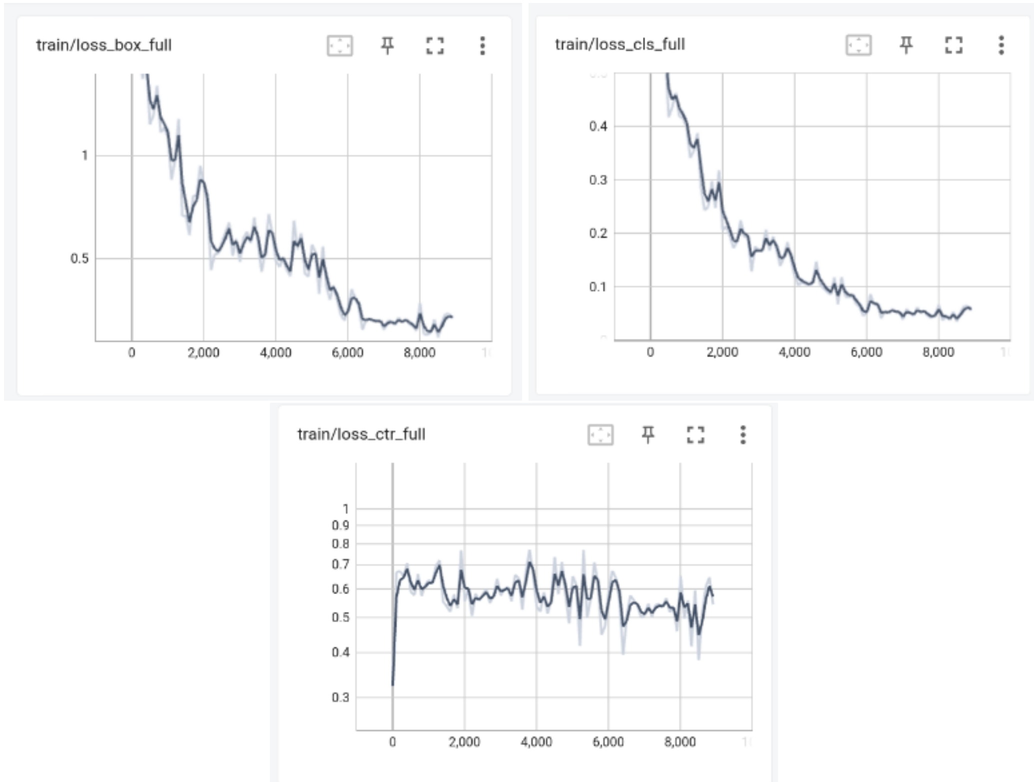


Figure 3.2: Full Training Curve

– Include the plot of the model's class-wise and average mAP. If everything is correct, your implementation should reach a mAP of at least 20.

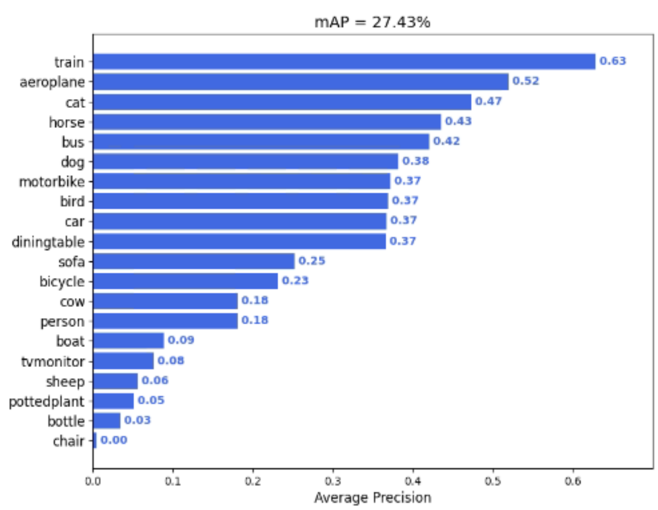


Figure 3.3: mAP plots

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


Figure 3.4: 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 think the overall result is acceptable, as the objects in the images are clearly identified and labeled, one little drawback is the model is not confident enough for most of the classification results, as I had to adjust the threshold to 0.5 to let the visualization happen. Future improvements include further fine-tuning for the hyperparameters and adding more data to the corresponding class which needs improvements, based on the mAP results are not stable across all classes.

**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")

   Feiya Zhu gave me some instructions about one hot indexing in loss calculation. ChatGPT gave me instructions about t-SNE plot creating and hyperparameter tuning.

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.