Assignment 3

Name(s): Taaha Kazi, Nidhish Kamath
NetID(s): tnkazi2, nkamath5

Part 1: Self-supervised Learning on CIFAR10

1) Rotation training

Initial Learning Rate = 0.01, decayed by a factor of 10 every 15 epochs.
Our batch size was 128.
Total number of epochs = 45.
We used the Adam optimizer with default beta values. We chose Adam over vanilla SGD to take advantage of its momentum to find a path to the minima faster, and this choice worked well (we saw that SGD had lower accuracies vs Adam for the first 5 epochs).
Apart from our choice of optimizer, the remainder of our hyperparameters were the same as suggested by the starter code by default - we were able to clear given benchmarks.

On the test set, we were able to achieve an accuracy of **78.72%** (after training for 45 epochs) for this rotation task.

2) Fine-tuning late layers for classification task (Last conv layer & last FC layer)

A) Using the Pre-trained Model (for rotation):
Initial Learning Rate = 0.01, decayed by a factor of 10 every 10 epochs.
Our batch size was 128.
Total number of epochs = 20.
Again, we used the Adam optimizer with default beta values.

B) Using randomly initialized weights:
Initial Learning Rate = 0.01, decayed by a factor of 10 every 10 epochs.
Our batch size was 128.
Total number of epochs = 20.
Again, we used the Adam optimizer with default beta values.

As expected, we found the ResNet pre-trained for the rotation task to work better than the one with randomly initialized weights. The former has learnt to look at images in four different orientations and thus seems to get its predictions right on images that might have objects of interest in different orientations.

Accuracy on test set after training for 20 epochs on model with pre-trained weights: **59.01%**
Acc. on test set after training for 20 epochs on model with randomly initialized weights: **45.74%**

Report the hyperparameters you used to fine-tune your model. Compare the performance between pre-trained model and randomly initialized model.

3) Fully supervised learning for classification

A) Using the Pre-trained Model (for rotation):
Initial Learning Rate = 0.01, decayed by a factor of 10 every 10 epochs.
Our batch size was 128.
Total number of epochs = 20.
Again, we used the Adam optimizer with default beta values.

B) Using randomly initialized weights:
Initial Learning Rate = 0.01, decayed by a factor of 10 every 10 epochs.
Our batch size was 128.
Total number of epochs = 20.
Again, we used the Adam optimizer with default beta values.

Accuracy on test set after training for 20 epochs on model with pre-trained weights: **83.84%**
Acc. on test set after training for 20 epochs on model with randomly initialized weights: **82.21%**

Overall, the performance of the fine-tuning on the full network turned out better than fine-tuning only on the last two layers.

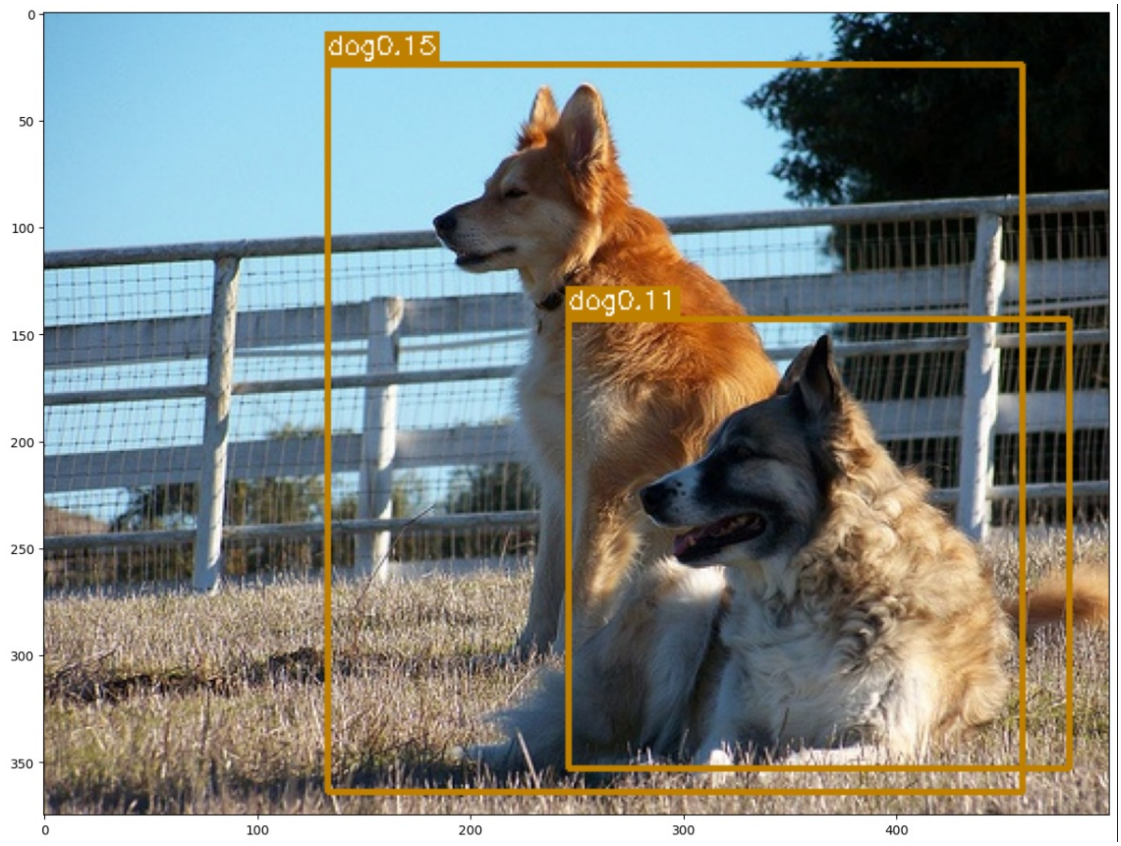
There is an interesting observation on comparing the relative performance of using pre-trained & randomly initialized models in sections (2) & (3).
We notice that the gain in accuracy when only the last two layers are fine-tuned (while freezing weights of other layers) is significant when going from randomly initialized models to pre-trained models. Whereas, when the whole network is trained, the accuracies in the two cases are actually comparable.
We hypothesize that when all (or more) parameters were open to being trained, we were able to find the values of parameters that minimize the loss function irrespective of where we started. This was possible with the Adam optimizer and over 20 epochs. However, when we froze a majority of the parameters, we were working to find the minimum in a subspace of all the parameters (these subspaces being different for the pre-trained and randomly initialized cases) and thus had lesser-than-optimal results vs. training all parameters. Additionally the minima found in the subspace of pre-trained weights turned out to be better than the minima in the subspace of randomly initialized weights, thus a difference in accuracy was observed.

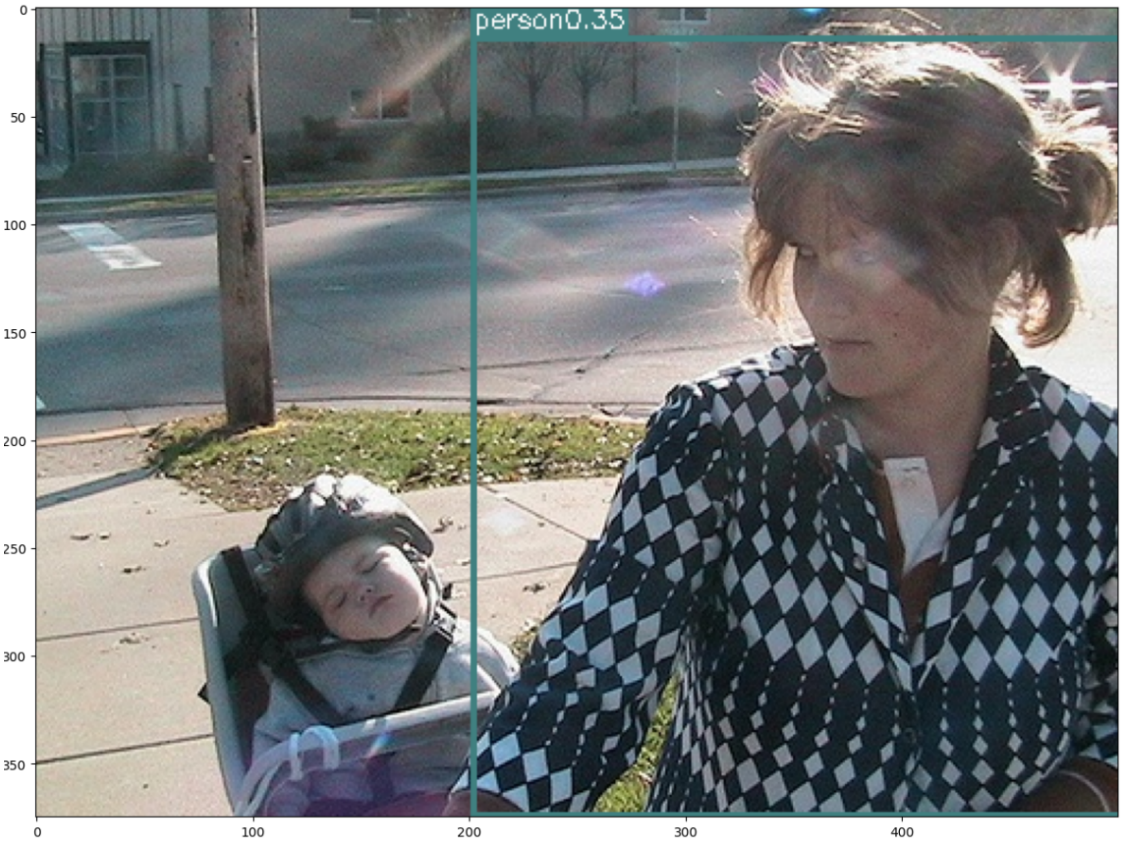
Report the hyperparameters you used to fine-tune your model. Compare the performance between pre-trained model and randomly initialized model. Discuss anything you find interesting comparing fine-tuning the late layers only in section (2) and fine-tuning the whole model in section (3).

4) Extra credit:
We replaced the Resnet18 to a ResNet50, hoping that more parameters might result in better performance.
While keeping the same hyperparameters, the Resnet50 did about the same (slightly lower than ResNet18) when training all layers either from pre-trained or randomly initialized weights (~82% and ~77%).
We then tried altering the decay frequency to every 9 epochs rather than 10, and got an improvement on accuracy, especially for the randomly initialized model - the pretrained model did **82.77%** & the rand_init model did **83.56%** on the test images.
We then also tried to change the hyperparameters for the rotation task, but could not find suitable values that could improve over the ResNet18 model - accuracies obtained on the rotation task were between 72-77%. We also noticed that the Resnet50 architecture did not do well on the classification task when only the last two layers were trained.

Part-2: Object Detection by YOLO

1. My best mAP value on Kaggle : 0.74
2. Did you upload final CSV file on Kaggle: Yes
3. My final loss value :
   Total: 1.158
   Reg: 0.828
   Containing_obj: 0.134
   No_obj: 0.04
   Class: 0.156
4. What did not work in my code(if anything): Code worked, weren't able to train above mAP of 0.294
5. Sample Images from my detector from PASCAL VOC:


dog0.15
dog0.11
0
50
100
150
200
250
300
350
0
100
200
300
400


person0.35
0
50
100
150
200
250
300
350
0
100
200
300
400


0
50
100
150
200
250
300
0
100
200
300
400


bicycle0.28
0
50
100
150
200
250
300
350
0
100
200
300
400


person0.17
0
50
100
150
200
250
300
350
0
100
200
300
400