Assignment 3

Name(s): Dushyant Singh Udawat, Pratikshit Singh
NetID(s): ds35, ps71

Part 1: Self-supervised Learning on CIFAR10

1) Rotation training

Report the hyperparameters you used to train your model. Discuss any particular implementation choices which caused significant performance increases.

We tried a combination of Init_lr = [0.01, 0.001] and optimizers = [Adam , SGD]. That makes it four combinations of hyperparameters tried.
I would say that SGD and Adam , showed similar performance while Adam was slightly better.
init _lr of 0.01 showed fast results in the beginning but it would not get to better accuracy soon because of high value of decay_epoch. Init_lr was slow to increase accuracy but was more accurate in the longer term.

2) Fine-tuning late layers

Report the hyperparameters you used to fine-tune your model. Compare the performance between pre-trained model and randomly initialized model.
We tried a combination of Init_lr = [0.01, 0.001, 0.002] and optimizers = [Adam , SGD]. That makes it six combinations of hyperparameters tried. We finally chose lr = 0.001, and Adam optimizer.
Comparison of performances between pre trained and randomly initialized model -

1. ***Final Accuracy of pre pre-trained model was around 65% where as that of the randomly initialized model was 45%. The reason is that the pretrained model has already learnt to recognize the low level features, while the randomly intialized model never got the chance to do it.***
2. ***The starting accuracy of the pretrained model was 55% which is already pretty decent, (the reason is the same as in point 1), whereas that of the randomly initialized model is 37%.***

3) Fully supervised learning

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).*

We used the following hyperparameters to train the pre-trained model as well as the randomly initialized model -
(lr: [0.01, 0.001], optimizer : [Adam] , decay_epoch: [5, 10] :: 4 total)
(Best hyperparamters - lr = 0.001, decay_epoch=10 :: 78% pretrained , ~82% randomly initialized)

Comparison between pretrained and randomly initialized models -

1. ***A high learning rate in the case of pretrained model is less useful than that in the case of randomly initialized model because the low level features have already been learnt.***
2. ***The randomly initialized model has better performance (accuracy) at the end of 20 epochs than pretrained one. Possible explanation - Maybe it is stuck in the local optima that was corresponding to the the pre trained model's dataset.***
3. ***The randomly initialized model has far less accuracy than the pretrained model. Possible reason - the model has already learnt low level features.***

Comparison between fine-tuning late layers and fine tuning the whole model -

1. ***One huge observation , which is perhaps the target learning of this MP is that the pretrained model has already learnt the low level features that helps it a lot to predict the right classes. It is because of this reason that the accuracy of model when fine tuning only the last layer is way higher than that of the randomly initialized model. In the other case where we train the whole model, the low level features are learnt by the model from scratch and these low level features are better helpful in predicting the classes that pretrained model features because they are based upon the true data rather than augmented data. Because of this , the randomly initialized model has better performance (accuracy) at the end of 20 epochs than pretrained one.***

4) Extra credit

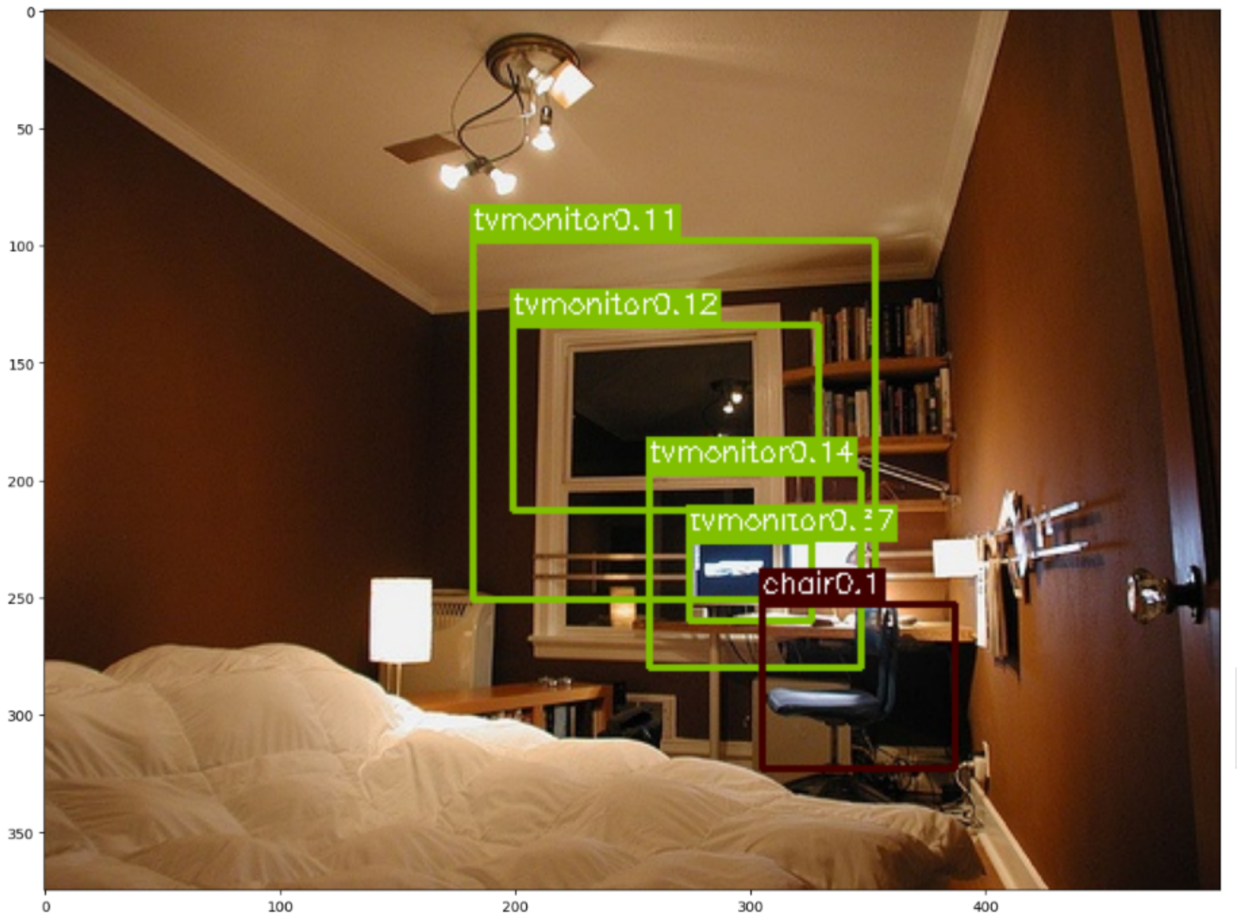
Part-2: Object Detection by YOLO

1. My best mAP value on Kaggle :
   0.48129
2. Did you upload final CSV file on Kaggle: **Yes, Dushyant Singh Udawat**
3. My final loss value :

Test loss - 2.56105
Train loss - 1.609,

4. What did not work in my code(if anything):
   I think Everything worked fine in my code.
5. Sample Images from my detector from PASCAL VOC:

```
predicting...
<matplotlib.image.AxesImage at 0x7ff149845310>
```



Extra Credit for YOLO :