Assignment 3

Name(s): Chiranjeevi Konduru, Rishabh Agarwal
NetID(s): konduru4, ra26

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.

Train a ResNet18 on the rotation task

Initially I was using the learning rate of 0.001 and optimizer as Adam but the accuracy was very low. So we increased the learning rate but still we were not getting the required accuracy. ***Then we used the SGD with 0.1 learning rate and got the required accuracy.*** *So SGD performed very well and it showed the significant performance increase.*

Optimizer	*Learning Rate*	*Decay epochs*	*Weight decay*	*Accuracy*
Adam	*0.001*	*10*	*5e-4*	*74.04%*
Adam	*0.01*	*15*	*1e-3*	*76.09%*
SGD	*0.001*	*10*	*5e-5*	*76.98%*
SGD	*0.1*	*15*	*5e-4*	*81.51%*

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.

For pre-trained model:

Optimizer	*Learning Rate*	*Decay epochs*		*Accuracy*
Adam	*0.001*	*15*		*47.32%*

Adam	0.01	10		50.21%
SGD	0.001	15		50.32%
SGD	0.1	10		57.34%

For randomly initialize model:

Optimizer	Learning Rate	Decay epochs		Accuracy
Adam	0.001	15		40.12%
Adam	0.01	10		42.21%
SGD	0.001	15		41.98%
SGD	0.1	10		44.53%

Clearly the pre-trained model has performed well and randomly initialize model is not performing well. We got only 44.53% of accuracy for the randomly initialize model.

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).

For pre-trained model:

Optimizer	Learning Rate	Decay epochs		Accuracy
Adam	0.001	5		64.65%
Adam	0.01	4		65.43%
SGD	0.001	10		74.23%
SGD	0.1	4		76.31%

For randomly initialize model:

Optimizer	Learning Rate	Decay epochs		Accuracy
Adam	0.001	15		72.76%
Adam	0.01	10		74.12%

SGD	*0.001*	*15*		*76.32%*
SGD	*0.1*	*10*		*78.81%*

Since we only tuned the last layer in the section 2 the accuracy was very low for both the task but when we re-trained the whole model the accuracy increased drastically. The interesting thing that we find is that even though we used the rotation model and trained only the last layer still we were able to achieve the accuracy of 57.34% and 44.53%. When we retrained the whole rotation model for the classification dataset our accuracy increased but the model with randomly initialised weights performed well.

4) Extra credit

b) Use a more advanced model than ResNet18 to try to get higher accuracy on the rotation prediction task, as well as for transfer to supervised CIFAR10 classification.

For rotation prediction task:

We used resnet50 with the parameters : num_epochs=45, decay_epochs=5, init_lr=0.01, task='rotation' and got the accuracy of 84.89% which is higher than resnet18 (81.51%).

For Classification task:

We used resnet50 with the parameters : num_epochs=20, decay_epochs=4, init_lr=0.1, task='classification' and got the accuracy of 68.70% which is higher than resnet18(57.43%).

c) If you have a good amount of compute at your disposal, try to train a rotation prediction model on the larger ImageNette dataset (still smaller than ImageNet, though).

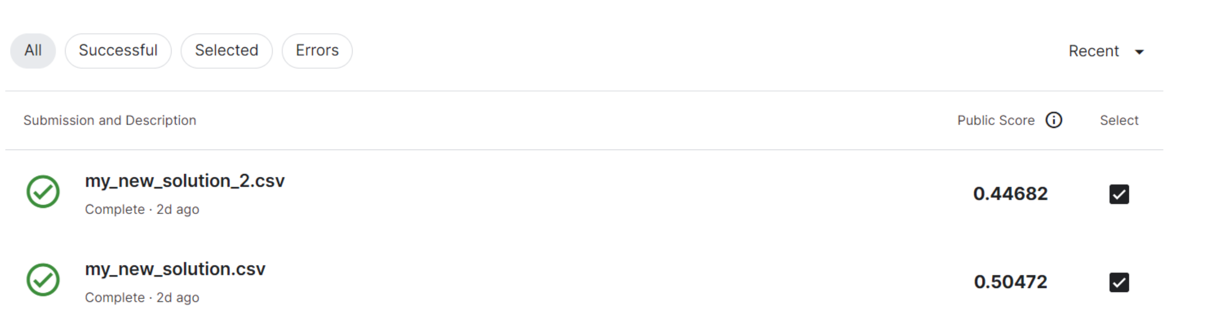
We have trained Imagenette and Imagenette-160. Since we have less compute resources so we used the Imagenette-160 and trained it on resnet50 model and we got the accuracy of 95.91.

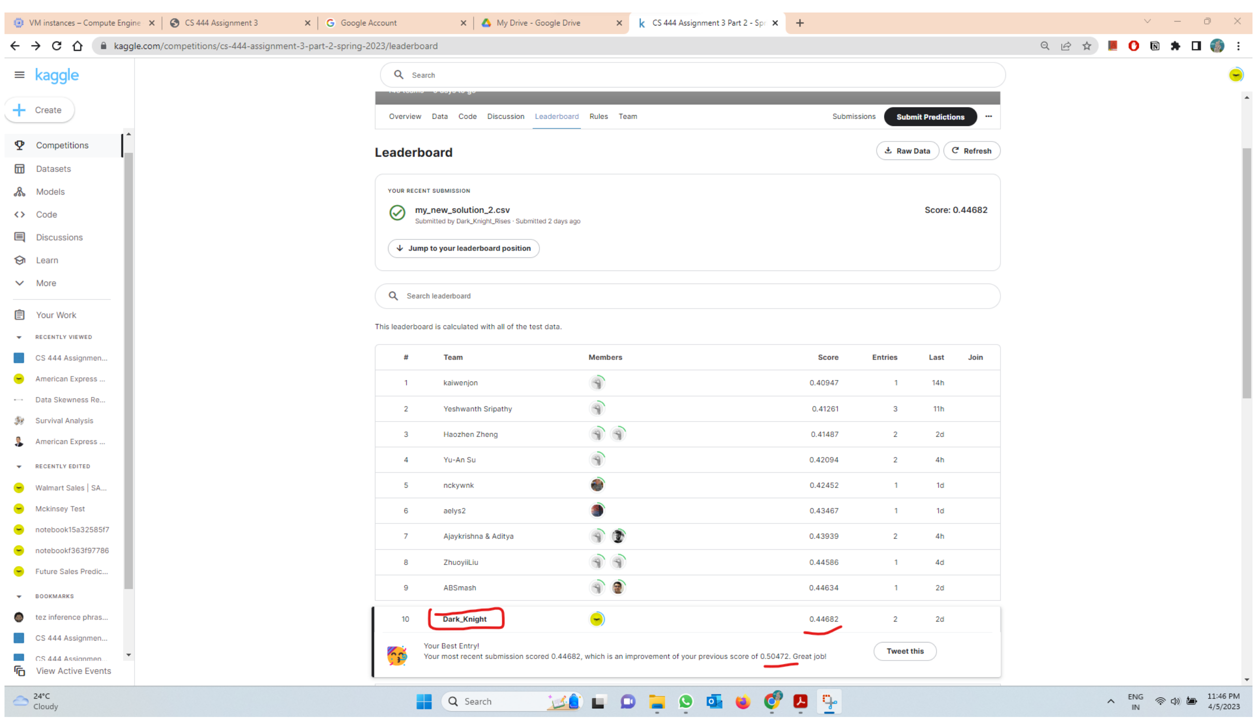
For the Imagenette data set we have written the complete code but due to lack of resources only 2 epochs were done and after 2 epochs our accuracy was 64.23%.

Part-2: Object Detection by YOLO

1. **My best mAP value on Kaggle :**
   0.44682 (Best mAP using Adam Optimizer).
   0.50472 (using SGD Optimizer)

2. **Did you upload final CSV file on Kaggle:**
   [Yes](#).



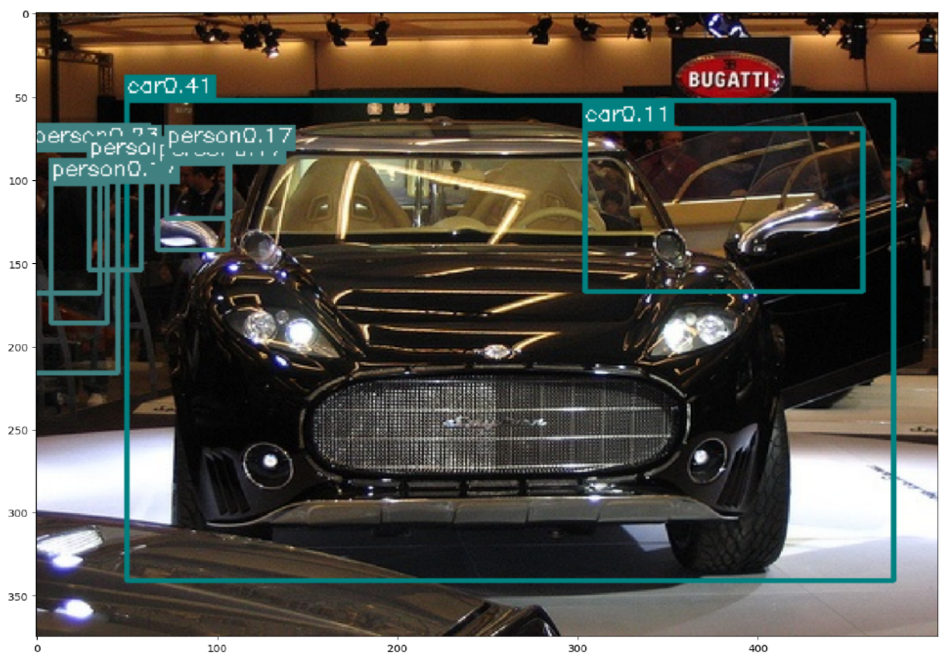


3. **My final loss value :**

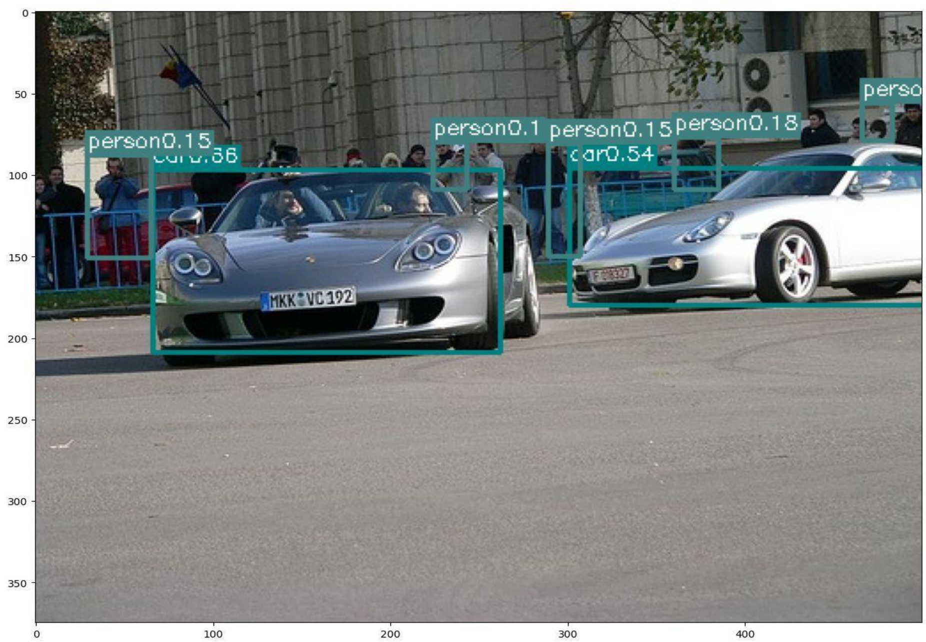
The best total loss is **2.69256** (Pg 23 in part 2 report output PDF). This is for SGD optimizer, however to get better predictions I used **Adam** optimizer with **1e^{-4}** lr starting from the previous checkpoint (49-th Epoch of SGD).

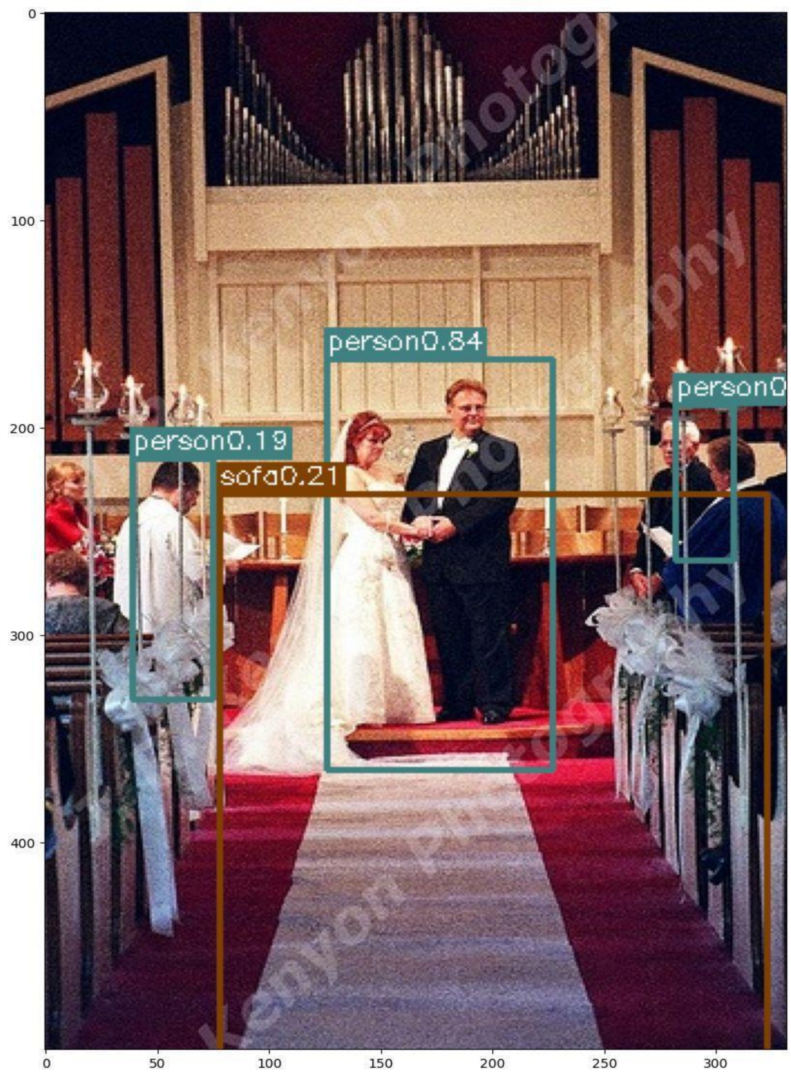
4. **What did not work in my code(if anything):**
Code worked properly, but initially I got low mAP values and convergence of total loss (plateau problem) took more time. Sometimes, initial test sample evaluations gave me 'no cls predictions', but later on after 30 epochs I was able to get some decent mAP values.

5. **Sample Images from my detector from PASCAL VOC:**
Check below the results of resnet50 model.




person0.71
0.83
person0.14
person0.2
0
50
100
150
200
250
300
350
400


person0.15
person0.1
person0.15
person0.18
car0.54
MKK VC 192
0
50
100
150
200
250
300
350
400


person0.84
person0.19
sofa0.21
person0
0
100
200
300
400
0
50
100
150
200
250
300

Extra Credit for YOLO :

> Try to replace the provided pre-trained network with a different one and train with the YOLO loss on top to attempt to get better accuracy.

I tried pretrained “***maskrcnn_resnet50_fpn_v2***” from torchvision.models.detections on top of my YOLO loss criteria. The initial mAPs were all 0s. The code files related to this extra credit can be found inside the extra_credit folder. To compare the performance, the initial resnet50() gave us better results than this maskrcnn. We’ve tried scheduled decaying and other learning_rate decay techniques by increasing the epochs (70), but still it didn’t perform as good as the initial model. I guess it should’ve been due to a plateau problem. We’ve also tried Retinnanet, Resnet152, and fasterrcnn, which reached the mAP value of 0.22 after 50 epochs. The former models didn’t perform well and the maximum mAP value it could achieve was 0.007. The checkpoints for these models were also added to the extra credit folder. The .ipynb file and yolo_resnet.py file, where the pretrained models were invoked are added in the folder ‘*mp3_part2_extra_credit*’. The .ipynb file here, will only contain the results of the latest model maskrcnn.