Assignment 3

Name(s): Girish Madhavan Venkataramani
NetID(s): gmv4
Team name on Kaggle leaderboard: Girish

For each of the sections below, your reported test accuracy should approximately match the accuracy reported on Kaggle. It is fine to have a small deviation. Report both accuracies if they are different.

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.

- Kaggle score : **79.49%**
- Notebook score : **79.36%**
- Hyperparameter setting :
    - Adam optimizer, lr = 0.001
    - Decay Epochs = 15
    - Total Epochs = 50

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.

- Kaggle score : **63.72%**
- Notebook score :
    - **63.72%** for model with pretraining
    - **42.31%** for randomly initialized model
- Hyperparameter setting : [Both cases with and without pretraining]
    - Adam Optimizer, lr = 0.01
    - Decay Epochs = 10
    - Total Epochs = 20

The pre-trained model gave better accuracy as compared to randomly initialized one. This happens because of freezing all the layers except two layers, giving the pretrained model a better chance to classify the images correctly as the model parameters have been exposed to the dataset, while in the randomly initialized case, the weights have no relation to the dataset

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).

- Kaggle score : **80.68%**
- Notebook score :
    - **80.68%** for model with pretraining
    - **80.10%** for randomly initialized model
- Hyperparameter setting : same as task 2

This time the accuracy between both the models are approximately the same, because of not freezing any layer, both models are allowed to adapt to the data during the training process

4) With a more advanced architecture

Report the details of your architecture and the hyperparameters used to train this model. Compare the performance between this model and the models in sections (1) and (3).

Additional Data augmentation : Gaussian blur and Color Jitter applied randomly

Rotation Classification : **Resnet34**

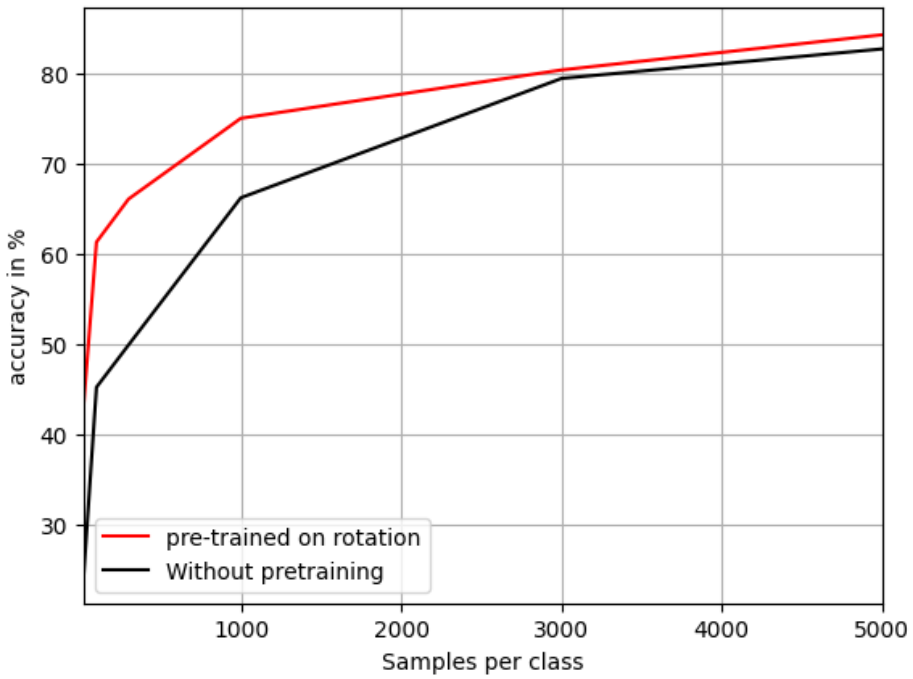
- Kaggle score : **86.32%**
- Notebook score : **85.97%**
- Hyperparameter setting :
    - Adam optimizer with weight decay
        - Lr = 0.0001
        - Weight decay = 1e-6
    - Lr scheduler : Cosine Annealing
        - Minimum lr = 1e-6
        - Transition time for decay = 30 epochs
    - Total epochs : ~130
- As compared to task 1, this gives an improvement of about **6%**

Image Classification : **Efficientnet v2 S**

- Kaggle score : **84.42%**
- Notebook score : **84.42%**
- Hyperparameter setting :
    - Adam optimizer with weight decay
        - Lr = 0.0001
        - Weight decay = 0.5 * 1e-5
    - Lr scheduler : Cosine Annealing
        - Minimum lr = 1e-7
        - Transition time for decay = 30 epochs
    - Total epochs : ~130
- As compared to task 3, this implementation gives about **3.74%** improvement

5) Extra credit

Dependence of model accuracy on varying amounts of samples for each label between randomly initialized and fine tuning on pretrained model



- Number of samples used per label : [20, 100, 300, 1000, 3000, 5000]
- A pattern similar to one mentioned by **Gidaris et al.** is observed, ie for lesser samples per class, the pre-trained model outperforms a randomly initialized model. But as the samples per class keep increasing, both models perform nearly the same
- For the fine-tuning process, the model from task 4a [**resnet34**] was used because of its higher accuracy. **Resnet34** is the model used in this case for with & without pretraining

ImageNette Rotation Classification :

- Network Architecture : **Efficientnet b2**
- Model Accuracy : **87.4%** after ~150 epochs of training, training stops when certain accuracy is achieved
- Model was trained on images from ImageNette dataset that were cropped down to 150 x 150 pixels from their original size to prevent memory overflow for GPU