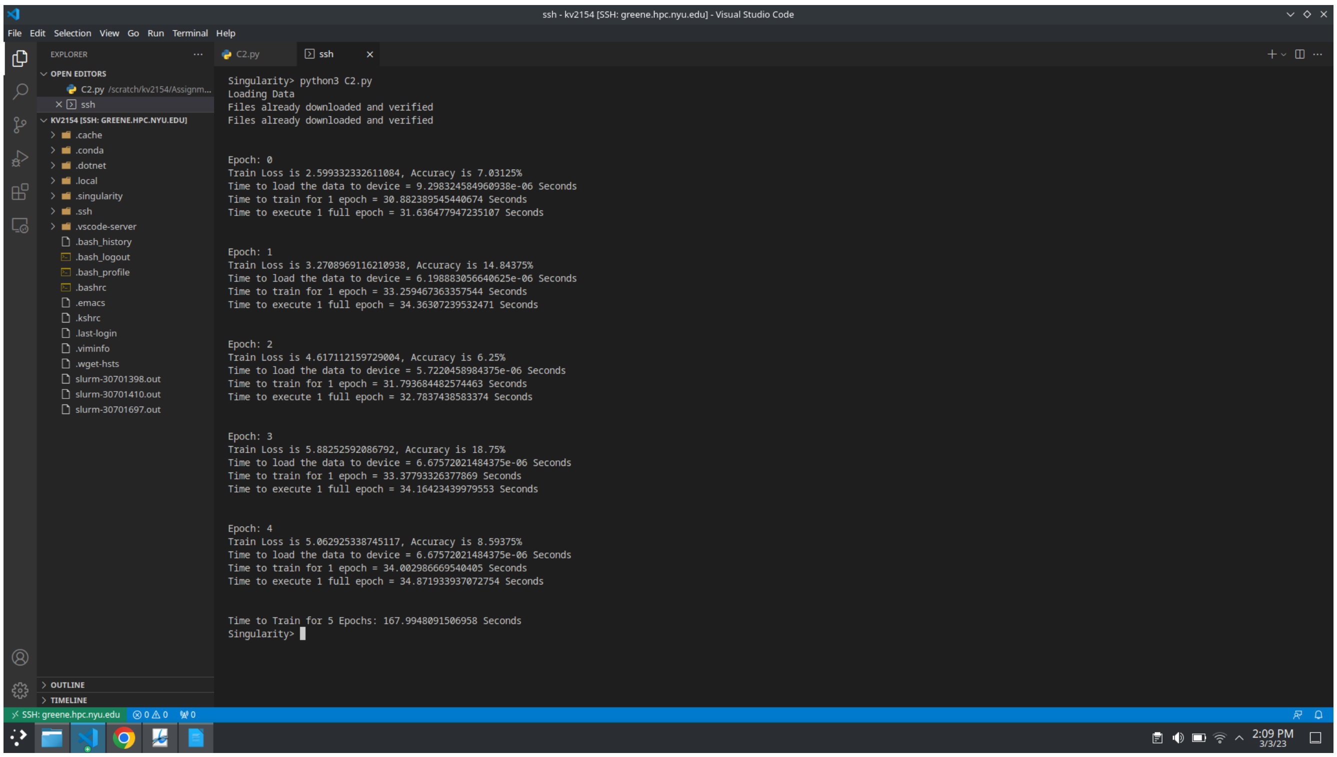
Karan Vora (kv2154)
ECE-GY 9143 Introduction to High-Performance Machine Learning Assignment 2

C1

The Code file is attached with the Final submission

==

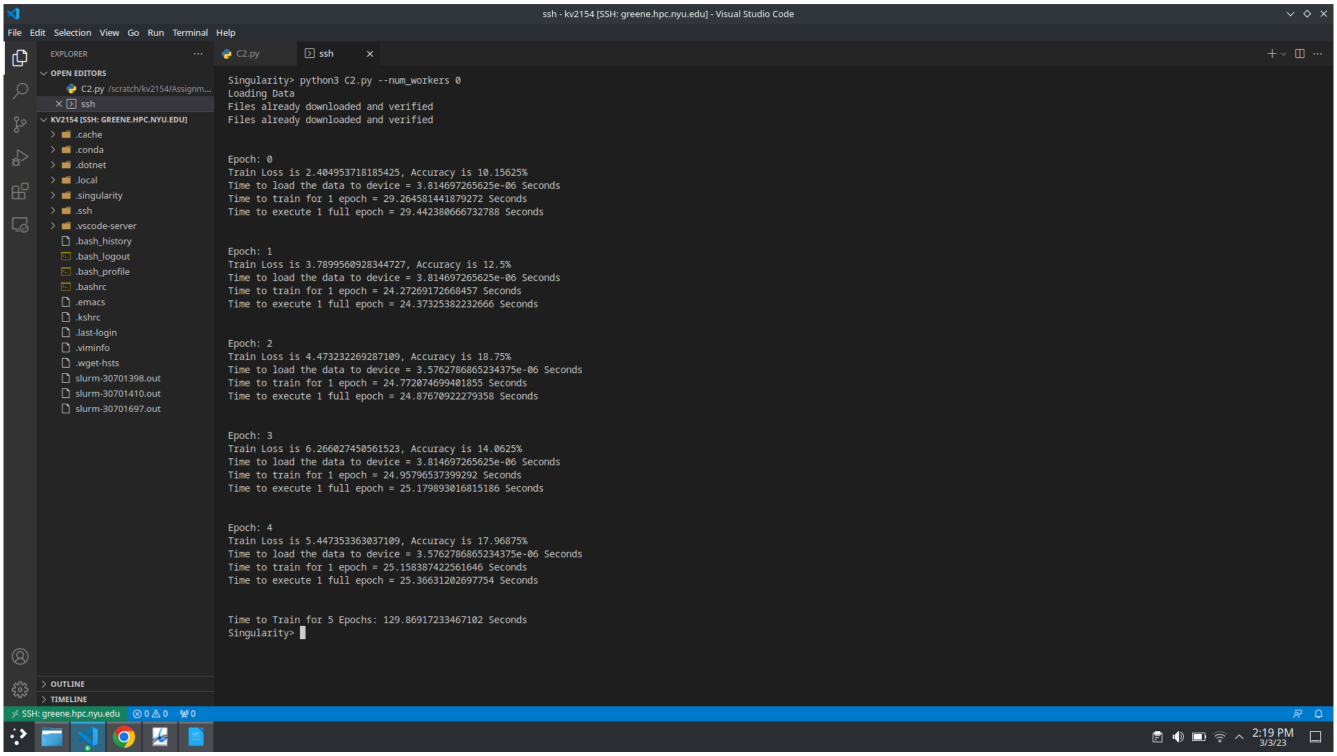
C2



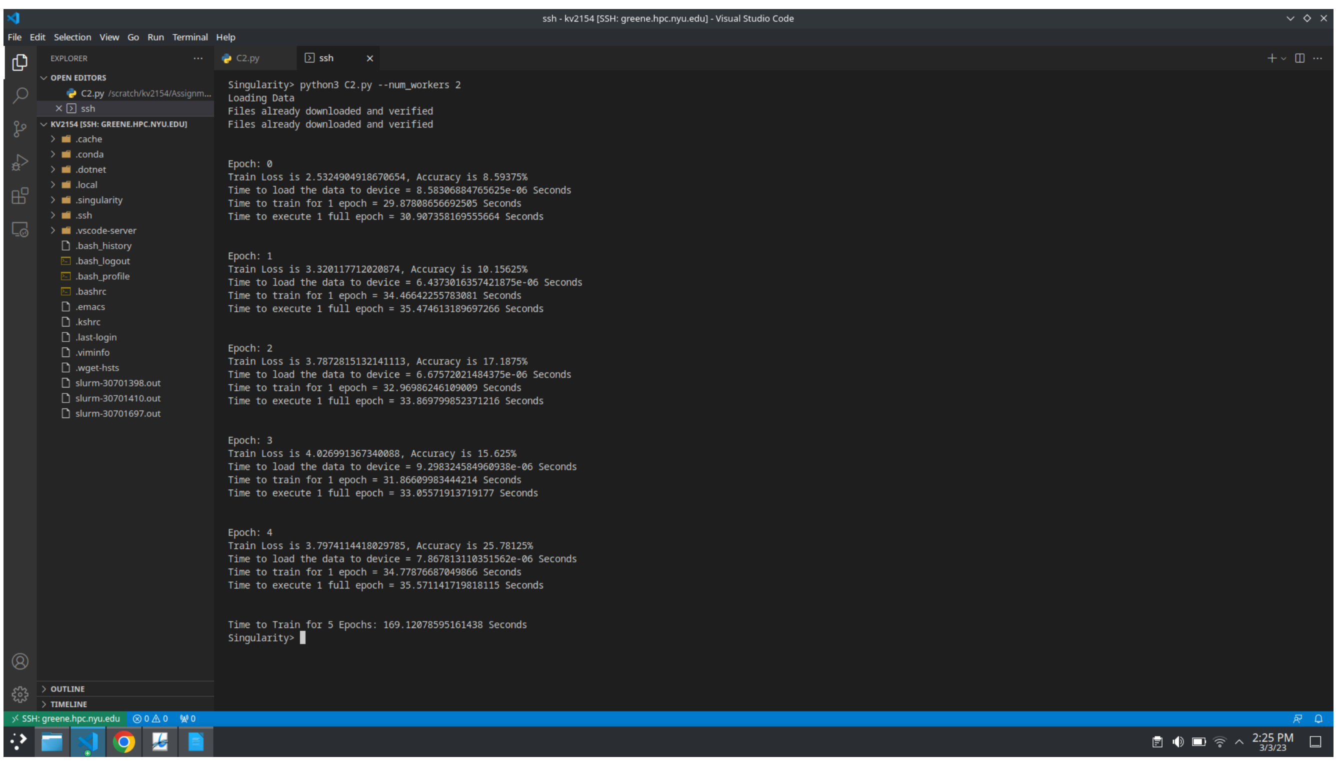
==

C3

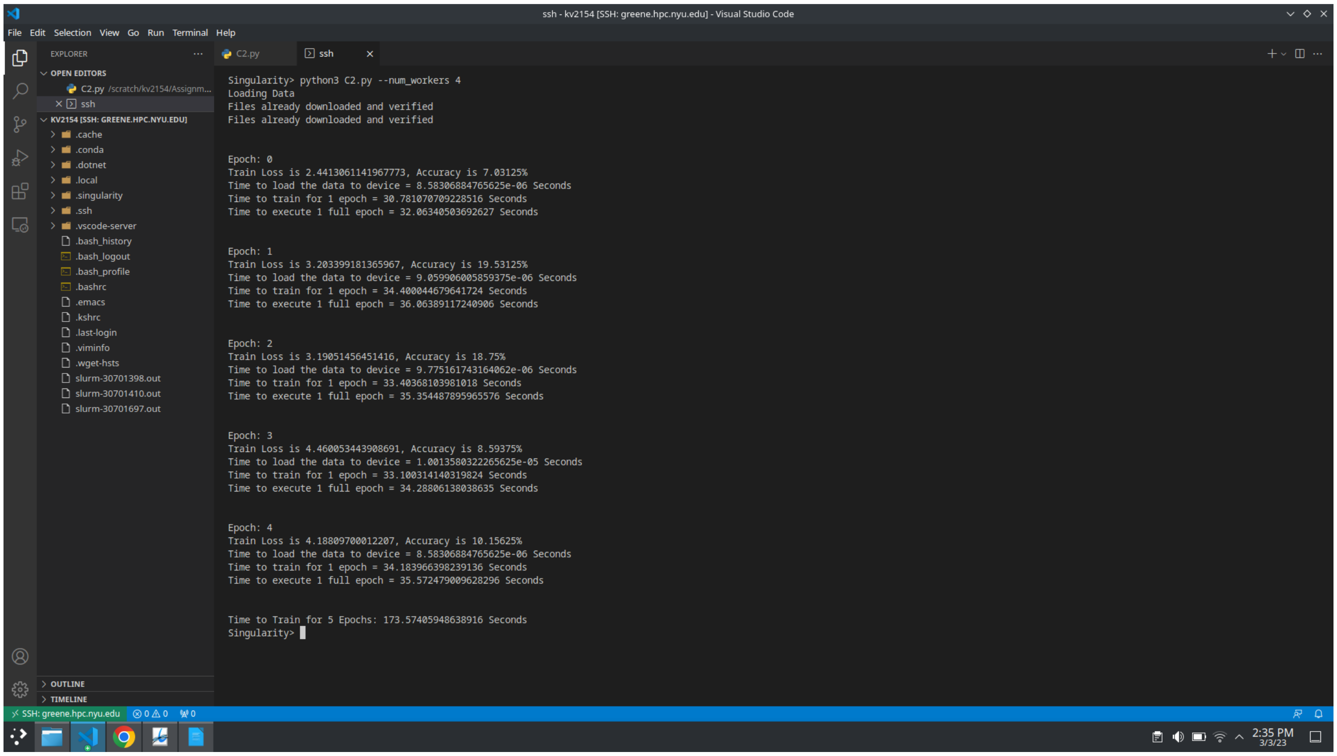
0 Workers



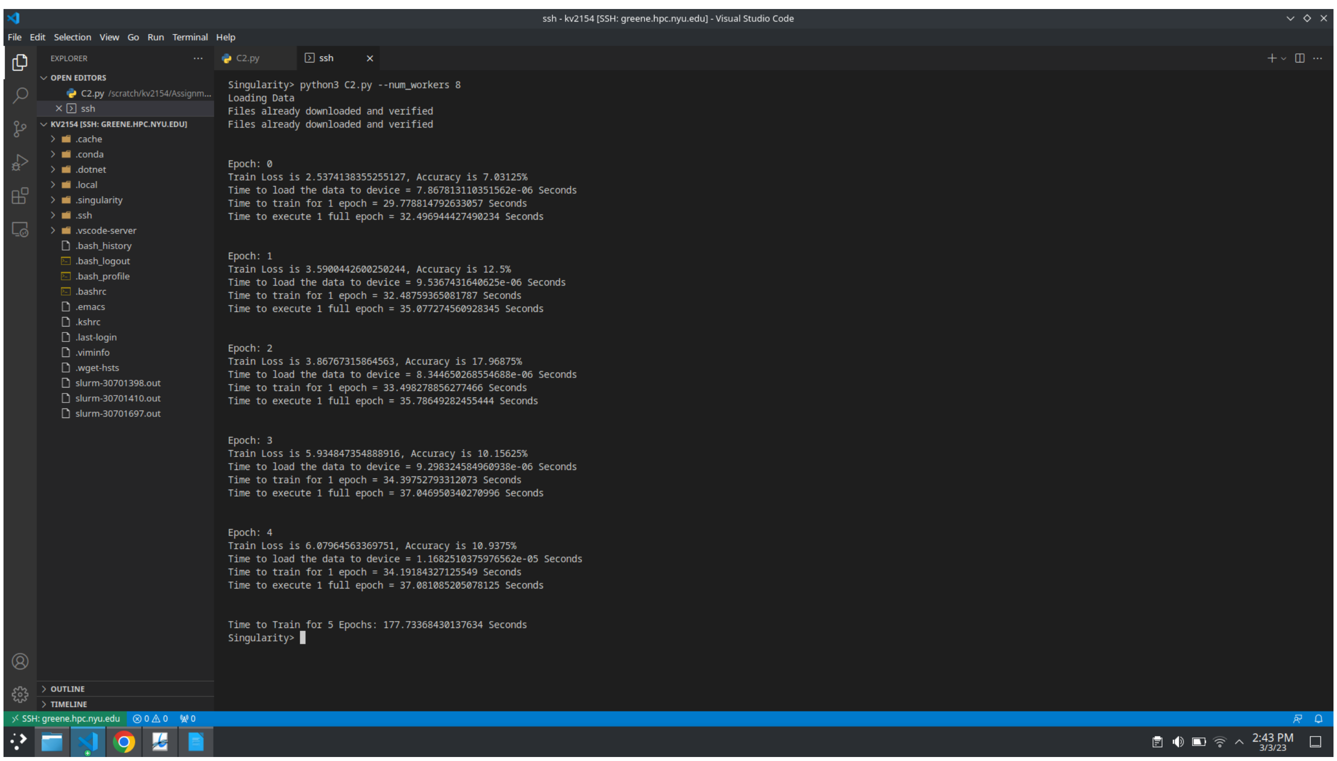
2 Workers



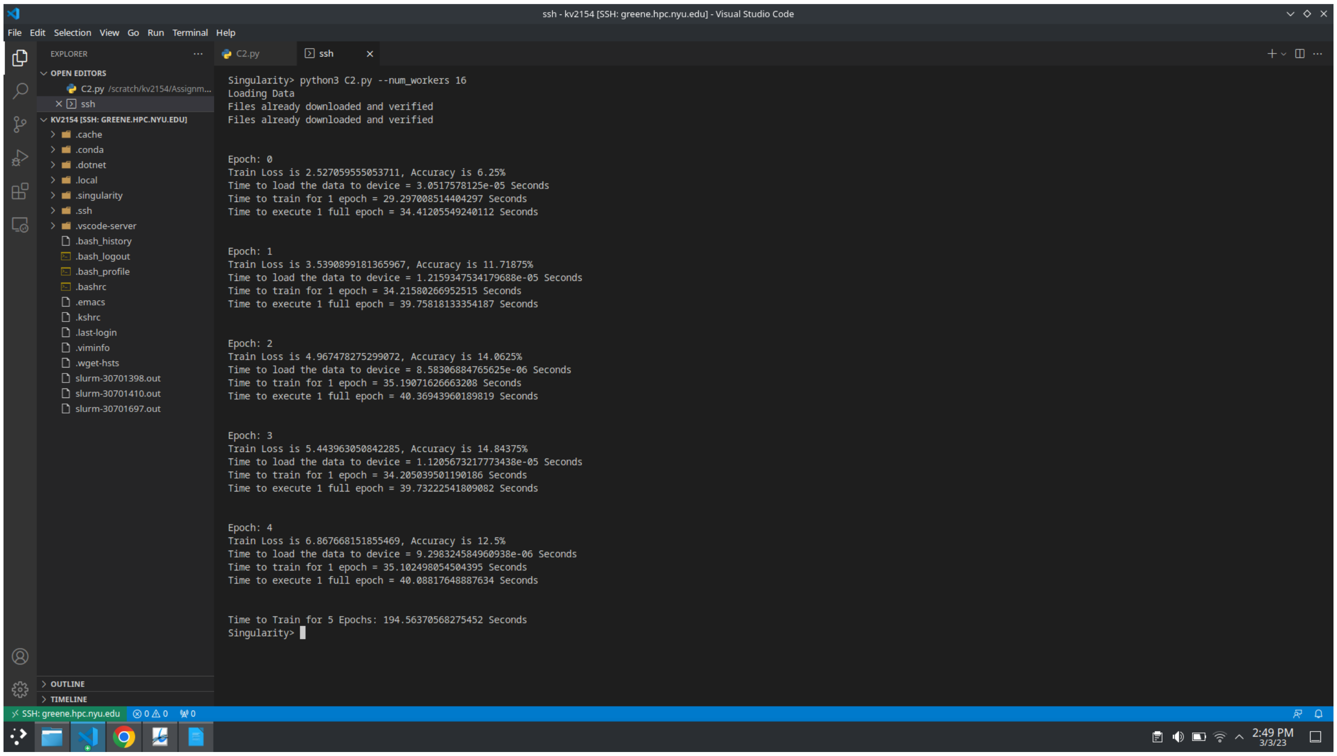
4 Workers



8 Workers



16 Workers

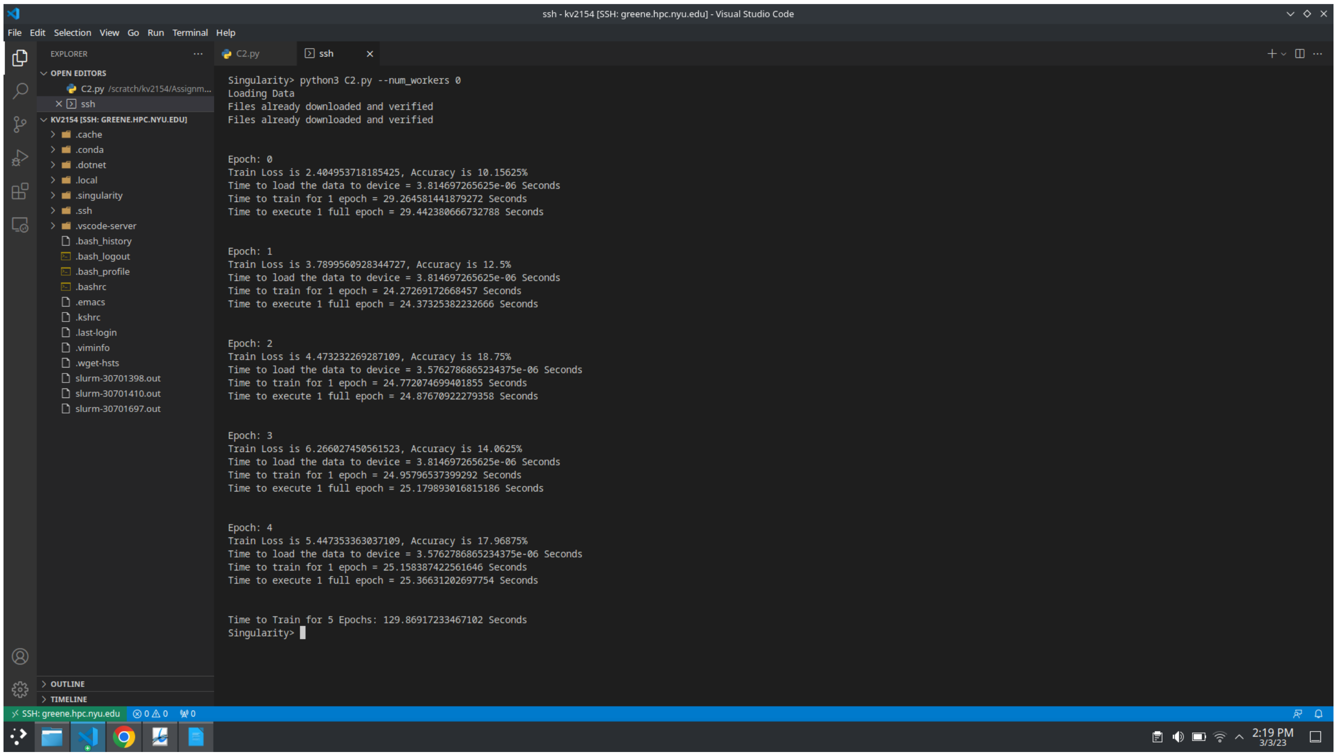


From the above mentioned results we can see that we get best results at 0 Workers

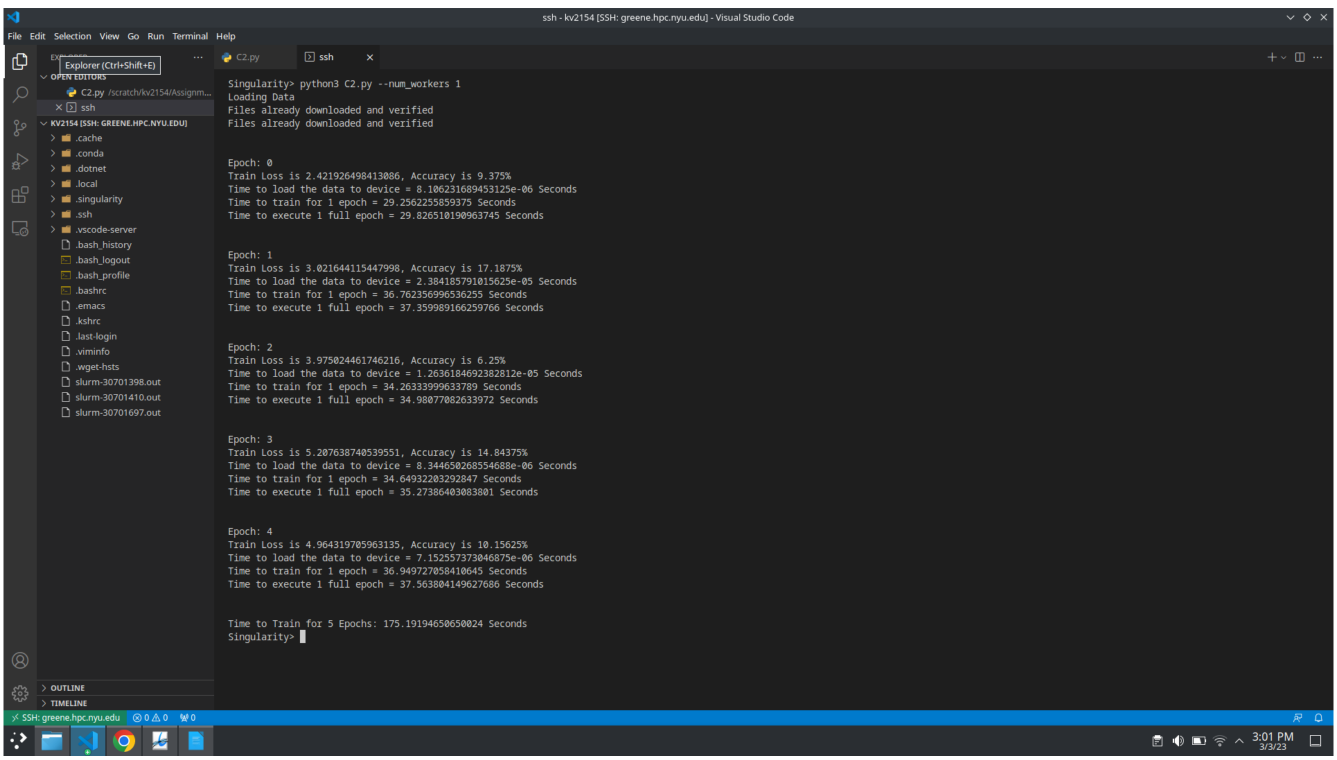
===

C4

At 0 Workers



At 1 Worker

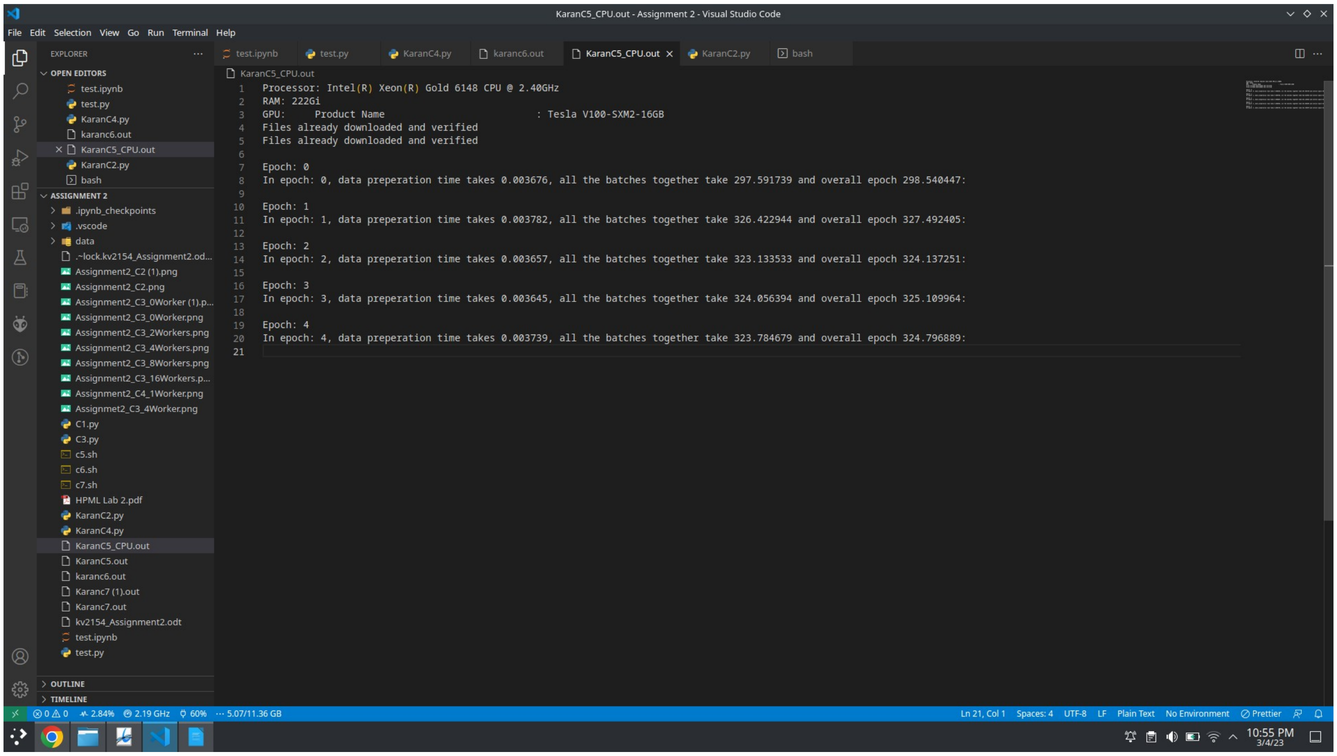


From the above mentioned output, we can see that code with 0 workers run faster than code with 1 worker by a significant amount

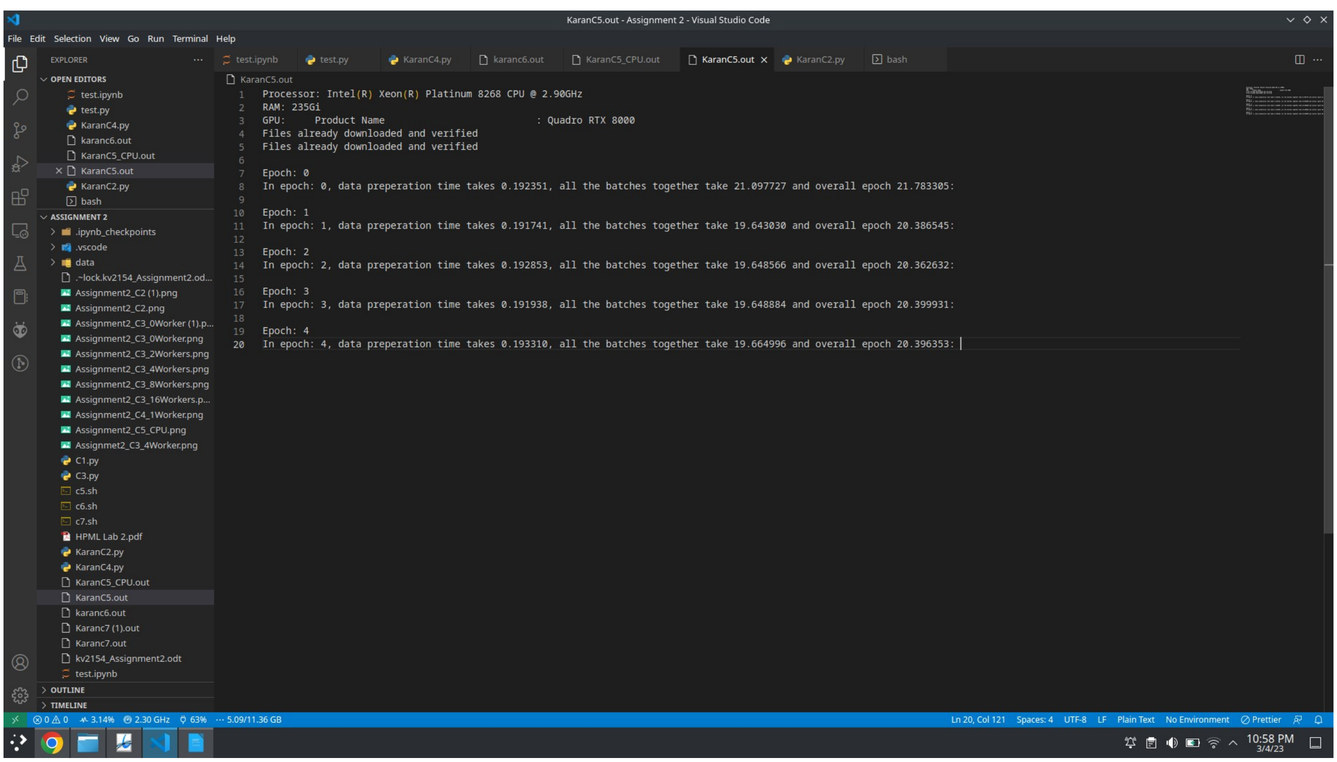
==

C5

CPU:


KaranC5_CPU.out - Assignment 2 - Visual Studio Code
Processor: Intel(R) Xeon(R) Gold 6148 CPU @ 2.40GHz
RAM: 222Gi
GPU: Product Name   : Tesla V100-SXM2-16GB
Files already downloaded and verified
Files already downloaded and verified
Epoch: 0
In epoch: 0, data preperation time takes 0.003676, all the batches together take 297.591739 and overall epoch 298.540447:
Epoch: 1
In epoch: 1, data preperation time takes 0.003782, all the batches together take 326.422944 and overall epoch 327.492405:
Epoch: 2
In epoch: 2, data preperation time takes 0.003657, all the batches together take 323.133533 and overall epoch 324.137251:
Epoch: 3
In epoch: 3, data preperation time takes 0.003645, all the batches together take 324.056394 and overall epoch 325.109964:
Epoch: 4
In epoch: 4, data preperation time takes 0.003739, all the batches together take 323.784679 and overall epoch 324.796889:

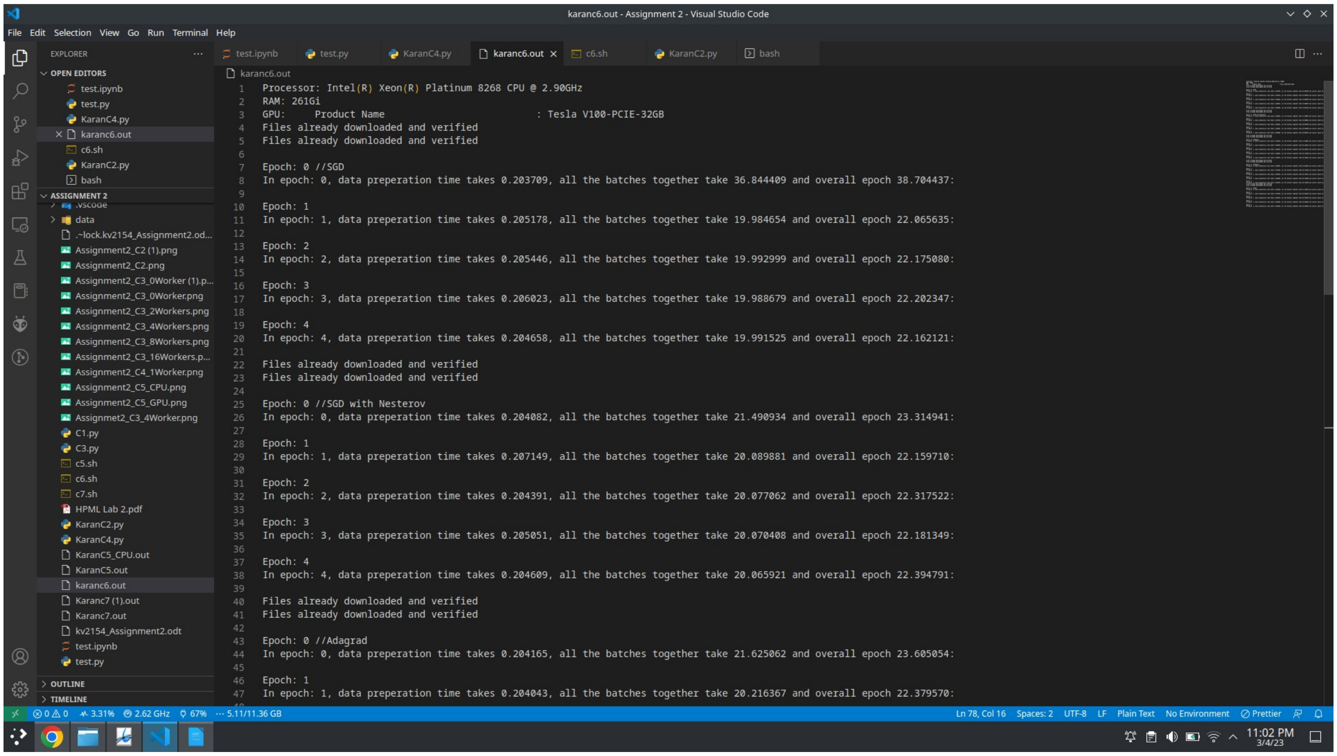
GPU:

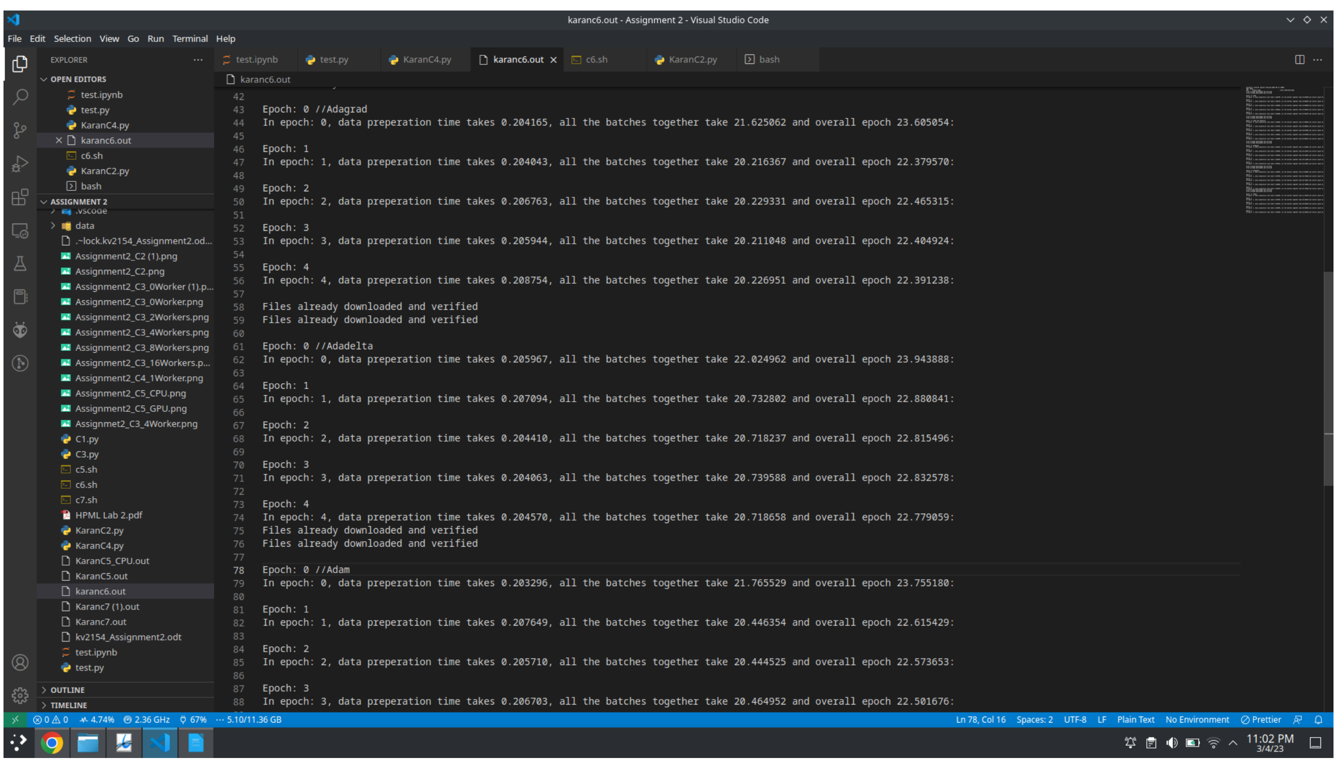

KaranC5.out - Assignment 2 - Visual Studio Code
Processor: Intel(R) Xeon(R) Platinum 8268 CPU @ 2.90GHz
RAM: 235Gi
GPU: Product Name   : Quadro RTX 8000
Files already downloaded and verified
Files already downloaded and verified
Epoch: 0
In epoch: 0, data preperation time takes 0.192351, all the batches together take 21.097727 and overall epoch 21.783305:
Epoch: 1
In epoch: 1, data preperation time takes 0.191741, all the batches together take 19.643030 and overall epoch 20.386545:
Epoch: 2
In epoch: 2, data preperation time takes 0.192853, all the batches together take 19.648566 and overall epoch 20.362632:
Epoch: 3
In epoch: 3, data preperation time takes 0.191938, all the batches together take 19.648884 and overall epoch 20.399931:
Epoch: 4
In epoch: 4, data preperation time takes 0.193310, all the batches together take 19.664996 and overall epoch 20.396353:

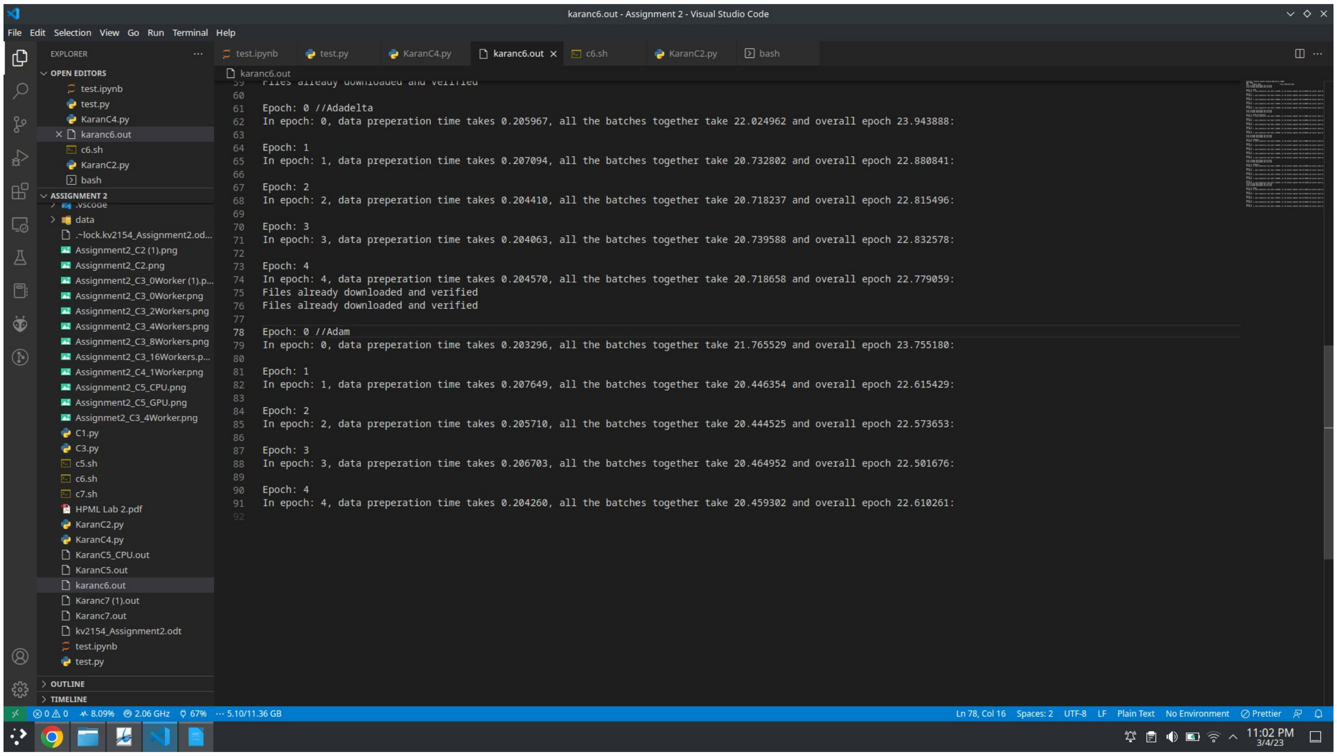
From the above mentioned results we can see that the average time to run on CPU is approximately 300 seconds while on GPU is approximately 20 seconds per epoch.

==

C6

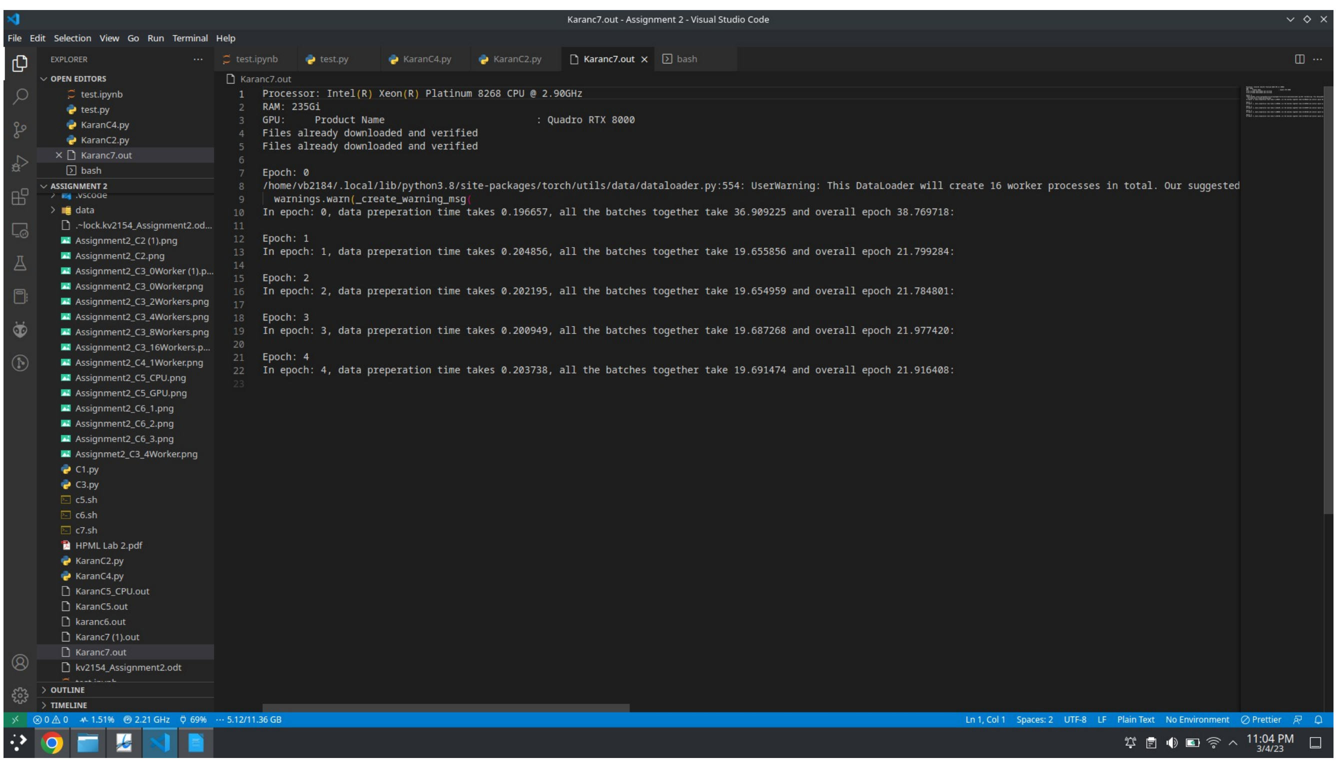





karanc6.out - Assignment 2 - Visual Studio Code
Epoch: 0 //Adadelta
In epoch: 0, data preperation time takes 0.205967, all the batches together take 22.024962 and overall epoch 23.943888:
Epoch: 1
In epoch: 1, data preperation time takes 0.207094, all the batches together take 20.732802 and overall epoch 22.880841:
Epoch: 2
In epoch: 2, data preperation time takes 0.204410, all the batches together take 20.718237 and overall epoch 22.815496:
Epoch: 3
In epoch: 3, data preperation time takes 0.204063, all the batches together take 20.739588 and overall epoch 22.832578:
Epoch: 4
In epoch: 4, data preperation time takes 0.204570, all the batches together take 20.718658 and overall epoch 22.779059:
Files already downloaded and verified
Files already downloaded and verified
Epoch: 0 //Adam
In epoch: 0, data preperation time takes 0.203296, all the batches together take 21.765529 and overall epoch 23.755180:
Epoch: 1
In epoch: 1, data preperation time takes 0.207649, all the batches together take 20.446354 and overall epoch 22.615429:
Epoch: 2
In epoch: 2, data preperation time takes 0.205710, all the batches together take 20.444525 and overall epoch 22.573653:
Epoch: 3
In epoch: 3, data preperation time takes 0.206703, all the batches together take 20.464952 and overall epoch 22.501676:
Epoch: 4
In epoch: 4, data preperation time takes 0.204260, all the batches together take 20.459302 and overall epoch 22.610261:

==

C7


Karanc7.out - Assignment 2 - Visual Studio Code
Processor: Intel(R) Xeon(R) Platinum 8268 CPU @ 2.90GHz
RAM: 235Gi
GPU: Product Name : Quadro RTX 8000
Files already downloaded and verified
Files already downloaded and verified
Epoch: 0
/home/vb2184/.local/lib/python3.8/site-packages/torch/utils/data/dataloader.py:554: UserWarning: This DataLoader will create 16 worker processes in total. Our suggested
warnings.warn(_create_warning_msg(
In epoch: 0, data preperation time takes 0.196657, all the batches together take 36.909225 and overall epoch 38.769718:
Epoch: 1
In epoch: 1, data preperation time takes 0.204856, all the batches together take 19.655856 and overall epoch 21.799284:
Epoch: 2
In epoch: 2, data preperation time takes 0.202195, all the batches together take 19.654959 and overall epoch 21.784801:
Epoch: 3
In epoch: 3, data preperation time takes 0.200949, all the batches together take 19.687268 and overall epoch 21.977420:
Epoch: 4
In epoch: 4, data preperation time takes 0.203738, all the batches together take 19.691474 and overall epoch 21.916408:

==

Q1

There are a total of 16 convolutional layers in the ResNet-18 model. These layers are arranged in a series of residual blocks, each containing two or three convolutional layers. Specifically, there are four stages of residual blocks, each with a different number of convolutional layers:

→ Stage 1: Contains one residual block with two convolutional layers

→ Stage 2: Contains two residual blocks, each with two convolutional layers

→ Stage 3: Contains two residual blocks, each with three convolutional layers

→ Stage 4: Contains two residual blocks, each with three convolutional layers

==

Q2

The last linear layer in ResNet-18 is a fully connected layer that is used to map the output of the previous layer to the number of classes in the classification task. In the case of ResNet-18, which is typically used for image classification on the ImageNet dataset, there are 1000 classes. Therefore, the input dimension of the last linear layer is 512, which is the number of features output by the preceding layer, and the output dimension is 1000, which is the number of classes to be predicted. So the input dimension of the last linear layer in ResNet-18 is 512.

==

Q3

ResNet-18 is a convolutional neural network architecture that is commonly used in image classification tasks. The number of trainable parameters and gradients in the ResNet-18 model depends on the specific implementation, but I will assume that we are using a standard implementation of ResNet-18 and the SGD optimizer.

First, let's define some terms:

- Trainable parameters: These are the parameters in the model that are learned during training. They include the weights and biases of the convolutional layers, fully connected layers, and any other learnable layers in the model.

- Gradients: These are the derivatives of the loss function with respect to the trainable parameters. They are used by the optimizer to update the parameters during training.

ResNet-18 consists of 18 layers, including 16 convolutional layers and 2 fully connected layers. The number of trainable parameters and gradients in each layer is as follows:

1. Convolutional layer with 64 filters, kernel size 3x3, and stride 1x1. Trainable parameters: 64 x 3 x 3 x 3 + 64 = 1,792 Gradients: Same as the number of trainable parameters.
2. Batch normalization layer with 64 channels. Trainable parameters: 128 (64 scales and 64 biases) Gradients: Same as the number of trainable parameters.
3. ReLU activation layer.
4. Convolutional layer with 64 filters, kernel size 3x3, and stride 1x1. Trainable parameters: 64 x 64 x 3 x 3 + 64 = 36,928 Gradients: Same as the number of trainable parameters.
5. Batch normalization layer with 64 channels. Trainable parameters: 128 (64 scales and 64 biases) Gradients: Same as the number of trainable parameters.
6. ReLU activation layer.
7. Convolutional layer with 64 filters, kernel size 3x3, and stride 1x1. Trainable parameters: 64 x 64 x 3 x 3 + 64 = 36,928 Gradients: Same as the number of trainable parameters.
8. Batch normalization layer with 64 channels. Trainable parameters: 128 (64 scales and 64 biases) Gradients: Same as the number of trainable parameters.
9. ReLU activation layer.
10. Convolutional layer with 128 filters, kernel size 3x3, and stride 2x2. Trainable parameters: 64 x 128 x 3 x 3 + 128 = 73,856 Gradients: Same as the number of trainable parameters.
11. Batch normalization layer with 128 channels. Trainable parameters: 256 (128 scales and 128 biases) Gradients: Same as the number of trainable parameters.
12. ReLU activation layer.
13. Convolutional layer with 128 filters, kernel size 3x3, and stride 1x1. Trainable parameters: 128 x 128 x 3 x 3 + 128 = 147,584 Gradients: Same as the number of trainable parameters.
14. Batch normalization layer with 128 channels. Trainable parameters: 256 (128 scales and 128 biases) Gradients: Same as the number of trainable parameters.
15. ReLU activation layer.
16. Convolutional layer with 128 filters, kernel size 3x3, and stride 1x1. Trainable parameters: 128

x 128 x 3 x 3 + 128 = 147,584 Gradients: Same as the number of trainable parameters.

17. Batch normalization layer with 128 channels. Trainable parameters: 256 (128 scales and 128 biases) Gradients: Same as the number of trainable parameters.

18. ReLU activation layer.

19. Convolutional layer with 256 filters, kernel size 3x3, and stride 2x2. Trainable parameters: 128 x 256 x 3 x 3 + 256 = 295,168 Gradients: Same as the number of trainable parameters.

20. Batch normalization layer with 256 channels. Trainable parameters: 512 (256 scales and 256 biases) Gradients: Same as the number of trainable parameters.

21. ReLU activation layer.

22. Convolutional layer with 256 filters, kernel size 3x3, and stride 1x1. Trainable parameters: 256 x 256 x 3 x 3 + 256 = 590,080 Gradients: Same as the number of trainable parameters.

23. Batch normalization layer with 256 channels. Trainable parameters: 512 (256 scales and 256 biases) Gradients: Same as the number of trainable parameters.

24. ReLU activation layer.

25. Convolutional layer with 256 filters, kernel size 3x3, and stride 1x1. Trainable parameters: 256 x 256 x 3 x 3 + 256 = 590,080 Gradients: Same as the number of trainable parameters.

26. Batch normalization layer with 256 channels. Trainable parameters: 512 (256 scales and 256 biases) Gradients: Same as the number of trainable parameters.

27. ReLU activation layer.

28. Convolutional layer with 512 filters, kernel size 3x3, and stride 2x2. Trainable parameters: 256 x 512 x 3 x 3 + 512 = 1,180,160 Gradients: Same as the number of trainable parameters.

29. Batch normalization layer with 512 channels. Trainable parameters: 1024 (512 scales and 512 biases) Gradients: Same as the number of trainable parameters.

30. ReLU activation layer.

31. Convolutional layer with 512 filters, kernel size 3x3, and stride 1x1. Trainable parameters: 512 x 512 x 3 x 3 + 512 = 2,359,808 Gradients: Same as the number of trainable parameters.

32. Batch normalization layer with 512 channels. Trainable parameters: 1024 (512 scales and 512 biases) Gradients: Same as the number of trainable parameters.

33. ReLU activation layer.
34. Convolutional layer with 512 filters, kernel size 3x3, and stride 1x1. Trainable parameters: 512 x 512 x 3 x 3 + 512 = 2,359,808 Gradients: Same as the number of trainable parameters.
35. Batch normalization layer with 512 channels. Trainable parameters: 1024 (512 scales and 512 biases) Gradients: Same as the number of trainable parameters.
36. ReLU activation layer.
37. Average pooling layer with kernel size 7x7.
38. Fully connected layer with 1000 outputs. Trainable parameters: 512 x 1000 + 1000 = 513,000 Gradients: Same as the number of trainable parameters.

The total number of trainable parameters in ResNet-18 is the sum of the trainable parameters in each layer:

1,792 + 128 + 36,928 + 128 + 36,928 + 128 + 36,928 + 256 + 147,584 + 256 + 147,584 + 256 + 295,168 + 512 + 590,080 + 512 + 590,080 + 512 + 1,180,160 + 1024 + 2,359,808 + 1024 + 2,359,808 + 1024 + 513,000 = 11,180,968 trainable parameters.

The total number of gradients is also the sum of the gradients in each layer, which is the same as the number of trainable parameters:

11,180,968 gradients.

==

Q4

The number of trainable parameters and gradients in a ResNet-18 model when using the Adam optimizer will be the same as when using the SGD optimizer. The Adam optimizer uses the same update rule for the trainable parameters as the SGD optimizer, but it also maintains a separate set of adaptive learning rates for each parameter. This means that the number of trainable parameters and gradients will be the same, but the learning rates used to update each parameter may be different.

1,792 + 128 + 36,928 + 128 + 36,928 + 128 + 36,928 + 256 + 147,584 + 256 + 147,584 + 256 + 295,168 + 512 + 590,080 + 512 + 590,080 + 512 + 1,180,160 + 1024 + 2,359,808 + 1024 + 2,359,808 + 1024 + 513,000 = 11,180,968 trainable parameters.And the total number of gradients is the same as the number of trainable parameters: 11,180,968 gradients.