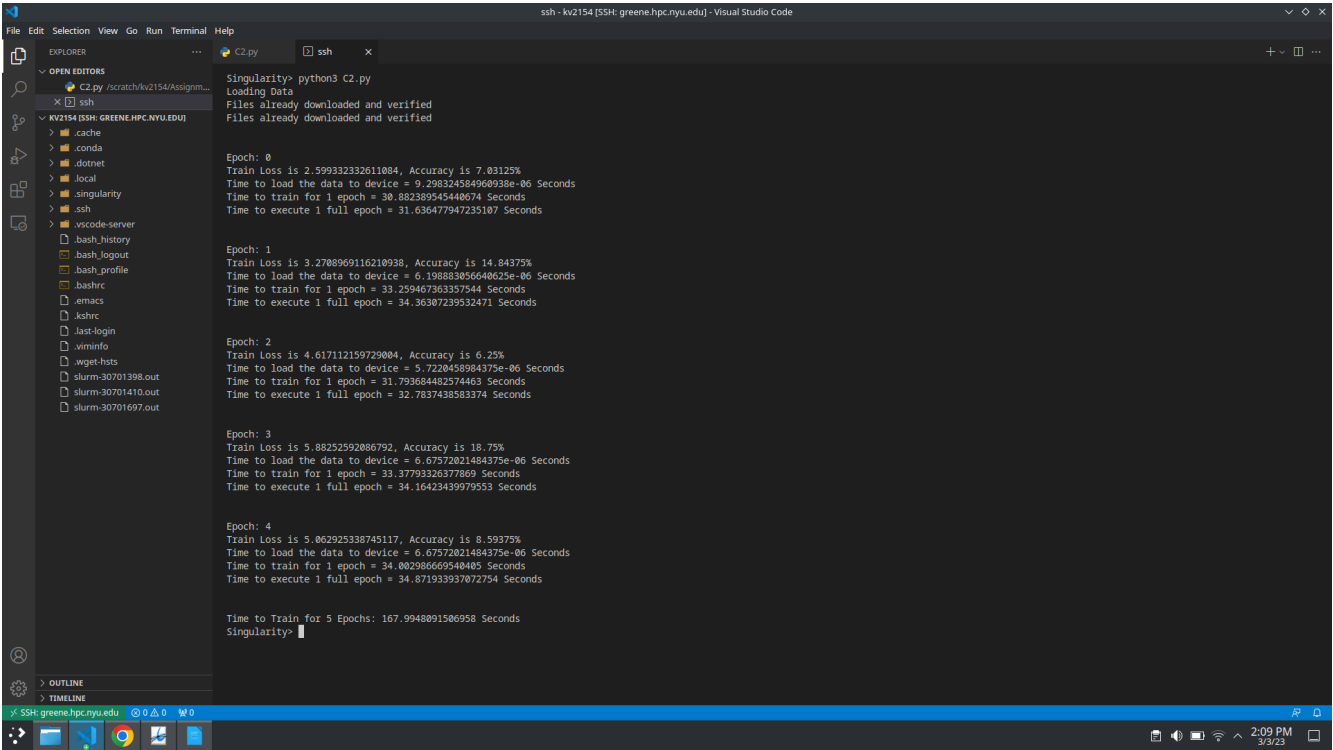


Karan Vora (kv2154)
ECE-GY 9143 Introduction to High-Performance Machine Learning Assignment 2

C1

The Code file is attached with the Final submission

C2



C3
0 Workers

The screenshot shows a Visual Studio Code window with a terminal connected to a remote host via SSH. The terminal displays the output of a Python script that runs a training process for 5 epochs with 0 workers. The script is `C2.py` and the command used is `python3 C2.py --num_workers 0`. The output shows the following training metrics for each epoch:

Epoch	Train Loss	Accuracy	Time to load the data to device	Time to train for 1 epoch	Time to execute 1 full epoch
0	2.404953718185425	10.15625%	3.814697265625e-06	29.264581441879272	29.442380666732788
1	3.7899500928344727	12.5%	3.814697265625e-06	24.27269172668457	24.37325382232666
2	4.473232269287109	18.75%	3.5762786865234375e-06	24.772074699481855	24.87678922279358
3	6.266027458561523	14.0625%	3.5762786865234375e-06	24.95796537399292	25.179893016815186
4	5.447353363837109	17.96875%	3.5762786865234375e-06	25.15838742561646	25.36631202697754

The total training time for 5 epochs is 129.86917233467182 seconds. The terminal also shows the command `ssh` and the file `C2.py` in the Explorer pane.

2 Workers

The screenshot shows a Visual Studio Code window with a terminal connected to a remote host via SSH. The terminal displays the output of a Python script that runs a training process for 5 epochs with 2 workers. The script is `C2.py` and the command used is `python3 C2.py --num_workers 2`. The output shows the following training metrics for each epoch:

Epoch	Train Loss	Accuracy	Time to load the data to device	Time to train for 1 epoch	Time to execute 1 full epoch
0	2.5324904918670654	8.59375%	8.58306884765625e-06	29.87808056692505	30.907358169555664
1	3.328117712828874	10.15625%	6.4373016357421875e-06	34.46642259783081	35.474613189097266
2	3.7872815132141113	17.1875%	6.67572021484375e-06	32.96986246190009	33.869799852371216
3	4.026991367340088	15.625%	9.298324584968938e-06	31.86609983444214	33.05571913719177
4	3.797414418029785	25.78125%	7.867813110351562e-06	34.77876687049866	35.571141719818115

The total training time for 5 epochs is 169.12078595161438 seconds. The terminal also shows the command `ssh` and the file `C2.py` in the Explorer pane.

4 Workers

The screenshot shows a Visual Studio Code window with a terminal connected to a remote host via SSH. The terminal displays the output of a Python script that runs a training process for 5 epochs with 4 workers. The script is `C2.py` and is executed with the command `python3 C2.py --num_workers 4`. The output shows the following training metrics for each epoch:

- Epoch: 0
Train Loss is 2.4413061141967773, Accuracy is 7.03125%
Time to load the data to device = 8.58306884765625e-06 Seconds
Time to train for 1 epoch = 30.781070709228516 Seconds
Time to execute 1 full epoch = 32.06340503692627 Seconds
- Epoch: 1
Train Loss is 3.203399181365967, Accuracy is 19.53125%
Time to load the data to device = 9.05990000859375e-06 Seconds
Time to train for 1 epoch = 34.400044679641724 Seconds
Time to execute 1 full epoch = 36.06389117240906 Seconds
- Epoch: 2
Train Loss is 3.19051456451416, Accuracy is 18.75%
Time to load the data to device = 9.775161743164062e-06 Seconds
Time to train for 1 epoch = 33.40368103981018 Seconds
Time to execute 1 full epoch = 35.354487895965576 Seconds
- Epoch: 3
Train Loss is 4.46003443908691, Accuracy is 8.59375%
Time to load the data to device = 1.001358032225625e-05 Seconds
Time to train for 1 epoch = 33.100314140319824 Seconds
Time to execute 1 full epoch = 34.28806138038635 Seconds
- Epoch: 4
Train Loss is 4.1880700012207, Accuracy is 10.15625%
Time to load the data to device = 8.58306884765625e-06 Seconds
Time to train for 1 epoch = 34.183966398239136 Seconds
Time to execute 1 full epoch = 35.572479009628296 Seconds

The total training time for 5 epochs is 173.57405948638916 Seconds. The terminal prompt is `Singularity>`.

8 Workers

The screenshot shows a Visual Studio Code window with a terminal connected to a remote host via SSH. The terminal displays the output of a Python script that runs a training process for 5 epochs with 8 workers. The script is `C2.py` and is executed with the command `python3 C2.py --num_workers 8`. The output shows the following training metrics for each epoch:

- Epoch: 0
Train Loss is 2.537413835525127, Accuracy is 7.03125%
Time to load the data to device = 7.867813110351562e-06 Seconds
Time to train for 1 epoch = 29.778814792633057 Seconds
Time to execute 1 full epoch = 32.496944427490234 Seconds
- Epoch: 1
Train Loss is 3.5900442600250244, Accuracy is 12.5%
Time to load the data to device = 9.5307451640625e-06 Seconds
Time to train for 1 epoch = 32.48795065081787 Seconds
Time to execute 1 full epoch = 35.077274508928345 Seconds
- Epoch: 2
Train Loss is 3.86767315064563, Accuracy is 17.96875%
Time to load the data to device = 8.34465026854688e-06 Seconds
Time to train for 1 epoch = 33.498278856277466 Seconds
Time to execute 1 full epoch = 35.78649282455444 Seconds
- Epoch: 3
Train Loss is 5.934847354888916, Accuracy is 10.15625%
Time to load the data to device = 9.298324584908938e-06 Seconds
Time to train for 1 epoch = 34.39752793312073 Seconds
Time to execute 1 full epoch = 37.046950340270996 Seconds
- Epoch: 4
Train Loss is 6.07964563369751, Accuracy is 10.9375%
Time to load the data to device = 1.1682510375976562e-05 Seconds
Time to train for 1 epoch = 34.19184327125549 Seconds
Time to execute 1 full epoch = 37.081085205078125 Seconds

The total training time for 5 epochs is 177.73368430137634 Seconds. The terminal prompt is `Singularity>`.

16 Workers

```
Singularity> python3 C2.py --num_workers 16
Loading Data
Files already downloaded and verified
Files already downloaded and verified

Epoch: 0
Train Loss is 2.527089555053711, Accuracy is 6.25%
Time to load the data to device = 3.0517578125e-05 Seconds
Time to train for 1 epoch = 29.297008514404297 Seconds
Time to execute 1 full epoch = 34.41205549248112 Seconds

Epoch: 1
Train Loss is 3.5390899181365967, Accuracy is 11.71875%
Time to load the data to device = 1.2159347534179688e-05 Seconds
Time to train for 1 epoch = 34.21580266952515 Seconds
Time to execute 1 full epoch = 39.75818133354187 Seconds

Epoch: 2
Train Loss is 4.967478275299072, Accuracy is 14.0625%
Time to load the data to device = 8.58306884765625e-06 Seconds
Time to train for 1 epoch = 35.19071626663208 Seconds
Time to execute 1 full epoch = 40.36943960189619 Seconds

Epoch: 3
Train Loss is 5.443963058842285, Accuracy is 14.84375%
Time to load the data to device = 1.1205673217773438e-05 Seconds
Time to train for 1 epoch = 34.205039581190186 Seconds
Time to execute 1 full epoch = 39.73222541809082 Seconds

Epoch: 4
Train Loss is 6.867668151855469, Accuracy is 12.5%
Time to load the data to device = 9.298324584968938e-06 Seconds
Time to train for 1 epoch = 35.102498054504395 Seconds
Time to execute 1 full epoch = 40.08817648887634 Seconds

Time to Train for 5 Epochs: 194.56370568275452 Seconds
Singularity>
```

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

=====

C4

At 0 Workers

```
ssh - kv2154 [SSH: greene.hpc.nyu.edu] - Visual Studio Code

C2.py  ssh

Singularity> python3 C2.py --num_workers 0
Loading Data
Files already downloaded and verified
Files already downloaded and verified

Epoch: 0
Train Loss is 2.404953718185425, Accuracy is 10.15625%
Time to load the data to device = 3.814697265625e-06 Seconds
Time to train for 1 epoch = 29.264581441879272 Seconds
Time to execute 1 full epoch = 29.442380666732788 Seconds

Epoch: 1
Train Loss is 3.7899500928344727, Accuracy is 12.5%
Time to load the data to device = 3.814697265625e-06 Seconds
Time to train for 1 epoch = 24.27269172668457 Seconds
Time to execute 1 full epoch = 24.37325382232666 Seconds

Epoch: 2
Train Loss is 4.473232269287109, Accuracy is 18.75%
Time to load the data to device = 3.5762786865234375e-06 Seconds
Time to train for 1 epoch = 24.772074699481855 Seconds
Time to execute 1 full epoch = 24.87678922279358 Seconds

Epoch: 3
Train Loss is 6.266027458561523, Accuracy is 14.0625%
Time to load the data to device = 3.814697265625e-06 Seconds
Time to train for 1 epoch = 24.95796537399292 Seconds
Time to execute 1 full epoch = 25.179893016815186 Seconds

Epoch: 4
Train Loss is 5.44735363837109, Accuracy is 17.96875%
Time to load the data to device = 3.5762786865234375e-06 Seconds
Time to train for 1 epoch = 25.15838742561646 Seconds
Time to execute 1 full epoch = 25.36631202697754 Seconds

Time to Train for 5 Epochs: 129.86917233467182 Seconds
Singularity>
```

At 1 Worker

```
ssh - kv2154 [SSH: greene.hpc.nyu.edu] - Visual Studio Code

C2.py  ssh

Singularity> python3 C2.py --num_workers 1
Loading Data
Files already downloaded and verified
Files already downloaded and verified

Epoch: 0
Train Loss is 2.421926498413086, Accuracy is 9.375%
Time to load the data to device = 8.106231689453125e-06 Seconds
Time to train for 1 epoch = 29.2562255859375 Seconds
Time to execute 1 full epoch = 29.826510190963745 Seconds

Epoch: 1
Train Loss is 3.021644115447998, Accuracy is 17.1875%
Time to load the data to device = 2.384185791015625e-05 Seconds
Time to train for 1 epoch = 36.70235499633625 Seconds
Time to execute 1 full epoch = 37.359989166239706 Seconds

Epoch: 2
Train Loss is 3.975824461746216, Accuracy is 6.25%
Time to load the data to device = 1.2636184692382812e-05 Seconds
Time to train for 1 epoch = 34.2633399633789 Seconds
Time to execute 1 full epoch = 34.98077082633972 Seconds

Epoch: 3
Train Loss is 5.207638748539551, Accuracy is 14.84375%
Time to load the data to device = 8.344650826854688e-06 Seconds
Time to train for 1 epoch = 34.64932203292847 Seconds
Time to execute 1 full epoch = 35.273864038083801 Seconds

Epoch: 4
Train Loss is 4.964319705963135, Accuracy is 10.15625%
Time to load the data to device = 7.152557373046875e-06 Seconds
Time to train for 1 epoch = 36.949727058410645 Seconds
Time to execute 1 full epoch = 37.5638084149627686 Seconds

Time to Train for 5 Epochs: 175.191946506508024 Seconds
Singularity>
```

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:

The screenshot shows the Visual Studio Code interface with the file 'KaranC5_CPU.out' open. The code displays system information and training progress for a CPU-based model. The system information includes: Processor: Intel(R) Xeon(R) Gold 6148 CPU @ 2.48GHz, RAM: 222Gi, GPU: Product Name : Tesla V100-SXM2-16GB. The training progress shows four epochs with the following data preparation times and overall epoch times:

Epoch	Data Preparation Time	Overall Epoch Time
0	0.003676	298.540447
1	0.003782	327.492485
2	0.003657	324.137251
3	0.003645	325.109964
4	0.003739	324.796889

The status bar at the bottom indicates the system is running at 2.84% CPU usage, 2.19 GHz frequency, and 60% power. The system clock shows 10:55 PM on 3/4/23.

GPU:

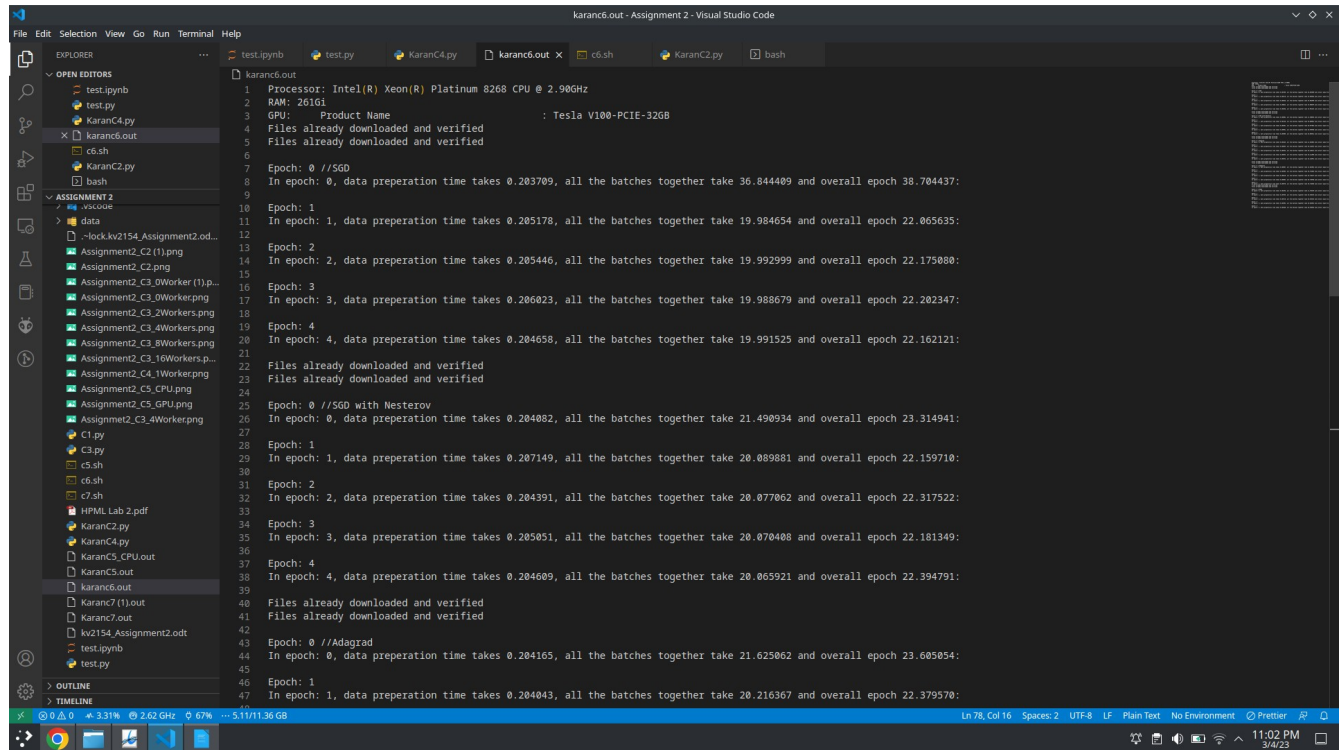
The screenshot shows the Visual Studio Code interface with the file 'KaranC5.out' open. The code displays system information and training progress for a GPU-based model. The system information includes: Processor: Intel(R) Xeon(R) Platinum 8268 CPU @ 2.90GHz, RAM: 235Gi, GPU: Product Name : Quadro RTX 8000. The training progress shows four epochs with the following data preparation times and overall epoch times:

Epoch	Data Preparation Time	Overall Epoch Time
0	0.192351	21.783305
1	0.191741	20.386545
2	0.192853	20.362632
3	0.191938	20.399931
4	0.193310	20.396353

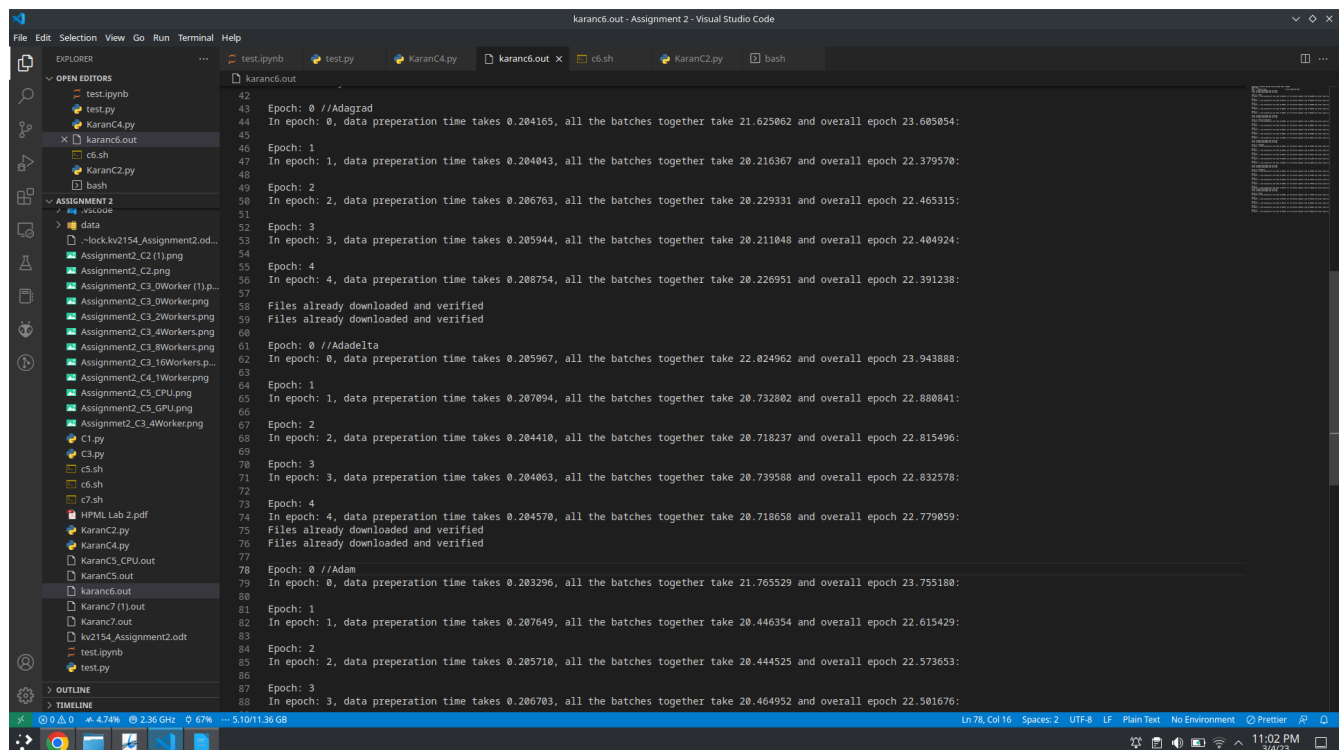
The status bar at the bottom indicates the system is running at 3.14% CPU usage, 2.30 GHz frequency, and 67% power. The system clock shows 10:58 PM on 3/4/23.

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



```
1 Processor: Intel(R) Xeon(R) Platinum 8268 CPU @ 2.90GHz
2 RAM: 261Gi
3 GPU: Product Name : Tesla V100-PCIE-32GB
4 Files already downloaded and verified
5 Files already downloaded and verified
6
7 Epoch: 0 //SGD
8 In epoch: 0, data preparation time takes 0.203709, all the batches together take 36.844409 and overall epoch 38.704437:
9
10 Epoch: 1
11 In epoch: 1, data preparation time takes 0.205178, all the batches together take 19.984654 and overall epoch 22.065635:
12
13 Epoch: 2
14 In epoch: 2, data preparation time takes 0.205446, all the batches together take 19.992999 and overall epoch 22.175080:
15
16 Epoch: 3
17 In epoch: 3, data preparation time takes 0.206023, all the batches together take 19.988679 and overall epoch 22.282347:
18
19 Epoch: 4
20 In epoch: 4, data preparation time takes 0.204658, all the batches together take 19.991525 and overall epoch 22.162121:
21
22 Files already downloaded and verified
23 Files already downloaded and verified
24
25 Epoch: 0 //SGD with Nesterov
26 In epoch: 0, data preparation time takes 0.204082, all the batches together take 21.490934 and overall epoch 23.314941:
27
28 Epoch: 1
29 In epoch: 1, data preparation time takes 0.207149, all the batches together take 20.089881 and overall epoch 22.159710:
30
31 Epoch: 2
32 In epoch: 2, data preparation time takes 0.204391, all the batches together take 20.077062 and overall epoch 22.317522:
33
34 Epoch: 3
35 In epoch: 3, data preparation time takes 0.205051, all the batches together take 20.070408 and overall epoch 22.181349:
36
37 Epoch: 4
38 In epoch: 4, data preparation time takes 0.204609, all the batches together take 20.065921 and overall epoch 22.394791:
39
40 Files already downloaded and verified
41 Files already downloaded and verified
42
43 Epoch: 0 //Adagrad
44 In epoch: 0, data preparation time takes 0.204165, all the batches together take 21.625062 and overall epoch 23.605054:
45
46 Epoch: 1
47 In epoch: 1, data preparation time takes 0.204043, all the batches together take 20.216367 and overall epoch 22.379570:
```



```
42
43 Epoch: 0 //Adagrad
44 In epoch: 0, data preparation time takes 0.204165, all the batches together take 21.625062 and overall epoch 23.605054:
45
46 Epoch: 1
47 In epoch: 1, data preparation time takes 0.204043, all the batches together take 20.216367 and overall epoch 22.379570:
48
49 Epoch: 2
50 In epoch: 2, data preparation time takes 0.206763, all the batches together take 20.229331 and overall epoch 22.465315:
51
52 Epoch: 3
53 In epoch: 3, data preparation time takes 0.205944, all the batches together take 20.211048 and overall epoch 22.404924:
54
55 Epoch: 4
56 In epoch: 4, data preparation time takes 0.208754, all the batches together take 20.226951 and overall epoch 22.391238:
57
58 Files already downloaded and verified
59 Files already downloaded and verified
60
61 Epoch: 0 //Adadelta
62 In epoch: 0, data preparation time takes 0.205967, all the batches together take 22.024962 and overall epoch 23.943888:
63
64 Epoch: 1
65 In epoch: 1, data preparation time takes 0.207094, all the batches together take 20.732802 and overall epoch 22.880841:
66
67 Epoch: 2
68 In epoch: 2, data preparation time takes 0.204410, all the batches together take 20.718237 and overall epoch 22.815496:
69
70 Epoch: 3
71 In epoch: 3, data preparation time takes 0.204063, all the batches together take 20.739588 and overall epoch 22.832578:
72
73 Epoch: 4
74 In epoch: 4, data preparation time takes 0.204570, all the batches together take 20.718658 and overall epoch 22.779059:
75
76 Files already downloaded and verified
77 Files already downloaded and verified
78
79 Epoch: 0 //Adam
80 In epoch: 0, data preparation time takes 0.203296, all the batches together take 21.765529 and overall epoch 23.755180:
81
82 Epoch: 1
83 In epoch: 1, data preparation time takes 0.207649, all the batches together take 20.446354 and overall epoch 22.615429:
84
85 Epoch: 2
86 In epoch: 2, data preparation time takes 0.205710, all the batches together take 20.444525 and overall epoch 22.573653:
87
88 Epoch: 3
89 In epoch: 3, data preparation time takes 0.206703, all the batches together take 20.464952 and overall epoch 22.581676:
```



```
karanc6.out
59 Files already downloaded and verified
60
61 Epoch: 0 //Adadelta
62 In epoch: 0, data preparation time takes 0.285967, all the batches together take 22.824962 and overall epoch 23.943888:
63
64 Epoch: 1
65 In epoch: 1, data preparation time takes 0.287894, all the batches together take 20.732882 and overall epoch 22.888841:
66
67 Epoch: 2
68 In epoch: 2, data preparation time takes 0.284410, all the batches together take 20.718237 and overall epoch 22.815496:
69
70 Epoch: 3
71 In epoch: 3, data preparation time takes 0.284863, all the batches together take 20.739588 and overall epoch 22.832578:
72
73 Epoch: 4
74 In epoch: 4, data preparation time takes 0.284578, all the batches together take 20.718658 and overall epoch 22.779059:
75 Files already downloaded and verified
76 Files already downloaded and verified
77
78 Epoch: 0 //Adam
79 In epoch: 0, data preparation time takes 0.283296, all the batches together take 21.765529 and overall epoch 23.755180:
80
81 Epoch: 1
82 In epoch: 1, data preparation time takes 0.287649, all the batches together take 20.446354 and overall epoch 22.615429:
83
84 Epoch: 2
85 In epoch: 2, data preparation time takes 0.285710, all the batches together take 20.444525 and overall epoch 22.573653:
86
87 Epoch: 3
88 In epoch: 3, data preparation time takes 0.286783, all the batches together take 20.464952 and overall epoch 22.581676:
89
90 Epoch: 4
91 In epoch: 4, data preparation time takes 0.284268, all the batches together take 20.459382 and overall epoch 22.610261:
92
```

C7

```
Karanc7.out
1 Processor: Intel(R) Xeon(R) Platinum 8268 CPU @ 2.90GHz
2 RAM: 235Gi
3 GPU: Product Name : Quadro RTX 8000
4 Files already downloaded and verified
5 Files already downloaded and verified
6
7 Epoch: 0
8 /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
9 warnings.warn(_create_warning_msg
10 In epoch: 0, data preparation time takes 0.196657, all the batches together take 36.989225 and overall epoch 38.769718:
11
12 Epoch: 1
13 In epoch: 1, data preparation time takes 0.284856, all the batches together take 19.655856 and overall epoch 21.799284:
14
15 Epoch: 2
16 In epoch: 2, data preparation time takes 0.282195, all the batches together take 19.654959 and overall epoch 21.784801:
17
18 Epoch: 3
19 In epoch: 3, data preparation time takes 0.280949, all the batches together take 19.687268 and overall epoch 21.977420:
20
21 Epoch: 4
22 In epoch: 4, data preparation time takes 0.283738, all the batches together take 19.691474 and overall epoch 21.916488:
23
```

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: $64 \times 3 \times 3 \times 3 + 64 = 1,792$ 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: $64 \times 64 \times 3 \times 3 + 64 = 36,928$ 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: $64 \times 64 \times 3 \times 3 + 64 = 36,928$ 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: $64 \times 128 \times 3 \times 3 + 128 = 73,856$ 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: $128 \times 128 \times 3 \times 3 + 128 = 147,584$ 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

$\times 128 \times 3 \times 3 + 128 = 147,584$ 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 3×3 , and stride 2×2 . Trainable parameters: $128 \times 256 \times 3 \times 3 + 256 = 295,168$ 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 3×3 , and stride 1×1 . Trainable parameters: $256 \times 256 \times 3 \times 3 + 256 = 590,080$ 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 3×3 , and stride 1×1 . Trainable parameters: $256 \times 256 \times 3 \times 3 + 256 = 590,080$ 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 3×3 , and stride 2×2 . Trainable parameters: $256 \times 512 \times 3 \times 3 + 512 = 1,180,160$ 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 3×3 , and stride 1×1 . Trainable parameters: $512 \times 512 \times 3 \times 3 + 512 = 2,359,808$ 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: $512 \times 512 \times 3 \times 3 + 512 = 2,359,808$ 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: $512 \times 1000 + 1000 = 513,000$ 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.