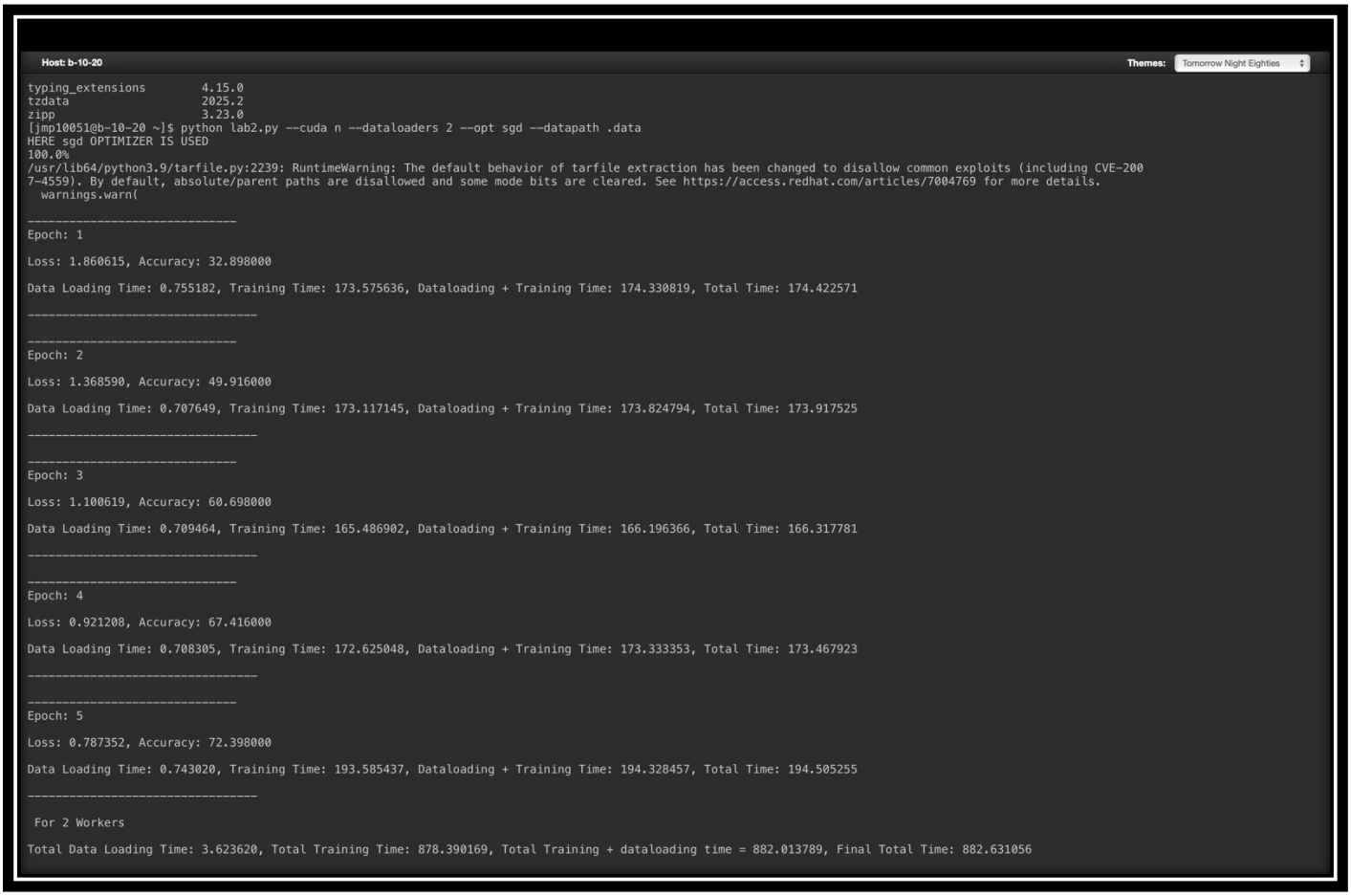
HPML Assignment 2

Name: Jayraj M. Pamnani
NetID: jmp10051

C1: In the lab2.py file.

C2: Time Measurement



C3: I/O optimization for code

C3.1:

(a) 0 workers

```
[jmp10051@b-10-20 ~]$ python lab2.py --cuda n --dataloaders 0 --opt sgd --datapath .data
HERE sgd OPTIMIZER IS USED

--------------------------------
Epoch: 1

Loss: 2.065489, Accuracy: 26.326000

Data Loading Time: 12.204592, Training Time: 130.676519, Dataloading + Training Time: 142.881110, Total Time: 142.959942

----------------------------------

--------------------------------
Epoch: 2

Loss: 1.533229, Accuracy: 42.956000

Data Loading Time: 11.922107, Training Time: 134.067319, Dataloading + Training Time: 145.989426, Total Time: 146.066802

----------------------------------

--------------------------------
Epoch: 3

Loss: 1.289574, Accuracy: 53.346000

Data Loading Time: 11.884516, Training Time: 133.361944, Dataloading + Training Time: 145.246460, Total Time: 145.322051

----------------------------------

--------------------------------
Epoch: 4

Loss: 1.072985, Accuracy: 61.764000

Data Loading Time: 11.848706, Training Time: 134.228885, Dataloading + Training Time: 146.077591, Total Time: 146.152744

----------------------------------

--------------------------------
Epoch: 5

Loss: 0.907638, Accuracy: 67.866000

Data Loading Time: 11.841454, Training Time: 135.310624, Dataloading + Training Time: 147.152078, Total Time: 147.227918

----------------------------------

 For 0 Workers

Total Data Loading Time: 59.701375, Total Training Time: 667.645292, Total Training + dataloading time = 727.346666, Final Total Time: 727.729456
[jmp10051@b-10-20 ~]$
```

(b) 4 workers

```
[jmp10051@b-10-20 ~]$ python lab2.py --cuda n --dataloaders 4 --opt sgd --datapath .data
HERE sgd OPTIMIZER IS USED

--------------------------------
Epoch: 1

Loss: 1.967847, Accuracy: 30.180000

Data Loading Time: 0.690580, Training Time: 165.673667, Dataloading + Training Time: 166.364248, Total Time: 166.450348

----------------------------------

--------------------------------
Epoch: 2

Loss: 1.446657, Accuracy: 46.602000

Data Loading Time: 0.707954, Training Time: 173.457828, Dataloading + Training Time: 174.165781, Total Time: 174.262491

----------------------------------

--------------------------------
Epoch: 3

Loss: 1.217369, Accuracy: 56.196000

Data Loading Time: 0.721406, Training Time: 179.546504, Dataloading + Training Time: 180.267910, Total Time: 180.385074

----------------------------------

--------------------------------
Epoch: 4

Loss: 1.019952, Accuracy: 63.582000

Data Loading Time: 0.765364, Training Time: 186.318561, Dataloading + Training Time: 187.083925, Total Time: 187.209729

----------------------------------

--------------------------------
Epoch: 5

Loss: 0.865086, Accuracy: 69.418000

Data Loading Time: 0.748715, Training Time: 162.973996, Dataloading + Training Time: 163.722711, Total Time: 163.849709

----------------------------------

 For 4 Workers

Total Data Loading Time: 3.634020, Total Training Time: 867.970556, Total Training + dataloading time = 871.604576, Final Total Time: 872.157350
[jmp10051@b-10-20 ~]$
```

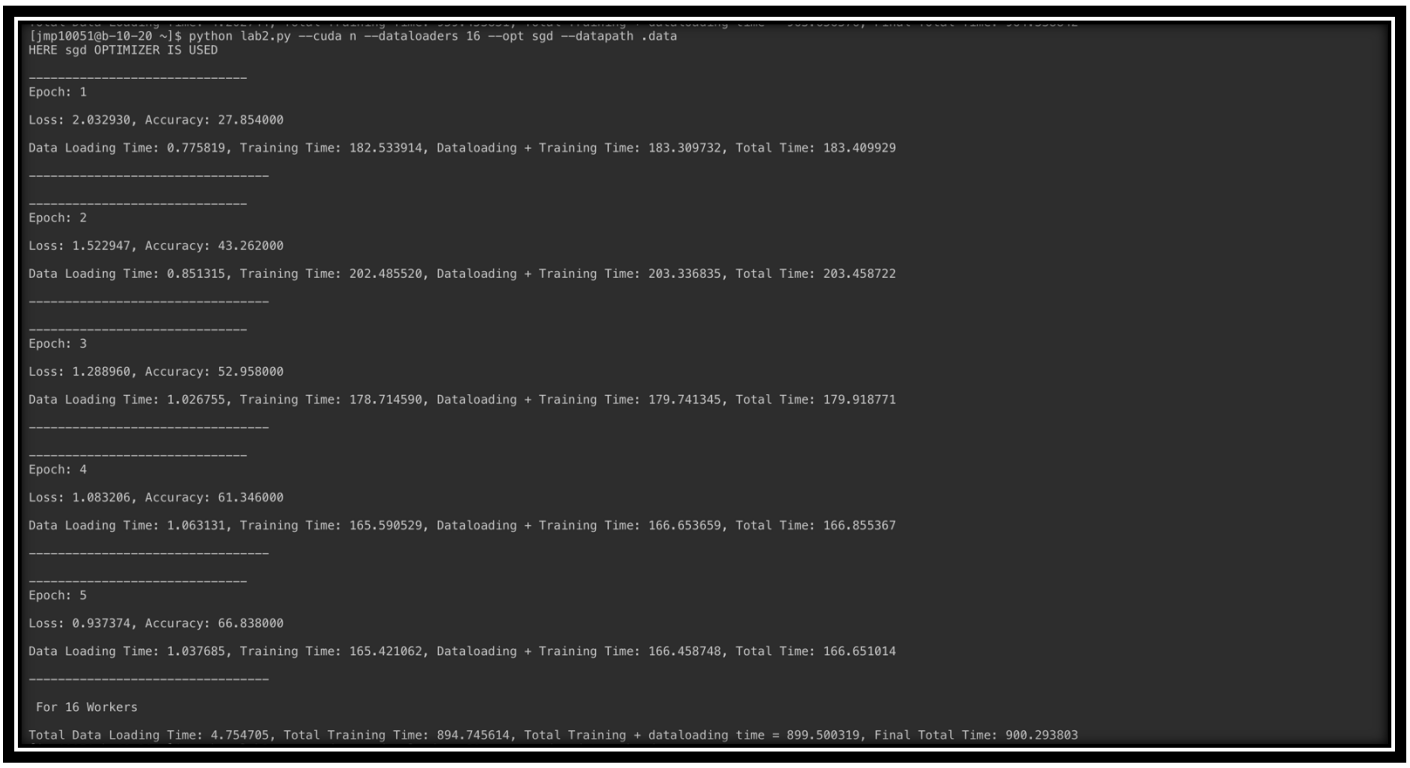
(c) 8 workers

```
[jmp10051@b-10-20 ~]$ python lab2.py --cuda n --dataloaders 8 --opt sgd --datapath .data
HERE sgd OPTIMIZER IS USED

--------------------------------
Epoch: 1

Loss: 2.012797, Accuracy: 27.812000

Data Loading Time: 0.693904, Training Time: 158.509670, Dataloading + Training Time: 159.203575, Total Time: 159.295259

----------------------------------

--------------------------------
Epoch: 2

Loss: 1.485887, Accuracy: 45.140000

Data Loading Time: 0.771884, Training Time: 167.771452, Dataloading + Training Time: 168.543336, Total Time: 168.645253

----------------------------------

--------------------------------
Epoch: 3

Loss: 1.208373, Accuracy: 56.274000

Data Loading Time: 0.822565, Training Time: 186.338576, Dataloading + Training Time: 187.161141, Total Time: 187.306729

----------------------------------

--------------------------------
Epoch: 4

Loss: 0.985520, Accuracy: 64.968000

Data Loading Time: 0.837965, Training Time: 182.156227, Dataloading + Training Time: 182.994192, Total Time: 183.153118

----------------------------------

--------------------------------
Epoch: 5

Loss: 0.818894, Accuracy: 71.356000

Data Loading Time: 0.844311, Training Time: 181.735133, Dataloading + Training Time: 182.579445, Total Time: 182.759328

----------------------------------

 For 8 Workers

Total Data Loading Time: 3.970629, Total Training Time: 876.511059, Total Training + dataloading time = 880.481688, Final Total Time: 881.159688
[jmp10051@b-10-20 ~]$
```

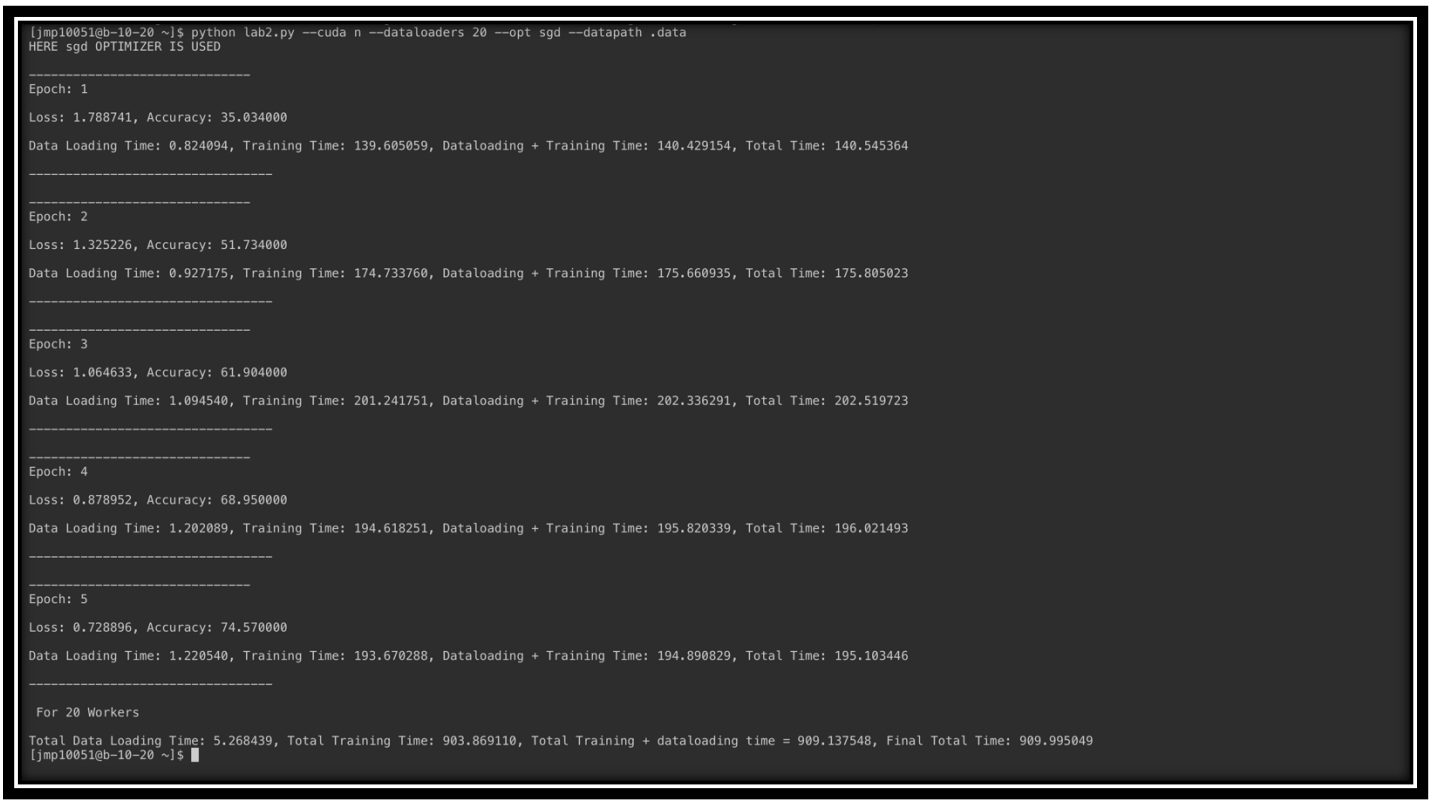
(d) 12 workers

```
[jmp10051@b-10-20 ~]$ python lab2.py --cuda n --dataloaders 12 --opt sgd --datapath .data
HERE sgd OPTIMIZER IS USED

--------------------------------
Epoch: 1

Loss: 1.865820, Accuracy: 31.202000

Data Loading Time: 0.747234, Training Time: 166.942699, Dataloading + Training Time: 167.689933, Total Time: 167.781527

----------------------------------

--------------------------------
Epoch: 2

Loss: 1.439344, Accuracy: 47.026000

Data Loading Time: 0.787791, Training Time: 172.108379, Dataloading + Training Time: 172.896170, Total Time: 173.005293

----------------------------------

--------------------------------
Epoch: 3

Loss: 1.180190, Accuracy: 57.448000

Data Loading Time: 0.880789, Training Time: 185.454586, Dataloading + Training Time: 186.335376, Total Time: 186.487479

----------------------------------

--------------------------------
Epoch: 4

Loss: 0.985625, Accuracy: 64.926000

Data Loading Time: 0.889018, Training Time: 209.061774, Dataloading + Training Time: 209.950792, Total Time: 210.108224

----------------------------------

--------------------------------
Epoch: 5

Loss: 0.860720, Accuracy: 69.744000

Data Loading Time: 0.897912, Training Time: 225.886393, Dataloading + Training Time: 226.784304, Total Time: 226.956319

----------------------------------

 For 12 Workers

Total Data Loading Time: 4.202744, Total Training Time: 959.453831, Total Training + dataloading time = 963.656576, Final Total Time: 964.338842
[jmp10051@b-10-20 ~]$
```

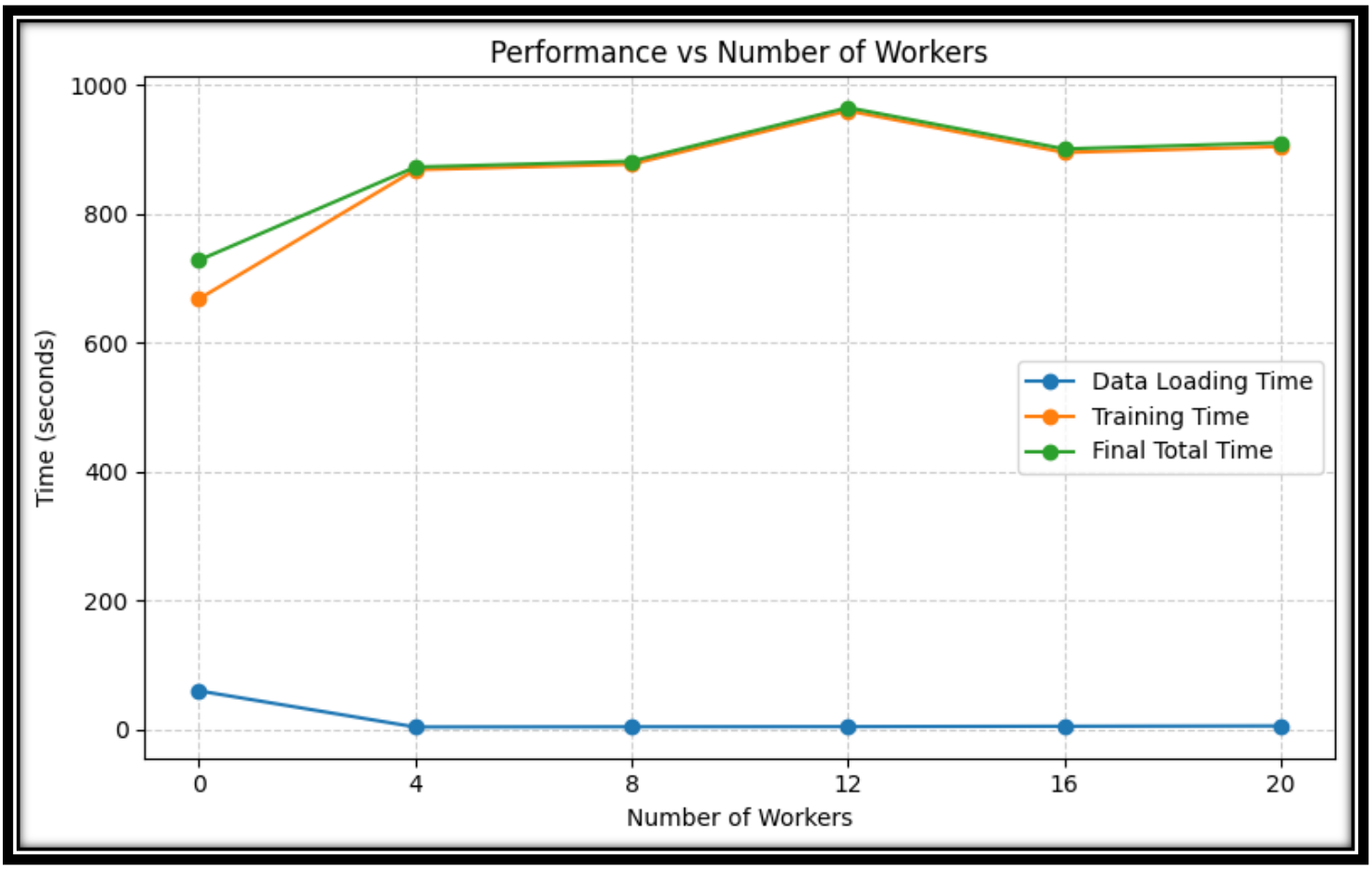
(e) 16 workers



(f) 20 workers



Graph:



C3.2: As we can see, the best runtime is found at 0 workers.

C4: Comparing performance of 0 worker and 1 worker

(a) 0 Worker

```
[jmp10051@b-10-20 ~]$ python lab2.py --cuda n --dataloaders 0 --opt sgd --datapath .data
HERE sgd OPTIMIZER IS USED

--------------------------------
Epoch: 1

Loss: 2.065489, Accuracy: 26.326000

Data Loading Time: 12.204592, Training Time: 130.676519, Dataloading + Training Time: 142.881110, Total Time: 142.959942

----------------------------------

--------------------------------
Epoch: 2

Loss: 1.533229, Accuracy: 42.956000

Data Loading Time: 11.922107, Training Time: 134.067319, Dataloading + Training Time: 145.989426, Total Time: 146.066802

----------------------------------

--------------------------------
Epoch: 3

Loss: 1.289574, Accuracy: 53.346000

Data Loading Time: 11.884516, Training Time: 133.361944, Dataloading + Training Time: 145.246460, Total Time: 145.322051

----------------------------------

--------------------------------
Epoch: 4

Loss: 1.072985, Accuracy: 61.764000

Data Loading Time: 11.848706, Training Time: 134.228885, Dataloading + Training Time: 146.077591, Total Time: 146.152744

----------------------------------

--------------------------------
Epoch: 5

Loss: 0.907638, Accuracy: 67.866000

Data Loading Time: 11.841454, Training Time: 135.310624, Dataloading + Training Time: 147.152078, Total Time: 147.227918

----------------------------------

 For 0 Workers

Total Data Loading Time: 59.701375, Total Training Time: 667.645292, Total Training + dataloading time = 727.346666, Final Total Time: 727.729456
[jmp10051@b-10-20 ~]$
```

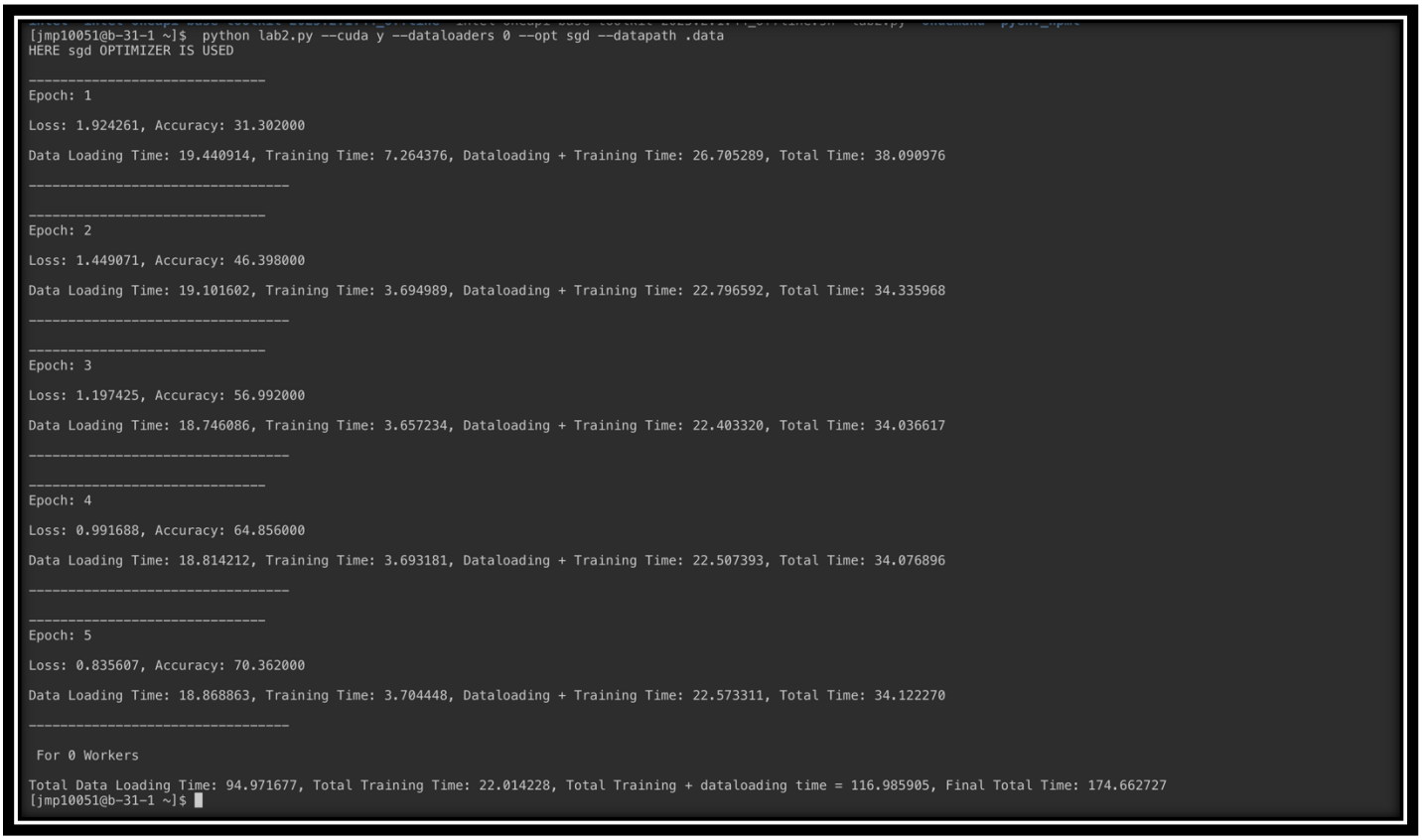
(b) 1 Worker

```
[jmp10051@b-10-20 ~]$ python lab2.py --cuda n --dataloaders 1 --opt sgd --datapath .data
HERE sgd OPTIMIZER IS USED

--------------------------------
Epoch: 1

Loss: 2.016264, Accuracy: 28.638000

Data Loading Time: 0.697321, Training Time: 205.417271, Dataloading + Training Time: 206.114592, Total Time: 206.206205

----------------------------------

--------------------------------
Epoch: 2

Loss: 1.473830, Accuracy: 45.846000

Data Loading Time: 0.694985, Training Time: 189.009725, Dataloading + Training Time: 189.704710, Total Time: 189.808759

----------------------------------

--------------------------------
Epoch: 3

Loss: 1.191866, Accuracy: 57.278000

Data Loading Time: 0.714199, Training Time: 170.592950, Dataloading + Training Time: 171.307149, Total Time: 171.443690

----------------------------------

--------------------------------
Epoch: 4

Loss: 0.970248, Accuracy: 65.710000

Data Loading Time: 0.727580, Training Time: 173.664183, Dataloading + Training Time: 174.391764, Total Time: 174.529810

----------------------------------

--------------------------------
Epoch: 5

Loss: 0.814219, Accuracy: 71.498000

Data Loading Time: 0.688038, Training Time: 168.002924, Dataloading + Training Time: 168.690962, Total Time: 168.826985

----------------------------------

 For 1 Workers

Total Data Loading Time: 3.522123, Total Training Time: 906.687053, Total Training + dataloading time = 910.209176, Final Total Time: 910.815450
[jmp10051@b-10-20 ~]$
```

We can observe that the total run time for 0 worker is much faster than 1 worker.

C5: Training GPU vs CPU

(a) GPU:



(b) CPU:

```
[jmp10051@b-10-20 ~]$ python lab2.py --cuda n --dataloaders 0 --opt sgd --datapath .data
HERE sgd OPTIMIZER IS USED

--------------------------------
Epoch: 1

Loss: 2.065489, Accuracy: 26.326000

Data Loading Time: 12.204592, Training Time: 130.676519, Dataloading + Training Time: 142.881110, Total Time: 142.959942

----------------------------------

--------------------------------
Epoch: 2

Loss: 1.533229, Accuracy: 42.956000

Data Loading Time: 11.922107, Training Time: 134.067319, Dataloading + Training Time: 145.989426, Total Time: 146.066802

----------------------------------

--------------------------------
Epoch: 3

Loss: 1.289574, Accuracy: 53.346000

Data Loading Time: 11.884516, Training Time: 133.361944, Dataloading + Training Time: 145.246460, Total Time: 145.322051

----------------------------------

--------------------------------
Epoch: 4

Loss: 1.072985, Accuracy: 61.764000

Data Loading Time: 11.848706, Training Time: 134.228885, Dataloading + Training Time: 146.077591, Total Time: 146.152744

----------------------------------

--------------------------------
Epoch: 5

Loss: 0.907638, Accuracy: 67.866000

Data Loading Time: 11.841454, Training Time: 135.310624, Dataloading + Training Time: 147.152078, Total Time: 147.227918

----------------------------------

 For 0 Workers

Total Data Loading Time: 59.701375, Total Training Time: 667.645292, Total Training + dataloading time = 727.346666, Final Total Time: 727.729456
[jmp10051@b-10-20 ~]$
```

C6: Implementing different optimizers

(a) SGD

```
[jmp10051@b-31-1 ~]$  python lab2.py --cuda y --dataloaders 0 --opt sgd --datapath .data
HERE sgd OPTIMIZER IS USED

--------------------------------
Epoch: 1

Loss: 1.924261, Accuracy: 31.302000

Data Loading Time: 19.440914, Training Time: 7.264376, Dataloading + Training Time: 26.705289, Total Time: 38.090976

----------------------------------

--------------------------------
Epoch: 2

Loss: 1.449071, Accuracy: 46.398000

Data Loading Time: 19.101602, Training Time: 3.694989, Dataloading + Training Time: 22.796592, Total Time: 34.335968

----------------------------------

--------------------------------
Epoch: 3

Loss: 1.197425, Accuracy: 56.992000

Data Loading Time: 18.746086, Training Time: 3.657234, Dataloading + Training Time: 22.403320, Total Time: 34.036617

----------------------------------

--------------------------------
Epoch: 4

Loss: 0.991688, Accuracy: 64.856000

Data Loading Time: 18.814212, Training Time: 3.693181, Dataloading + Training Time: 22.507393, Total Time: 34.076896

----------------------------------

--------------------------------
Epoch: 5

Loss: 0.835607, Accuracy: 70.362000

Data Loading Time: 18.868863, Training Time: 3.704448, Dataloading + Training Time: 22.573311, Total Time: 34.122270

----------------------------------

 For 0 Workers

Total Data Loading Time: 94.971677, Total Training Time: 22.014228, Total Training + dataloading time = 116.985905, Final Total Time: 174.662727
[jmp10051@b-31-1 ~]$
```

(b) Sgdnesterov

```
[jmp10051@b-31-1 ~]$  python lab2.py --cuda y --dataloaders 0 --opt sgdnesterov --datapath .data
HERE sgdnesterov OPTIMIZER IS USED

--------------------------------
Epoch: 1

Loss: 1.820007, Accuracy: 33.784000

Data Loading Time: 19.209811, Training Time: 5.006053, Dataloading + Training Time: 24.215864, Total Time: 35.905136

----------------------------------

--------------------------------
Epoch: 2

Loss: 1.308248, Accuracy: 52.390000

Data Loading Time: 19.075580, Training Time: 3.715100, Dataloading + Training Time: 22.790680, Total Time: 34.584356

----------------------------------

--------------------------------
Epoch: 3

Loss: 1.033548, Accuracy: 63.224000

Data Loading Time: 18.638670, Training Time: 3.673876, Dataloading + Training Time: 22.312546, Total Time: 34.127415

----------------------------------

--------------------------------
Epoch: 4

Loss: 0.869650, Accuracy: 69.140000

Data Loading Time: 18.571750, Training Time: 3.585731, Dataloading + Training Time: 22.157480, Total Time: 34.037791

----------------------------------

--------------------------------
Epoch: 5

Loss: 0.745725, Accuracy: 73.962000

Data Loading Time: 18.554339, Training Time: 3.604089, Dataloading + Training Time: 22.158427, Total Time: 34.055433

----------------------------------

 For 0 Workers

Total Data Loading Time: 94.050149, Total Training Time: 19.584849, Total Training + dataloading time = 113.634998, Final Total Time: 172.710131
[jmp10051@b-31-1 ~]$
```

(c) Adadelta

```
[jmp10051@b-31-1 ~]$  python lab2.py --cuda y --dataloaders 0 --opt adadelta --datapath .data
HERE adadelta OPTIMIZER IS USED

--------------------------------
Epoch: 1

Loss: 1.359245, Accuracy: 50.146000

Data Loading Time: 19.504380, Training Time: 5.399201, Dataloading + Training Time: 24.903582, Total Time: 37.370283

----------------------------------

--------------------------------
Epoch: 2

Loss: 0.860174, Accuracy: 69.510000

Data Loading Time: 19.520624, Training Time: 4.010388, Dataloading + Training Time: 23.531012, Total Time: 36.127645

----------------------------------

--------------------------------
Epoch: 3

Loss: 0.671880, Accuracy: 76.408000

Data Loading Time: 19.439833, Training Time: 4.015188, Dataloading + Training Time: 23.455021, Total Time: 36.005968

----------------------------------

--------------------------------
Epoch: 4

Loss: 0.567265, Accuracy: 80.262000

Data Loading Time: 19.245256, Training Time: 3.958511, Dataloading + Training Time: 23.203768, Total Time: 35.813909

----------------------------------

--------------------------------
Epoch: 5

Loss: 0.500229, Accuracy: 82.656000

Data Loading Time: 18.939286, Training Time: 3.939099, Dataloading + Training Time: 22.878384, Total Time: 35.514244

----------------------------------

 For 0 Workers

Total Data Loading Time: 96.649380, Total Training Time: 21.322388, Total Training + dataloading time = 117.971768, Final Total Time: 180.832049
```

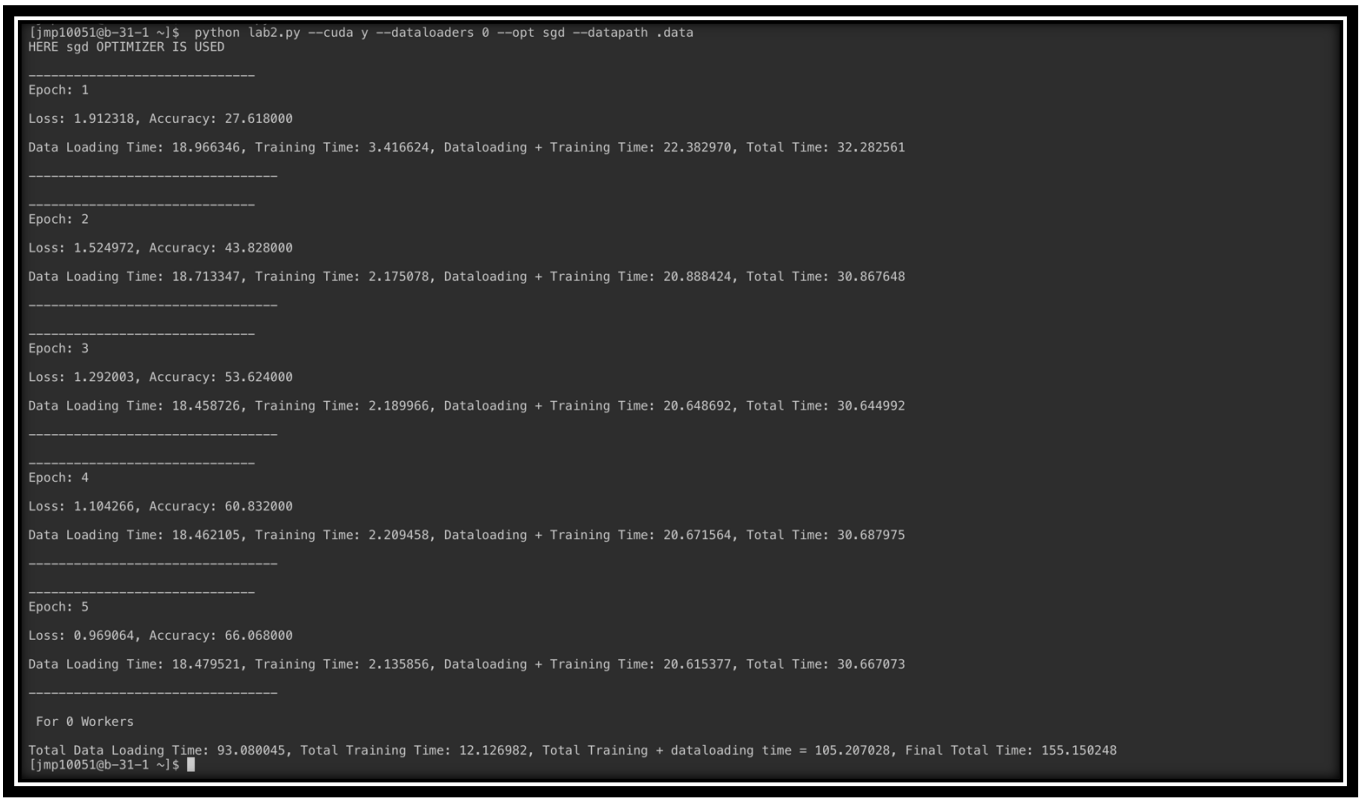
(d) Adagrad

```
[jmp10051@b-31-1 ~]$  python lab2.py --cuda y --dataloaders 0 --opt adagrad --datapath .data
HERE adagrad OPTIMIZER IS USED

--------------------------------
Epoch: 1

Loss: 2.161710, Accuracy: 24.984000

Data Loading Time: 19.231862, Training Time: 5.158088, Dataloading + Training Time: 24.389951, Total Time: 36.089702

----------------------------------

--------------------------------
Epoch: 2

Loss: 1.647262, Accuracy: 38.196000

Data Loading Time: 19.152587, Training Time: 3.795993, Dataloading + Training Time: 22.948580, Total Time: 34.725770

----------------------------------

--------------------------------
Epoch: 3

Loss: 1.414044, Accuracy: 47.974000

Data Loading Time: 19.063723, Training Time: 3.784065, Dataloading + Training Time: 22.847788, Total Time: 34.568351

----------------------------------

--------------------------------
Epoch: 4

Loss: 1.202188, Accuracy: 56.308000

Data Loading Time: 19.110550, Training Time: 3.825210, Dataloading + Training Time: 22.935760, Total Time: 34.613485

----------------------------------

--------------------------------
Epoch: 5

Loss: 1.035707, Accuracy: 62.876000

Data Loading Time: 19.046902, Training Time: 3.804514, Dataloading + Training Time: 22.851416, Total Time: 34.515355

----------------------------------

 For 0 Workers

Total Data Loading Time: 95.605623, Total Training Time: 20.367871, Total Training + dataloading time = 115.973494, Final Total Time: 174.512662
[jmp10051@b-31-1 ~]$
```

(e) Adam

```
[jmp10051@b-31-1 ~]$  python lab2.py --cuda y --dataloaders 0 --opt adam --datapath .data
HERE adam OPTIMIZER IS USED

--------------------------------
Epoch: 1

Loss: 2.230027, Accuracy: 22.100000

Data Loading Time: 19.374107, Training Time: 5.313177, Dataloading + Training Time: 24.687284, Total Time: 36.377403

----------------------------------

--------------------------------
Epoch: 2

Loss: 1.866349, Accuracy: 28.334000

Data Loading Time: 19.329797, Training Time: 4.012411, Dataloading + Training Time: 23.342208, Total Time: 34.984629

----------------------------------

--------------------------------
Epoch: 3

Loss: 1.830058, Accuracy: 29.590000

Data Loading Time: 19.317120, Training Time: 4.018960, Dataloading + Training Time: 23.336080, Total Time: 34.863164

----------------------------------

--------------------------------
Epoch: 4

Loss: 1.815095, Accuracy: 30.530000

Data Loading Time: 19.298339, Training Time: 3.968577, Dataloading + Training Time: 23.266916, Total Time: 34.798296

----------------------------------

--------------------------------
Epoch: 5

Loss: 1.801504, Accuracy: 30.614000

Data Loading Time: 19.063774, Training Time: 3.949497, Dataloading + Training Time: 23.013271, Total Time: 34.509087

----------------------------------

 For 0 Workers

Total Data Loading Time: 96.383137, Total Training Time: 21.262623, Total Training + dataloading time = 117.645759, Final Total Time: 175.532579
[jmp10051@b-31-1 ~]$
```

C7: Implementing without Batch Norm



Theoretical Questions:

Q1: How many convolutional layers are in the ResNet-18 model?
Ans: There are 18 convolutional layers in ResNet-18 model.

Q2: What is the input dimension of the last linear layer?
Ans: The input dimension of the last linear layer is (1*512).

Q3: How many trainable parameters and how many gradients are in the ResNet-18 model that you build (please show both the answer and the code that you use to count them) when using the SGD optimizer?
Ans: The model summary for net with an input shape of (3, 32, 32) shows:

```
python count_params.py
======================================================================
ResNet-18 Parameter and Gradient Count (with SGD optimizer)
======================================================================

Trainable parameters: 11,164,362
Number of gradients: 11,164,362

======================================================================
Verification:
Trainable parameters == Gradients: True
======================================================================

Layer-wise Parameter Count:
Layer Name                              Shape                  Parameters
--------------------------------------------------------------------------
conv1.weight                            torch.Size([64, 3, 3, 3])        1,728
layer1.0.conv1.weight                   torch.Size([64, 64, 3, 3])      36,864
layer1.0.conv2.weight                   torch.Size([64, 64, 3, 3])      36,864
layer1.1.conv1.weight                   torch.Size([64, 64, 3, 3])      36,864
layer1.1.conv2.weight                   torch.Size([64, 64, 3, 3])      36,864
layer2.0.conv1.weight                   torch.Size([128, 64, 3, 3])      73,728
layer2.0.conv2.weight                   torch.Size([128, 128, 3, 3])    147,456
layer2.0.shortcut.0.weight              torch.Size([128, 64, 1, 1])       8,192
layer2.1.conv1.weight                   torch.Size([128, 128, 3, 3])    147,456
layer2.1.conv2.weight                   torch.Size([128, 128, 3, 3])    147,456
layer3.0.conv1.weight                   torch.Size([256, 128, 3, 3])    294,912
layer3.0.conv2.weight                   torch.Size([256, 256, 3, 3])    589,824
layer3.0.shortcut.0.weight              torch.Size([256, 128, 1, 1])     32,768
layer3.1.conv1.weight                   torch.Size([256, 256, 3, 3])    589,824
layer3.1.conv2.weight                   torch.Size([256, 256, 3, 3])    589,824
layer4.0.conv1.weight                   torch.Size([512, 256, 3, 3])  1,179,648
layer4.0.conv2.weight                   torch.Size([512, 512, 3, 3])  2,359,296
layer4.0.shortcut.0.weight              torch.Size([512, 256, 1, 1])    131,072
layer4.1.conv1.weight                   torch.Size([512, 512, 3, 3])  2,359,296
layer4.1.conv2.weight                   torch.Size([512, 512, 3, 3])  2,359,296
linear.weight                           torch.Size([10, 512])             5,120
linear.bias                             torch.Size([10])                     10
[jmp10051@b-31-1 ~]$
```

Code:

```
cat > count_params.py << 'EOF'
import torch
import torch.nn as nn
import torch.optim as optim
from lab2 import ResNet18

# Create ResNet-18 model
model = ResNet18()

# Create SGD optimizer (as specified in the question)
optimizer = optim.SGD(model.parameters(), lr=0.1, momentum=0.9, weight_decay=5e-4)

print("="*70)
print("ResNet-18 Parameter and Gradient Count (with SGD optimizer)")
print("="*70)

# Count trainable parameters
trainable_params = sum(p.numel() for p in model.parameters() if p.requires_grad)
print(f"\nTrainable parameters: {trainable_params:,}")

# Perform forward and backward pass to compute gradients
```

```
dummy_input = torch.randn(1, 3, 32, 32)  # CIFAR-10 input size
dummy_target = torch.tensor([0])
criterion = nn.CrossEntropyLoss()

# Forward pass
output = model(dummy_input)
loss = criterion(output, dummy_target)

# Backward pass (computes gradients)
optimizer.zero_grad()
loss.backward()

# Count gradients
num_gradients = sum(p.numel() for p in model.parameters() if p.grad is not None)
print(f"Number of gradients: {num_gradients:,}")

print("\n" + "="*70)
print("Verification:")
print(f"Trainable parameters == Gradients: {trainable_params == num_gradients}")
print("="*70)

# Show layer-wise breakdown
print("\nLayer-wise Parameter Count:")
print(f"{'Layer Name':<40} {'Shape':<20} {'Parameters':>12}")
print("-"*75)
for name, param in model.named_parameters():
    if param.requires_grad:
        print(f"{name:<40} {str(param.shape):<20} {param.numel():>12,}")
EOF
python count_params.py
```

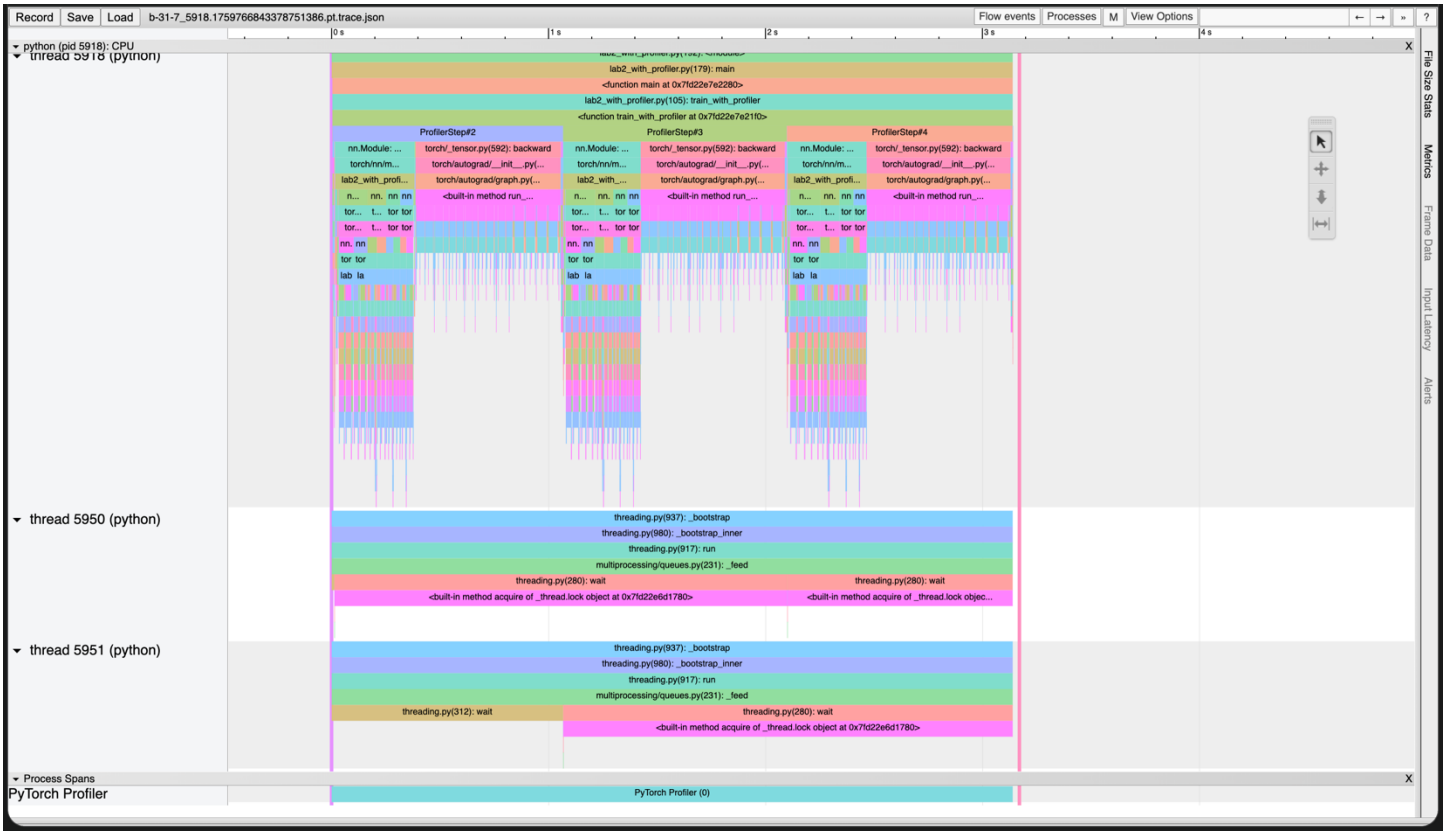
Q4: Same question as Q3, except now using Adam (only the answer is required, not the code).

Ans:

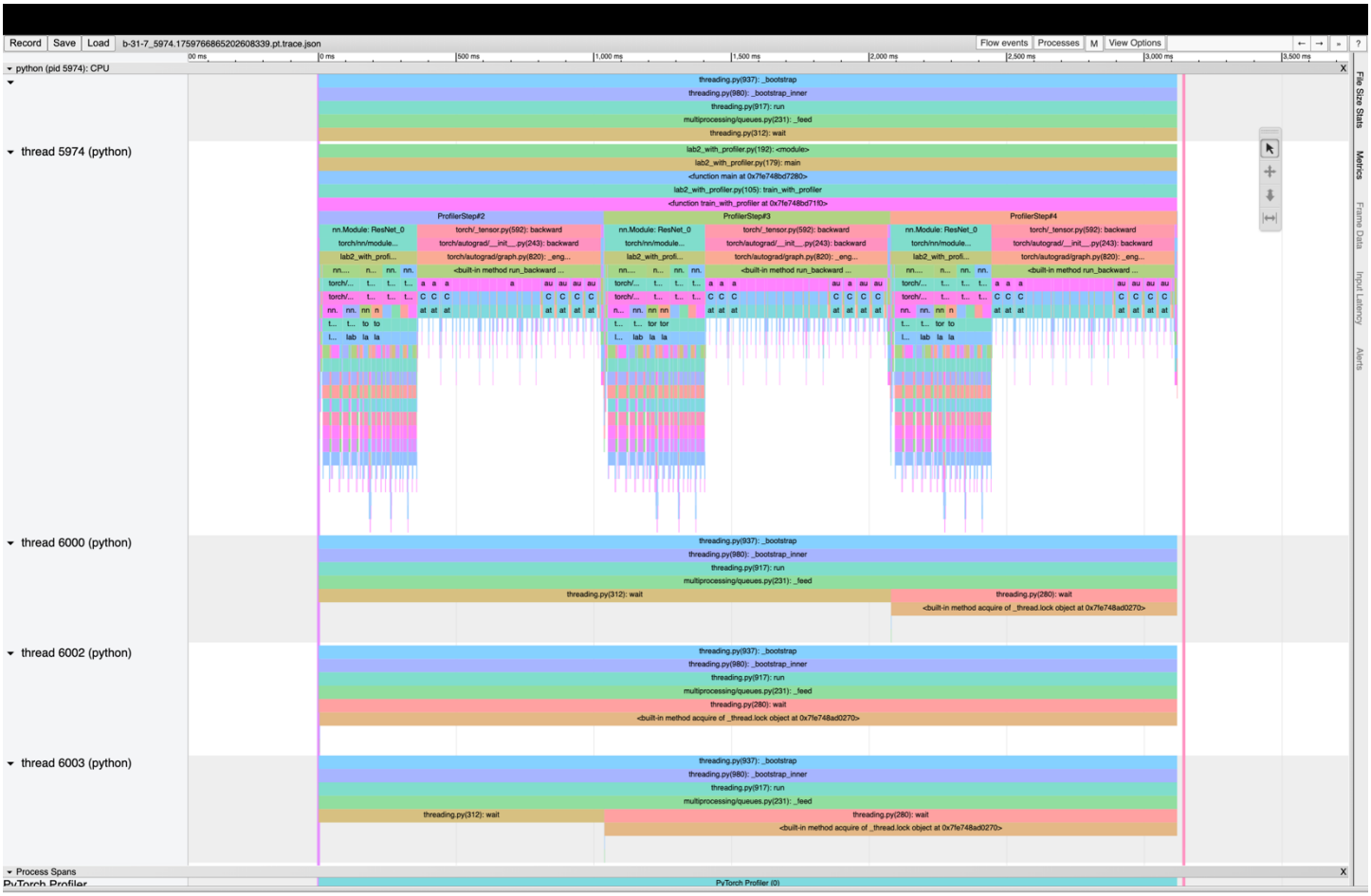
```
python count_params_adam.py
======================================================================
ResNet-18 Parameter and Gradient Count (with Adam optimizer)
======================================================================

Trainable parameters: 11,164,362
Number of gradients: 11,164,362

======================================================================
Verification:
Trainable parameters == Gradients: True
======================================================================

Layer-wise Parameter Count:
Layer Name                              Shape                  Parameters
----------------------------------------------------------------------
conv1.weight                            torch.Size([64, 3, 3, 3])        1,728
layer1.0.conv1.weight                   torch.Size([64, 64, 3, 3])      36,864
layer1.0.conv2.weight                   torch.Size([64, 64, 3, 3])      36,864
layer1.1.conv1.weight                   torch.Size([64, 64, 3, 3])      36,864
layer1.1.conv2.weight                   torch.Size([64, 64, 3, 3])      36,864
layer2.0.conv1.weight                   torch.Size([128, 64, 3, 3])     73,728
layer2.0.conv2.weight                   torch.Size([128, 128, 3, 3])   147,456
layer2.0.shortcut.0.weight              torch.Size([128, 64, 1, 1])      8,192
layer2.1.conv1.weight                   torch.Size([128, 128, 3, 3])   147,456
layer2.1.conv2.weight                   torch.Size([128, 128, 3, 3])   147,456
layer3.0.conv1.weight                   torch.Size([256, 128, 3, 3])   294,912
layer3.0.conv2.weight                   torch.Size([256, 256, 3, 3])   589,824
layer3.0.shortcut.0.weight              torch.Size([256, 128, 1, 1])    32,768
layer3.1.conv1.weight                   torch.Size([256, 256, 3, 3])   589,824
layer3.1.conv2.weight                   torch.Size([256, 256, 3, 3])   589,824
layer4.0.conv1.weight                   torch.Size([512, 256, 3, 3])  1,179,648
layer4.0.conv2.weight                   torch.Size([512, 512, 3, 3])  2,359,296
layer4.0.shortcut.0.weight              torch.Size([512, 256, 1, 1])    131,072
layer4.1.conv1.weight                   torch.Size([512, 512, 3, 3])  2,359,296
layer4.1.conv2.weight                   torch.Size([512, 512, 3, 3])  2,359,296
linear.weight                           torch.Size([10, 512])             5,120
linear.bias                             torch.Size([10])                     10
[jmp10051@b-31-1 ~]$
```

Extra Credit:

(a) Profiling C1:

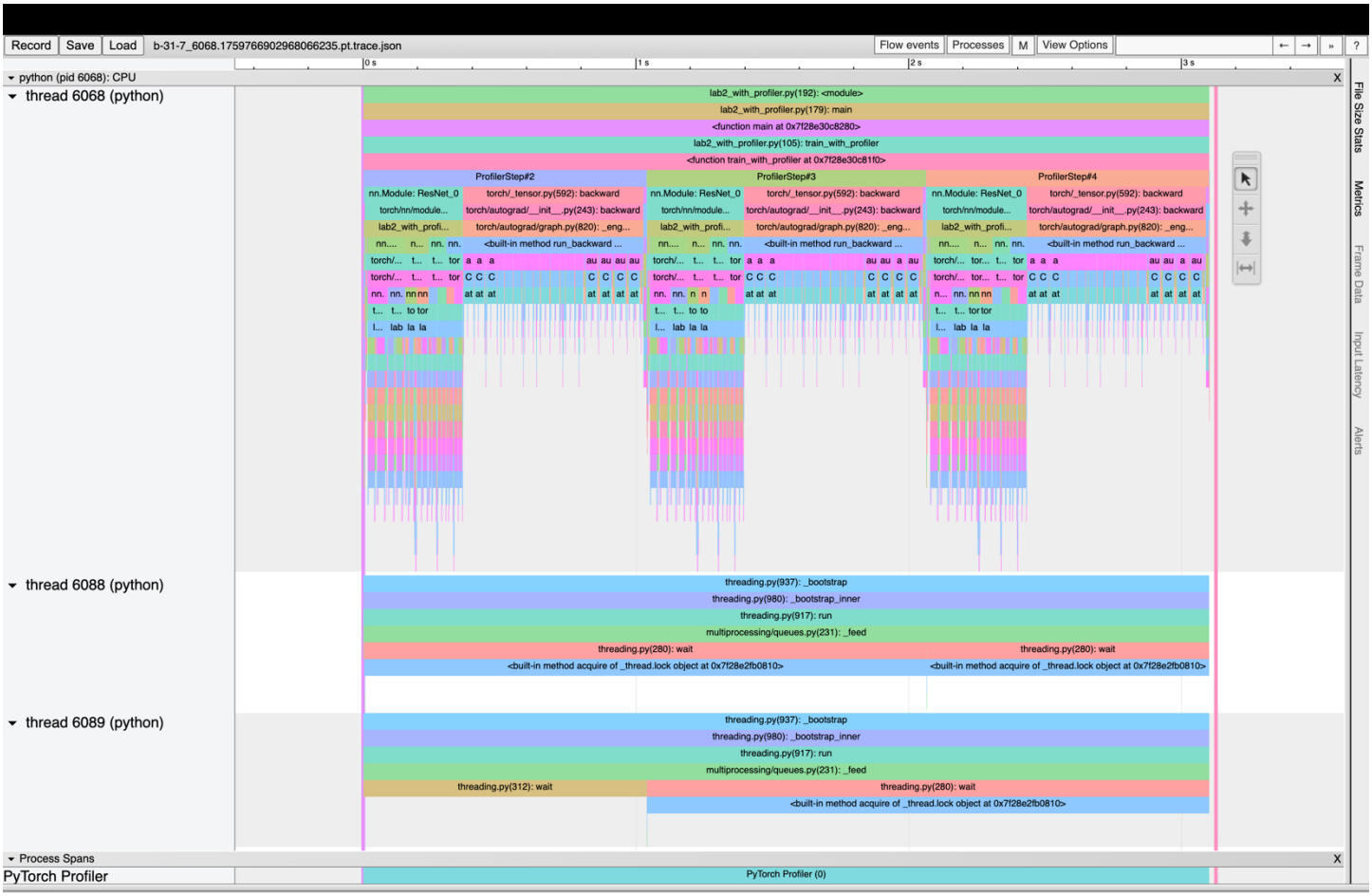


(b) Profiling C2:



(c) Profiling C3:

(d) Profiling C4:



(e) Profiling C5: