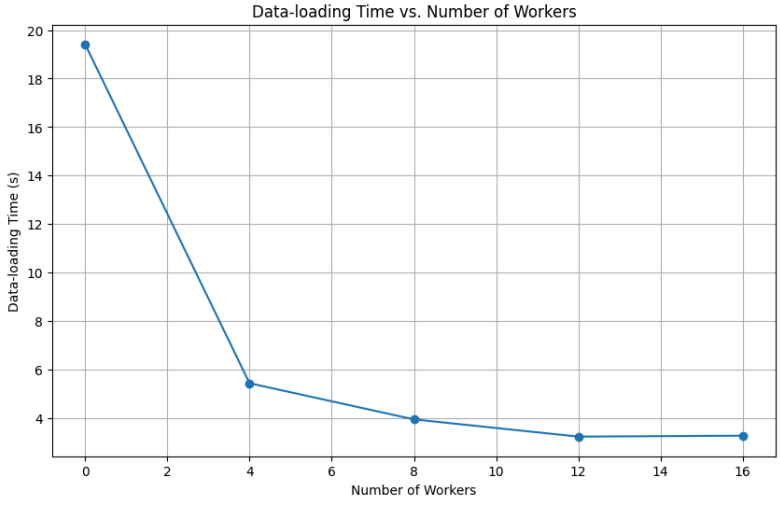
Dylan Soemitro
drs2199

C1:
Epoch 1, Loss: 2.1340176178061445, Accuracy: 25.406%
Epoch 2, Loss: 1.6184561941629785, Accuracy: 39.594%
Epoch 3, Loss: 1.3609117263418329, Accuracy: 50.078%
Epoch 4, Loss: 1.1279869196969834, Accuracy: 59.244%
Epoch 5, Loss: 0.9601353047144078, Accuracy: 65.684%C

C2:
Epoch 1, Loss: 1.786299714956747, Accuracy: 34.426%, Data-Loading Time: 0.000s, Training Time: 0.012s, Total Time: 10.589s
Epoch 2, Loss: 1.3060501967854512, Accuracy: 52.23%, Data-Loading Time: 0.002s, Training Time: 0.012s, Total Time: 10.537s
Epoch 3, Loss: 1.0247635429777453, Accuracy: 63.462%, Data-Loading Time: 0.003s, Training Time: 0.012s, Total Time: 10.360s
Epoch 4, Loss: 0.8660560797547441, Accuracy: 69.384%, Data-Loading Time: 0.000s, Training Time: 0.012s, Total Time: 10.565s
Epoch 5, Loss: 0.7277301362408396, Accuracy: 74.474%, Data-Loading Time: 0.000s, Training Time: 0.012s, Total Time: 10.453s

C3:
Num Workers: 0, Data-loading Time: 19.414s
Num Workers: 4, Data-loading Time: 5.446s
Num Workers: 8, Data-loading Time: 3.957s
Num Workers: 12, Data-loading Time: 3.247s
Num Workers: 16, Data-loading Time: 3.286s



C4:
1 Worker - Data-loading Time: 0.235s, Computing Time: 3.872s
Best (12 Workers) - Data-loading Time: 1.075s, Computing Time: 4.561s
This is not what I expected. With a single worker, the data-loading time is significantly shorter, which is counter-intuitive as the DataLoader can prepare multiple batches in parallel when the number of workers is significantly higher than one. The reason that the data-loading time could

be higher could be due to some hardware limitations or overhead of multiple workers, where there is increased contention for shared resources. The same reason could mean why the computing time is higher as well.

C5:
Average GPU running time over 5 epochs: 6.402s
Average CPU running time over 5 epochs: 426.343s

C6:
Training with SGD
Epoch: 1, Loss: 1.7659275217739212, Accuracy: 35.914%
Epoch: 2, Loss: 1.2256216632435695, Accuracy: 55.646%
Epoch: 3, Loss: 0.9763161212282108, Accuracy: 65.266%
Epoch: 4, Loss: 0.8179961032879627, Accuracy: 71.294%
Epoch: 5, Loss: 0.6966557004262725, Accuracy: 75.784%
Optimizer: SGD, Average Time: 7.301s, Loss: 0.6967, Accuracy: 75.78%

Training with SGD_nesterov
Epoch: 1, Loss: 1.9678074097084572, Accuracy: 29.468%
Epoch: 2, Loss: 1.3785853675564232, Accuracy: 49.138%
Epoch: 3, Loss: 1.0917203482764457, Accuracy: 60.98%
Epoch: 4, Loss: 0.9097430792915852, Accuracy: 67.846%
Epoch: 5, Loss: 0.7905158498098174, Accuracy: 72.328%
Optimizer: SGD_nesterov, Average Time: 7.282s, Loss: 0.7905, Accuracy: 72.33%

Training with Adagrad
Epoch: 1, Loss: 2.1477326468738447, Accuracy: 24.898%
Epoch: 2, Loss: 1.6599454980372164, Accuracy: 37.466%
Epoch: 3, Loss: 1.4438958036930054, Accuracy: 46.332%
Epoch: 4, Loss: 1.2379069846609365, Accuracy: 55.032%
Epoch: 5, Loss: 1.0706467918117943, Accuracy: 61.514%
Optimizer: Adagrad, Average Time: 7.418s, Loss: 1.0706, Accuracy: 61.51%

Training with Adadelta
Epoch: 1, Loss: 1.3689329561674992, Accuracy: 49.722%
Epoch: 2, Loss: 0.8828335668119933, Accuracy: 68.47%
Epoch: 3, Loss: 0.682237489677756, Accuracy: 76.312%
Epoch: 4, Loss: 0.5747742302277509, Accuracy: 80.016%
Epoch: 5, Loss: 0.5000553467237127, Accuracy: 82.638%
Optimizer: Adadelta, Average Time: 7.520s, Loss: 0.5001, Accuracy: 82.64%

Training with Adam
Epoch: 1, Loss: 2.2658108494165914, Accuracy: 20.79%
Epoch: 2, Loss: 1.9324488618489726, Accuracy: 25.18%

Dylan Soemitro
drs2199

Epoch: 3, Loss: 1.9179610980441197, Accuracy: 25.828%
Epoch: 4, Loss: 1.9202007116259212, Accuracy: 26.328%
Epoch: 5, Loss: 1.9197533661142334, Accuracy: 26.238%
Optimizer: Adam, Average Time: 7.554s, Loss: 1.9198, Accuracy: 26.24%

C7:
Epoch: 1, Loss: 1.9672720234107484, Accuracy: 25.84%
Epoch: 2, Loss: 1.5937652843992423, Accuracy: 41.16%
Epoch: 3, Loss: 1.385637001613217, Accuracy: 49.862%
Epoch: 4, Loss: 1.1870046395170109, Accuracy: 57.83%
Epoch: 5, Loss: 1.0309576146742876, Accuracy: 63.814%

Q1:
Initially, we have 1+4+4+4+4=17 conv layers. We also have shortcuts for the second, third, and fourth sets, so we have a total of 20 convolutional layers.

Q2:
```
self.layer1 = self._make_layer(block, 64, num_blocks[0], stride=1)
        self.layer2 = self._make_layer(block, 128, num_blocks[1],
stride=2)
        self.layer3 = self._make_layer(block, 256, num_blocks[2],
stride=2)
        self.layer4 = self._make_layer(block, 512, num_blocks[3],
stride=2)
        self.linear = nn.Linear(512 * block.expansion, num_classes)
```
We can see that it is 512.

Q3
```
model = ResNet18()
num_trainable_params = sum(p.numel() for p in model.parameters() if
p.requires_grad)
print(f"Number of trainable parameters: {num_trainable_params}")

dummy_input = torch.randn(1, 3, 32, 32)
optimizer = optim.SGD(model.parameters(), lr=0.1)
output = model(dummy_input)
criterion = nn.CrossEntropyLoss()
target = torch.tensor([1])
loss = criterion(output, target)
loss.backward()
num_gradients = sum(torch.count_nonzero(p.grad) for p in
model.parameters() if p.grad is not None)
```

Dylan Soemitro
drs2199

```
print(f"Number of non-zero gradients: {num_gradients}")
```

Number of trainable parameters: 11164362
Number of non-zero gradients: 8920240

Q4:

```
Number of trainable parameters: 11164362
Number of non-zero gradients: 8896249
```