HPML Lab2

am14661

**C1: Training in PyTorch (20 points)**

**C2: Time Measurement of Code in C1 (10 points)**

* Measure and report:
  1. Data-loading time for each epoch.
  2. Training time (mini-batch calculation) for each epoch.
  3. Total running time for each epoch.

A screenshot of a computer

Description automatically generated

**C3: I/O Optimization (10 points)**

**0-workers**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A graph with a blue line

Description automatically generated

**C4: Profiling (10 points)**

* Compare data-loading and computing time for runs using 1 worker vs. the optimal number of workers (from C3).

• Explain any differences in performance.

A screenshot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

**Compare data-loading and computing time for runs using 1 worker and the number of workers needed for best performance found in C3 and explain (in a few words) the differences if there is any.**

As we are dealing with a 32X32 image which doesn’t demand much computational power. Here I think , a single worker can load data fast enough that adding more workers doesn’t significantly improve the speed.

And other reason can be that 1 worker is there is no communication overhead involved where as while using 20 the process might be fast but due to communication overhead it is almost same.

**C5: Training in GPUs vs CPUs (10 points)**

• Compare the average running time over 5 epochs for training on GPU vs. CPU (using the optimal number of workers from C3).

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**C6: Experimenting with Different Optimizers (10 points)**

* Train the model for 5 epochs using different optimizers (SGD, SGD with Nesterov, Adagrad, Adadelta, Adam).

• Report the average training time, training loss, and top-1 training accuracy for each optimizer.

A screenshot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A comparison of blue bars

Description automatically generated

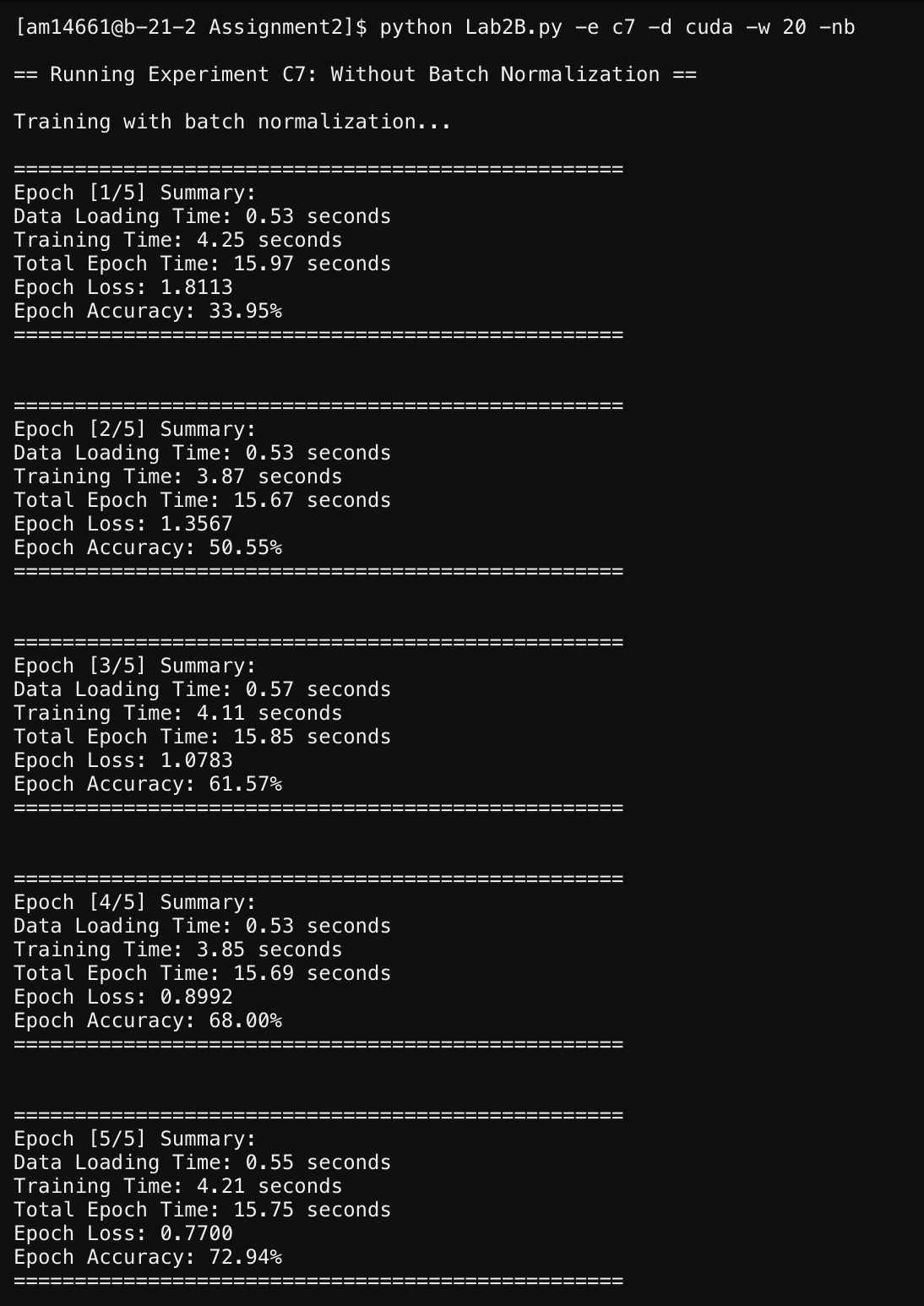
A graph of blue rectangular bars

Description automatically generated

**C7: Experimenting Without Batch Norm (10 points)**

* Train the model for 5 epochs without batch normalization layers.

• Report the average training loss and top-1 training accuracy.



A screenshot of a computer

Description automatically generated

**Q1 )How many convolutional layers are in the ResNet-18 model?**

**Sol:-** The Breakdown of Layers in ResNet-18 is following:

Initial Convolution Layer: 1 (7×7 Conv)

Residual Blocks (each containing 2 convolutional layers):

* + Conv2\_x: 2 blocks × 2 conv layers = 4
  + Conv3\_x: 2 blocks × 2 conv layers = 4
  + Conv4\_x: 2 blocks × 2 conv layers = 4
  + Conv5\_x: 2 blocks × 2 conv layers = 4

Final Fully Connected Layer: 1 (FC layer)

Total = 1 + 4 + 4 + 4 + 4 + 1 = 18 Layers

* Convolutional layers: 17
* Fully connected layer: 1
* Total trainable layers: 18

**Q2 ) What is the input dimension of the last linear layer?**

**Sol:-**

1. **Input Image**: 32x32
2. **Spatial Reduction**:
   * conv1: No reduction (32x32)
   * layer2: Stride 2 → 16x16
   * layer3: Stride 2 → 8x8
   * layer4: Stride 2 → 4x4
3. **Global Average Pooling**: 4x4 → 1x1 (kernel size 4)
4. **Flattened Features**: 512 channels → **512-dimensional input** to linear layer.

The final input dimension becomes 512\*1=512.

**Q3) How many trainable parameters and how many gradients in the ResNet-18 model that you build (please show both the answer and the code that you use to count them), when using SGD optimizer?**

def count\_parameters\_and\_gradients(use\_bn=True):

model = ResNet18(use\_bn=use\_bn)

total\_params = sum(p.numel() for p in model.parameters() if p.requires\_grad)

print(f"\nTrainable Parameters: {total\_params}")

# Dummy forward/backward pass to generate gradients

dummy\_input = torch.randn(1, 3, 32, 32)

dummy\_target = torch.randint(0, 10, (1,))

output = model(dummy\_input)

loss = F.cross\_entropy(output, dummy\_target)

loss.backward()

total\_gradients = sum(p.grad.numel() for p in model.parameters() if p.requires\_grad)

print(f"Gradients: {total\_gradients}\n")

and have called this function in main :-

count\_parameters\_and\_gradients(use\_bn=not args.no\_batchnorm)

A black background with white text

Description automatically generated

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

A black background with white text

Description automatically generated