

MP-3

Name: Savya Saachi Verma

NetID: ssverma2

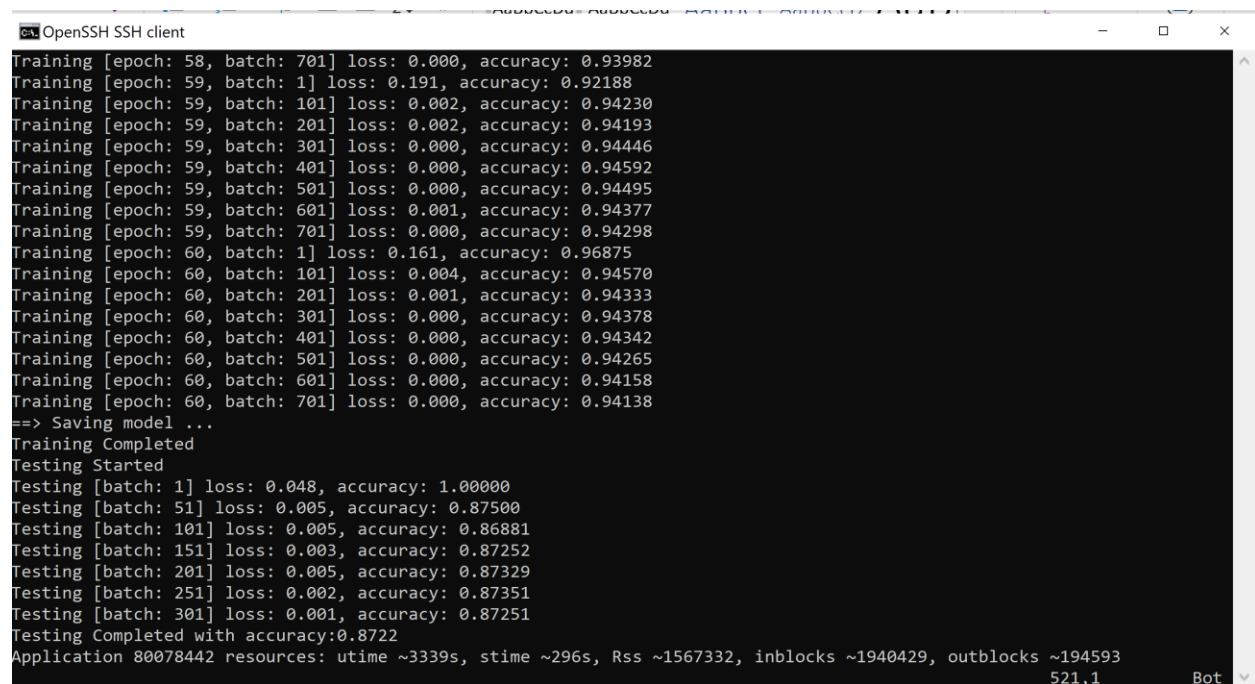
Task: Train a deep convolution network on a GPU with PyTorch for the CIFAR10 dataset.

Goal: Achieve 80-90% accuracy on the test data set.

RESULTS →

Test Accuracy: 87.22%

Train Accuracy: 94.138%



```
OpenSSH SSH client
Training [epoch: 58, batch: 701] loss: 0.000, accuracy: 0.93982
Training [epoch: 59, batch: 1] loss: 0.191, accuracy: 0.92188
Training [epoch: 59, batch: 101] loss: 0.002, accuracy: 0.94230
Training [epoch: 59, batch: 201] loss: 0.002, accuracy: 0.94193
Training [epoch: 59, batch: 301] loss: 0.000, accuracy: 0.94446
Training [epoch: 59, batch: 401] loss: 0.000, accuracy: 0.94592
Training [epoch: 59, batch: 501] loss: 0.000, accuracy: 0.94495
Training [epoch: 59, batch: 601] loss: 0.001, accuracy: 0.94377
Training [epoch: 59, batch: 701] loss: 0.000, accuracy: 0.94298
Training [epoch: 60, batch: 1] loss: 0.161, accuracy: 0.96875
Training [epoch: 60, batch: 101] loss: 0.004, accuracy: 0.94570
Training [epoch: 60, batch: 201] loss: 0.001, accuracy: 0.94333
Training [epoch: 60, batch: 301] loss: 0.000, accuracy: 0.94378
Training [epoch: 60, batch: 401] loss: 0.000, accuracy: 0.94342
Training [epoch: 60, batch: 501] loss: 0.000, accuracy: 0.94265
Training [epoch: 60, batch: 601] loss: 0.000, accuracy: 0.94158
Training [epoch: 60, batch: 701] loss: 0.000, accuracy: 0.94138
==> Saving model ...
Training Completed
Testing Started
Testing [batch: 1] loss: 0.048, accuracy: 1.00000
Testing [batch: 51] loss: 0.005, accuracy: 0.87500
Testing [batch: 101] loss: 0.005, accuracy: 0.86881
Testing [batch: 151] loss: 0.003, accuracy: 0.87252
Testing [batch: 201] loss: 0.005, accuracy: 0.87329
Testing [batch: 251] loss: 0.002, accuracy: 0.87351
Testing [batch: 301] loss: 0.001, accuracy: 0.87251
Testing Completed with accuracy:0.8722
Application 80078442 resources: utime ~3339s, stime ~296s, Rss ~1567332, inblocks ~1940429, outblocks ~194593
521,1 Bot
```

Fig.1: xxxx.bw.out screen shot

Model: A deep convolutional neural network.

Loss/Objective Function: torch's cross entropy loss (SoftMax + log-likelihood)

Optimizer: torch's ADAM optimizer as it takes advantages of both Momentum and RMSDrop.

Hyper-parameters:

Learning Rate	0.001
Epochs	60
Test Batch Size	32
Train Batch Size	64

Layers:

Convolution layer 1: 64 channels, $k = 4$; $s = 1$; $P = 2$.

Batch normalization

ReLU

Convolution layer 2: 64 channels, $k = 4$; $s = 1$; $P = 2$.

Max Pooling: $s = 2$, $k = 2$.

Dropout

Convolution layer 3: 64 channels, $k = 4$; $s = 1$; $P = 2$.

Batch normalization

ReLU

Convolution layer 4: 64 channels, $k = 4$; $s = 1$; $P = 2$.

Max Pooling: $k=2$, $s=2$, non-overlapping

Dropout $p=0.05$

Convolution layer 5: 64 channels, $k = 4$; $s = 1$; $P = 2$.

Batch normalization

ReLU

Convolution layer 6: 64 channels, $k = 3$; $s = 1$; $P = 0$.

Dropout $p=0.05$

Convolution layer 7: 64 channels, $k = 3$; $s = 1$; $P = 0$.

Batch normalization

ReLU

Convolution layer 8: 64 channels, $k = 3$; $s = 1$; $P = 0$.

Batch normalization

ReLU

Dropout $p=0.05$

Fully connected layer 1: 500 units.

ReLU

Fully connected layer 2: 500 units.

ReLU

Dropout $p=0.08$

Fully connected layer 3: 10 units

Notes:

- Followed the neural network structure provided in class.
- Tried experimenting with the dropout rates [0.03, 0.05, 0.1], found that 0.05 was optimal.
- Adding ReLU layers helped improve accuracy, most probably helping with vanishing gradients.

Project Structure:

MP3

- Data
- Checkpoint
- main.py
- ConvNeuralNetModel.py
- run.pbs