CS498 AML,AMO HW10 Report

WORAWICH CHAIYAKUNAPRUK, Worapol Setwipatanachai

TOTAL POINTS

100 / 100

QUESTION 1

1 Part 0 **5 / 5**

✓ **- 0 pts** Correct

QUESTION 2

2 Part 1 **25 / 25**

✓ **- 0 pts** Correct

QUESTION 3

3 Part 2 **35 / 35**

✓ **- 0 pts** Correct

QUESTION 4

4 Part 3 **35 / 35**

✓ **- 0 pts** Correct

QUESTION 5

5 Late penalty **0 / 0**

✓ **- 0 pts** Correct

HW10 Report

Part 0: Imports and Basic Setup (5 Points)

Part 1: Fully connected neural networks (25 Points)

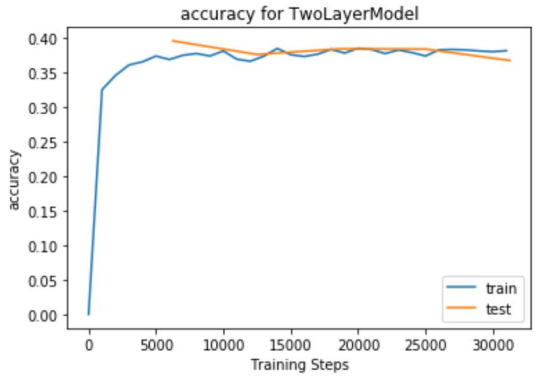
Test (on validation set) accuracy (5 Points): `0.368100`

Test loss (5 Points): `1.747959`

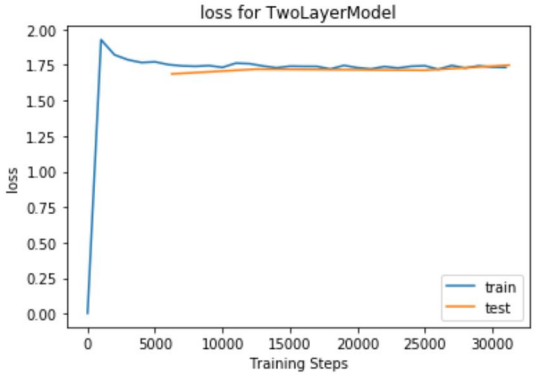
Training time (5 Points): `86.6s`

Plots:

Plot a graph of accuracy on validation set vs training steps (5 Points)



Plot a graph of loss on validation set vs training steps (5 Points)



Part 2: Convolution Network (Basic) (35 Points)

Tensor dimensions: A good way to debug your network for size mismatches is to print the dimension of output after every layer: (10 Points)

Output dimension after 1st conv layer: (8, 16, 32, 32)

Output dimension after 1st max pooling: (8, 16, 16, 16)

Output dimension after 2nd conv layer: (8, 16, 16, 16)

1 Part 0 5 / 5

✓ - 0 pts Correct

HW10 Report

Part 0: Imports and Basic Setup (5 Points)

Part 1: Fully connected neural networks (25 Points)

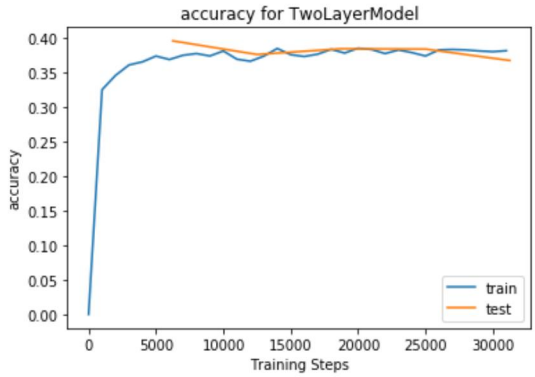
Test (on validation set) accuracy (5 Points): `0.368100`

Test loss (5 Points): `1.747959`

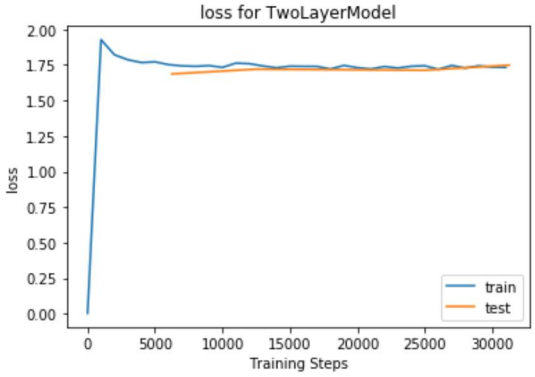
Training time (5 Points): `86.6s`

Plots:

Plot a graph of accuracy on validation set vs training steps (5 Points)



Plot a graph of loss on validation set vs training steps (5 Points)



Part 2: Convolution Network (Basic) (35 Points)

Tensor dimensions: A good way to debug your network for size mismatches is to print the dimension of output after every layer: (10 Points)

Output dimension after 1st conv layer: (8, 16, 32, 32)

Output dimension after 1st max pooling: (8, 16, 16, 16)

Output dimension after 2nd conv layer: (8, 16, 16, 16)

2 Part 1 25 / 25

✓ - 0 pts Correct

HW10 Report

Part 0: Imports and Basic Setup (5 Points)

Part 1: Fully connected neural networks (25 Points)

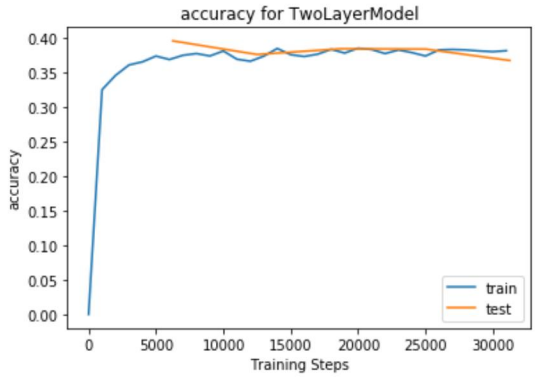
Test (on validation set) accuracy (5 Points): `0.368100`

Test loss (5 Points): `1.747959`

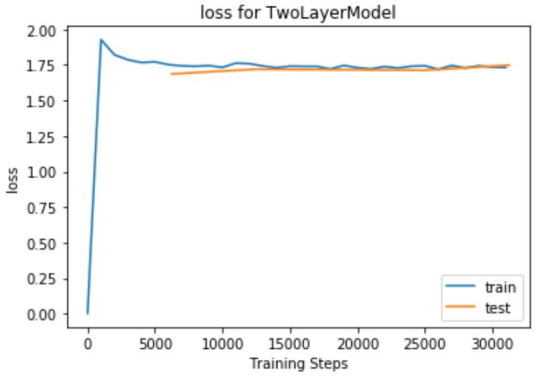
Training time (5 Points): `86.6s`

Plots:

Plot a graph of accuracy on validation set vs training steps (5 Points)



Plot a graph of loss on validation set vs training steps (5 Points)



Part 2: Convolution Network (Basic) (35 Points)

Tensor dimensions: A good way to debug your network for size mismatches is to print the dimension of output after every layer: (10 Points)

Output dimension after 1st conv layer: (8, 16, 32, 32)

Output dimension after 1st max pooling: (8, 16, 16, 16)

Output dimension after 2nd conv layer: (8, 16, 16, 16)

Output dimension after flatten layer: (8, 4096)

Output dimension after 1st fully connected layer: (8, 64)

Output dimension after 2nd fully connected layer: (8, 10)

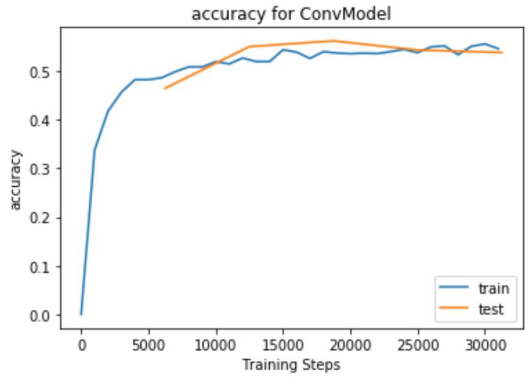
Test (on validation set) Accuracy (5 Points): `0.537800`

Test loss (5 Points): `1.281966`

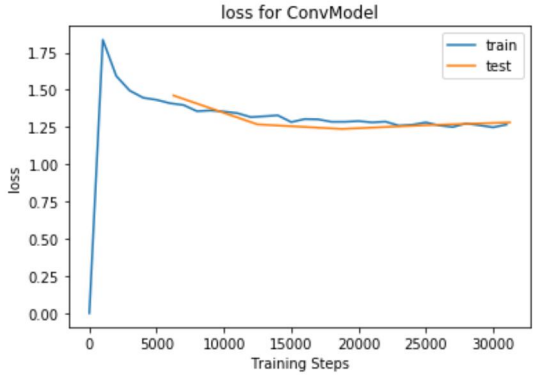
Training time (5 Points): `125.5s`

Plots:

Plot a graph of accuracy on validation set vs training steps (5 Points)



Plot a graph of loss on validation set vs training steps (5 Points)



Part 3: Convolution Network (Add one or more suggested changes) (35 Points)

Describe the additional changes implemented, your intuition for as to why it works, you may also describe other approaches you experimented with (10 Points):

We used BatchNorm2d after each Conv2d layer to normalize data. After two Conv2d-ReLU-BatchNorm2d series, MaxPool2d was added to extract significant information from convolutional data followed by a Dropout layer to decrease overfitting. The whole process was repeated two more times before Flatten() and 2 fully-connected Linear layers. We trained for 60 epochs to ensure maximum test accuracy. However, it seems like the model started overfitting and test accuracy stopped increasing just

3 Part 2 35 / 35

✓ - 0 pts Correct

Output dimension after flatten layer: (8, 4096)

Output dimension after 1st fully connected layer: (8, 64)

Output dimension after 2nd fully connected layer: (8, 10)

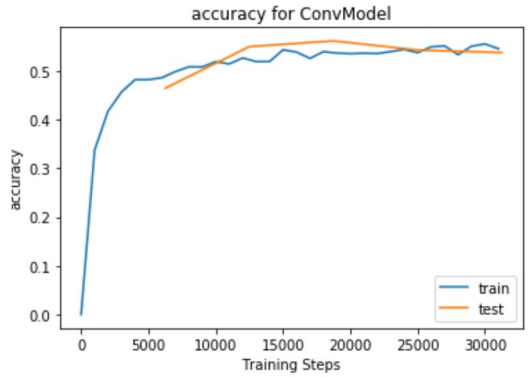
Test (on validation set) Accuracy (5 Points): `0.537800`

Test loss (5 Points): `1.281966`

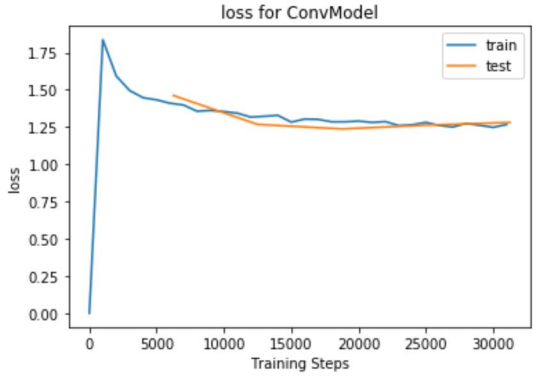
Training time (5 Points): `125.5s`

Plots:

Plot a graph of accuracy on validation set vs training steps (5 Points)



Plot a graph of loss on validation set vs training steps (5 Points)



Part 3: Convolution Network (Add one or more suggested changes) (35 Points)

Describe the additional changes implemented, your intuition for as to why it works, you may also describe other approaches you experimented with (10 Points):

We used BatchNorm2d after each Conv2d layer to normalize data. After two Conv2d-ReLU-BatchNorm2d series, MaxPool2d was added to extract significant information from convolutional data followed by a Dropout layer to decrease overfitting. The whole process was repeated two more times before Flatten() and 2 fully-connected Linear layers. We trained for 60 epochs to ensure maximum test accuracy. However, it seems like the model started overfitting and test accuracy stopped increasing just

after 50,000 steps. The optimizer was also changed to Adam optimizer due to its general effectiveness in reducing loss in computer vision applications. We also experimented on adding another set of convolutional layers and Linear layer, but these approaches did not improve the test accuracy.

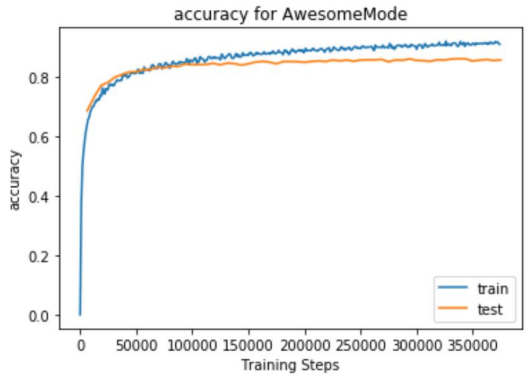
Test (on validation set) Accuracy (5 Points): `0.856400`

Test loss (5 Points): `0.474526`

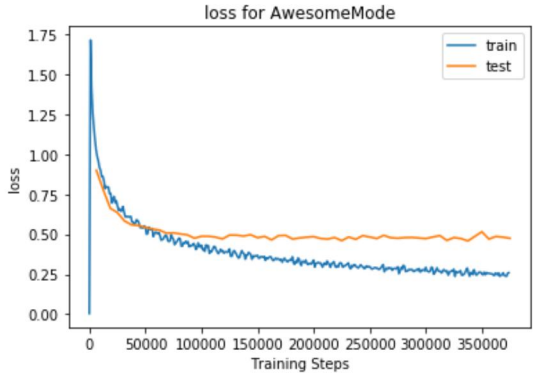
Training time (5 Points): `3282.2s`

Plots:

Plot a graph of accuracy on validation set vs training steps (5 Points)



Plot a graph of loss on validation set vs training steps (5 Points)



4 Part 3 35 / 35

✓ - 0 pts Correct

5 Late penalty **0 / 0**

✓ **- 0 pts** **Correct**