Deep Learning Homework 3
PyTorch CIFAR-10 Image Classification

Yathartha Tuladhar

Part 1: Without Batch Normalization

Maximum validation accuracy: 75.2%
Maximum train accuracy: 96.8%
Minimum validation loss: 0.76
Minimum train loss: 0.095

Epochs = 10
Optimizer = SGD (lr = 0.01, momentum = 0.9)

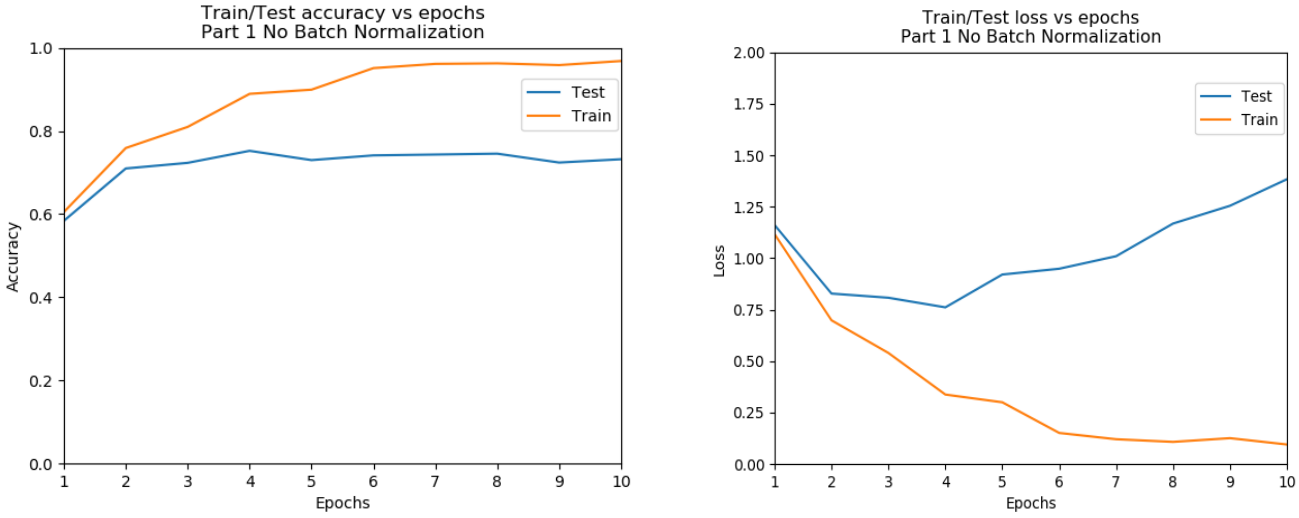

Figure 1: Results for the provided model (Part 1)

The model quickly achieved a steady validation accuracy of about 75% after 4 epochs.
As you can see in the loss in Figure 1 (right), the model starts over-fitting after 4th epoch.

Note: In all the plots in this report, when I say test accuracy, I am actually referring to the validation accuracy.

Part 1: With Batch Normalization

Maximum validation accuracy: 78.5%
Maximum train accuracy: 99.2%
Minimum validation loss: 0.665
Minimum train loss: 0.034

Epochs = 10
Optimizer = SGD (lr = 0.01, momentum = 0.9)

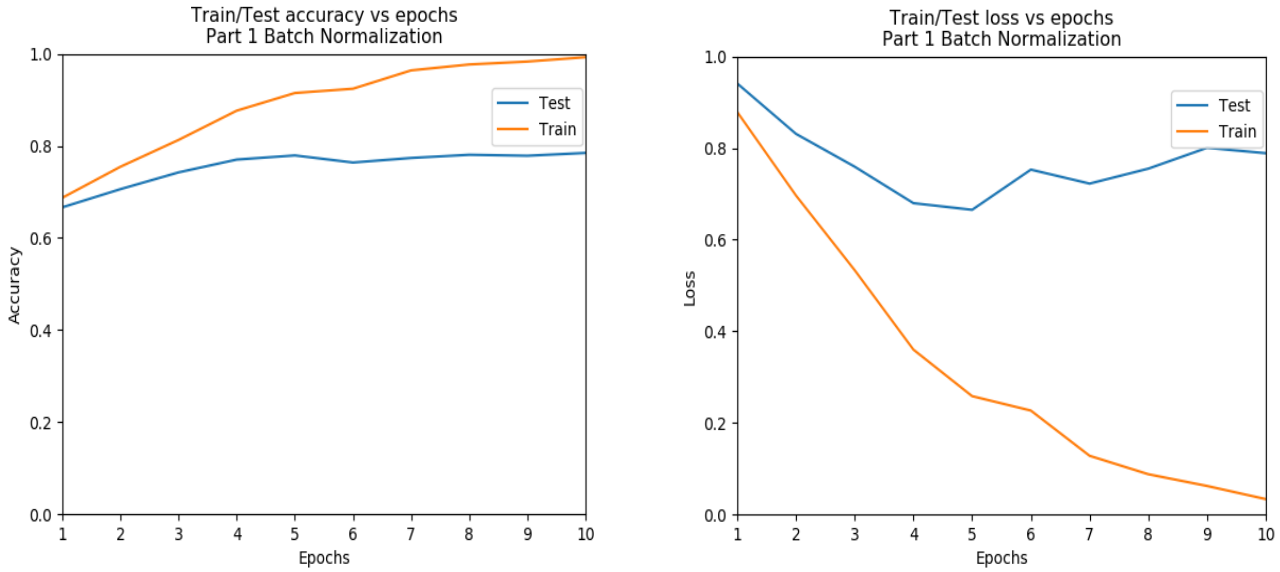

Figure 2: Results for the provided model + Batch Normalization (Part 1)

The model quickly achieved a steady validation accuracy of about 78% after 5 epochs.
As you can see in the loss in Figure 2 (right), the model starts over-fitting after 5th epoch.
There was ~ 3% improvement by adding the Batch Normalization after the first fully connected layer.

Part 2: With added Fully Connected (512) layer, Batch Normalization, Pre-trained Weights

Maximum validation accuracy: 78.3%
Maximum train accuracy: 99.4%
Minimum validation loss: 0.98
Minimum train loss: 0.017

Epochs = 10
Optimizer = SGD (lr = 0.01, momentum = 0.9)

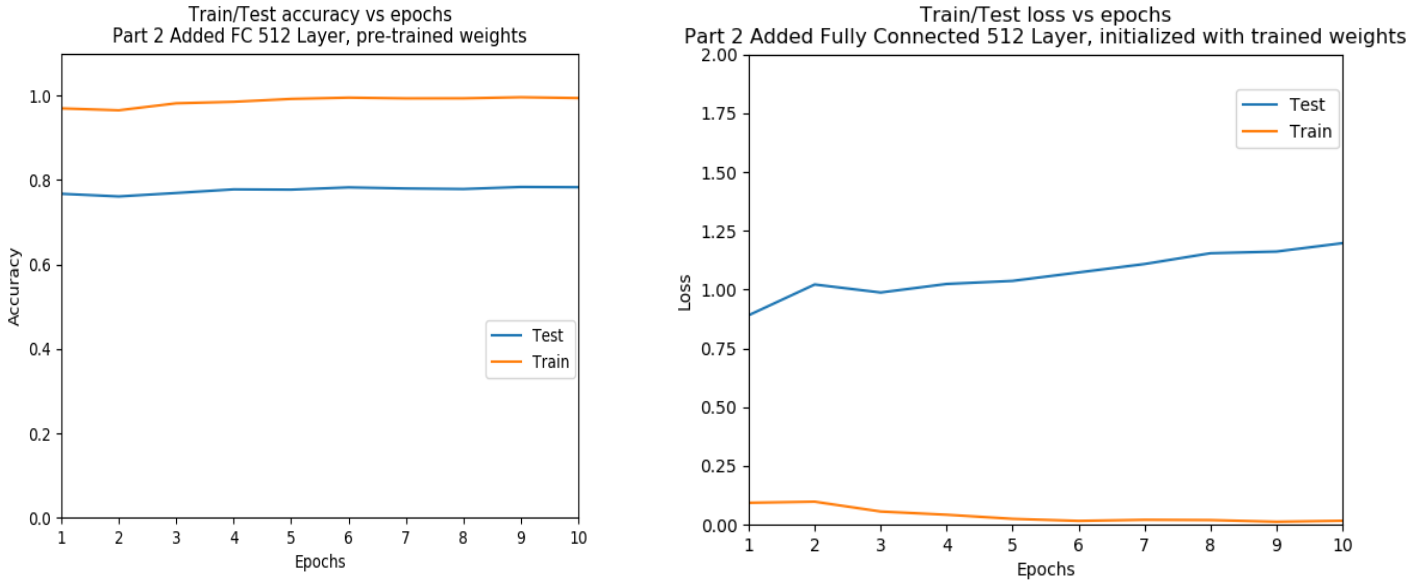

Figure 3: Results for the added Fully Connected layer + Batch Normalization

The model starts with pre-trained weights, thus the validation accuracy is already about 78%.
As you can see in the loss in Figure 3 (right), the model starts over-fitting from the first epoch itself.
There was no improvement by adding a fully connected (512) layer after FC1, compared to that of the original model with added Batch Normalization in Part 1.

Part 3: Model with RMSprop Optimizer

Maximum validation accuracy: 73.7%
Maximum train accuracy: 85.4%
Minimum validation loss: 0.83
Minimum train loss: 0.412

Epochs = 10
Optimizer = RMSprop (lr=1e-2, alpha=0.99)

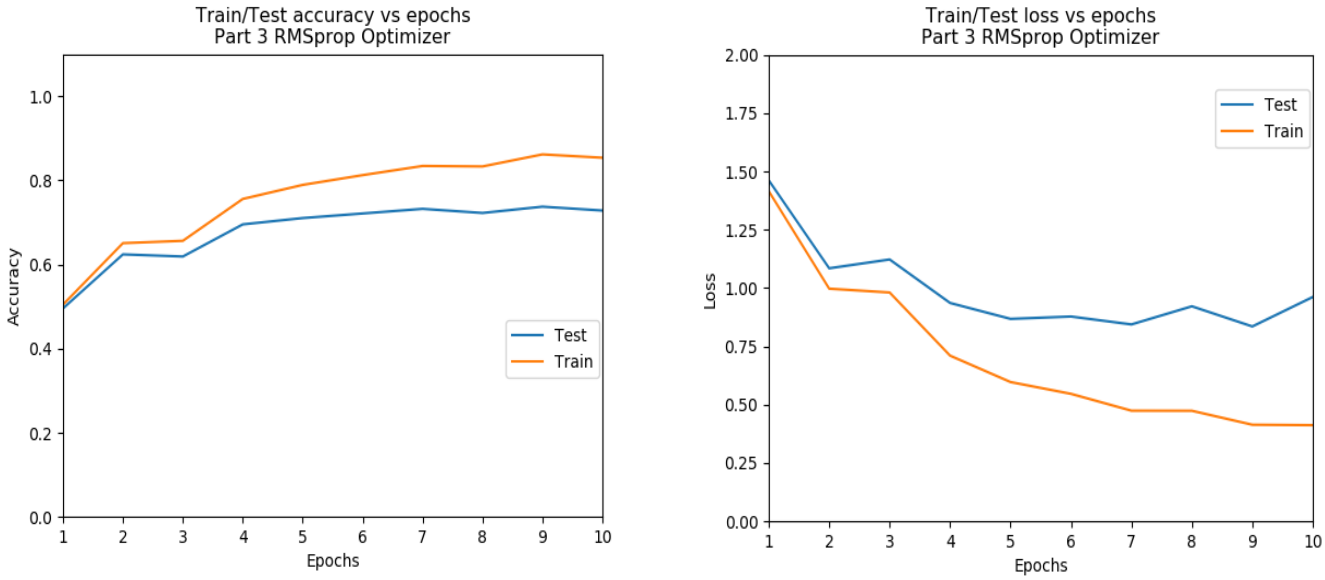

Figure 4: Results for original model with RMSprop as optimizer

The model quickly achieved a steady validation accuracy of about 72% after 6 epochs.
There was no improvement by choosing RMSprop (lr=1e-2, alpha=0.99) as the optimizer.

Part 4: (Modification 1) Model with added Batch Normalization, FC(512), Adam Optimizer, and more Epochs

Maximum validation accuracy: 78.3%
Maximum train accuracy: 99.1%
Minimum validation loss: 0.65
Minimum train loss: 0.02

Epochs = 15
Optimizer = Adam (lr=1e-3, betas=(0.9, 0.999))

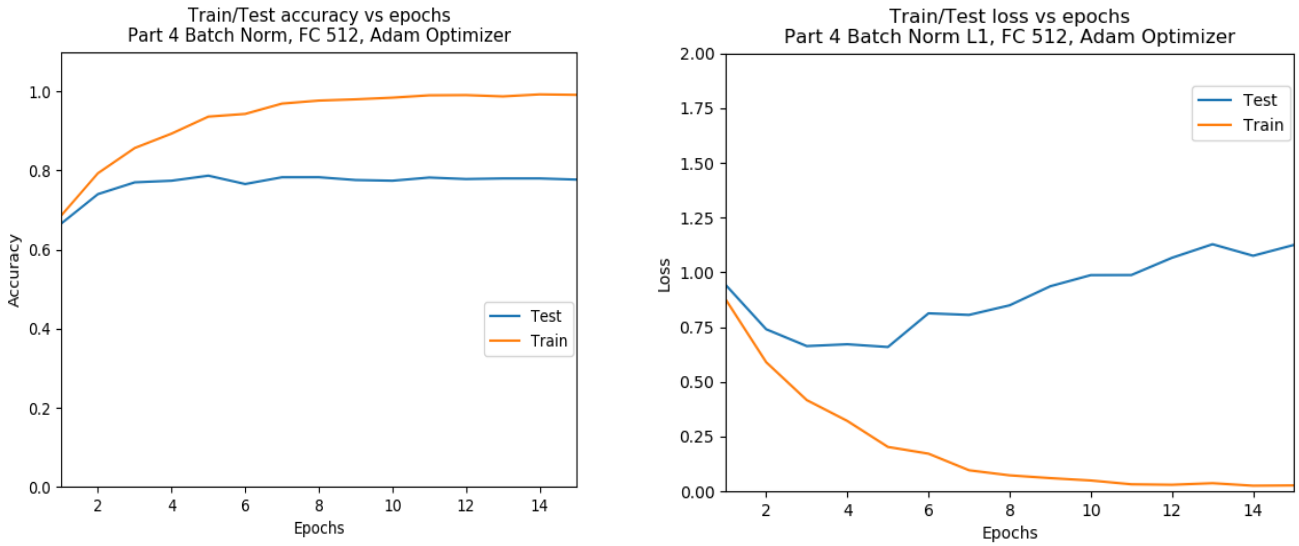

Figure 5: Results for model with added Batch Normalization after FC1, added FC(512), Adam Optimizer and 15 epochs

The model quickly achieves validation accuracy about 77%.
As you can see in the loss in Figure 5 (right), the model starts over-fitting after 5 epochs.
There was no improvement by using more Epochs and Adam optimizer with the given parameters, when compared to that of the original model with added Batch Normalization in Part 1.

Part 4: (Modification 2) Model with two added Convolution Layers + Pooling, Batch Normalization, FC(512), Adam Optimizer, and more Epochs

Two convolutional layers with 128 filters were added (after conv4 in the original model). This was following by max-pooling (2x2)

Maximum validation accuracy: 80.35%
Maximum train accuracy: 98.9%
Minimum validation loss: 0.628
Minimum train loss: 0.034

Epochs = 15
Optimizer = Adam (lr=1e-3, betas=(0.9, 0.999))

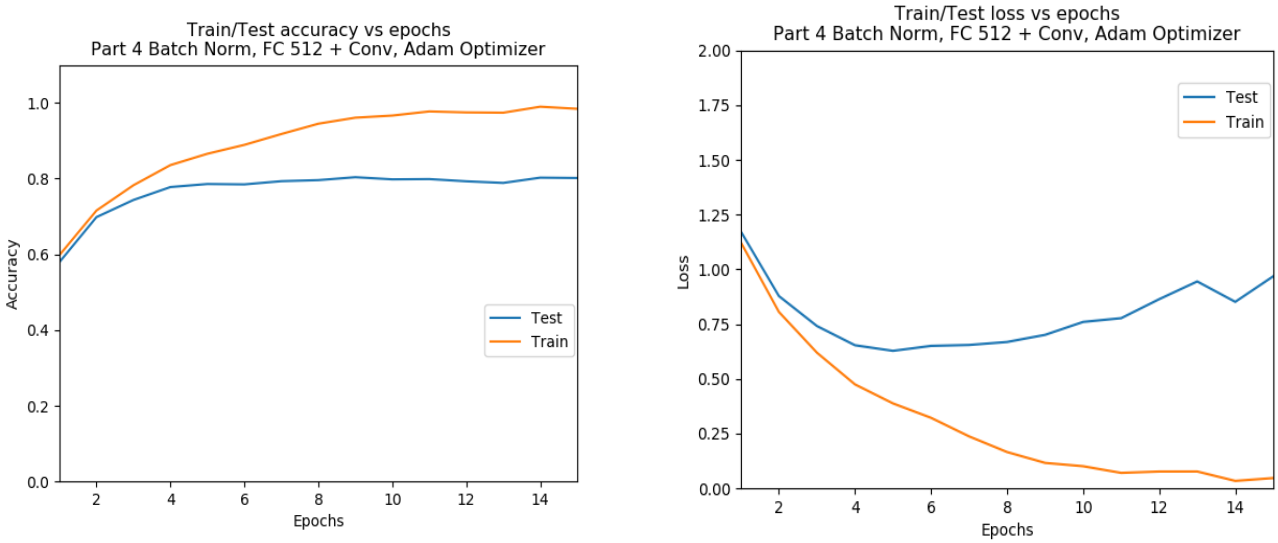

Figure 6: Results for model with added two Convolutional Layers, Batch Normalization after FC1, added FC(512), Adam Optimizer and 15 epochs

The model quickly achieves validation accuracy about 79%.
As you can see in the loss in Figure 6 (right), the model starts over-fitting after 5 epochs.
There was a slight ~2% improvement by adding two convolutional layers with 128 filters each, when compared to that of the original model with added Batch Normalization in Part 1.