Introduction to Deep Learning for Computer Vision
Assignment 7: Convolutional Neural Networks II

Paul Molina-Plant

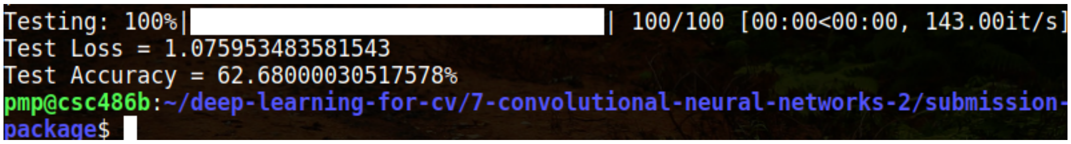
March 19, 2019

Abstract

In this assignment I completed the convolutional neural network implementation from the last assignement and improved the performance.

1 Reproducing default architecture results

Following the assignment specification I finished the implementing the init, conv, and forward pass of the model. Then I verified my testing results matched the expected accuracy of 62.68%.

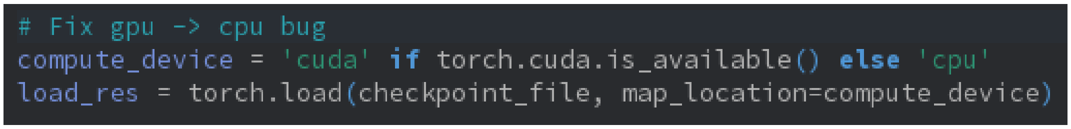


```
Testing: 100%|████████████████████████████████| 100/100 [00:00<00:00, 143.00it/s]
Test Loss = 1.075953483581543
Test Accuracy = 62.68000030517578%
pmp@csc486b:~/deep-learning-for-cv/7-convolutional-neural-networks-2/submission-
package$
```

Figure 1: Testing on provided model.

2 Debugging

Too address the gpu-cpu issue, I added an argument to load the model onto the appropriate computation device for the platform.

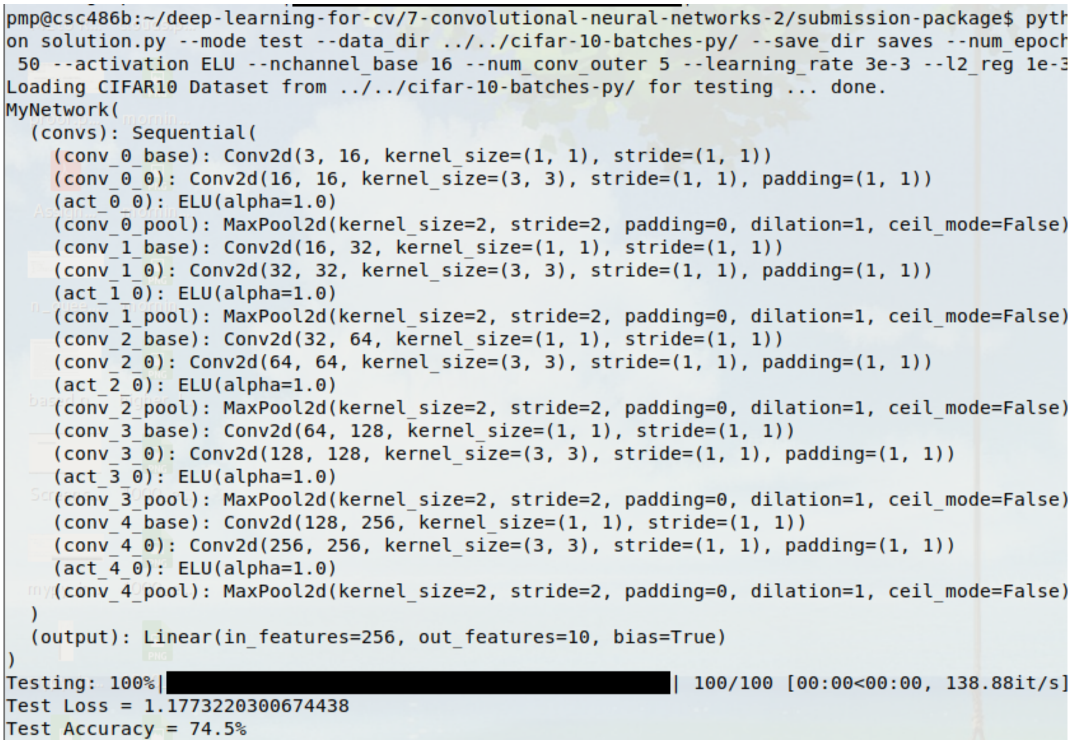


```
# Fix gpu -> cpu bug
compute_device = 'cuda' if torch.cuda.is_available() else 'cpu'
load_res = torch.load(checkpoint_file, map_location=compute_device)
```

Figure 2: Model loads correctly on cpu only platforms.

3 Going beyond

I tried many different architectures when attempting to improve the model's accuracy on the test set. First I tried increasing the iterations, but this plateaued at around 60%. Next I tried adding additional layers and neurons to the network, this peaked around 66%. I tried the ReLU, Leaky ReLU, and ELU activation functions, and found ELU provided the best results. Next I tried increasing the learning rate and l2 regularization to reduce overfitting, this peaked around 70%. Then I added additional input channels and increased the size of the input convolutional layer.
My final model uses ELU for activation, 16 base channels, an outer convolutional size of 5, a learning rate of $3e-3$, and an l2 regularization stength of $1e-3$. The model was trained for 150 epochs. The final accuracy achived was 74.5%.
When testing the model, please note the directory used for the saved model is only "save" with no subdirectory.



```
pmp@csc486b:~/deep-learning-for-cv/7-convolutional-neural-networks-2/submission-package$ pyth
on solution.py --mode test --data_dir ../../cifar-10-batches-py/ --save_dir saves --num_epoch
 50 --activation ELU --nchannel_base 16 --num_conv_outer 5 --learning_rate 3e-3 --l2_reg 1e-3
Loading CIFAR10 Dataset from ../../cifar-10-batches-py/ for testing ... done.
MyNetwork(
  (convs): Sequential(
    (conv_0_base): Conv2d(3, 16, kernel_size=(1, 1), stride=(1, 1))
    (conv_0_0): Conv2d(16, 16, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (act_0_0): ELU(alpha=1.0)
    (conv_0_pool): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (conv_1_base): Conv2d(16, 32, kernel_size=(1, 1), stride=(1, 1))
    (conv_1_0): Conv2d(32, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (act_1_0): ELU(alpha=1.0)
    (conv_1_pool): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (conv_2_base): Conv2d(32, 64, kernel_size=(1, 1), stride=(1, 1))
    (conv_2_0): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (act_2_0): ELU(alpha=1.0)
    (conv_2_pool): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (conv_3_base): Conv2d(64, 128, kernel_size=(1, 1), stride=(1, 1))
    (conv_3_0): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (act_3_0): ELU(alpha=1.0)
    (conv_3_pool): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (conv_4_base): Conv2d(128, 256, kernel_size=(1, 1), stride=(1, 1))
    (conv_4_0): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (act_4_0): ELU(alpha=1.0)
    (conv_4_pool): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  )
  (output): Linear(in_features=256, out_features=10, bias=True)
)
Testing: 100%|████████████████████████████████████████| 100/100 [00:00<00:00, 138.88it/s]
Test Loss = 1.1773220300674438
Test Accuracy = 74.5%
```

Figure 3: Model testing output.

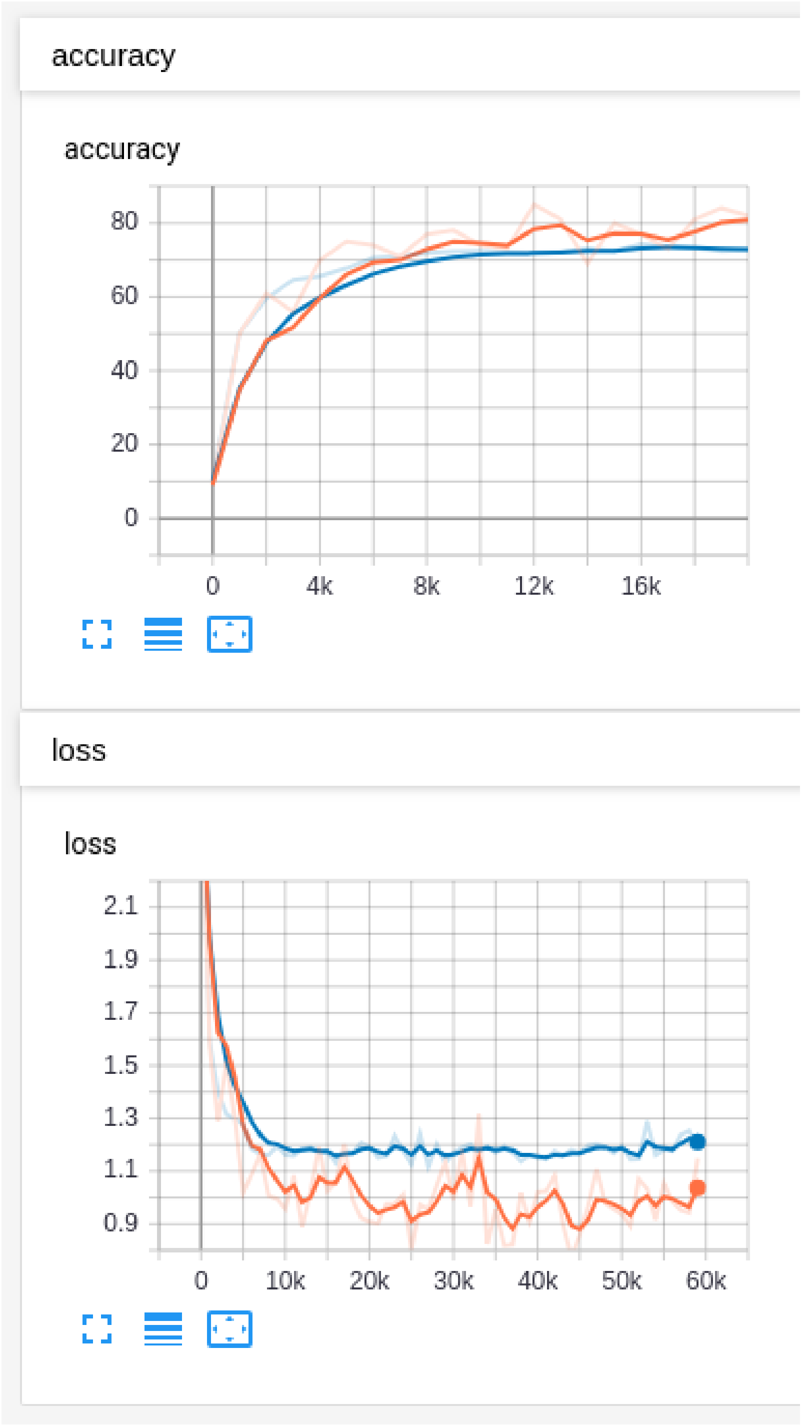


Figure 4: Model training graphs.