Pattern Recognition

HW5: Deep Neural Network

Part 1, Coding

Implementation details

I referenced to Kaiming He's paper about Deep Residual Learning for Image Recognition and implemented a simple ResNet network.
I used PyTorch as my implementation tool and trained my networks on GPUs via Google Colab for weeks.
I augmented the data by flipping the original images. Additionally, I added Gaussian noise to both original and flipped images. Therefore, a total of 20,000 images are used during training.
When predicting an image, I fed in the original image and also the flipped image. The network will predict by choosing the maximum of the torch summed up of the two output torch. This method can improve the network results with a 2% of increase in accuracy on test set.

Hyperparameters

Below are the parameters I have used in this lab. Those values in blue gave better results.
Norm layer: used, not used
batch_size: 128, 1024
learning_rate: 0.01, 0.001, 0.0001, 0.00001
momentum: 0.9
According to the paper, there are 4 types of convolution layers, conv2, conv3, conv4, conv5. The input widths of these layers are 64, 128, 256, 512, respectively. I used a list [a, b, c, d] to represent the count of residual blocks in these layers respectively.
[a, b, c, d]: [1, 1, 1, 1], [1, 1, 2, 1], [3, 4, 6, 3] (ResNet-50), [3, 4, 23, 6] (ResNet-101), [3, 8, 36, 3] (ResNet-152)

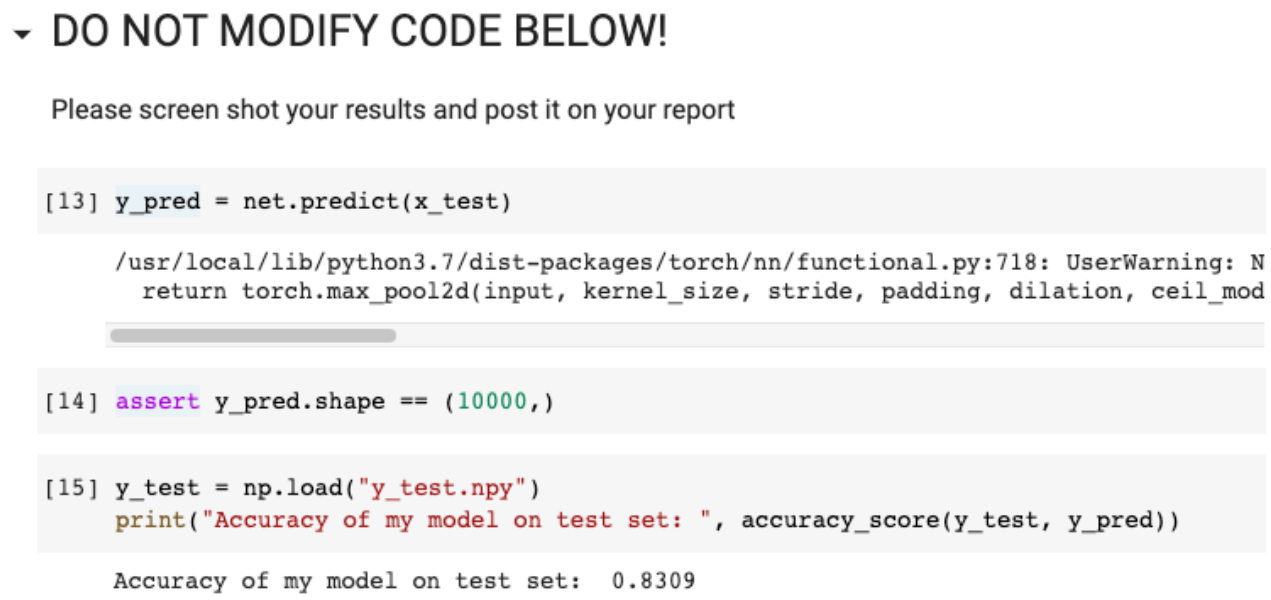
Trained weights

The best trained weights, cifar_resnet_1_1_1_1_RELEASE_8309.pth, is provided in the submitted zip file.
You can also find all trained weights of good results in the link below:
https://drive.google.com/drive/folders/1iM5NR6oikefuGYNiCKKCYB37lRofmEg8?usp=sharing
All the results of the listed checkpoints should be better than 80% of accuracy.
Note that the checkpoints are named under the following format: cifar_resnet_{a_b_c_d}_RELEASE_{acc}.pth
For example, checkpoint cifar_resnet_1_1_1_1_RELEASE_8309.pth was trained by [a, b, c, d] = [1, 1, 1, 1], so you should construct the network by

```
net = ResNet(layer_cnts=[1, 1, 1, 1], class_cnt=10)
```

And the accuracy of this checkpoint should be 83.09%.

Results



Conclusion and some thoughts

This is the first time for me to completely build up a neural network. At first, I constructed an extremely simple toy example network and results in approximately 70% accuracy. I then read the ResNet paper and rebuilt my network based on the structures mentioned in the paper, expecting it to result in better accuracy. The ResNet networks did have better results; however, they all struggled to increase accuracy after reaching 78%. This was not what I was expecting.

Additionally, as I searched for hyper parameters, I realized that the more layers and residual blocks I used, the worse was the accuracy. Since the paper mentioned the opposite, I think that the following reasons should have caused the consequence: insufficient training data, training not yet converged, or my networks happened to have some mistakes.

Training on Google Colab with GPUs is frequently blocked due to exceeding the use of time. Therefore, I could not search for all possible parameters. I hope that I can have access to some GPU resources, have a better understanding toward ResNet architecture, and optimize my current network in the future.