Deep Learning HW2 Code

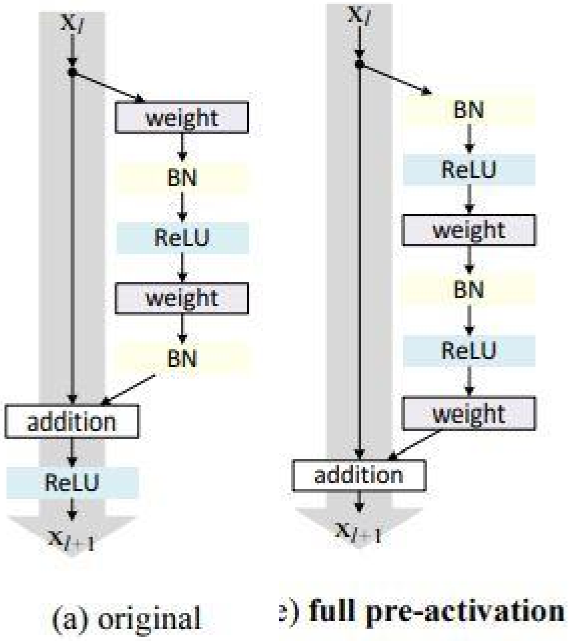
Rohit Sah

6

6.a CIFAR-10 Dataset https://www.cs.toronto.edu/ kriz/cifar-10-python.tar.gz Data Reader Data Reader class uses Pickle to load the files and then iterate over the 5 training files to store in variables x train numpy array of shape [50000, 3072] and labels y train: An numpy array of shape [50000,]. Similary 1 test file is used to store in variables x test numpy array of shape [10000, 3072] and labels y test: An numpy array of shape [10000,].

6.b Here, the [3072] size flattened vector is reshaped to [3, 32, 32]. some of the numpy methods such as pad and flip is used for the data augmentation. For cropping images, random index is used to create a shorter subsection out of the original image.

6.c



(a) original

:) full pre-activation

Network Architecture

Standard Residual Block (resnet size = 3)

ResNet(

```
  (conv1): Conv2d(3, 64, kernel_size=(7, 7), stride=(1, 1), padding=(3, 3), bias=False)
  (batch_norm_relu_start): batch_norm_relu_layer(
    (batch_norm_function): BatchNorm2d(64, eps=1e-05, momentum=0.997, affine=True,
track_running_stats=True)
    (relu_function): ReLU()
  )
  (stack_layers): ModuleList(
    (0): stack_layer(
      (proj_shortcut): Conv2d(32, 64, kernel_size=(1, 1), stride=(2, 2))
      (std_blocks): ModuleList(
        (0): standard_block(
          (conv1): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
          (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
          (batch_norm_function1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
          (batch_norm_function2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
          (relu_function): ReLU()
        )
        (1): standard_block(
          (conv1): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
          (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
          (batch_norm_function1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
          (batch_norm_function2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
          (relu_function): ReLU()
        )
        (2): standard_block(
          (conv1): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
          (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
          (batch_norm_function1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
          (batch_norm_function2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
          (relu_function): ReLU()
        )
      )
    )
    (1): stack_layer(
      (proj_shortcut): Conv2d(64, 128, kernel_size=(1, 1), stride=(2, 2))
      (std_blocks): ModuleList(
        (0): standard_block(
          (proj_shortcut): Conv2d(64, 128, kernel_size=(1, 1), stride=(2, 2))
          (conv1): Conv2d(64, 128, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), bias=False)
          (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
          (batch_norm_function1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
```

```
          (batch_norm_function2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
          (relu_function): ReLU()
        )
        (1): standard_block(
          (conv1): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
          (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
          (batch_norm_function1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
          (batch_norm_function2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
          (relu_function): ReLU()
        )
        (2): standard_block(
          (conv1): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
          (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
          (batch_norm_function1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
          (batch_norm_function2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
          (relu_function): ReLU()
        )
      )
    )
    (2): stack_layer(
      (proj_shortcut): Conv2d(128, 256, kernel_size=(1, 1), stride=(2, 2))
      (std_blocks): ModuleList(
        (0): standard_block(
          (proj_shortcut): Conv2d(128, 256, kernel_size=(1, 1), stride=(2, 2))
          (conv1): Conv2d(128, 256, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), bias=False)
          (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
          (batch_norm_function1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
          (batch_norm_function2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
          (relu_function): ReLU()
        )
        (1): standard_block(
          (conv1): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
          (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
          (batch_norm_function1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
          (batch_norm_function2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
          (relu_function): ReLU()
        )
        (2): standard_block(
          (conv1): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
          (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
          (batch_norm_function1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
          (batch_norm_function2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
          (relu_function): ReLU()
        )
      )
    )
  )
  (output_layer): output_layer(
    (avg_pool): AdaptiveAvgPool2d(output_size=(1, 1))
    (fc_layer): Linear(in_features=256, out_features=10, bias=True)
  )
)
```

Full pre-activation residual block (resnet size = 2)

```
ResNet(
  (conv1): Conv2d(3, 64, kernel_size=(7, 7), stride=(1, 1), padding=(3, 3), bias=False)
  (stack_layers): ModuleList(
    (0): stack_layer(
      (proj_shortcut): Conv2d(32, 64, kernel_size=(1, 1), stride=(2, 2))
      (std_blocks): ModuleList(
        (0): bottleneck_block(
          (bottle_conv1): Conv2d(64, 16, kernel_size=(1, 1), stride=(1, 1), bias=False)
          (bottle_BN1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
          (bottle_BN2): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
          (bottle_conv2): Conv2d(16, 16, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
          (bottle_BN3): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
          (bottle_conv3): Conv2d(16, 64, kernel_size=(1, 1), stride=(1, 1), bias=False)
          (bottle_batch_relu): ReLU()
        )
        (1): bottleneck_block(
          (bottle_conv1): Conv2d(64, 16, kernel_size=(1, 1), stride=(1, 1), bias=False)
          (bottle_BN1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
          (bottle_BN2): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
          (bottle_conv2): Conv2d(16, 16, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
          (bottle_BN3): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
```

```
          (bottle_conv3): Conv2d(16, 64, kernel_size=(1, 1), stride=(1, 1), bias=False)
          (bottle_batch_relu): ReLU()
        )
      )
    )
    (1): stack_layer(
      (proj_shortcut): Conv2d(64, 128, kernel_size=(1, 1), stride=(2, 2))
      (std_blocks): ModuleList(
        (0): bottleneck_block(
          (bottle_conv1): Conv2d(64, 32, kernel_size=(1, 1), stride=(2, 2), bias=False)
          (bottle_BN1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
          (proj_shortcut): Conv2d(64, 128, kernel_size=(1, 1), stride=(2, 2))
          (bottle_BN2): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
          (bottle_conv2): Conv2d(32, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
          (bottle_BN3): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
          (bottle_conv3): Conv2d(32, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
          (bottle_batch_relu): ReLU()
        )
        (1): bottleneck_block(
          (bottle_conv1): Conv2d(128, 32, kernel_size=(1, 1), stride=(1, 1), bias=False)
          (bottle_BN1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
          (bottle_BN2): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
          (bottle_conv2): Conv2d(32, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
          (bottle_BN3): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
          (bottle_conv3): Conv2d(32, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
          (bottle_batch_relu): ReLU()
        )
      )
    )
    (2): stack_layer(
      (proj_shortcut): Conv2d(128, 256, kernel_size=(1, 1), stride=(2, 2))
      (std_blocks): ModuleList(
        (0): bottleneck_block(
          (bottle_conv1): Conv2d(128, 64, kernel_size=(1, 1), stride=(2, 2), bias=False)
          (bottle_BN1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
          (proj_shortcut): Conv2d(128, 256, kernel_size=(1, 1), stride=(2, 2))
          (bottle_BN2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
          (bottle_conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
          (bottle_BN3): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
          (bottle_conv3): Conv2d(64, 256, kernel_size=(1, 1), stride=(1, 1), bias=False)
          (bottle_batch_relu): ReLU()
        )
        (1): bottleneck_block(
          (bottle_conv1): Conv2d(256, 64, kernel_size=(1, 1), stride=(1, 1), bias=False)
          (bottle_BN1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
          (bottle_BN2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
          (bottle_conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
          (bottle_BN3): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
          (bottle_conv3): Conv2d(64, 256, kernel_size=(1, 1), stride=(1, 1), bias=False)
          (bottle_batch_relu): ReLU()
        )
      )
    )
  )
  (output_layer): output_layer(
    (bn_relu): batch_norm_relu_layer(
      (batch_norm_function): BatchNorm2d(256, eps=1e-05, momentum=0.997, affine=True,
track_running_stats=True)
      (relu_function): ReLU()
    )
    (avg_pool): AdaptiveAvgPool2d(output_size=(1, 1))
    (fc_layer): Linear(in_features=256, out_features=10, bias=True)
  )
)
```

6.d.

For crossEntropyLoss, **torch.optim** pytorch package is used. The optimizer takes the parameters we want to update, the learning rate we want to use (and possibly many other parameters as well, and performs the updates through its step() method. **SGD with learning_rate = 0.01,weight_decay=0.0001, momentum=0.9**
For each 100 epoch, the learning rate is reduced by a factor of 10.

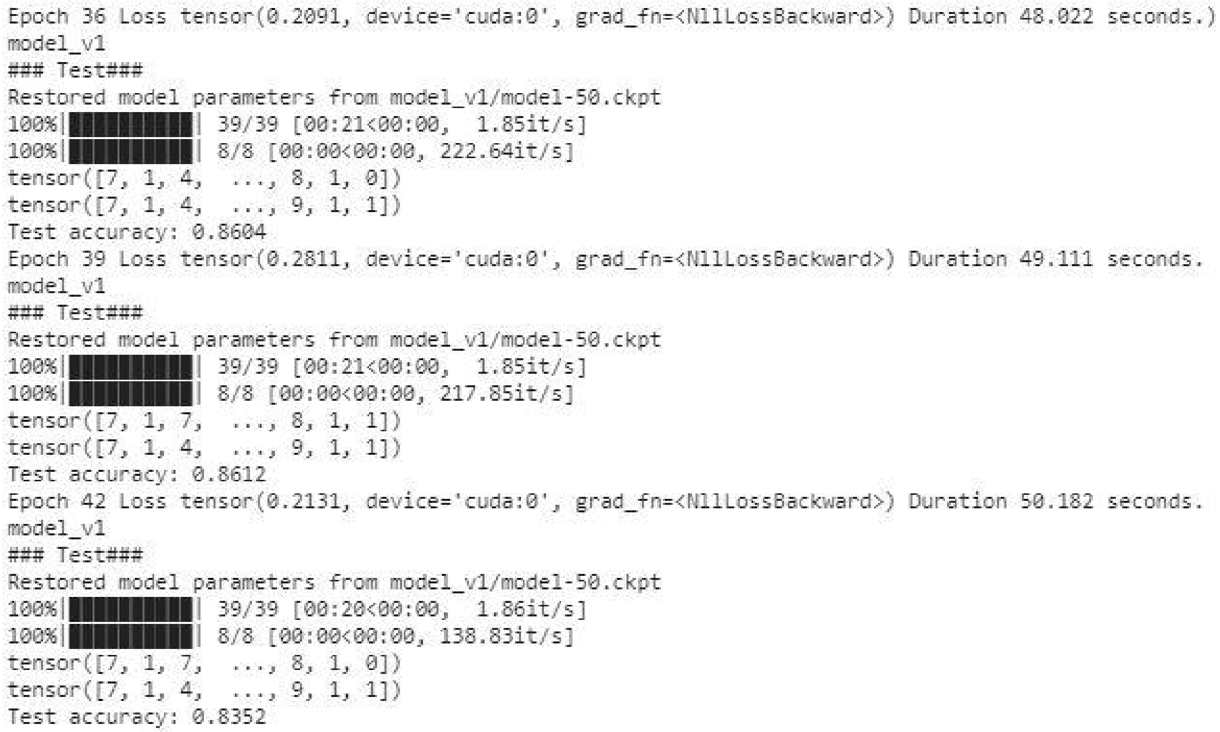
6.e. THe best training accuracy arrives for residual block with resnet size = 18.

Hyperparameters:

Standard residual block

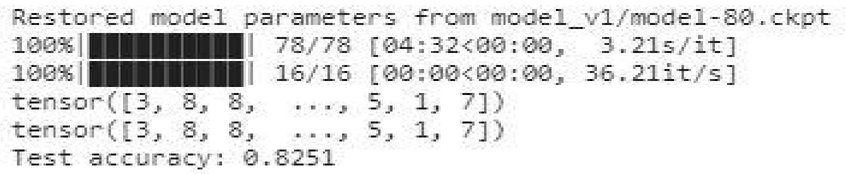
Learning rate: 0.0001
Weight decay: 2e-4
No of epoch - 50

Batch size :128
Resnet size :18



```
Epoch 36 Loss tensor(0.2091, device='cuda:0', grad_fn=<NllLossBackward>) Duration 48.022 seconds.)
model_v1
### Test###
Restored model parameters from model_v1/model-50.ckpt
100%|██████████| 39/39 [00:21<00:00,  1.85it/s]
100%|██████████| 8/8 [00:00<00:00, 222.64it/s]
tensor([7, 1, 4,  ..., 8, 1, 0])
tensor([7, 1, 4,  ..., 9, 1, 1])
Test accuracy: 0.8604
Epoch 39 Loss tensor(0.2811, device='cuda:0', grad_fn=<NllLossBackward>) Duration 49.111 seconds.
model_v1
### Test###
Restored model parameters from model_v1/model-50.ckpt
100%|██████████| 39/39 [00:21<00:00,  1.85it/s]
100%|██████████| 8/8 [00:00<00:00, 217.85it/s]
tensor([7, 1, 7,  ..., 8, 1, 1])
tensor([7, 1, 4,  ..., 9, 1, 1])
Test accuracy: 0.8612
Epoch 42 Loss tensor(0.2131, device='cuda:0', grad_fn=<NllLossBackward>) Duration 50.182 seconds.
model_v1
### Test###
Restored model parameters from model_v1/model-50.ckpt
100%|██████████| 39/39 [00:20<00:00,  1.86it/s]
100%|██████████| 8/8 [00:00<00:00, 138.83it/s]
tensor([7, 1, 7,  ..., 8, 1, 0])
tensor([7, 1, 4,  ..., 9, 1, 1])
Test accuracy: 0.8352
```

Full-pre activation residual block



```
Restored model parameters from model_v1/model-80.ckpt
100%|██████████| 78/78 [04:32<00:00,  3.21s/it]
100%|██████████| 16/16 [00:00<00:00, 36.21it/s]
tensor([3, 8, 8,  ..., 5, 1, 7])
tensor([3, 8, 8,  ..., 5, 1, 7])
Test accuracy: 0.8251
```