BME69500DL ECE695DL
Spring 2020

Homework 3

Deadline : Tuesday, March 3, 2020 , 11:00 am

1 Introduction

The goal of this homework is to gain a deeper understanding of issues related to boundary effects in constructing the convolutional layers on a CNN.

Another important goal of this homework is to understand how to account for the changing resolution in the image array as it is pushed up the resolution hierarchy of a CNN and how to estimate the number of nodes you need in the first fully connected layer.

Yet another goal of this homework is to understand the role played by padding in the convolutional layers.

Your appreciation of these goals would depend on your understanding of the Week 6 lecture material dealing with convolutions in PyTorch.

2 Tasks

First note that the programming part of this homework is meant to be carried out on the CIFAR-10 dataset. Your last task is about registering yourself at the ImageNet website so that you can use those images for future homework assignments.

Here are the tasks for your Homework 3:

Task 1: In the following network, you will notice that we are constructing instance of `torch.nn.Conv2d` in the mode in which it only uses the valid pixels for the convolutions. But, as you now know based on my Week 6 lecture (and slides), this is going to cause the image to shrink as it goes up the convolutional stack. In what has been shown below, this reduction in the image size has already been accounted for when we go from the convolutional layer to the fully-connected layer.

Your first task is to run the network as shown. To make sure that your code is working, at the end of the first epoch, your loss should be roughly 1.137 with a batch size of 4, learning rate of 10^{-3}, and momentum of 0.9. The precise value of your loss at the end of the first epoch would obviously also depend on how the network was initialized by the random number generator.

```
class TemplateNet(nn.Module):
    def __init__(self):
        super(TemplateNet, self).__init__()
        self.conv1 = nn.Conv2d(3, 128, 3)                                  ## (A)
        self.conv2 = nn.Conv2d(128, 128, 3)                                ## (B)
        self.pool = nn.MaxPool2d(2, 2)
        self.fc1 = nn.Linear(28800, 1000)                                  ## (C)
        self.fc2 = nn.Linear(1000, 10)

    def forward(self, x):
        x = self.pool(F.relu(self.conv1(x)))
        ##  Uncomment the next statement and see what happens to the
        ##  performance of your classifier with and without padding.
```

```
        ##  Note that you will have to change the first arg in the
        ##  call to Linear in line (C) above and in the line (E)
        ##  shown below.  After you have done this experiment, see
        ##  if the statement shown below can be invoked twice with
        ##  and without padding.  How about three times?
#       x = self.pool(F.relu(self.conv2(x)))                          ## (D)
        x = x.view(-1, 28800)                                         ## (E)
        x = F.relu(self.fc1(x))
        x = self.fc2(x)
        return x
```

In order to experiment with a network like the one shown above, your training function can be as simple as:

```
def run_code_for_training(net):
    net = net.to(device)
    criterion = torch.nn.CrossEntropyLoss()
    optimizer = torch.optim.SGD(net.parameters(), lr=1e-3, momentum=0.9)
    for epoch in range(epochs):
        running_loss = 0.0
        for i, data in enumerate(train_data_loader):
            inputs, labels = data
            inputs = inputs.to(device)
            labels = labels.to(device)
            optimizer.zero_grad()
            outputs = net(inputs)
            loss = criterion(outputs, labels)
            loss.backward()
            optimizer.step()
            running_loss += loss.item()
            if i % 2000 == 1999:
                print("\n[epoch:%d, batch:%5d] loss: %.3f" %
                            (epoch + 1, i + 1, running_loss / float(2000)))
                running_loss = 0.0
```

where the parameter `net` is an instance of `TemplateNet`.

Task 2: Uncomment line (D). This will now give you two convolutional layers in the network. To make the network function with this change, you must also change the number of nodes in lines (C) and (E). Line (C) defines the first part of the fully-connected neural network that will sit above the convolutional layers. With the two convolutional layers and the changed fully-connected layer, train the classifier again and see what happens to the loss.

Task 3: In the TemplateNet class as shown, we used the class `torch.nn.Conv2d` class without padding. In this task, construct instances of this class with padding. Specifically, add a padding of one to the first convolutional layer. Now calculate the loss again and compare with the loss for the case when no padding was used.

Task 4: Write your own code for calculating the confusion matrix for the testing part of your homework. For the CIFAR-10 dataset, your confusion matrix will be a 10×10 array of numbers, with both the rows and the columns standing for the 10 classes in the dataset. The numbers in each row should show how the test samples corresponding to that class were correctly and incorrectly classified.

Task 5: One or more of your future homework assignments will involve the images from the ImageNet dataset. Please visit the `http://www.image-net.org/` website and register yourself for access to that dataset. It usually takes them a few days to send you the

permission info that you need before you can download any images. In my case, it took them a week, and that too after I sent a reminder to the email address listed for "support".

3 Output Format

In order to evaluate your homework, please use the following lines in your code to initialize the seeds:

```
torch.manual_seed(0)
np.random.seed(0)
torch.backends.cudnn.deterministic = True
torch.backends.cudnn.benchmark = False
```

Construct a file named output.txt and use the following format to structure your output. For the first three lines of the output file, print the losses of tasks 1, 2 and 3. Specifically, print the loss for the 12000th batch of the first epoch. Finally, print the confusion matrix of task 4. Here is a sample of the expected output to help you format your output:

```
[epoch:1, batch:12000] loss: 1.415
[epoch:1, batch:12000] loss: 1.211
[epoch:1, batch:12000] loss: 1.007
tensor([[116.,   5.,  29.,   5.,   9.,   5.,   0.,   6.,  24.,  30.],
        [ 26., 427.,   7.,   4.,   1.,   1.,   3.,   5.,  45., 116.],
        [ 42.,   4., 210.,  24.,  56.,  18.,  24.,  81.,   9.,   3.],
        [ 19.,   6.,  33., 244.,  55.,  48.,  12.,  22.,  11.,   7.],
        [  1.,   0.,   5.,   5., 121.,   5.,   0.,   7.,   0.,   2.],
        [  9.,   1.,  23., 191.,  24., 256.,  18., 106.,   7.,   7.],
        [ 12.,   9.,  77.,  86., 128.,  39., 438.,  17.,   9.,  71.],
        [  4.,   1.,   7.,   3.,  33.,   8.,   1., 305.,   2.,   7.],
        [ 41.,  13.,   6.,   8.,  13.,   5.,   2.,   3., 369.,  17.],
        [ 77.,  25.,   1.,   0.,   2.,   3.,   7.,  11.,  18., 419.]])
```

Note that the numbers you will have in your program will vary from the sample output above.

4 Submission

- Make sure to submit your code in Python 3.x and not Python 2.x.
- Name your main Python file as hw03.py and your output file as output.txt
- All homework to be submitted on-line through the *turnin* command on the min.ecn.purdue.edu[1][2] server.
- Log into min.ecn.purdue.edu server using your Purdue career account.
- From min.ecn.purdue.edu, go to the directory where your homework files are. e.g., if your files are at /home/hw1 directory then go to /home/hw1. Obviously, if you don't have your files on the server, move them there using, for example, the *scp* command.
- Type in the following command:

  `turnin -c ecedl -p hw<double digits hw#> <your files separated by a space>`

[1] If you are an undergraduate student, use the shay.ecn.purdue.edu server instead of the **min** server
[2] If you registered this class as the BME695 course then use the weldon.ecn.purdue.edu server

- As an example, for this homework you will enter the command:

  ```
  turnin -c ecedl -p hw03 hw03.py output.txt
  ```

- You should get a statement that says "Your files have been submitted to ecedl, hw03 for grading". You can verify your submission by typing:

  ```
  turnin -c ecedl -p hw<double digits homework number> -v .
  ```