Deep Learning 2020 **(Due: [25 Nov])**

Submission Assignment #[3]

Instructor: Jakub Tomczak *Name:* [Ho,Yu-Feng], *Netid:* [2689918]

1 Problem statement

- Implement a convolutional neural network as a module class in PyTorch
- Learn the CNN using the ADAM optimizer and analyze the performance of the CNN
- Analyze learning curves during training, and calculate the test classification error.
- Run the ADAM optimizer multiple times and plot an average and a standard deviation of the training loss in each iteration

2 Methodology

- PyTorch
  A Python-based scientific computing package targeted at replacement for NumPy to use the power of GPUs and a deep learning research platform that provides maximum flexibility and speed.
- CNN architecture

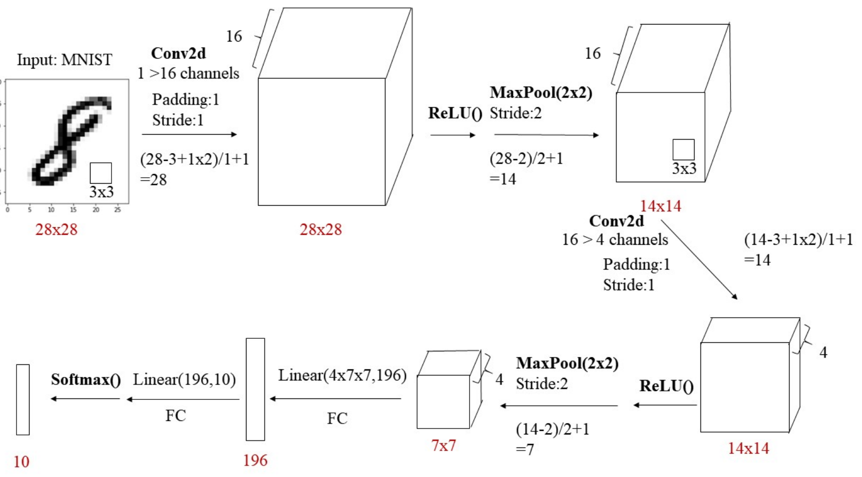

Figure 1: convolutional neural network architecture for assignment 3

- ADAM optimizer
  Adam, adaptive moment estimation is an adaptive learning rate optimization algorithm that's been designed specifically for training deep neural networks. It extracts the concept from *momentum* and combines with *RMSprop*.

$$m^0, v^0 = 0$$

$$W_i^{t+1} = W_i^t - \alpha \cdot \frac{\hat{m^t}}{\sqrt{\hat{v^t}} + \epsilon}$$

where

$$m^{t+1} = m^t \cdot \beta_1 + (1 - \beta_1) \cdot g^t$$

and

$$v^{t+1} = \beta_2 \cdot v^t + (1 - \beta_2) \cdot (g^t)^2$$

and

$$\hat{m^t} = \frac{m^t}{1-\beta_1}, \hat{v^t} = \frac{v^t}{1-\beta_2}, g^t = \frac{\partial L}{\partial w_i}(W^t)$$

3 Experiments

- The influences of different learning rates (step sizes) towards the final performance
- Personally hand-written arabic numeral test

4 Results and discussion

- Basic model training result

  After training the MNIST data, accuracy test is implemented using the *testloader*. The accuracy of the network on the 10000 test images is 97%, and the training loss converges at around 1.495. Furthermore, to better interpret the loss curve, I calculated the mean and standard deviation in each epoch (Figure 2).

- Different learning rates

  In this experiment, I tried four different learning rates (0.01,0.05,1e-4,1e-6) for 5 epochs, and draw the loss curves with mean and standard deviation of mini-batch (Figure 3). Firstly, when learning rate = 1e-4, it performed well during training. The loss dropped steeply at the beginning and slowly converge and stabilize at around 10^th batches. Secondly, regarding bigger learning rates (0.01 and 0.05), the training loss remained stable all the time at 2.4 of training loss. Thirdly, with a lower learning rate of 1e-6, the training loss started to decline at around 15^th batch and did not converge when the training process finished.

  ADAM can mitigate oscillation, escape local optimum, and adjust the learning rate to speed up trainingBock et al. (2018). In my cases, the four different learning rates expressed three patterns. On the one hand, higher learning rate would be inclined to learn faster but converge at higher training loss as a side effect. Therefore, to look into the graph of learning rate =0.01 and 0.05. Their training process might end with one batch and converge at higher loss in the end. On the other hand, lower learning rate causes to learn slowly at the beginning. It is obvious in the example of learning rate = 1e-6. However, Adam optimizer is proved to adjust learning rate, and it might be the reason that the training process took bigger strides forward after 15 batches.

- Hand-written arabic numeral test Based on my model with 97% up accuracy. I generated a hand-written numeral by myself and processed forward to predict or identify a certain numeral. The numeral I put in is actually "1", but the model identified as 3 (Figure 4). Besides, I also made a visual evaluation, compared the out-of-range numeral and the numeral inside the MNNIST dataset, and then the model can give the correct answer to each example (Figure 5). From this test, we could know that after the well-training with high accuracy, the model still cannot identify the data which is out of range but is rather understandable for human.

- Plot result

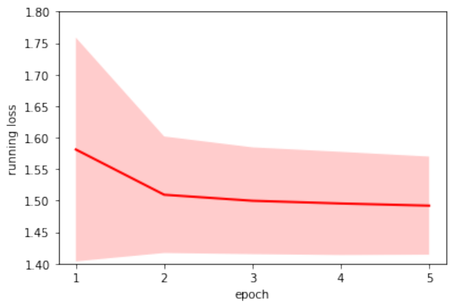

Figure 2: Learning curves during training with mean and standard deviation for 5 epochs (learning rate = 1e-4)

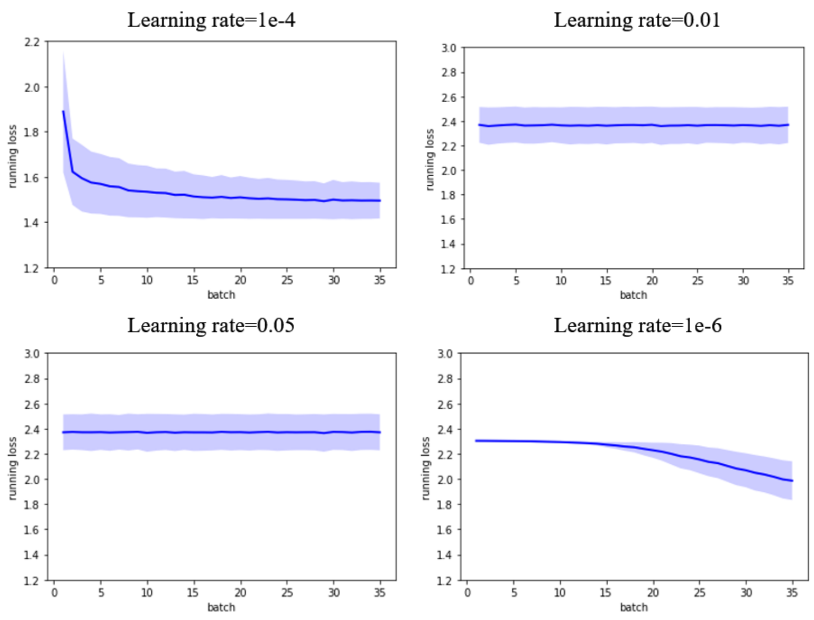

Figure 3: Training loss curves according to four different learning rates

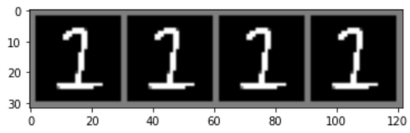

Figure 4: Hand-written arabic numeral by myself and its prediction by CNN model

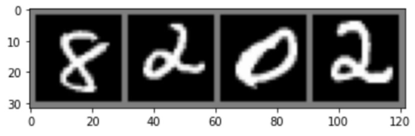

Figure 5: Hand-written arabic numerals from MNIST and their prediction by CNN model

5 A code snippet

- CNN architecture

```
class Net(nn.Module):

    def __init__(self):
        super(Net, self).__init__()
        # 1 input image channel, 16 output channels, 3x3 square convolution
        # kernel
        self.conv1 = nn.Conv2d(1, 16, 3,padding =1)
        self.conv2 = nn.Conv2d(16, 4, 3,padding =1)
        # an affine operation: y = Wx + b
        self.fc1 = nn.Linear(4 * 7 * 7, 196)  # 6*6 from image dimension
        self.fc2 = nn.Linear(196, 10)

    def forward(self, x):
        # Max pooling over a (2, 2) window
        x = F.max_pool2d(F.relu(self.conv1(x)), (2, 2))
        x = F.max_pool2d(F.relu(self.conv2(x)), (2, 2))
        x = x.view(-1, self.num_flat_features(x))
        x = self.fc1(x)
        x = F.softmax(self.fc2(x),dim=1)
        return x

    def num_flat_features(self, x):
        size = x.size()[1:]  # all dimensions except the batch dimension
        num_features = 1
        for s in size:
            num_features *= s
        return num_features
```

- Loading and adjustment of personal hand-written numeral (picture)

```
# Use dataloader to load my handwritten picture(.png)
mydataloader =torch.utils.data.DataLoader(mydata, batch_size=4,
                                          shuffle=True, num_workers=0)
dataiter = iter(mydataloader)
image_raw, labels = dataiter.next()
image_rightsize = F.interpolate(image_raw, size=28)
#change the tensor size to the feed-in data size of CNN
image = image_rightsize[0][0].unsqueeze(0).unsqueeze(0).expand(4,1,28,28)
# show images
imshow(torchvision.utils.make_grid(image))

net = Net()
print(net)
```

References

Bock, S., Goppold, J., and Weiß, M. (2018). An improvement of the convergence proof of the adam-optimizer. *arXiv preprint arXiv:1804.10587*.