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

Submission Assignment #3

Instructor: Jakub Tomczak *Name:* Asif Anwar, *Netid:* aar216

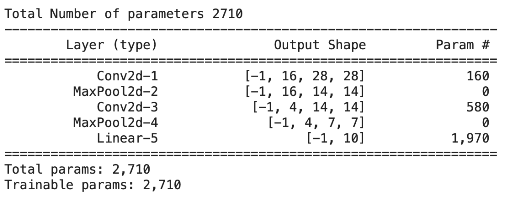
The following section layout is a suggestion for the structure of the report. If it doesn't quite fit, you can use whatever structure does work.

1 Part 1

Question 1 *The experiment has been implemented in Python and pytorch library as mentioned in the guide. The implementation is done in Jupiter notebook in google Colab environment.*

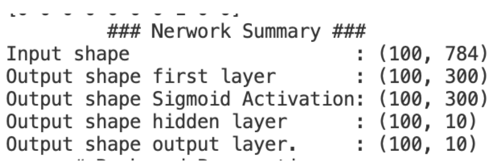
Question 2 *The network the element-wise application of f' applied to the elements of X, multiplied by the gradient of the loss with respect to the outputs.*

Question 3 *The total trainable parameters of the CNN are 2,710 where the total number of weights are 1,970. The detials of the network can be seen below in Figure 1. On the other hand the total number of parameters of the network created in assignment 1 had 238,510 parameters including the biases, the total number of weights in the linear network had 238,200. We can see that the convolution network has really small number of parameters in compare to linear network*



```
Total Number of parameters 2710
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1           [-1, 16, 28, 28]             160
         MaxPool2d-2           [-1, 16, 14, 14]               0
            Conv2d-3            [-1, 4, 14, 14]             580
         MaxPool2d-4              [-1, 4, 7, 7]               0
            Linear-5                   [-1, 10]           1,970
================================================================
Total params: 2,710
Trainable params: 2,710
```

(a) CNN



```
         ### Nerwork Summary ###
Input shape                       : (100, 784)
Output shape first layer          : (100, 300)
Output shape Sigmoid Activation: (100, 300)
Output shape hidden layer         : (100, 10)
Output shape output layer.        : (100, 10)
```

(b) Linear Network

Figure 1: Weights compression of two network

Question 4 *As instructed adam optimizer has been used*

Question 5 *MNIST dataset has been used for the experiment*

2 Part2: Problem statement

The goal of this experiment is to analyse the performance of the CNN Network by using ADAM optimizer over MNIST Dataset. Below are the items covered in this experiment

- Analyzing learning curve and calculate the test classification error.
- Finding Average μ and standard deviation σ of Loss and Accuracy
- Analyzing in performance of network for change in learning rate of ADAM optimizer.

3 Methodology

Overview: *To perform the experiment a three layer network was designed. The first two layers of the network are convolution layers. The relu activation functions were used to add nonlinearity to these layers. Then 2D maxpooling layers are added after each convolution layers. The last layer is a linear layer for the output. Softmax function is used for determining the output. The network was then trained over MNIST dataset.*

Strategy *The training script was build based on the given to perform the experiment. Three functions were created to perform each part of the experiment.* ***fit function*** *was created to training the model. This function takes few parameters like number of epochs, learning rate.* ***test_eval and valid_eval*** *functions were created find the test and validation loss and accuracy. Below strategies were followed to perform the experiment*

- Train the network with a selected running rate and verify the training and validation loss. Validation accuracy was also calculated for further analyse. Then accuracy of the test set has been checked.
- Running the network by calling **fit function** multiple times to get the mean μ and standard deviation σ of the training loss and accuracy.
- A script was developed to use different step size and then the impact was analyzed.

4 Experiments

CNN with ADAM optimizer: *The experiment scrip was designed to perform training over MNIST dataset using Adam optimizer. For the testing below steps were taken.*

- **Dataset:** for the testing the MNIST dataset was loaded into trainload, validload and testload. Where train and validation sets were the 80/20 split of the train dataset. The test dataset was a separate set of 1000 instances.
- **Optimizer:** as instructed ADAM optimizer was used for the training using **learning rate = 0.01**.
- **Epochs:** total 3 epochs were used for train the experiment.
- **Loss:** the negative logarithm of the likelihood function was used to determine the training, validation and test losses.
- **Evaluation:** the loss and accuracy value of the validation set was used for training evaluation. The measurements were taken after each 10 iterations. And after the training completion the accuracy was measured using test dataset.

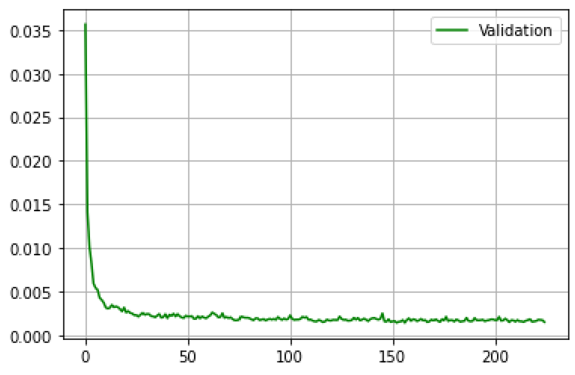
Analyzing mean loss and standard deviation: *The impact of initial performance of the network due to random initialization, the network was trained for 25 batches with different seeds(to reproduce the randomness). Then ther performance was measured by analyzing the mean of loss and its standard deviation by plotting in a alpha plot.*

Analyzing Learning rate: *To Analyze the learning rate step sizes of 0.00001, 0.0001, 0.001, 0.01 and 0.05 were considered. The network was training for one epochs and validation dataset was used to verify the loss and accuracy of the training.*

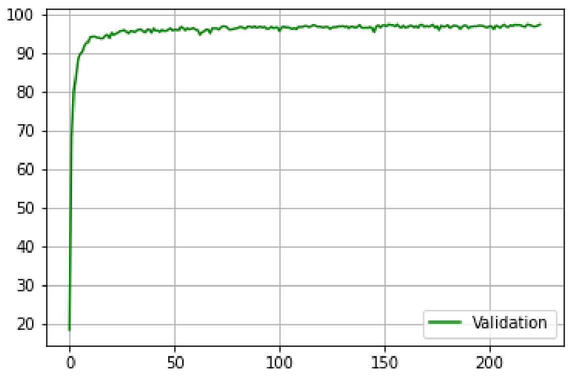
5 Results and discussion

Analyzing Lerning Curve *As mentioned in the Methodology the designed CNN was trained using a trainset and performance was verified using a validation set. From the the figure 2 we can observed that validation set performs really well for the ADAM optimizer with 0.01 step size. In the figure 2(a) we can notice that after only 40 batches the validation loss drops rapidly from 0.035 to 0.0025. however, after 50 batches the validation loss does not improve much. On the other hand validation accuracy also raises rapidly after only 20 batches and remain around 92% to 96% 200 batches. Accuracy slightly increases to 97% after 200 batches in epoch 3.*

*The result of the test set also performs really well with this network with current setup. The test accuracy reaches to **97%** where the rate was only 10% before the training.*



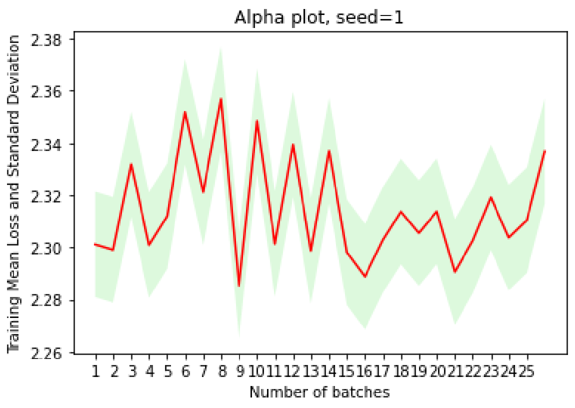
(a) Validation loss



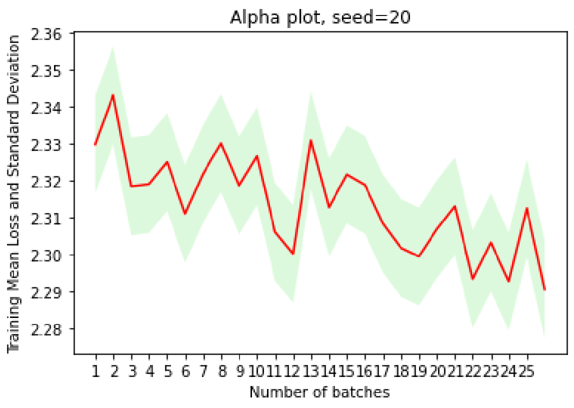
(b) Validation accuracy

Figure 2: Loss and Accuracy over 3 epochs

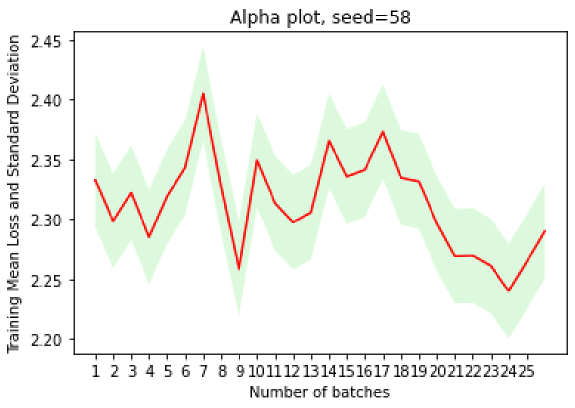
Analyzing mean loss and standard deviation: *In figure 3, the average loss is plotted with respective standard deviation. It can be observed that given the same parameters, the initial average training loss fluctuate a lot. Although for seed 20 and 94 we can observed the downwards train, but for seed 1 and 58 the trend is more random. however from the training results of different learning rate in next experiment we can see that despite of the initial weights, the network trend to overcome the local minima and achive targeted performance standard given proper step size.*



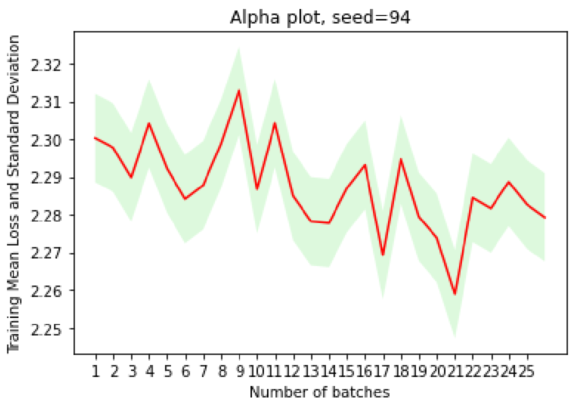
(a) Validation loss



(b) Validation accuracy



(c) Validation loss



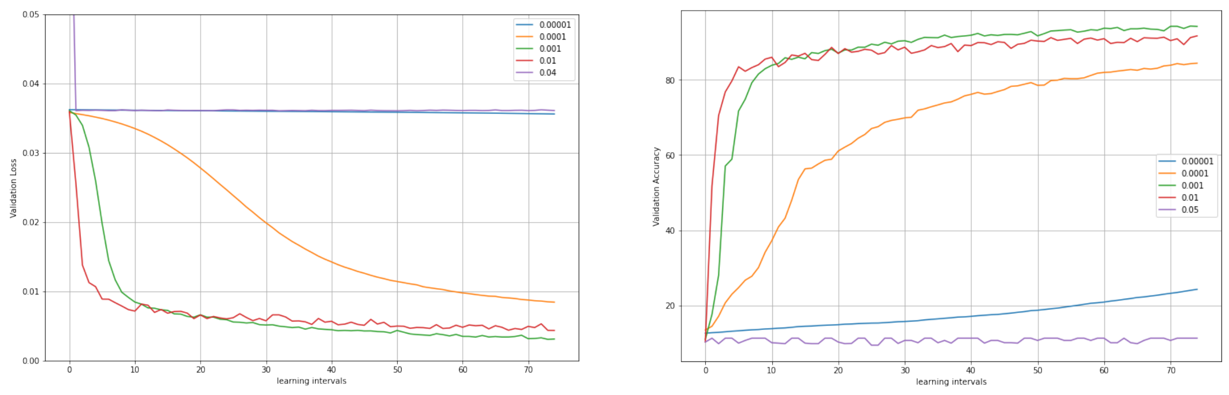
(d) Validation accuracy

Figure 3: Mean and standard deviation of training loss

Analyzing Learning rate *The impact of different step size was vital for the training. In the figure 4 we can observed that there are major changes in both validation loss and accuracy due to change in the step size of the ADAM optimizer. More details are explained below for each step size.*

- **0.00001:** the learning rate for step size 0.00001 is really slow. we can observed that in the figure 4. The loss curve is almost flat at 0.035 and the accuracy curve slightly have upload trend in the end.
- **0.0001:** the performance of the step size of 0.0001 is quite better than 0.00001. We can see that the validation loss smoothly reduces to 0.009 and validation accuracy raises to above 80
- **0.001:** the step size of 0.001 for adam optimizer does the best for this network. we can see that the validation loss is quite smooth and rapidly drops and keep going downwards gradually. Similarly the validation accuracy for this step size is also shows the best results. The accuracy raises rapidly and finally gets around 97% accuracy' which is the heights.
- **0.01:** Like 0.001 the step size 0.01 also performs well in both validation loss and accuracy. The initial performance of this step size is the best, however soon it falls behind 0.001.

Outcome: From all the results we can determine that step size of 0.001 performs the best among all the tested step size for ADAM optimizer given the designed network for MNIST dataset.



(a) Loss per epoch (b) Accuracy epochs

Figure 4: Loss and Accuracy for Momentum

6 A code snippet

6.1 Network Design

```
class NET(nn.Module):
    def __init__(self):
      super(NET, self).__init__()
      self.conv1 = nn.Conv2d(1, 16, kernel_size=3, stride=1, padding=1)
      self.conv2 = nn.Conv2d(16, 4, kernel_size=3, stride=1, padding=1)
      self.fc1   = nn.Linear(196, 10)
      self.pool  = nn.MaxPool2d(2,2)
    def forward(self, x):
      x = self.pool(F.relu(self.conv1(x)))
      x = self.pool(F.relu(self.conv2(x)))
      x = x.view(-1, 196)
      x = self.fc1(x)
      x  = F.log_softmax(x, dim=1)
      return x
    def count_parameters(self):
      return sum(p.numel() for p in self.parameters() if p.requires_grad)
```

References

NA

Appendix

The Jupiter Notebook of the experiment is published in the below GIST for easier viewing.
https://colab.research.google.com/gist/tamal2000/5b63672cee91a556f30f64ca987a6839/dl-assignment-3-final.ipynb