CPSC 483 Assignment-4 Report - Tommy Vu
Assignment #4

This assignment can be completed individually or by a team.
Total score: 100
Due date: 12/11

Objective: Handwritten Digit Recognition

In this assignment, you are asked to design and develop artificial neural network models that can classify handwritten digits from the MNIST dataset. The MNIST dataset consists of 60,000 training images and 10,000 testing images. Each image is a small 28x28 (784) pixels grayscale digit between 0 and 9. You can use a Python library to download this dataset.

From this assignment, you will also learn how to process and compute images, and design multilayer backpropagation perceptron (MLP) and Convolutional Neural Network (CNN) architectures.

Write an analysis report that describes the following elements:

MLP Model

(1) For your best MLP model:
(a) # of layers: 10
(b) the total number of neurons: 50
(c) activation function(s): reLU
(d) loss function(s): CrossEntropyLoss
(e) gradient methods (Adagrad, Adadelta, RMSprop, Adam): Stochastic Gradient Descent (SGD)
(f) hyperparameters (learning rate, momentum):
Learning Rate
Number of Epochs
Hidden Layers and Hidden Units
Activations Functions

(g) training methods (pretraining,sequential training,batch training,SGD,minibatch):
Stochastic Gradient Descent (SGD)

(h) training time: Depends on method used
(i) the accuracy in percentage: 100 samples, 80 correct prediction, 0.8 or 80% accuracy.
(j) brief justification of why your choice of design and parameters are the best for your model:

For my MLP model, the number of layers used are 10 with a total of 50 neurons. The activation function used is reLU as there are hidden layers. The loss function utilized is CrossEntropyLoss and the gradient method is Stochastic Gradient Descent (SGD). SGD is used for both training and gradient methods because it is more efficient than others if our model is not too complex and has fewer features. The hyper parameters used are learning rate, number of epochs, hidden layers/units and activation functions. Training time will vary based on algorithms such as Adam or SGD. As mentioned before, SGD is more efficient if the model is not too complex, but if there are large data points, then other methods such as Adam or Nadam are better due to faster convergence. The accuracy for the MLP model is 100 samples with 80 correct predictions, so 0.8 or 80% accuracy.

```
.............Training MLP.................
...Modeling using CPU...
Epoch=1/1, batch=100/600, loss=2.138658046722412
Epoch=1/1, batch=200/600, loss=2.0081803798675537
Epoch=1/1, batch=300/600, loss=1.7423105239868164
Epoch=1/1, batch=400/600, loss=1.6005222797393799
Epoch=1/1, batch=500/600, loss=1.471463680267334
Epoch=1/1, batch=600/600, loss=1.2211238145828247
.............Testing MLP model.................
> Input digits:
tensor([8, 2, 4, 2, 5, 9, 7, 1, 6, 8, 9, 2, 2, 7, 3, 3, 4, 4, 3, 1, 1, 9, 5, 0,
        8, 6, 9, 4, 5, 2, 8, 2, 9, 1, 4, 4, 6, 3, 0, 5, 5, 5, 1, 1, 2, 8, 9, 7,
        3, 5, 5, 1, 8, 9, 7, 7, 7, 5, 0, 4, 4, 5, 1, 7, 2, 4, 2, 5, 0, 8, 1, 3,
        7, 7, 6, 7, 9, 9, 3, 6, 0, 4, 2, 7, 7, 4, 5, 7, 7, 8, 8, 1, 6, 5, 6, 2,
        5, 5, 4, 8])
batch=100/100
> Number of samples= 100 number of correct prediction= 80 accuracy= 0.8
```

(2) For your best CNN model,
(a) # of convolutional layers: 4
(b) the kernel (filter) size: Ranges from 3-5
(c) the stride: 1
(d) # of filters: 32
(e) # of pooling layers: 4
(f) the pooling size: 32
(g) pooling method (min, max, avg): (1, 2, 3)
(h) the number of layers and neurons for the fully connected layer: 10 layers, 50 neurons
(i) activation function(s): reLU
(j) loss function(s): CrossEntropyLoss
(k) gradient methods (Adagrad, Adadelta, RMSprop, Adam): Adam
(l) hyperparameters (learning rate, momentum):
Learning Rate
Number of Epochs
Hidden Layers and Hidden Units
Activations Functions
(m) training methods (pretraining, sequential training, batch training, SGD, minibatch):
Stochastic Gradient Descent (SGD)
(n) training time: Training Time 23.34 seconds

(o) the accuracy in percentage Number of samples= 100 number of correct prediction= 97 accuracy= 0.97
(p) brief justification of why your choice of design and parameters are the best for your model:

For my CNN model, the number of convolutional layers is 4. The kernel (filter) size ranges from 3-5. The stride is 1. Number of filters is 32 and # of pooling layers is 4. The min, max and average is (1,2,3). The number of layers and neurons for the fully connected layer is 10 and 50. The activation functions are similar to the MLP model, which are reLU and the loss function used is CrossEntorpyCross. Hyper parameters are Learning Rate, Number of Epochs, Hidden Layers and Hidden Units, and Activations Functions. Training method used is Stochastic Gradient Descent (SGD). Training time received is 23.34 seconds and accuracy is 97% which is very efficient.

```
............Training CNN.................
...Modeling using CPU...
Epoch=1/1, batch=100/600, loss=0.6996995806694031
Epoch=1/1, batch=200/600, loss=0.34212639927864075
Epoch=1/1, batch=300/600, loss=0.32669126987457275
Epoch=1/1, batch=400/600, loss=0.11941561847925186
Epoch=1/1, batch=500/600, loss=0.12589633464813232
Epoch=1/1, batch=600/600, loss=0.11272335052490234
Training Time 23.34 seconds
............Testing CNN model.................
batch=100/100
> Number of samples= 100 number of correct prediction= 97 accuracy= 0.97
```

(3) A brief description of your own conclusion from the study results and recommendation of the design and parameters, if any

Overall, both models utilized similar parameters such as training methods, activation layers, and total number of neurons. As included above,SGD is more efficient if the model is not too complex, but if there are large data points, then other methods such as Adam or Nadam are better due to faster convergence. From the data gathered, it can be seen that CNN is more accurate than MLP. Ultimately, it is recommended that MLP be used for simple image classification, whereas CNN be used in complicated image classification.