Assignment 3: Neural Networks

Nitesh Jaiswal
2018400
CSB

Q2. Use the MNIST dataset for training and testing the neural network model created in Question 1 ONLY. Use the training dataset for calculating the training error and the test dataset for calculating the testing error.

1. Use the following architecture [#input, 256, 128, 64, #output], learning rate=0.1, and number of epochs=100. Use normal weight initialization as defined in the first question. Save the weights of the trained model separately for each activation function defined above and report the test accuracy.

1.**Relu Accuracy**

```
6]  1 print(NN.score(X_test, y_test))

   0.9695238095238096
```

2.**Sigmoid Accuracy**

```
]  1 print(NN.score(X_test, y_test))

  0.9268571428571428
```

3.**Linear Accuracy**

```
16]  1 print(NN.score(X_test, y_test))

   0.9064761904761904
```

4.**tanh Accuracy**

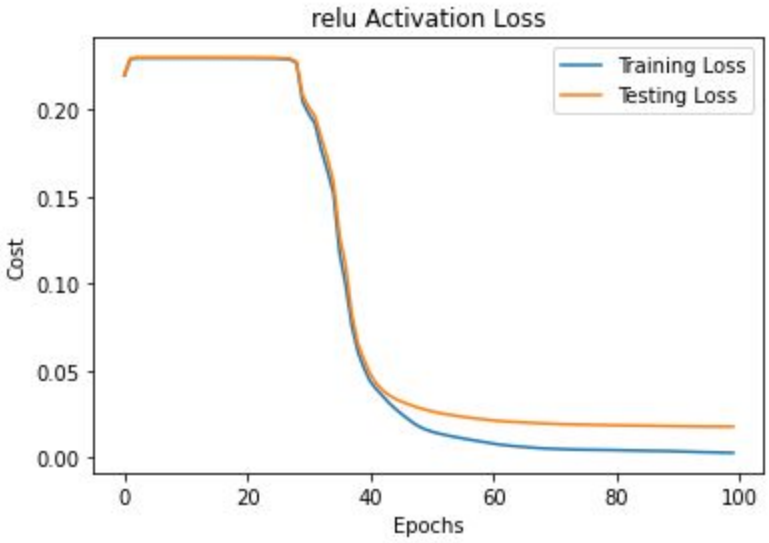
```
[589]  1 print(NN.score(X_test, y_test))

      0.9699047619047619
```

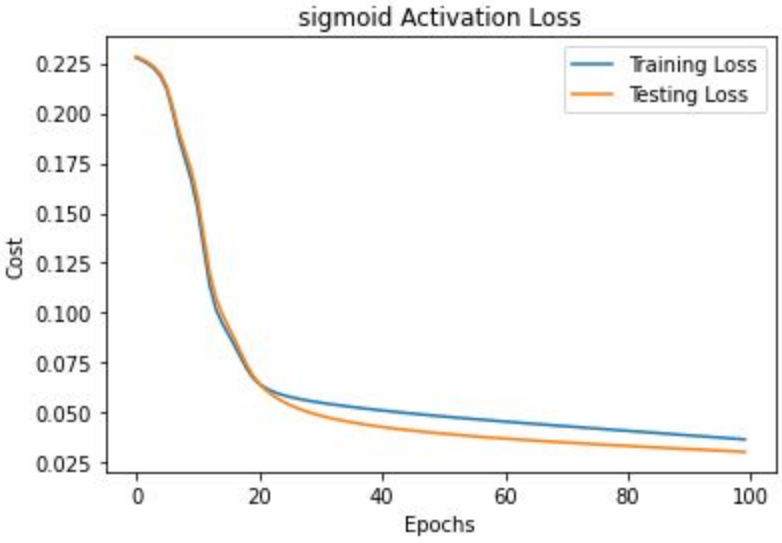
2. Plot training error vs epoch curve and testing error vs epoch curve for ReLU, sigmoid, linear, and tanh activation function.

Training Error

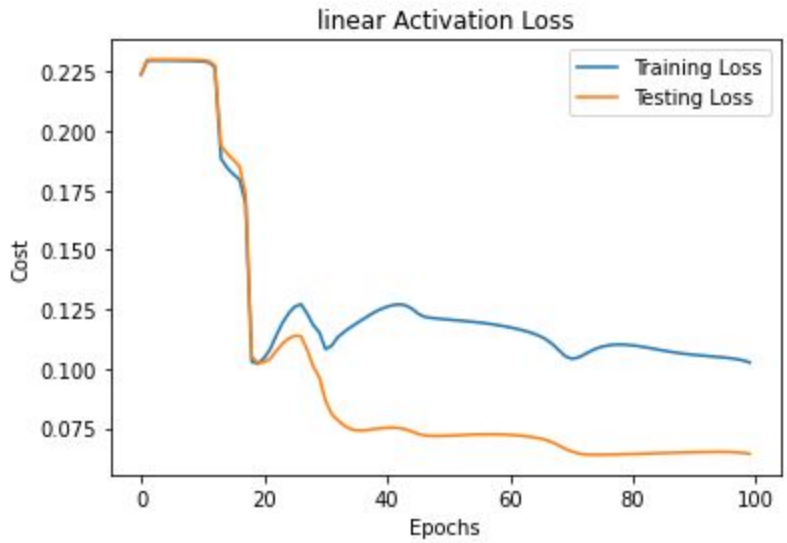
1.Relu



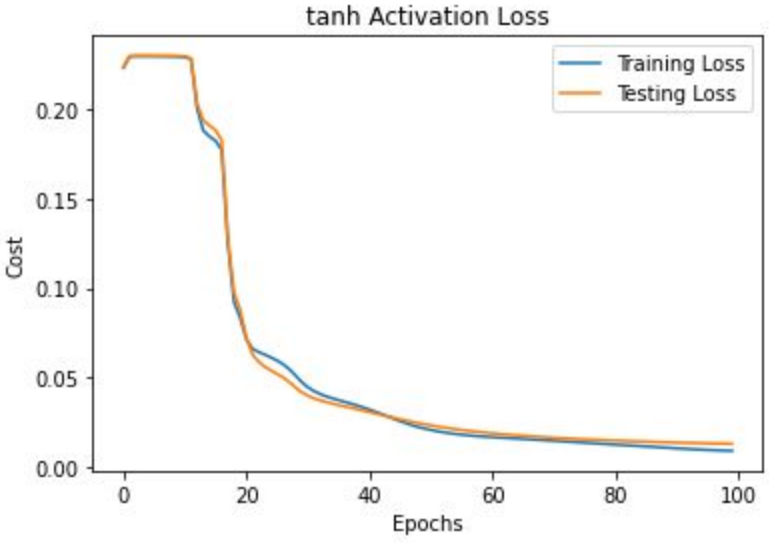
2.Sigmoid (multiply normal initialization by 0.1)



3.Linear



4. Tanh



3. In every case, what should be the activation function for the output layer? Give

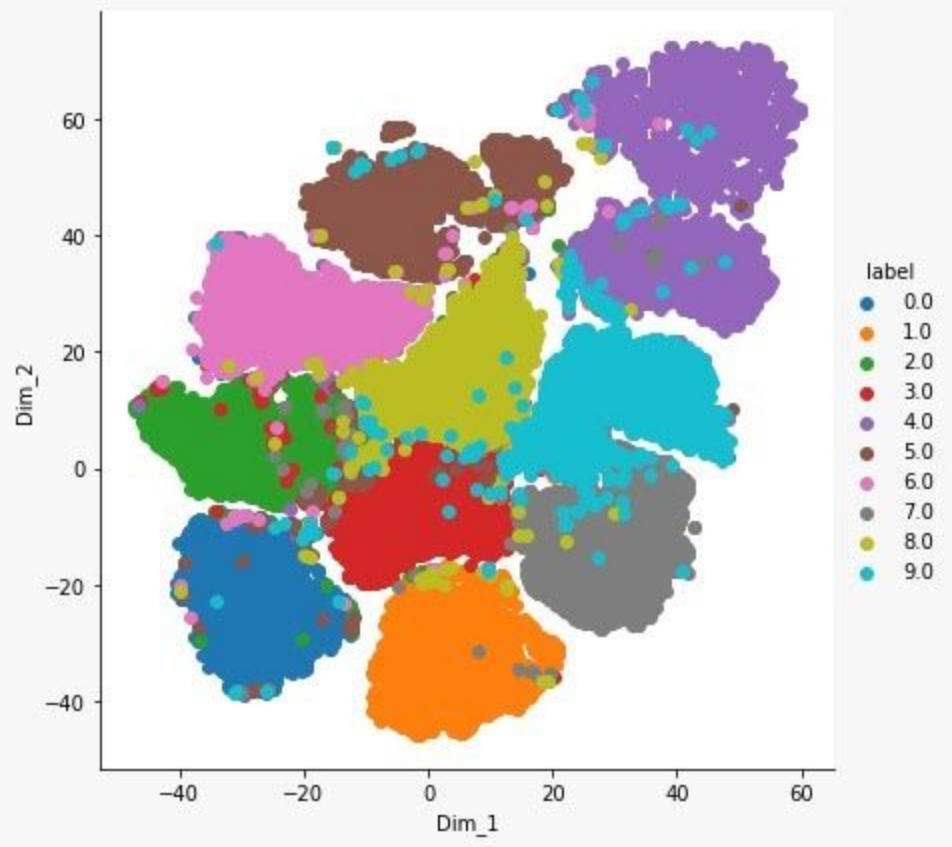
reasons to support your answer.

Answer - It can be sigmoid activation or softmax function because they return values in the range between 0 and 1 which is suitable for loss function and also prevent invalid log error or underflow.

4. What are the total number of layers and the number of hidden layers in this case?

Answer- Total number of layer - 5 [87,256,128,64,10]
Hidden Layer -3 -> [256,28,,64]

5. Visualize the features of the final hidden layer by creating tSNE plots for the model with the highest test accuracy. You can use sklearn for visualization.



6. Now, use sklearn with the same parameters defined above and report the test accuracy obtained using ReLU, sigmoid, linear, and tanh activation functions. Comment on the differences in accuracy, if any.

```
identity  -  0.8973333333333333
logistic  -  0.9286666666666666
tanh  -  0.9146666666666666
relu  -  0.9533333333333334
```

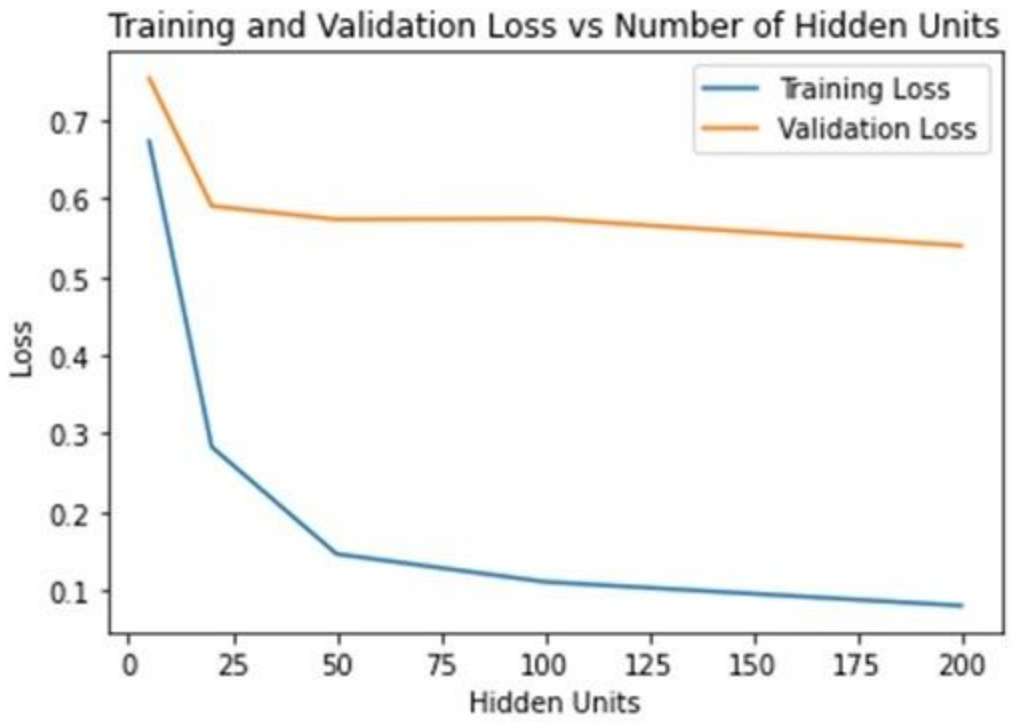
There is a difference between accuracy between my accuracy and sklearn implementation. It happens because I use the sigmoid function in the last layer and sklearn uses the softmax function in the last layer activation function.

I also checked by using the softmax function and it shows the same result.

Q-3 For the DATASET provided, use the following hyperparameter settings to train each neural network (using PyTorch) described below-

Answer - 1. Hidden Units
For the hyperparameters mentioned in the table above except the number of hidden units, train a single hidden layer neural network changing the value of the number of hidden units to 5, 20, 50, 100, and 200. Run the optimization for 100 epochs each time.

(a). Plot the average training cross-entropy (sum of the cross-entropy terms over the training set divided by the total number of training examples) on the y-axis vs a number of hidden units on the x-axis. In the same figure, plot the average validation cross-entropy.



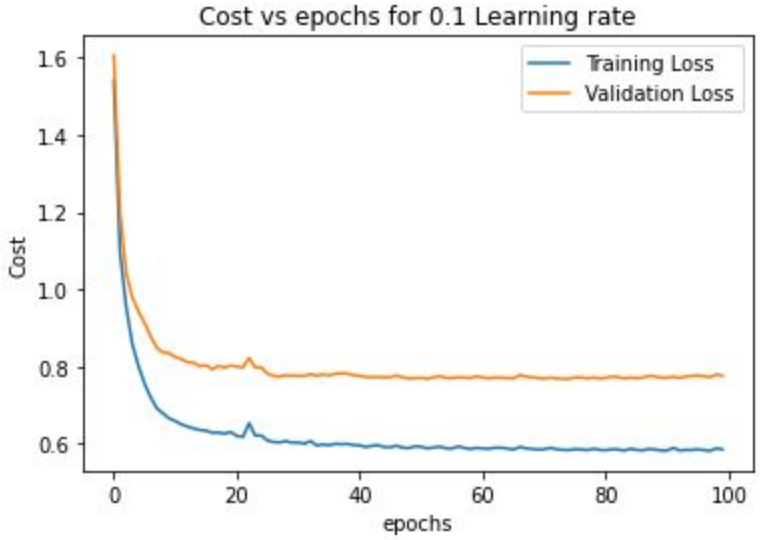
(b). Examine and comment on the plots of training and validation cross-entropy. What is the effect of changing the number of hidden units?

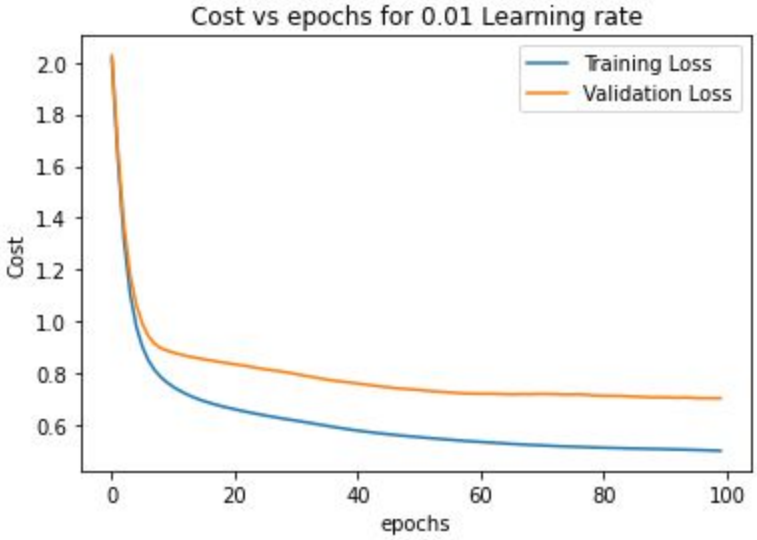
Answer - On increasing the number of hidden units training loss is continuously decreasing but in the case of validation there is a slight upliftment of loss and then continue to reduce. But overall on increasing hidden units both training and validation loss reduces.

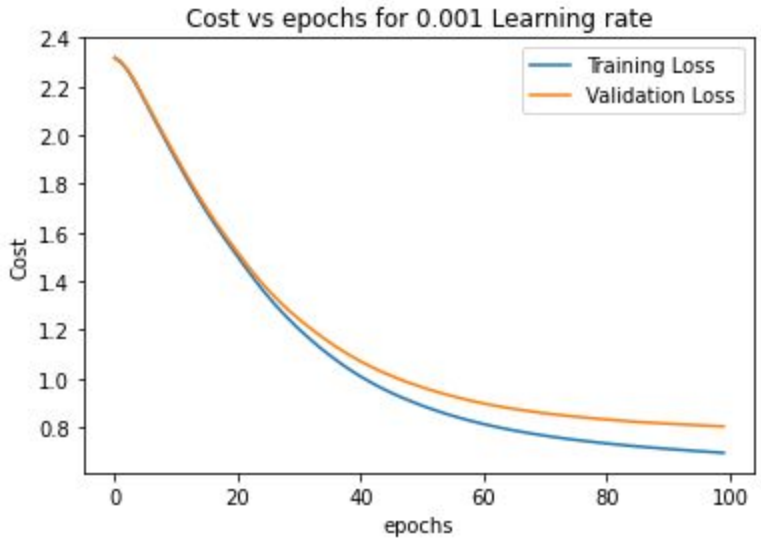
2. Learning Rate:

For the hyperparameters mentioned in the table above except the learning rate, train a single hidden layer neural network changing the value of the learning rate to 0.1, 0.01, and 0.001. Run the optimization for 100 epochs each time.

(a). Plot the average training cross-entropy on the y-axis vs the number of epochs on the x-axis for the mentioned learning rates. In the same figure, plot the average validation cross-entropy loss. Make a separate figure for each learning rate.


Cost vs epochs for 0.1 Learning rate
Training Loss
Validation Loss
Cost
epochs
1.6
1.4
1.2
1.0
0.8
0.6
0
20
40
60
80
100


Cost vs epochs for 0.01 Learning rate
Training Loss
Validation Loss
Cost
epochs
2.0
1.8
1.6
1.4
1.2
1.0
0.8
0.6
0
20
40
60
80
100

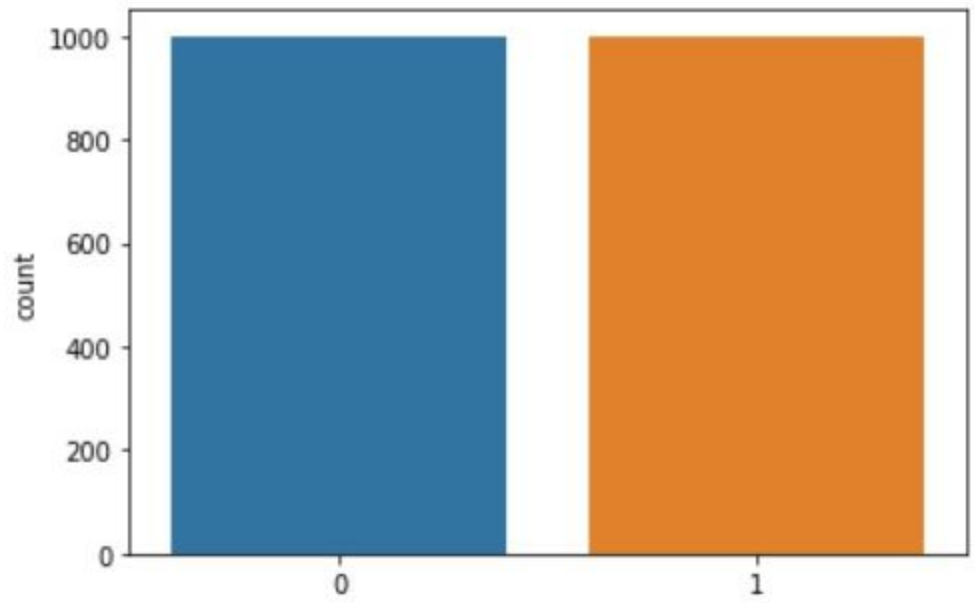


(b). Examine and comment on the plots of training and validation cross-entropy. How does adjusting the learning rate affect the convergence of cross-entropy of each Dataset?

Answer - In the case of 0.1: Overshoot occur sometime at a lower costs
In the case of 0.01: It softly converges in 100 epochs
In the case of 0.001: It does to able to converge completely in 100 epochs
Slowing the learning rate rates down the process to converging.
Higher the value of the learning rate it overshoots sometimes.

4. (20 points) Use the binary CIFAR 10 subset for this part.
1 Conduct Exploratory Data Analysis (EDA) on the CIFAR-10 dataset. Report the class distribution.

Answer- Label Distribution is equal



2. (10) Use the existing AlexNet Model from PyTorch (pre-trained on ImageNet) as a feature extractor for the images in the CIFAR subset. You should use the fc8 layer as the feature, which gives a 1000 dimensional feature vector for each image.

Answer -

```
[209]  1 AlexNet_model = torch.hub.load('pytorch/vision:v0.6.0', 'alexnet', pretrained=True)
```

```
Using cache found in /root/.cache/torch/hub/pytorch_vision_v0.6.0
```

```
1 X=AlexNet_Model(X_train)
```

3. (5) Train a Neural Network with 2 hidden layers of sizes 512 and 256 and use the fc8 layer as input to this Neural Network for the classification task.

Answer -

```
1 AlexNet_model.classifier[4] = nn.Linear(4096,512)
2 AlexNet_model.classifier[6] = nn.Linear(512,256)
```

4. (2) Report the test accuracy along with the confusion matrix and the ROC curve.

Taking a lot of time
Submitted collab