Assignment 3

---

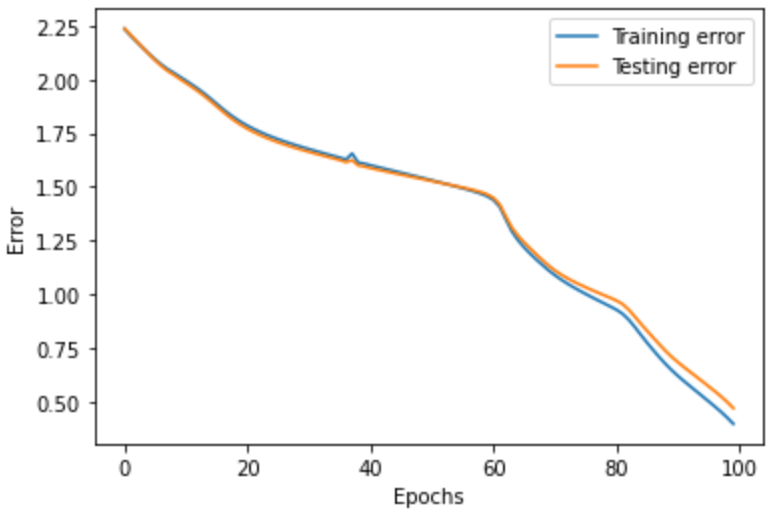
1. Code is attached

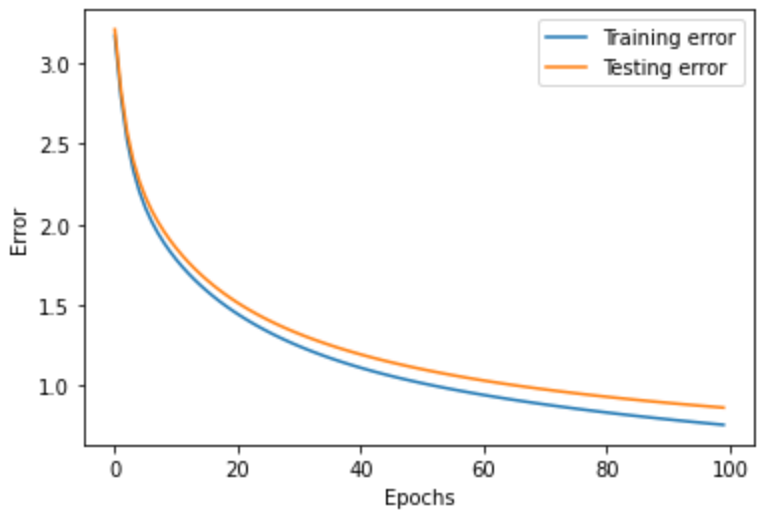
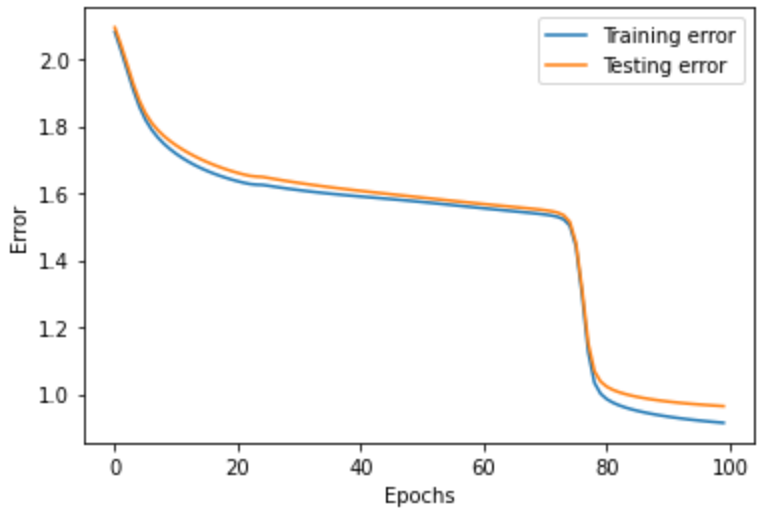
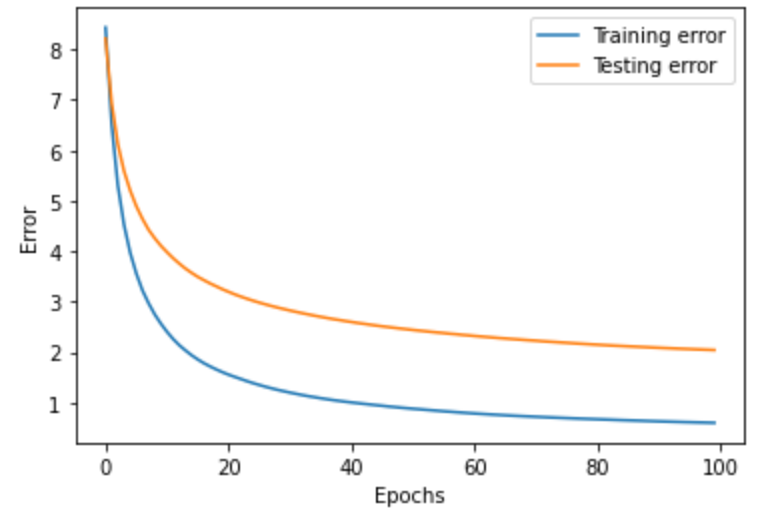
2.

a.

Test Accuracy			
ReLU	Sigmoid	Linear	Tanh
0.9323	0.74083	0.6893	0.9381

b.

Training error vs Epoch and Testing error vs Epoch curve	
ReLU	

Sigmoid	
Linear	
Tanh	

c. In every case the output layer activation function should be softmax. This is because MNIST dataset is multi class classification problem which requires the probabilities of each class to get trained correctly. And since softmax

uses the value of every other class in order to predict the probability of a single class.

$$\text{softmax}(\mathbf{y})_i = \frac{\exp(\mathbf{y}_i)}{\sum_j \exp(\mathbf{y}_j)}$$

d. In this case as specified in part A,
the total number of layers = 4
Number of hidden layers = 3

e.

```
act,pre=classifier.feed_forward(X_train)
a=act[2]
a.shape
```

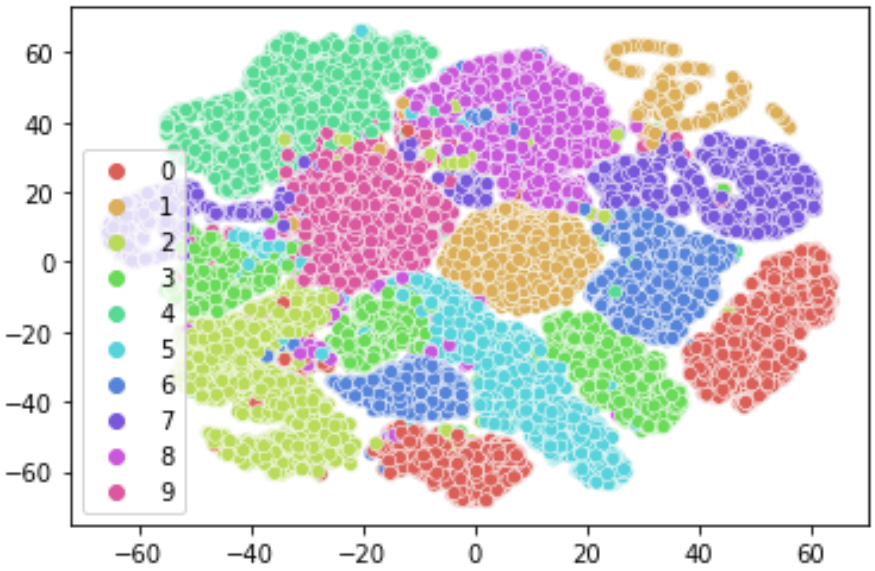
```
54000, 64)
```

```
tsne = TSNE(n_components=2, verbose=2, n_iter=1000)
tsne_results = tsne.fit_transform(a)

plt.figure(figsize=(16,10))

```

```
sns.scatterplot(
    x=tsne_results[:,0], y=tsne_results[:,1],
    hue=y_train.argmax(axis=1),
    palette=sns.color_palette("hls", 10),
    legend="full"
)
```



f.

Test Accuracy			
ReLU	Sigmoid	Linear	Tanh
0.9706	0.9578	0.9041	0.9235

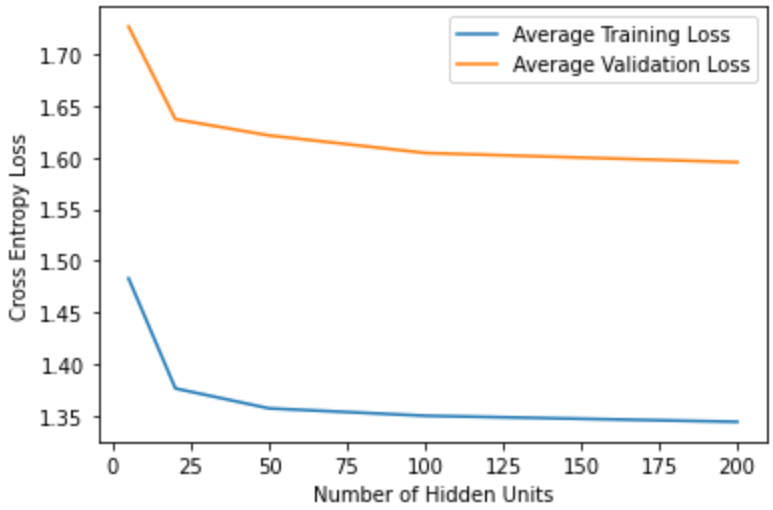
Observations:

- ReLU performed best in both the implementations
- Linear performed worst in both the cases
- Tanh was better in My implementation but slightly worse in sklearn and opposite is true for sigmoid

3.

a.

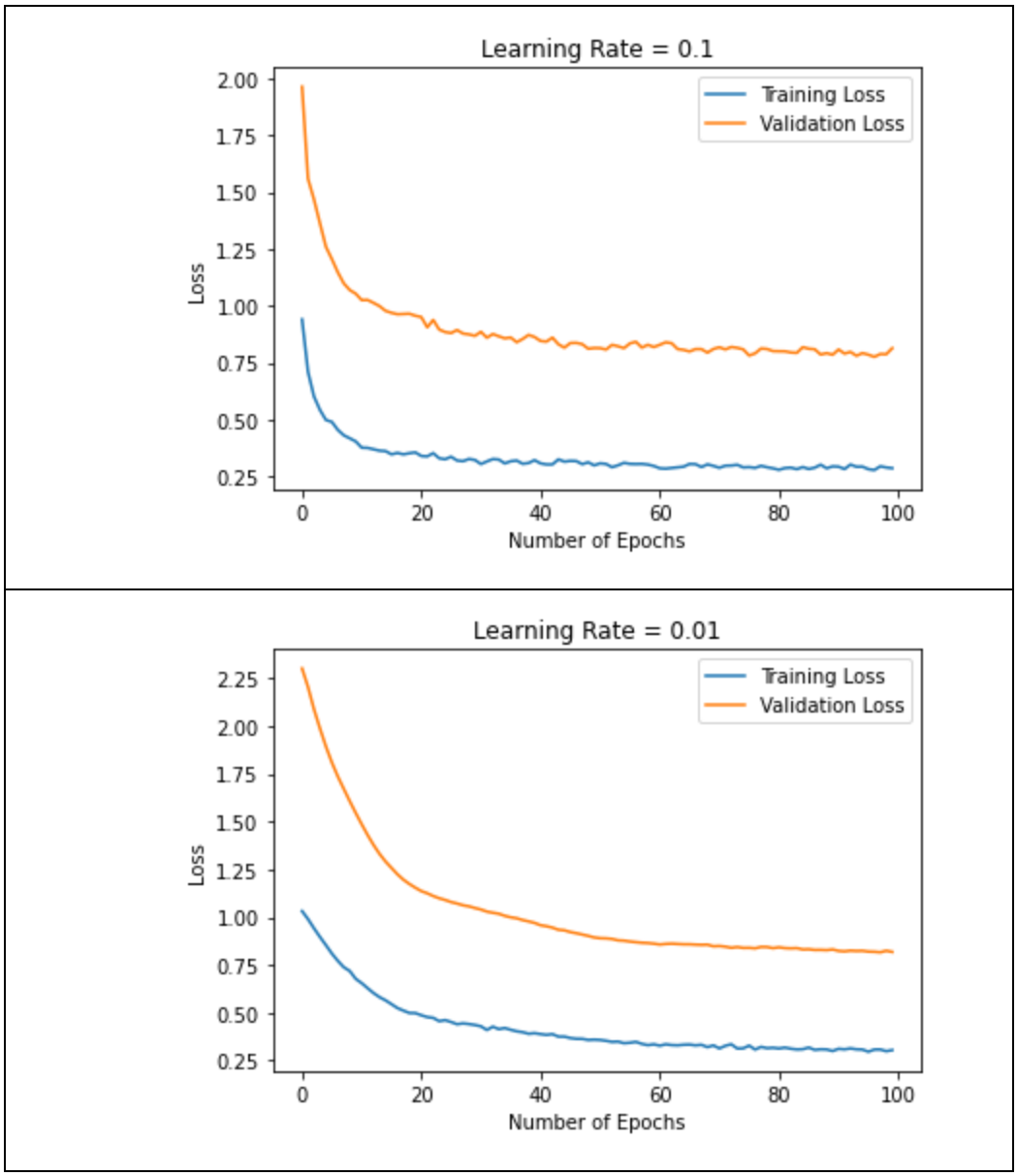
i. Hidden Units = [5, 20, 50, 100 ,200]

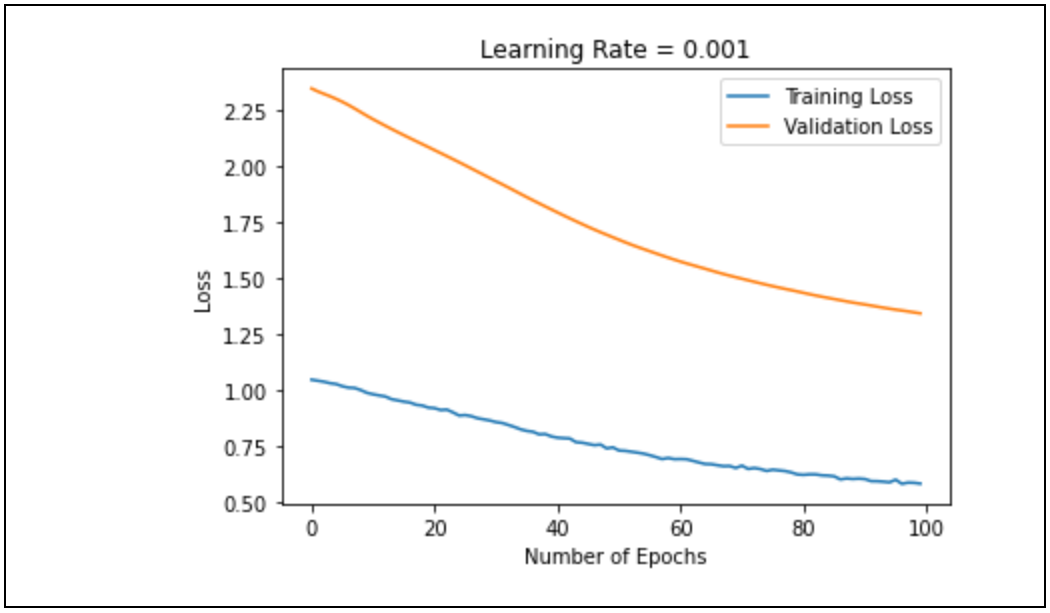


ii. By increasing the hidden the units then model is able to train much better and hence both average training and average validation loss is decreasing with the increase in hidden units.

b.

i.





ii. Observations:

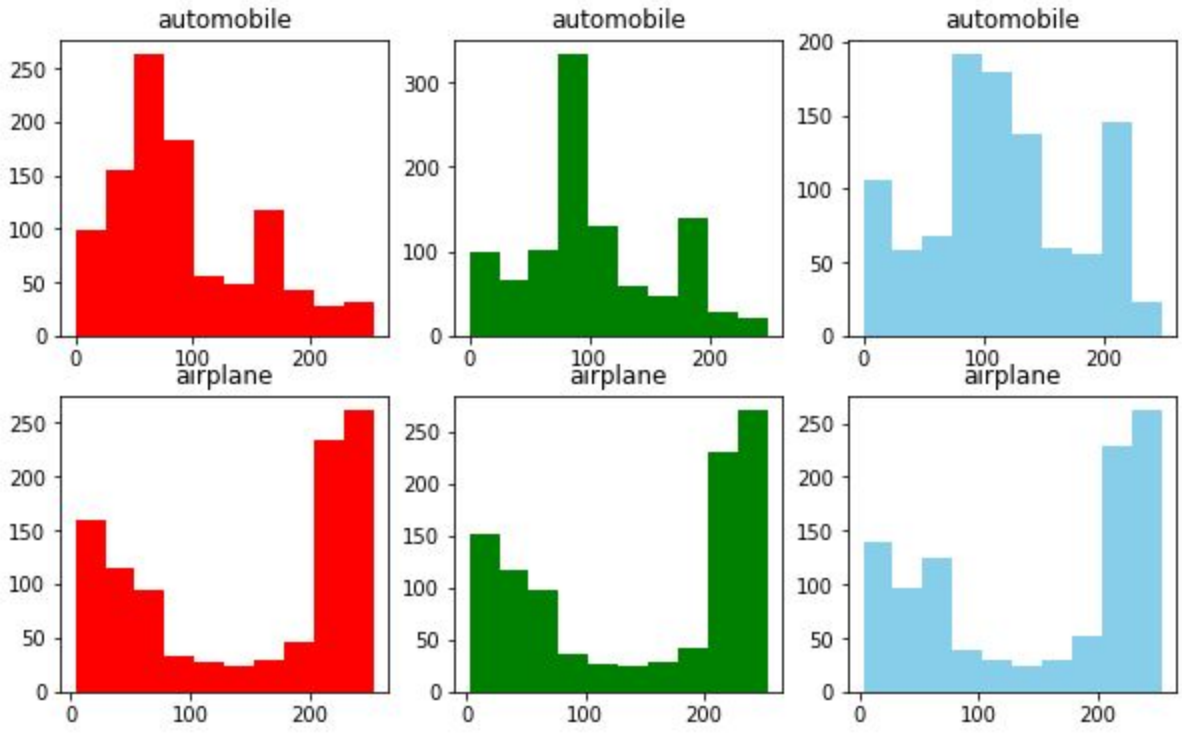
1. In all plots the validation loss is always greater than training loss
2. As the learning rate decreases the the slope of graph becomes more smooth
3. For learning rate 0.001 the line the model has not achieved the minima and hence model is not completely trained
4. For learning rate 0.1 the graph has lot of spikes which means the model is not learning properly.

4.

a. EDA

- CIFAR 10 dataset had 10000 samples each of them belonging to either label 0 or 1
- 16 random images visualised from the dataset

- Frequency distribution of colors of one of the automobile and car dataset



- Class Distribution: the dataset contains only two kinds of label 0 and 1 i.e of airplane and automobile. And both of them had equal number of datasets i.e 5000.

```
1  np.unique(y_train, return_counts=True)
```

```
(array([0, 1]), array([5000, 5000]))
```

b.

```
AlexNet(
  (features): Sequential(
    (0): Conv2d(3, 64, kernel_size=(11, 11), stride=(4, 4), padding=(2, 2))
    (1): ReLU(inplace=True)
    (2): MaxPool2d(kernel_size=3, stride=2, padding=0, dilation=1, ceil_mode=False)
    (3): Conv2d(64, 192, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
    (4): ReLU(inplace=True)
    (5): MaxPool2d(kernel_size=3, stride=2, padding=0, dilation=1, ceil_mode=False)
    (6): Conv2d(192, 384, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (7): ReLU(inplace=True)
    (8): Conv2d(384, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (9): ReLU(inplace=True)
    (10): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (11): ReLU(inplace=True)
    (12): MaxPool2d(kernel_size=3, stride=2, padding=0, dilation=1, ceil_mode=False)
  )
  (avgpool): AdaptiveAvgPool2d(output_size=(6, 6))
  (classifier): Sequential(
    (0): Dropout(p=0.5, inplace=False)
    (1): Linear(in_features=9216, out_features=4096, bias=True)
    (2): ReLU(inplace=True)
    (3): Dropout(p=0.5, inplace=False)
    (4): Linear(in_features=4096, out_features=4096, bias=True)
    (5): ReLU(inplace=True)
    (6): Linear(in_features=4096, out_features=1000, bias=True)
  )
)
```

```
train_transform_aug = transforms.Compose([
    transforms.Resize(256),
    transforms.CenterCrop(224),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])
    ]
  )
train_data=MyDataset1(X_train,y_train,train_transform_aug)

train_loader = data.DataLoader(dataset=train_data,
                               batch_size=X_train.shape[0],
                               shuffle=True)

test_data=MyDataset1(X_test,y_test,train_transform_aug)

test_loader = data.DataLoader(dataset=test_data,
                               batch_size=X_test.shape[0],
                               shuffle=True)
```

```
for x,y in train_loader:
  output=alexnet(x)
  print(output.size())

for x,y in test_loader:
  output_test=alexnet(x)
  print(output_test.size())
```

c. Neural network created using the sklearn library

```
mlp = MLPClassifier(hidden_layer_sizes=(512,256), max_iter=200, solver='sgd')
mlp.fit(X_new, y_train)
mlp.score(X_test_new,y_test)
```

d.

Test accuracy = *0.5025*

Confusion Matrix:

```
1  y_pred=mlp.predict(X_test_new)
2  metrics.confusion_matrix(y,y_pred)
```

```
array([[627, 373],
       [620, 380]])
```

ROC Curve: