CSCI 5922 - 001 Neural Networks and Deep Learning
Lab Assignment 1

Experimental Design:

Here the main aim of this assignment is to test how the hyper parameters like 'no. of hidden layers (hidden layer architecture)', 'batch size' and 'activation function' have an effect on the performance of the Multi Layer Perceptron Model by training it in a Multi Class Classification task using MNIST Handwritten digits Classification Dataset.

Checking measures like:

1. Will adding more hidden layers improves the accuracy of the model?
2. Will changing the batch size have any effect in the training and will it be useful to change an overfitting/underfitting model to a generalized model?
3. Will the activation functions like ReLU and Leaky ReLU influence the speed/time taken for model learning and the accuracy?

Steps for evaluating the impact of different architectures:

1. Changing the hidden layer configuration
   I implemented for five different network architectures

```
a.      [784, 128, 10],
b.      [784, 256, 10],
c.      [784, 512, 10],
d.      [784, 128, 64, 10],
e.      [784, 256, 128, 10]
```

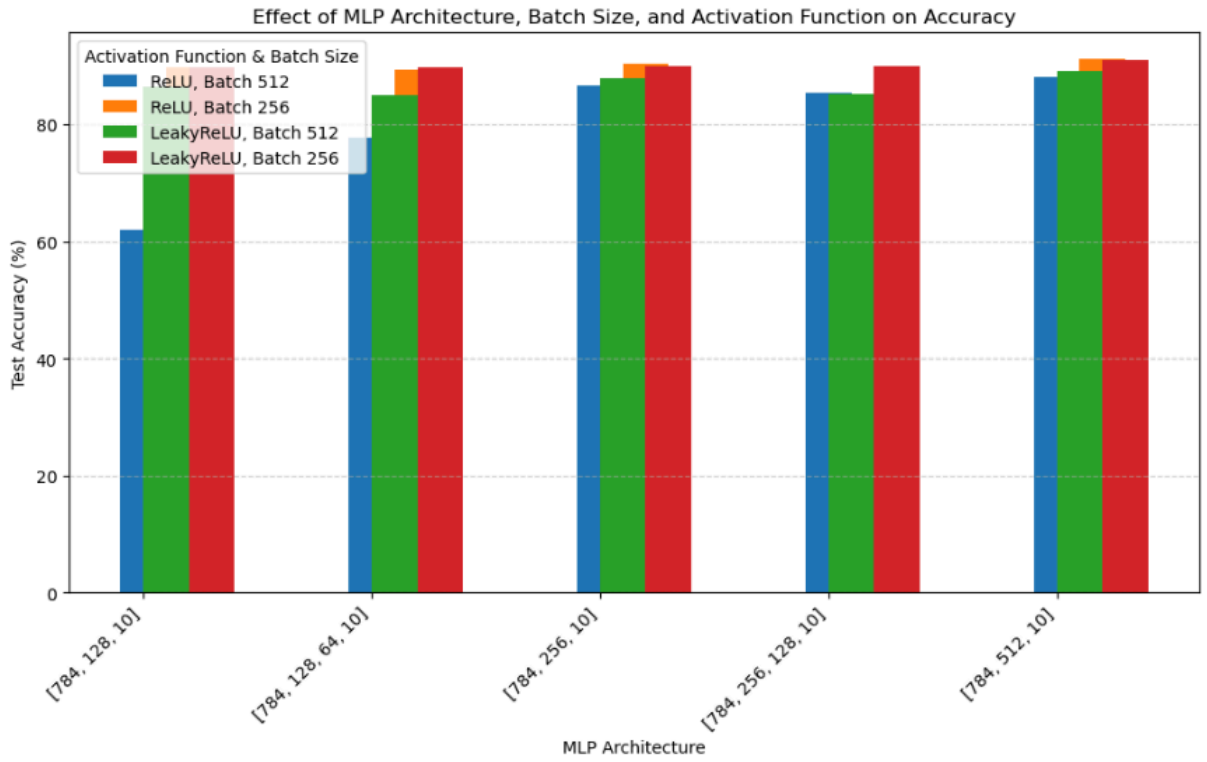
2. Learning rate, 1e-3 (fixed) because this is the learning rate in which I got better accuracy during testing with the base architecture.
3. Varying the batch size (ie. 512 & 256)
4. Two activation functions (ReLU / Leaky ReLU)
5. Each = model is trained with a fixed number of epochs (ie.25)
6. Normalized dataset is used for training to ensure fairness on the prediction.
7. Model Performance is evaluated using the Test accuracy.

Hypothesis:

- The neural network architecture with more hidden layers ie. Deep Network yield better accuracy.
- Smaller batch sizes yield better generalization.
- Leaky ReLU > ReLU (performance) in solving Vanishing Gradient Problem.

Results:

	Architecture	Batch Size	Activation Function	Test Loss	Test Accuracy (%)
0	[784, 128, 10]	512	ReLU	NaN	62.068257
1	[784, 128, 10]	512	LeakyReLU	NaN	86.461760
2	[784, 128, 10]	256	ReLU	NaN	89.683494
3	[784, 128, 10]	256	LeakyReLU	NaN	89.793670
4	[784, 256, 10]	512	ReLU	NaN	86.708470
5	[784, 256, 10]	512	LeakyReLU	NaN	87.911184
6	[784, 256, 10]	256	ReLU	NaN	90.354567
7	[784, 256, 10]	256	LeakyReLU	NaN	89.993990
8	[784, 512, 10]	512	ReLU	NaN	88.106497
9	[784, 512, 10]	512	LeakyReLU	NaN	89.000822
10	[784, 512, 10]	256	ReLU	NaN	91.075721
11	[784, 512, 10]	256	LeakyReLU	NaN	90.935497
12	[784, 128, 64, 10]	512	ReLU	NaN	77.641859
13	[784, 128, 64, 10]	512	LeakyReLU	NaN	84.960938
14	[784, 128, 64, 10]	256	ReLU	NaN	89.362981
15	[784, 128, 64, 10]	256	LeakyReLU	NaN	89.633413
16	[784, 256, 128, 10]	512	ReLU	NaN	85.402961
17	[784, 256, 128, 10]	512	LeakyReLU	NaN	85.259046
18	[784, 256, 128, 10]	256	ReLU	NaN	79.597356
19	[784, 256, 128, 10]	256	LeakyReLU	NaN	90.024038



Analysis:

1. The neural network architecture with more hidden layers ie. Deep Network yield better accuracy.
Proof from Analysis:
Model Architecture with [784,128,10] achieved an accuracy of 62.06% but in [784,512,10] it got improved to 88%. However, increasing the depth of two hidden layers [784,256,128,10] didn't improve the accuracy as i got 85% compared to the max accuracy of 90.02%. Too many layers will lead to overfitting and slower training.

2. Leaky ReLU outperforms the ReLU.
Proof from Analysis:
In all the architectures, Leaky ReLU had a significant higher accuracy compared to ReLU.
For example: in architecture [784,128,10] ReLU - 62.06% but with Leaky ReLU - 86.46% Leaky ReLU mitigates the vanishing gradient problem which helps in better weight updation.

3. Smaller batch size yield better generalization.
Proof from Analysis:
All the architectures with batch size of 256 performed slightly better than the 512 batch size.
For example: in architecture [784,128,64,10] batch size 512 - 77.64% but with batch size 512 - 89.36% Smaller batch sizes will introduce noise into the gradient updation steps which prevents the overfitting issue.

4. The model architecture [784,512,10] yielded the highest accuracy of 90.93% using Leaky ReLU and batch size of 256.

Comparing with Hypothesis:

1. The neural network architecture with more hidden layers ie. Deep Network yield better accuracy.
   (Yes this hypothesis is supported by the results but the accuracy diminishes beyond two hidden layers)
2. Smaller batch sizes yield better generalization.
   (Yes this hypothesis is supported and got a good improvement in accuracy)
3. Leaky ReLU > ReLU (performance) in solving Vanishing Gradient Problem.
   (Yes this hypothesis is supported and Leaky ReLU consistently provided a significantly better accuracy improvement)

Extra Credit:

These changes I made had a varying effects on my model performance.

- When using the Tanh activation function, the test accuracy dropped from 87.21 to 77.05% So tanh is not suitable for this architecture since it slows down the learning.
- When shifted to the Sigmoid Activation function, here also there is a drop in the accuracy of the model ie.87.21 to 47.82% and this may be due to the vanishing gradient problem where the smaller gradients tend to slow down the training process so it made hard for the hidden layers to update the weights.
- Without changing the activation function, the activation function was fixed with ReLU and added L2 regularization penalty (lambda = 0.001) has improved the accuracy from 87.21% to 87.58% for the architecture[784,256,10] with ReLU batch size 512 and lr = 1e-3. This L2 regularization technique helps in preventing the overfitting by penalizing the larger weights and the model will become more generalized.
- Overall, here the addition of L2 regularization parameter is a good addition to the model, while changing the activation function to Sigmoid & tanh is not an effective approach to this architecture.

Screenshots:

With default hyperparameters,

```python
def main():
    """
    Main function to train and evaluate the MLP model on M
    """
    x_train, y_train, x_test, y_test = normalize_mnist()


    model = MLP([784, 256, 10])
    model.initialize()
    model.set_hp(lr=1e-3, bs=512, activation=ReLU)

    E = 25
    for _ in range(E):
        TrainMLP(model, x_train, y_train)
        TestMLP(model, x_test, y_test)


if __name__ == "__main__":
    main()
```

✓ 9.3s

```
Using device: cuda
100%|██████████| 117/117 [00:00<00:00, 286.05it/s]
Train Loss: tensor(2.4927, device='cuda:0')
100%|██████████| 19/19 [00:00<00:00, 439.44it/s]
Test Loss: nan, Test Accuracy: 16.19%
100%|██████████| 117/117 [00:00<00:00, 423.14it/s]
Train Loss: tensor(2.2616, device='cuda:0')
100%|██████████| 19/19 [00:00<00:00, 487.55it/s]
Test Loss: nan, Test Accuracy: 22.76%
100%|██████████| 117/117 [00:00<00:00, 443.96it/s]
Train Loss: tensor(2.1270, device='cuda:0')
100%|██████████| 19/19 [00:00<00:00, 683.49it/s]
Test Loss: nan, Test Accuracy: 29.14%
100%|██████████| 117/117 [00:00<00:00, 487.38it/s]
Train Loss: tensor(2.0020, device='cuda:0')
100%|██████████| 19/19 [00:00<00:00, 797.17it/s]
Test Loss: nan, Test Accuracy: 37.53%
100%|██████████| 117/117 [00:00<00:00, 457.31it/s]
Train Loss: tensor(1.8385, device='cuda:0')
100%|██████████| 19/19 [00:00<00:00, 728.30it/s]
Test Loss: nan, Test Accuracy: 49.77%
100%|██████████| 117/117 [00:00<00:00, 490.60it/s]
Train Loss: tensor(1.6241, device='cuda:0')
100%|██████████| 19/19 [00:00<00:00, 730.19it/s]
Test Loss: nan, Test Accuracy: 60.53%
100%|██████████| 117/117 [00:00<00:00, 443.96it/s]
Train Loss: tensor(1.4065, device='cuda:0')
100%|██████████| 19/19 [00:00<00:00, 622.33it/s]
Test Loss: nan, Test Accuracy: 67.50%
100%|██████████| 117/117 [00:00<00:00, 459.58it/s]
Train Loss: tensor(1.2290, device='cuda:0')
100%|██████████| 19/19 [00:00<00:00, 700.53it/s]
Test Loss: nan, Test Accuracy: 71.60%
100%|██████████| 117/117 [00:00<00:00, 460.58it/s]
Train Loss: tensor(1.0949, device='cuda:0')
100%|██████████| 19/19 [00:00<00:00, 722.64it/s]
Test Loss: nan, Test Accuracy: 75.04%
100%|██████████| 117/117 [00:00<00:00, 407.92it/s]
Train Loss: tensor(0.9938, device='cuda:0')
100%|██████████| 19/19 [00:00<00:00, 463.11it/s]
Test Loss: nan, Test Accuracy: 77.22%
100%|██████████| 117/117 [00:00<00:00, 399.29it/s]
Train Loss: tensor(0.9153, device='cuda:0')
```

```
100%|██████████| 117/117 [00:00<00:00, 399.29it/s]
Train Loss: tensor(0.9153, device='cuda:0')
100%|██████████| 19/19 [00:00<00:00, 789.61it/s]
Test Loss: nan, Test Accuracy: 79.29%
100%|██████████| 117/117 [00:00<00:00, 469.48it/s]
Train Loss: tensor(0.8529, device='cuda:0')
100%|██████████| 19/19 [00:00<00:00, 612.58it/s]
Test Loss: nan, Test Accuracy: 80.61%
100%|██████████| 117/117 [00:00<00:00, 469.19it/s]
Train Loss: tensor(0.8012, device='cuda:0')
100%|██████████| 19/19 [00:00<00:00, 697.22it/s]
Test Loss: nan, Test Accuracy: 81.55%
100%|██████████| 117/117 [00:00<00:00, 482.72it/s]
Train Loss: tensor(0.7583, device='cuda:0')
100%|██████████| 19/19 [00:00<00:00, 530.08it/s]
Test Loss: nan, Test Accuracy: 82.46%
100%|██████████| 117/117 [00:00<00:00, 497.93it/s]
Train Loss: tensor(0.7218, device='cuda:0')
100%|██████████| 19/19 [00:00<00:00, 354.36it/s]
Test Loss: nan, Test Accuracy: 83.22%
100%|██████████| 117/117 [00:00<00:00, 412.30it/s]
Train Loss: tensor(0.6902, device='cuda:0')
100%|██████████| 19/19 [00:00<00:00, 524.06it/s]
Test Loss: nan, Test Accuracy: 83.61%
100%|██████████| 117/117 [00:00<00:00, 469.12it/s]
Train Loss: tensor(0.6626, device='cuda:0')
100%|██████████| 19/19 [00:00<00:00, 706.47it/s]
Test Loss: nan, Test Accuracy: 84.32%
100%|██████████| 117/117 [00:00<00:00, 481.27it/s]
Train Loss: tensor(0.6380, device='cuda:0')
100%|██████████| 19/19 [00:00<00:00, 673.32it/s]
Test Loss: nan, Test Accuracy: 84.93%
100%|██████████| 117/117 [00:00<00:00, 492.33it/s]
Train Loss: tensor(0.6166, device='cuda:0')
100%|██████████| 19/19 [00:00<00:00, 746.58it/s]
Test Loss: nan, Test Accuracy: 85.30%
100%|██████████| 117/117 [00:00<00:00, 436.13it/s]
Train Loss: tensor(0.5971, device='cuda:0')
100%|██████████| 19/19 [00:00<00:00, 623.80it/s]
Test Loss: nan, Test Accuracy: 85.67%
100%|██████████| 117/117 [00:00<00:00, 476.11it/s]
Train Loss: tensor(0.5796, device='cuda:0')
100%|██████████| 19/19 [00:00<00:00, 712.71it/s]
Test Loss: nan, Test Accuracy: 86.00%
100%|██████████| 117/117 [00:00<00:00, 471.01it/s]
Train Loss: tensor(0.5640, device='cuda:0')
100%|██████████| 19/19 [00:00<00:00, 668.50it/s]
Test Loss: nan, Test Accuracy: 86.48%
100%|██████████| 117/117 [00:00<00:00, 478.46it/s]
Train Loss: tensor(0.5495, device='cuda:0')
100%|██████████| 19/19 [00:00<00:00, 571.54it/s]
Test Loss: nan, Test Accuracy: 86.74%
100%|██████████| 117/117 [00:00<00:00, 441.60it/s]
Train Loss: tensor(0.5364, device='cuda:0')
100%|██████████| 19/19 [00:00<00:00, 479.80it/s]
Test Loss: nan, Test Accuracy: 86.96%
100%|██████████| 117/117 [00:00<00:00, 417.94it/s]
Train Loss: tensor(0.5243, device='cuda:0')
100%|██████████| 19/19 [00:00<00:00, 712.00it/s]
Test Loss: nan, Test Accuracy: 87.21%
```

With Sigmoid Activation Function:

```python
def main():
    """
    Main function to train and evaluate the MLP model on MNIST using GPU.
    """
    x_train, y_train, x_test, y_test = normalize_mnist()


    model = MLP([784, 256, 10])
    model.initialize()
    model.set_hp(lr=1e-3, bs=512, activation=Sigmoid)

    E = 25
    for _ in range(E):
        TrainMLP(model, x_train, y_train)
        TestMLP(model, x_test, y_test)


if __name__ == "__main__":
    main()
```

✓ 11.1s

```
Using device: cuda
100%|██████████| 117/117 [00:00<00:00, 238.62it/s]
Train Loss: tensor(2.3170, device='cuda:0')
100%|██████████| 19/19 [00:00<00:00, 495.17it/s]
Test Loss: nan, Test Accuracy: 13.59%
100%|██████████| 117/117 [00:00<00:00, 393.66it/s]
Train Loss: tensor(2.3106, device='cuda:0')
100%|██████████| 19/19 [00:00<00:00, 590.09it/s]
Test Loss: nan, Test Accuracy: 14.05%
100%|██████████| 117/117 [00:00<00:00, 408.75it/s]
Train Loss: tensor(2.3043, device='cuda:0')
100%|██████████| 19/19 [00:00<00:00, 557.36it/s]
Test Loss: nan, Test Accuracy: 14.77%
100%|██████████| 117/117 [00:00<00:00, 385.18it/s]
Train Loss: tensor(2.2979, device='cuda:0')
100%|██████████| 19/19 [00:00<00:00, 580.69it/s]
Test Loss: nan, Test Accuracy: 15.76%
100%|██████████| 117/117 [00:00<00:00, 343.02it/s]
Train Loss: tensor(2.2915, device='cuda:0')
100%|██████████| 19/19 [00:00<00:00, 605.69it/s]
Test Loss: nan, Test Accuracy: 17.07%
100%|██████████| 117/117 [00:00<00:00, 405.55it/s]
Train Loss: tensor(2.2852, device='cuda:0')
100%|██████████| 19/19 [00:00<00:00, 482.50it/s]
Test Loss: nan, Test Accuracy: 18.17%
100%|██████████| 117/117 [00:00<00:00, 382.03it/s]
Train Loss: tensor(2.2789, device='cuda:0')
100%|██████████| 19/19 [00:00<00:00, 603.85it/s]
Test Loss: nan, Test Accuracy: 19.52%
100%|██████████| 117/117 [00:00<00:00, 393.84it/s]
Train Loss: tensor(2.2726, device='cuda:0')
100%|██████████| 19/19 [00:00<00:00, 601.57it/s]
Test Loss: nan, Test Accuracy: 20.87%
100%|██████████| 117/117 [00:00<00:00, 373.17it/s]
Train Loss: tensor(2.2664, device='cuda:0')
100%|██████████| 19/19 [00:00<00:00, 602.80it/s]
Test Loss: nan, Test Accuracy: 22.13%
100%|██████████| 117/117 [00:00<00:00, 427.01it/s]
Train Loss: tensor(2.2602, device='cuda:0')
100%|██████████| 19/19 [00:00<00:00, 621.33it/s]
Test Loss: nan, Test Accuracy: 23.26%
100%|██████████| 117/117 [00:00<00:00, 388.90it/s]
Train Loss: tensor(2.2540, device='cuda:0')
```

```
Train Loss: tensor(2.2540, device='cuda:0')
100%|██████████| 19/19 [00:00<00:00, 589.95it/s]
Test Loss: nan, Test Accuracy: 24.67%
100%|██████████| 117/117 [00:00<00:00, 387.52it/s]
Train Loss: tensor(2.2480, device='cuda:0')
100%|██████████| 19/19 [00:00<00:00, 547.56it/s]
Test Loss: nan, Test Accuracy: 26.19%
100%|██████████| 117/117 [00:00<00:00, 378.14it/s]
Train Loss: tensor(2.2419, device='cuda:0')
100%|██████████| 19/19 [00:00<00:00, 590.68it/s]
Test Loss: nan, Test Accuracy: 27.85%
100%|██████████| 117/117 [00:00<00:00, 398.96it/s]
Train Loss: tensor(2.2359, device='cuda:0')
100%|██████████| 19/19 [00:00<00:00, 611.18it/s]
Test Loss: nan, Test Accuracy: 29.45%
100%|██████████| 117/117 [00:00<00:00, 400.28it/s]
Train Loss: tensor(2.2300, device='cuda:0')
100%|██████████| 19/19 [00:00<00:00, 563.64it/s]
Test Loss: nan, Test Accuracy: 31.52%
100%|██████████| 117/117 [00:00<00:00, 391.11it/s]
Train Loss: tensor(2.2240, device='cuda:0')
100%|██████████| 19/19 [00:00<00:00, 597.10it/s]
Test Loss: nan, Test Accuracy: 33.70%
100%|██████████| 117/117 [00:00<00:00, 372.94it/s]
Train Loss: tensor(2.2182, device='cuda:0')
100%|██████████| 19/19 [00:00<00:00, 581.92it/s]
Test Loss: nan, Test Accuracy: 35.84%
100%|██████████| 117/117 [00:00<00:00, 355.76it/s]
Train Loss: tensor(2.2125, device='cuda:0')
100%|██████████| 19/19 [00:00<00:00, 517.61it/s]
Test Loss: nan, Test Accuracy: 37.84%
100%|██████████| 117/117 [00:00<00:00, 323.71it/s]
Train Loss: tensor(2.2067, device='cuda:0')
100%|██████████| 19/19 [00:00<00:00, 674.29it/s]
Test Loss: nan, Test Accuracy: 39.63%
100%|██████████| 117/117 [00:00<00:00, 378.80it/s]
Train Loss: tensor(2.2011, device='cuda:0')
100%|██████████| 19/19 [00:00<00:00, 658.69it/s]
Test Loss: nan, Test Accuracy: 41.14%
100%|██████████| 117/117 [00:00<00:00, 374.62it/s]
Train Loss: tensor(2.1955, device='cuda:0')
100%|██████████| 19/19 [00:00<00:00, 521.05it/s]
Test Loss: nan, Test Accuracy: 42.77%
100%|██████████| 117/117 [00:00<00:00, 371.81it/s]
Train Loss: tensor(2.1900, device='cuda:0')
100%|██████████| 19/19 [00:00<00:00, 625.25it/s]
Test Loss: nan, Test Accuracy: 44.14%
100%|██████████| 117/117 [00:00<00:00, 385.95it/s]
Train Loss: tensor(2.1844, device='cuda:0')
100%|██████████| 19/19 [00:00<00:00, 394.59it/s]
Test Loss: nan, Test Accuracy: 45.34%
100%|██████████| 117/117 [00:00<00:00, 385.31it/s]
Train Loss: tensor(2.1791, device='cuda:0')
100%|██████████| 19/19 [00:00<00:00, 644.11it/s]
Test Loss: nan, Test Accuracy: 46.74%
100%|██████████| 117/117 [00:00<00:00, 358.40it/s]
Train Loss: tensor(2.1737, device='cuda:0')
100%|██████████| 19/19 [00:00<00:00, 597.86it/s]
Test Loss: nan, Test Accuracy: 47.82%
```

With Tanh activation function:

```python
def main():
    """
    Main function to train and evaluate the MLP model on MNIST using GPU.
    """
    x_train, y_train, x_test, y_test = normalize_mnist()


    model = MLP([784, 256, 10])
    model.initialize()
    model.set_hp(lr=1e-3, bs=512, activation=Tanh)

    E = 25
    for _ in range(E):
        TrainMLP(model, x_train, y_train)
        TestMLP(model, x_test, y_test)


if __name__ == "__main__":
    main()
```

✓ 12.0s

```
Using device: cuda
100%|██████████| 117/117 [00:00<00:00, 227.12it/s]
Train Loss: tensor(2.1271, device='cuda:0')
100%|██████████| 19/19 [00:00<00:00, 482.62it/s]
Test Loss: nan, Test Accuracy: 37.97%
100%|██████████| 117/117 [00:00<00:00, 330.52it/s]
Train Loss: tensor(1.8806, device='cuda:0')
100%|██████████| 19/19 [00:00<00:00, 483.13it/s]
Test Loss: nan, Test Accuracy: 51.79%
100%|██████████| 117/117 [00:00<00:00, 367.57it/s]
Train Loss: tensor(1.7484, device='cuda:0')
100%|██████████| 19/19 [00:00<00:00, 595.23it/s]
Test Loss: nan, Test Accuracy: 59.92%
100%|██████████| 117/117 [00:00<00:00, 353.85it/s]
Train Loss: tensor(1.6745, device='cuda:0')
100%|██████████| 19/19 [00:00<00:00, 529.90it/s]
Test Loss: nan, Test Accuracy: 64.94%
100%|██████████| 117/117 [00:00<00:00, 342.46it/s]
Train Loss: tensor(1.6295, device='cuda:0')
100%|██████████| 19/19 [00:00<00:00, 532.31it/s]
Test Loss: nan, Test Accuracy: 68.40%
100%|██████████| 117/117 [00:00<00:00, 331.43it/s]
Train Loss: tensor(1.5997, device='cuda:0')
100%|██████████| 19/19 [00:00<00:00, 559.74it/s]
Test Loss: nan, Test Accuracy: 71.08%
100%|██████████| 117/117 [00:00<00:00, 351.77it/s]
Train Loss: tensor(1.5786, device='cuda:0')
100%|██████████| 19/19 [00:00<00:00, 543.67it/s]
Test Loss: nan, Test Accuracy: 72.45%
100%|██████████| 117/117 [00:00<00:00, 365.76it/s]
Train Loss: tensor(1.5628, device='cuda:0')
100%|██████████| 19/19 [00:00<00:00, 553.42it/s]
Test Loss: nan, Test Accuracy: 73.55%
100%|██████████| 117/117 [00:00<00:00, 357.43it/s]
Train Loss: tensor(1.5506, device='cuda:0')
100%|██████████| 19/19 [00:00<00:00, 522.42it/s]
Test Loss: nan, Test Accuracy: 74.54%
100%|██████████| 117/117 [00:00<00:00, 317.32it/s]
Train Loss: tensor(1.5409, device='cuda:0')
100%|██████████| 19/19 [00:00<00:00, 602.05it/s]
Test Loss: nan, Test Accuracy: 75.17%
100%|██████████| 117/117 [00:00<00:00, 358.70it/s]
Train Loss: tensor(1.5332, device='cuda:0')
```

```
100%|██████████| 117/117 [00:00<00:00, 358.70it/s]
Train Loss: tensor(1.5332, device='cuda:0')
100%|██████████| 19/19 [00:00<00:00, 562.99it/s]
Test Loss: nan, Test Accuracy: 75.61%
100%|██████████| 117/117 [00:00<00:00, 371.11it/s]
Train Loss: tensor(1.5268, device='cuda:0')
100%|██████████| 19/19 [00:00<00:00, 565.50it/s]
Test Loss: nan, Test Accuracy: 75.86%
100%|██████████| 117/117 [00:00<00:00, 365.83it/s]
Train Loss: tensor(1.5215, device='cuda:0')
100%|██████████| 19/19 [00:00<00:00, 557.36it/s]
Test Loss: nan, Test Accuracy: 76.16%
100%|██████████| 117/117 [00:00<00:00, 315.27it/s]
Train Loss: tensor(1.5171, device='cuda:0')
100%|██████████| 19/19 [00:00<00:00, 579.72it/s]
Test Loss: nan, Test Accuracy: 76.30%
100%|██████████| 117/117 [00:00<00:00, 347.91it/s]
Train Loss: tensor(1.5137, device='cuda:0')
100%|██████████| 19/19 [00:00<00:00, 574.58it/s]
Test Loss: nan, Test Accuracy: 76.48%
100%|██████████| 117/117 [00:00<00:00, 360.74it/s]
Train Loss: tensor(1.5106, device='cuda:0')
100%|██████████| 19/19 [00:00<00:00, 501.27it/s]
Test Loss: nan, Test Accuracy: 76.57%
100%|██████████| 117/117 [00:00<00:00, 300.83it/s]
Train Loss: tensor(1.5082, device='cuda:0')
100%|██████████| 19/19 [00:00<00:00, 617.41it/s]
Test Loss: nan, Test Accuracy: 76.76%
100%|██████████| 117/117 [00:00<00:00, 319.17it/s]
Train Loss: tensor(1.5060, device='cuda:0')
100%|██████████| 19/19 [00:00<00:00, 445.54it/s]
Test Loss: nan, Test Accuracy: 76.80%
100%|██████████| 117/117 [00:00<00:00, 367.67it/s]
Train Loss: tensor(1.5043, device='cuda:0')
100%|██████████| 19/19 [00:00<00:00, 591.62it/s]
Test Loss: nan, Test Accuracy: 76.93%
100%|██████████| 117/117 [00:00<00:00, 359.95it/s]
Train Loss: tensor(1.5026, device='cuda:0')
100%|██████████| 19/19 [00:00<00:00, 503.90it/s]
Test Loss: nan, Test Accuracy: 76.93%
100%|██████████| 117/117 [00:00<00:00, 336.17it/s]
Train Loss: tensor(1.5013, device='cuda:0')
100%|██████████| 19/19 [00:00<00:00, 582.44it/s]
Test Loss: nan, Test Accuracy: 76.83%
100%|██████████| 117/117 [00:00<00:00, 347.11it/s]
Train Loss: tensor(1.5003, device='cuda:0')
100%|██████████| 19/19 [00:00<00:00, 555.82it/s]
Test Loss: nan, Test Accuracy: 76.95%
100%|██████████| 117/117 [00:00<00:00, 309.74it/s]
Train Loss: tensor(1.4994, device='cuda:0')
100%|██████████| 19/19 [00:00<00:00, 611.15it/s]
Test Loss: nan, Test Accuracy: 76.98%
100%|██████████| 117/117 [00:00<00:00, 364.80it/s]
Train Loss: tensor(1.4986, device='cuda:0')
100%|██████████| 19/19 [00:00<00:00, 539.87it/s]
Test Loss: nan, Test Accuracy: 77.07%
100%|██████████| 117/117 [00:00<00:00, 338.22it/s]
Train Loss: tensor(1.4979, device='cuda:0')
100%|██████████| 19/19 [00:00<00:00, 606.75it/s]
Test Loss: nan, Test Accuracy: 77.05%
```

With l2 Regularization penalty:

```python
def main():
    """
    Main function to train and evaluate the MLP model on MNIST using GPU.
    """
    x_train, y_train, x_test, y_test = normalize_mnist()


    model = MLP([784, 256, 10])
    model.initialize()
    model.set_hp(lr=1e-3, bs=512, activation=ReLU, l2 = 0.001)

    E = 25
    for _ in range(E):
        TrainMLP(model, x_train, y_train)
        TestMLP(model, x_test, y_test)


if __name__ == "__main__":
    main()
```

✓ 10.6s

```
Using device: cuda
100%|██████████| 117/117 [00:00<00:00, 218.30it/s]
Train Loss: tensor(2.3631, device='cuda:0')
100%|██████████| 19/19 [00:00<00:00, 452.42it/s]
Test Loss: nan, Test Accuracy: 27.05%
100%|██████████| 117/117 [00:00<00:00, 352.68it/s]
Train Loss: tensor(2.0340, device='cuda:0')
100%|██████████| 19/19 [00:00<00:00, 689.89it/s]
Test Loss: nan, Test Accuracy: 42.62%
100%|██████████| 117/117 [00:00<00:00, 402.01it/s]
Train Loss: tensor(1.7416, device='cuda:0')
100%|██████████| 19/19 [00:00<00:00, 591.83it/s]
Test Loss: nan, Test Accuracy: 54.79%
100%|██████████| 117/117 [00:00<00:00, 373.95it/s]
Train Loss: tensor(1.5009, device='cuda:0')
100%|██████████| 19/19 [00:00<00:00, 622.69it/s]
Test Loss: nan, Test Accuracy: 63.85%
100%|██████████| 117/117 [00:00<00:00, 375.35it/s]
Train Loss: tensor(1.3047, device='cuda:0')
100%|██████████| 19/19 [00:00<00:00, 778.47it/s]
Test Loss: nan, Test Accuracy: 69.94%
100%|██████████| 117/117 [00:00<00:00, 430.43it/s]
Train Loss: tensor(1.1554, device='cuda:0')
100%|██████████| 19/19 [00:00<00:00, 745.10it/s]
Test Loss: nan, Test Accuracy: 73.62%
100%|██████████| 117/117 [00:00<00:00, 365.58it/s]
Train Loss: tensor(1.0425, device='cuda:0')
100%|██████████| 19/19 [00:00<00:00, 464.30it/s]
Test Loss: nan, Test Accuracy: 76.04%
100%|██████████| 117/117 [00:00<00:00, 380.09it/s]
Train Loss: tensor(0.9545, device='cuda:0')
100%|██████████| 19/19 [00:00<00:00, 786.78it/s]
Test Loss: nan, Test Accuracy: 77.99%
100%|██████████| 117/117 [00:00<00:00, 431.62it/s]
Train Loss: tensor(0.8842, device='cuda:0')
100%|██████████| 19/19 [00:00<00:00, 648.84it/s]
Test Loss: nan, Test Accuracy: 79.48%
100%|██████████| 117/117 [00:00<00:00, 360.97it/s]
Train Loss: tensor(0.8268, device='cuda:0')
100%|██████████| 19/19 [00:00<00:00, 734.97it/s]
Test Loss: nan, Test Accuracy: 80.89%
100%|██████████| 117/117 [00:00<00:00, 395.25it/s]
Train Loss: tensor(0.7790, device='cuda:0')
```

```
Train Loss: tensor(0.7790, device='cuda:0')
100%|██████████| 19/19 [00:00<00:00, 560.65it/s]
Test Loss: nan, Test Accuracy: 81.93%
100%|██████████| 117/117 [00:00<00:00, 333.75it/s]
Train Loss: tensor(0.7387, device='cuda:0')
100%|██████████| 19/19 [00:00<00:00, 745.91it/s]
Test Loss: nan, Test Accuracy: 82.78%
100%|██████████| 117/117 [00:00<00:00, 407.80it/s]
Train Loss: tensor(0.7039, device='cuda:0')
100%|██████████| 19/19 [00:00<00:00, 723.70it/s]
Test Loss: nan, Test Accuracy: 83.43%
100%|██████████| 117/117 [00:00<00:00, 404.61it/s]
Train Loss: tensor(0.6742, device='cuda:0')
100%|██████████| 19/19 [00:00<00:00, 606.90it/s]
Test Loss: nan, Test Accuracy: 84.12%
100%|██████████| 117/117 [00:00<00:00, 402.24it/s]
Train Loss: tensor(0.6481, device='cuda:0')
100%|██████████| 19/19 [00:00<00:00, 721.45it/s]
Test Loss: nan, Test Accuracy: 84.39%
100%|██████████| 117/117 [00:00<00:00, 378.56it/s]
Train Loss: tensor(0.6246, device='cuda:0')
100%|██████████| 19/19 [00:00<00:00, 723.02it/s]
Test Loss: nan, Test Accuracy: 84.93%
100%|██████████| 117/117 [00:00<00:00, 410.07it/s]
Train Loss: tensor(0.6039, device='cuda:0')
100%|██████████| 19/19 [00:00<00:00, 674.61it/s]
Test Loss: nan, Test Accuracy: 85.33%
100%|██████████| 117/117 [00:00<00:00, 394.16it/s]
Train Loss: tensor(0.5856, device='cuda:0')
100%|██████████| 19/19 [00:00<00:00, 716.28it/s]
Test Loss: nan, Test Accuracy: 85.58%
100%|██████████| 117/117 [00:00<00:00, 343.53it/s]
Train Loss: tensor(0.5689, device='cuda:0')
100%|██████████| 19/19 [00:00<00:00, 711.54it/s]
Test Loss: 0.5462266802787781, Test Accuracy: 85.92%
100%|██████████| 117/117 [00:00<00:00, 423.19it/s]
Train Loss: tensor(0.5539, device='cuda:0')
100%|██████████| 19/19 [00:00<00:00, 764.66it/s]
Test Loss: nan, Test Accuracy: 86.18%
100%|██████████| 117/117 [00:00<00:00, 349.62it/s]
Train Loss: tensor(0.5397, device='cuda:0')
100%|██████████| 19/19 [00:00<00:00, 730.28it/s]
Test Loss: nan, Test Accuracy: 86.47%
100%|██████████| 117/117 [00:00<00:00, 378.54it/s]
Train Loss: tensor(0.5270, device='cuda:0')
100%|██████████| 19/19 [00:00<00:00, 728.76it/s]
Test Loss: nan, Test Accuracy: 86.78%
100%|██████████| 117/117 [00:00<00:00, 389.00it/s]
Train Loss: tensor(0.5151, device='cuda:0')
100%|██████████| 19/19 [00:00<00:00, 704.44it/s]
Test Loss: nan, Test Accuracy: 87.08%
100%|██████████| 117/117 [00:00<00:00, 406.94it/s]
Train Loss: tensor(0.5044, device='cuda:0')
100%|██████████| 19/19 [00:00<00:00, 717.64it/s]
Test Loss: nan, Test Accuracy: 87.27%
100%|██████████| 117/117 [00:00<00:00, 338.75it/s]
Train Loss: tensor(0.4944, device='cuda:0')
100%|██████████| 19/19 [00:00<00:00, 805.28it/s]
Test Loss: nan, Test Accuracy: 87.58%
```