MNIST DATASET

Name: Valentin Gundorov

Id:336490271

Description:

In this project, I developed an artificial neural network (ANN) model capable of accurately classifying handwritten digits using the MNIST dataset. The MNIST dataset comprises 60,000 training images and 10,000 test images of handwritten digits, each sized 28x28 pixels. Our objective was to construct a reliable ANN model capable of achieving high accuracy on this standard dataset.

1. *Data Preparation:*

   - *The MNIST dataset was downloaded and preprocessed for training and testing purposes.*
   - *The data was divided into training and testing sets with an 80% to 20% ratio.*

```
Dataset MNIST
    Number of datapoints: 60000
    Root location: Data
    Split: Train
    StandardTransform
Transform: ToTensor()
```

```
Dataset MNIST
    Number of datapoints: 10000
    Root location: Data
    Split: Test
    StandardTransform
Transform: ToTensor()
```

2. *Examine a training record and exploring data:*

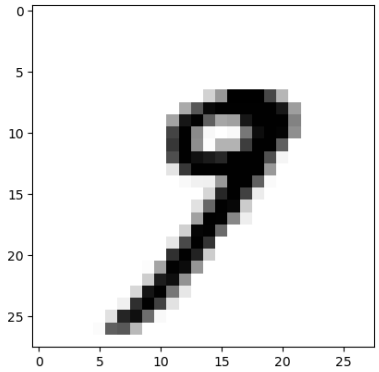
```
          [0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000,
           0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000,
           0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000,
           0.0000, 0.0000, 0.0000, 0.0000]]]),
9)
```

```
image.shape
```

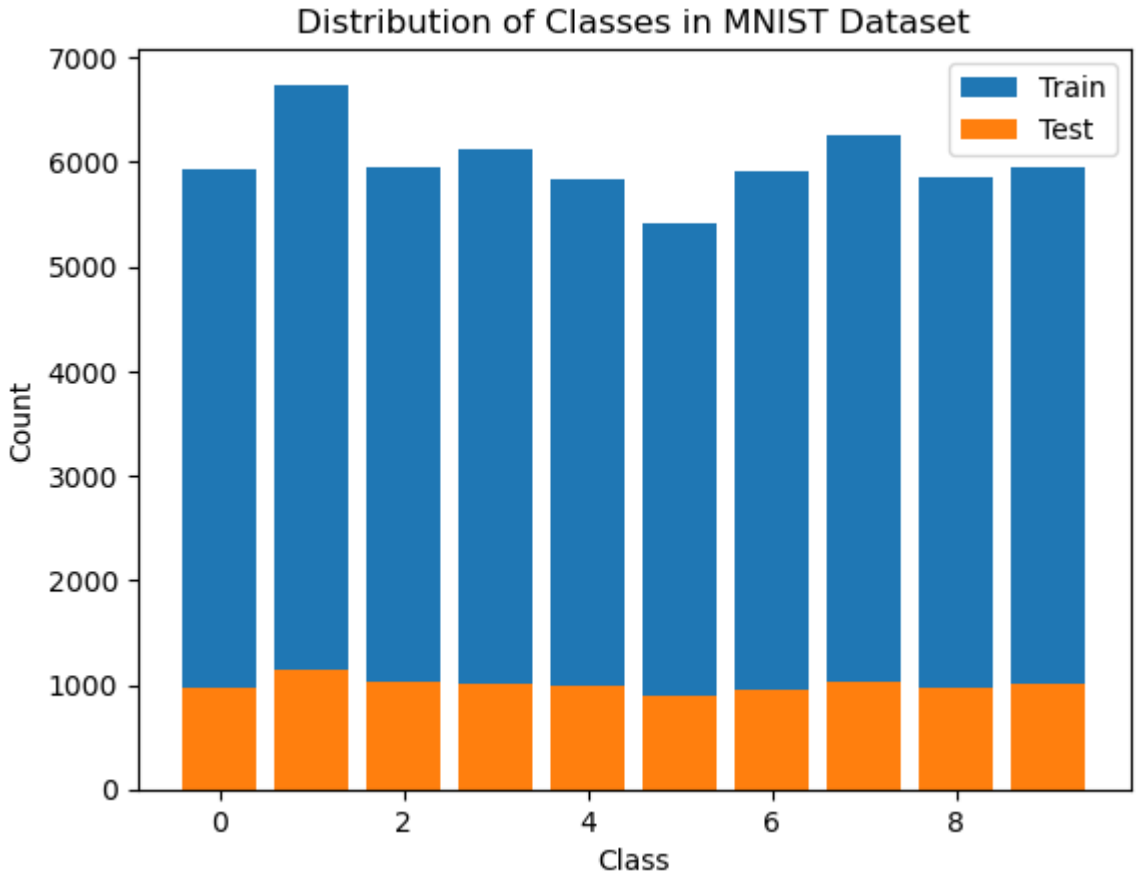
```
torch.Size([1, 28, 28])
```

```
label
```

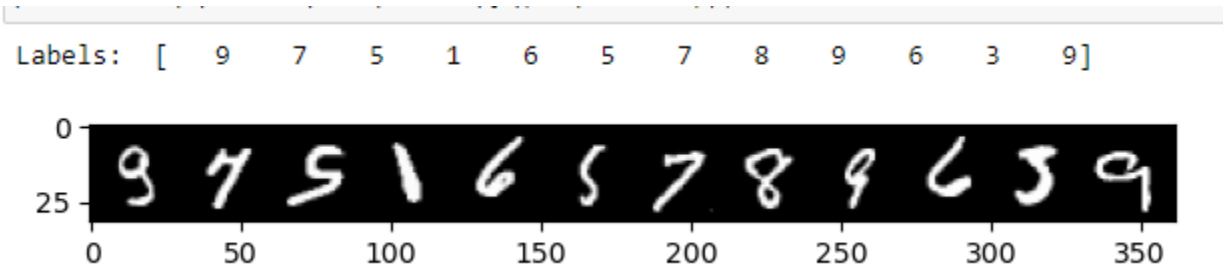
```
9
```



3. Class distribution.



4. *View examples*



5. *Model Architecture:*

Note: Due to the fact that this is my first work with neural networks, I decided to study ANN, remembering that the second project will be on CNN.

- *I designed an artificial neural network (ANN) architecture consisting of multiple fully connected layers.*
- *The choice of ANN was based on its ability to learn complex relationships between input and output data.*
- *Various configurations of layers and hyperparameters were experimented with to optimize the model's performance.*

6. *Count Params:*

```
94080
120
10080
84
840
10
```

94080: all conections from 784 to 120

120: biases

10080 :all connectors into next layer

84: biases

840:all connections into the next layer 84 prevoius to 10 output neurons

10: output neurons

sum of those 105,214 total parameters that are being adjusted by the network

- *Model Complexity Assessment: The number of parameters gives an idea of the model's complexity.*
- *Controlling Model Size: Knowing the number of parameters helps estimate the model's size and its memory requirements*
- *Task Complexity Assessment: The number of parameters in the model can also provide an idea of the task's complexity in machine learning.*

7. *Model Training:*
    - *The model was trained on the training dataset using backpropagation and gradient descent algorithms.*
    - *Techniques such as dropout regularization were applied to prevent overfitting.*
    - *Optimal model parameters such as learning rate and number of epochs were fine-tuned using cross-validation methods.*

```
Epoch 0  Training Loss: 0.33522 Training Accuracy: 79.83%
Epoch 1  Training Loss: 0.36547 Training Accuracy: 90.51%
Epoch 2  Training Loss: 0.27985 Training Accuracy: 92.32%
Epoch 3  Training Loss: 0.25639 Training Accuracy: 93.23%
Epoch 4  Training Loss: 0.25415 Training Accuracy: 93.65%
Epoch 5  Training Loss: 0.18530 Training Accuracy: 94.14%
Epoch 6  Training Loss: 0.23994 Training Accuracy: 94.42%
Epoch 7  Training Loss: 0.32552 Training Accuracy: 94.63%
Epoch 8  Training Loss: 0.14443 Training Accuracy: 94.94%
Epoch 9  Training Loss: 0.07804 Training Accuracy: 95.06%
Epoch 10  Training Loss: 0.12332 Training Accuracy: 95.18%
```

    - *as you can see, after the 5th epoch the accuracy almost does not change, so I decided to stop at 10.*

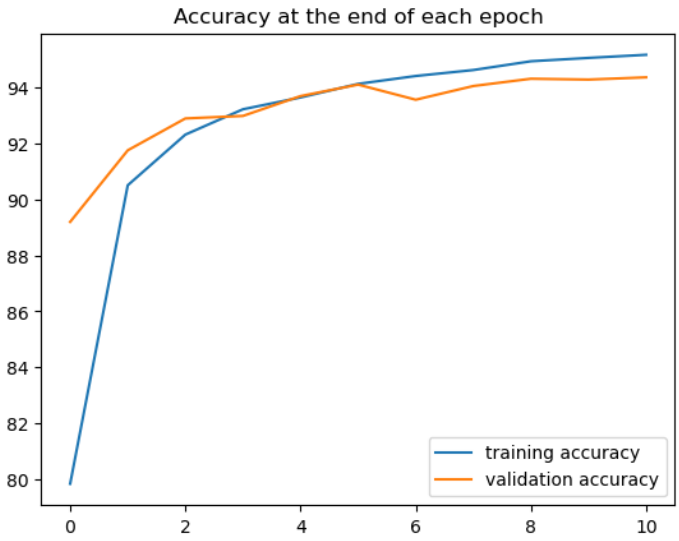
8. Plot the loss and accuracy comparisons: - *Model Evaluation:*

- *We can see that we're not performing as well on our test and validation set as we are on the training set. This is data that the network has never seen before. We wouldn't expect it to do as well on the test or validation set as it did on the training set since it doesn't actually get to adjust it's weights and biases based off the test set. We said before "torch.no_grad()".That information never seen before and it doesn't really get to adjust to it.*
- *We see that validation loss is doing down, but it looks like it's leveling off after 8-9 epochs.*
- *"Dynamics of Loss: The graph depicts the change in loss on both training and validation data during the training of an Artificial Neural Network (ANN) on the MNIST dataset. After an initial increase, the validation loss decreases, indicating an improvement in the model's performance.*

- *Model Training: Decreasing loss on the validation data suggests that the model is successfully learning and adapting to the data, enhancing its generalization ability."*

```
Test accuracy: 97.210%
```

- *The achieved test accuracy of 97.210% reflects commendable performance, showing the model's proficiency in accurately classifying data. Such high precision signifies a robust understanding and classification ability, indicating substantial learning and effective generalization of patterns within the dataset.*



- *Train accuracy is going to increase and get bigger and better as we train for more epochs.*

- *High model trainability: The increase in training accuracy to 96% by the 10th epoch indicates that the model learns well and can predict class labels accurately on the training dataset.*

- *Relatively high generalization ability: The initial value of validation accuracy at 80%, which increases to 94% by the 6th epoch, suggests that the model generalizes well to new data, despite experiencing slight improvement after the initial period.*

- *Lack of overfitting: The fact that validation accuracy remains stable or slowly increases after reaching a certain level may indicate that the model is not suffering from overfitting to the training data.*

9. *Results:*

- *My trained ANN model achieved high accuracy on the test dataset, demonstrating its capability to accurately recognize handwritten digits.*

Examine the misses

```
Total number of misses: 569
First 10 misses indices: [  18   33   62   72   83   84  124  133  175  195]
```

```
misses[:20]
```

```
array([  18,   33,   62,   72,   83,   84,  124,  133,  175,  195,  233,
        241,  245,  257,  259,  264,  266,  290,  313,  321], dtype=int64)
```

```
Indices:     [  18   33   62   72   83   84  124  133  175  195]
Labels:      [   3    4    9    2    7    8    7    7    7    3]
Predictions: [   8    0    8    3    9    5    4    3    3    5]
```

- *By examining the misses, we delving into the nuanced intricacies of the model's performance. This process serves a twofold purpose: first, it offers valuable insights into the specific patterns or instances where the model struggles, potentially highlighting areas for refinement or augmentation. Second, it provides an opportunity for iterative learning, allowing for the refinement of the model through targeted adjustments or feature engineering based on the identified shortcomings. In essence, delving into the misses is a strategic endeavor aimed at enhancing the model's overall efficacy and ensuring its adaptability.*

10. *Conclusion:*

In the MNIST project using ANN, a high classification accuracy was achieved on the test dataset (97.210%). Analyzing the losses, it's evident that the model consistently improves its performance with each training epoch. Additionally, error analysis helps identify areas for further model enhancement. I am pleased with these results, which confirm the effectiveness of using artificial neural networks in handwritten digit recognition tasks and open opportunities for further research and optimization.

Thank you for your attention and goodbye!
Valentine.