Project 5 – Recogmition using Deep Networks

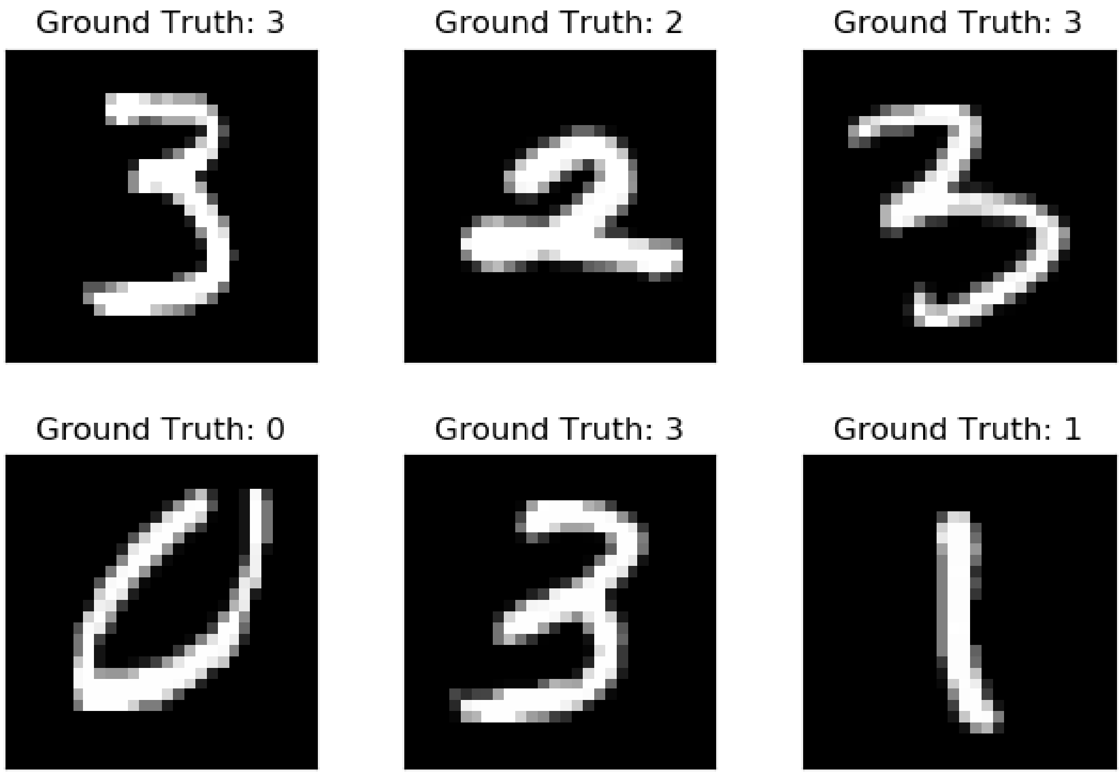
Abinav Anantharaman | NUID: 002774223

Satwik Shridhar Bhandiwad | NUID: 002920338

Introduction

This project focuses on building, training, analyzing, and modifying a deep neural network for the recognition of handwritten digits using the MNIST dataset. The goal is to demonstrate the potential of deep learning by creating a model that can accurately identify handwritten digits. The project is designed to be accessible without the need for a GPU and provides a good example of the capabilities of deep neural networks. Throughout the project, various techniques will be explored to improve the model's performance, providing insight into the process of building and improving deep learning models.

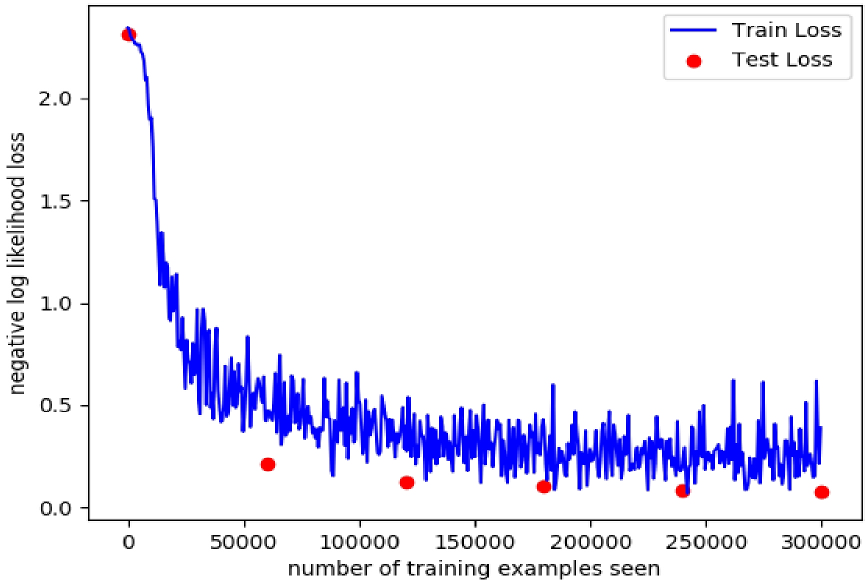
Task 1A: Get the MNIST digit data set



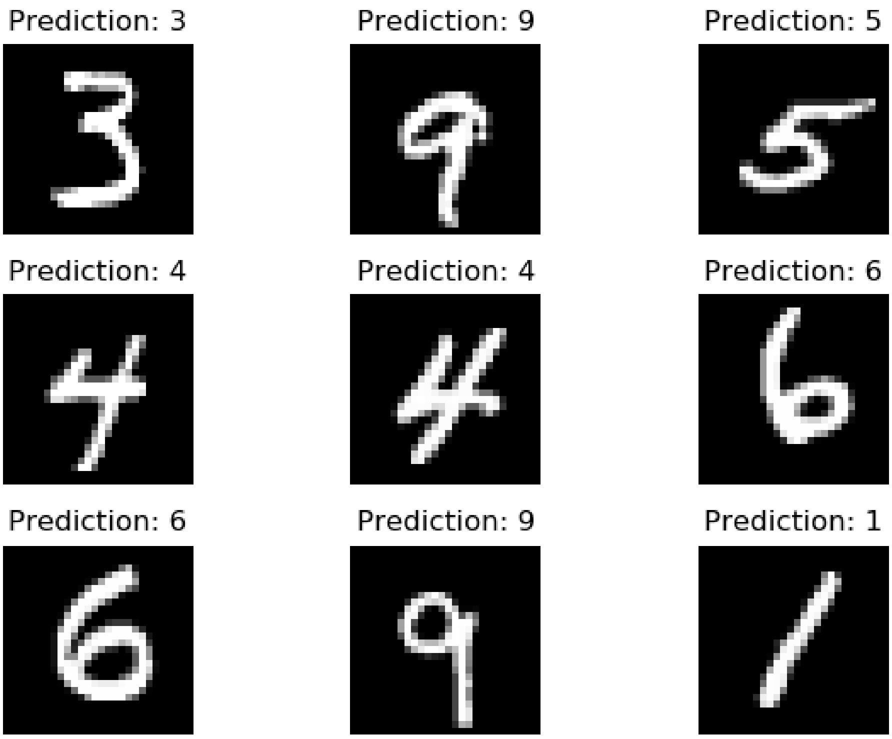
Task 1C: Build a network model

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1           [-1, 10, 24, 24]             260
            Conv2d-2             [-1, 20, 8, 8]           5,020
         Dropout2d-3             [-1, 20, 8, 8]               0
            Linear-4                   [-1, 50]          16,050
            Linear-5                   [-1, 10]             510
================================================================
Total params: 21,840
Trainable params: 21,840
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.00
Forward/backward pass size (MB): 0.06
Params size (MB): 0.08
Estimated Total Size (MB): 0.15
----------------------------------------------------------------
Net(
  (conv1): Conv2d(1, 10, kernel_size=(5, 5), stride=(1, 1))
  (conv2): Conv2d(10, 20, kernel_size=(5, 5), stride=(1, 1))
  (conv2_drop): Dropout2d(p=0.5, inplace=False)
  (flatten): Flatten(start_dim=1, end_dim=-1)
  (fc1): Linear(in_features=320, out_features=50, bias=True)
  (fc2): Linear(in_features=50, out_features=10, bias=True)
)
```

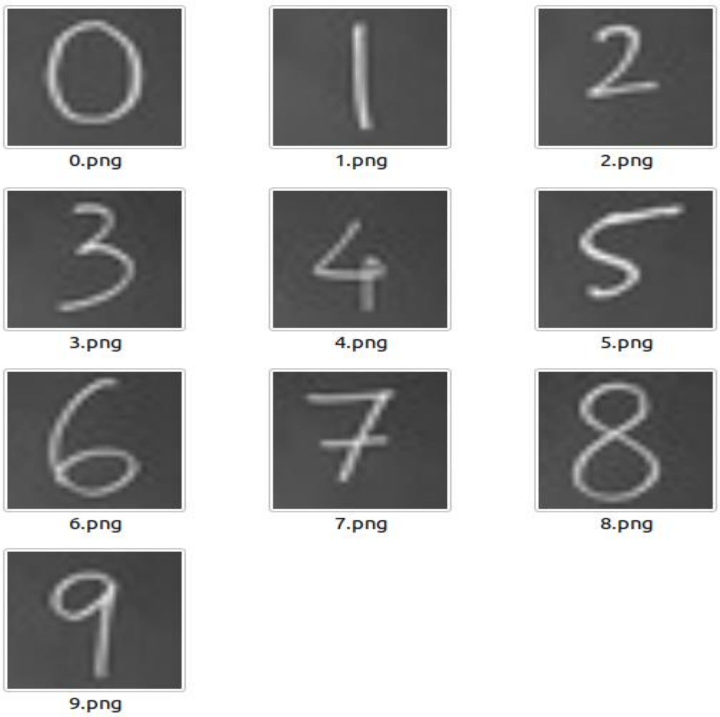
Task 1D: Train the model



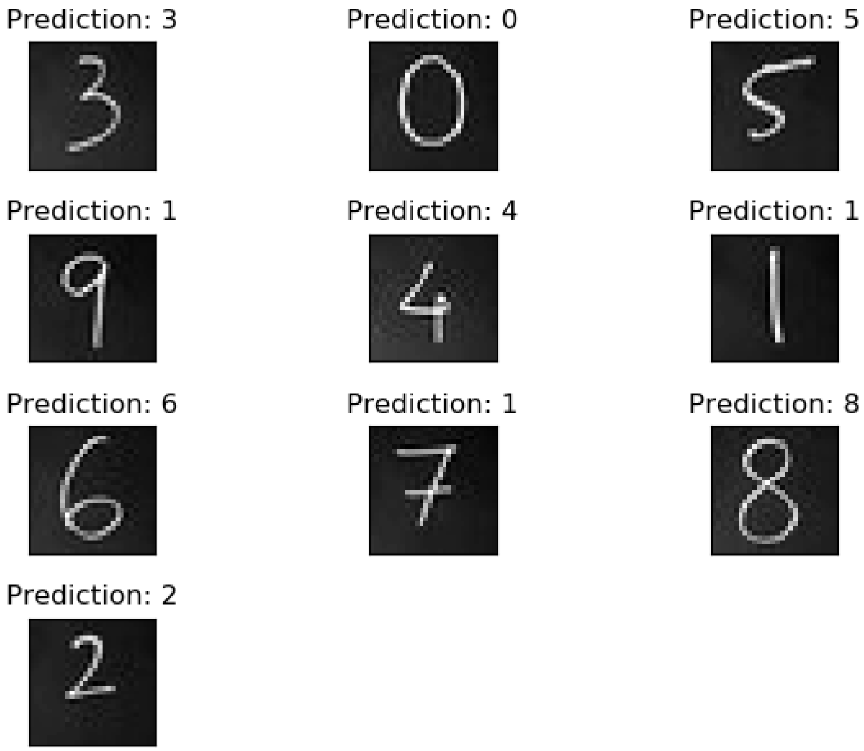
Task 1F: Read the network and run it on the test set



Task 1G: Test the network on new inputs

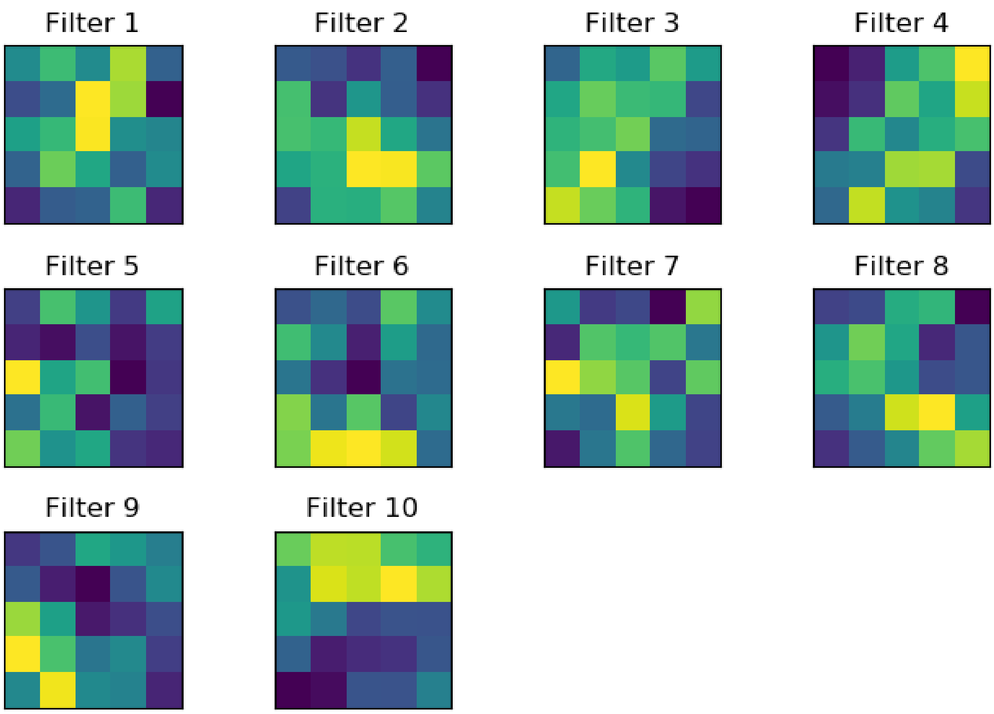


Handwritten digits resized to 28 x 28 and converted to grayscale and inverted.

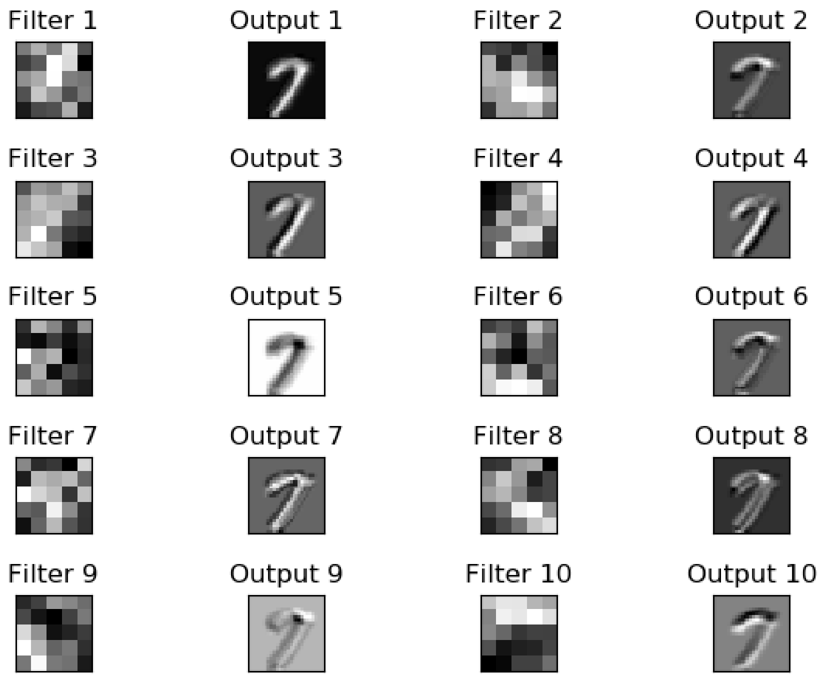


Got 80 percent accuracy on above handwritten image dataset.

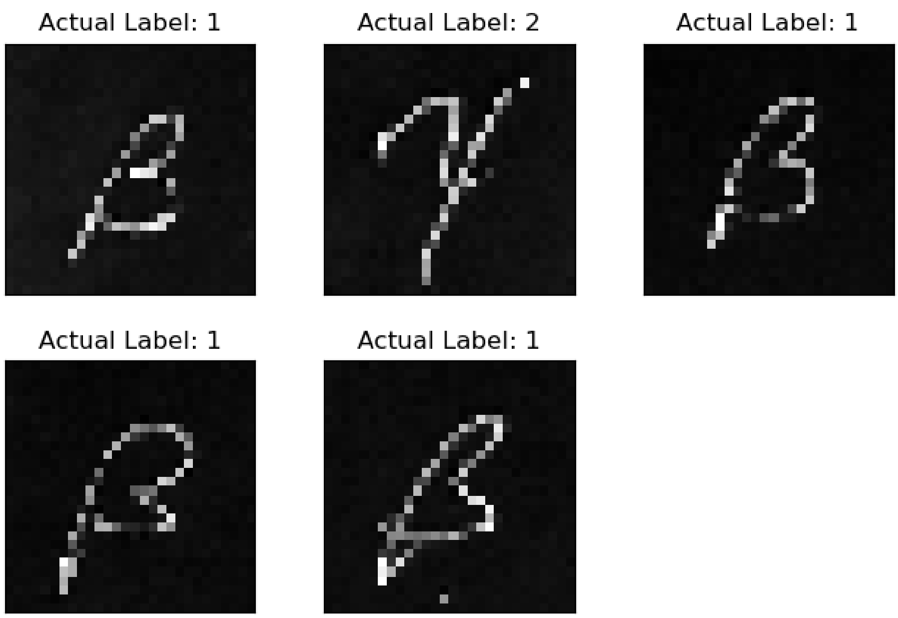
Task 2A: Analyze the first layer



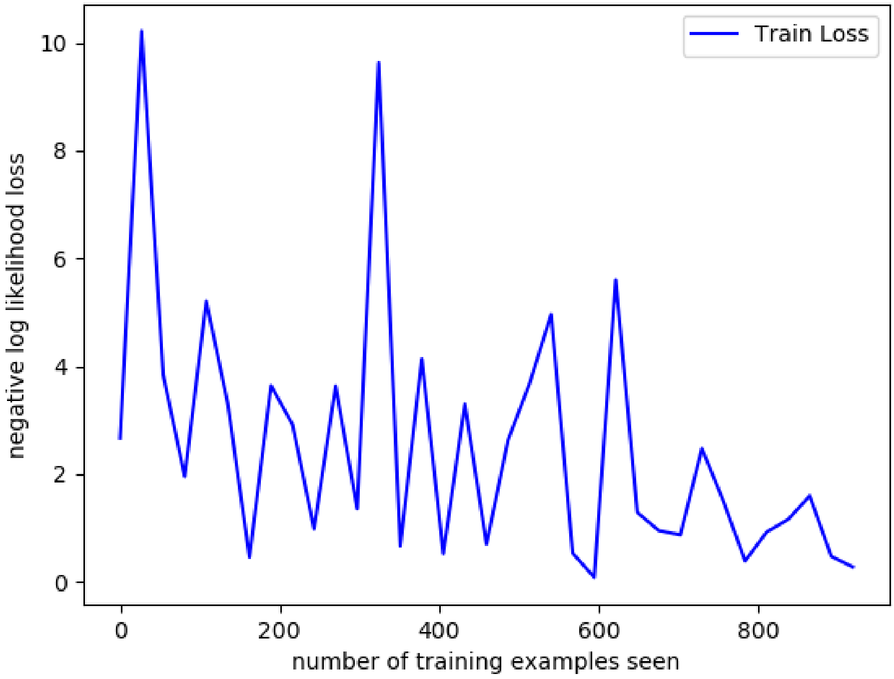
Task 2B: Show the effect of filters



Task 3: Transfer Learning on Greek Letters



Above are the sample images used for training

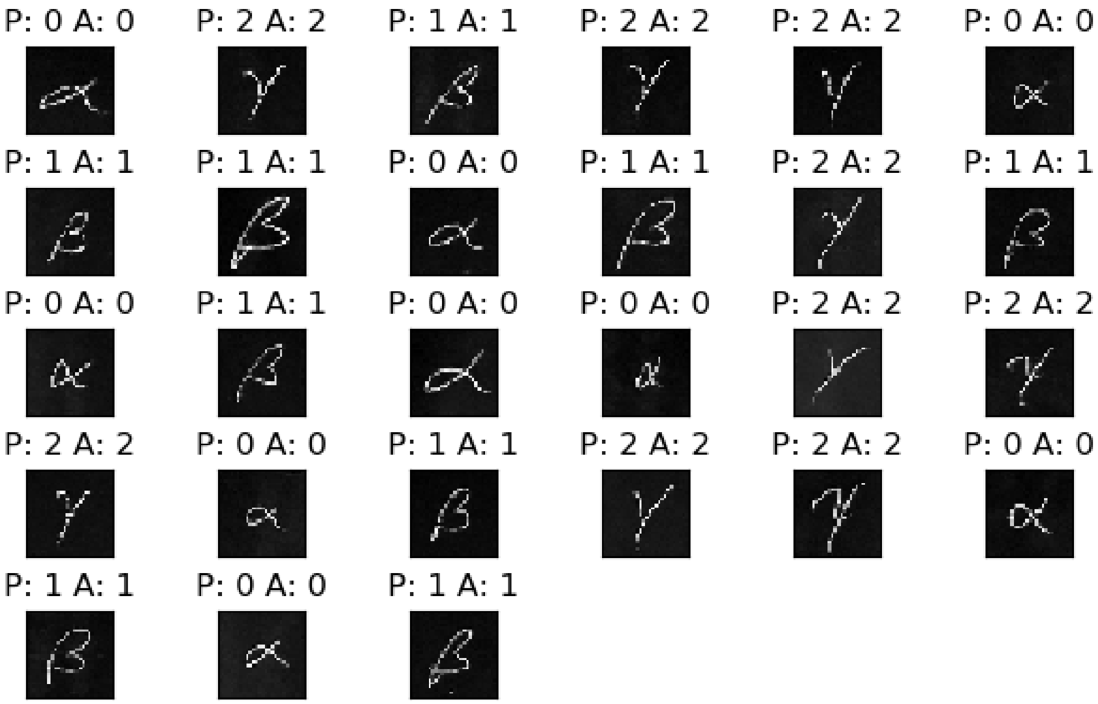


Above is the training Loss for Greek letters.

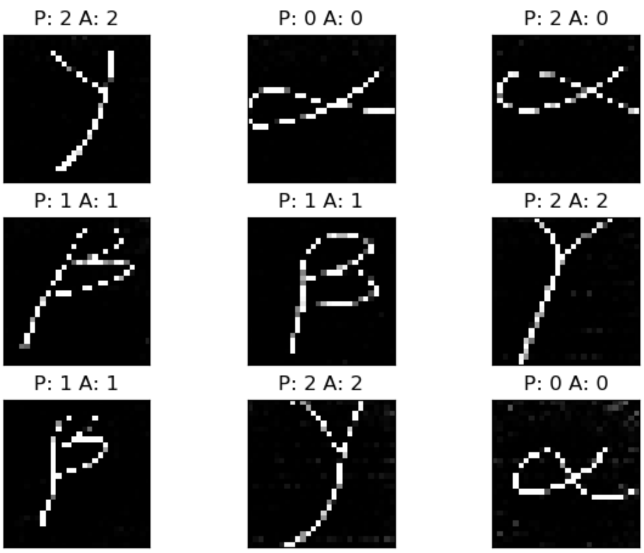
```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1           [-1, 10, 24, 24]             260
            Conv2d-2             [-1, 20, 8, 8]           5,020
         Dropout2d-3             [-1, 20, 8, 8]               0
            Linear-4                   [-1, 50]          16,050
            Linear-5                    [-1, 3]             153
================================================================
Total params: 21,483
Trainable params: 153
Non-trainable params: 21,330
----------------------------------------------------------------
Input size (MB): 0.00
Forward/backward pass size (MB): 0.06
Params size (MB): 0.08
Estimated Total Size (MB): 0.15
----------------------------------------------------------------


Net(
  (conv1): Conv2d(1, 10, kernel_size=(5, 5), stride=(1, 1))
  (conv2): Conv2d(10, 20, kernel_size=(5, 5), stride=(1, 1))
  (conv2_drop): Dropout2d(p=0.5, inplace=False)
  (flatten): Flatten(start_dim=1, end_dim=-1)
  (fc1): Linear(in_features=320, out_features=50, bias=True)
  (fc2): Linear(in_features=50, out_features=3, bias=True)
)
```

Above is the Network Diagram



Got 100 percent accuracy on above test data.
P = Predicted. A = Actual
0 = Alpha, 1 = Beta, 2 = Gamma



Got 88.88 percent result for above handwritten dataset.

Task 4A: Develop a plan

We changed 5 dimensions, which are:

- The number of convolution layers = 1 to 4
- The size of the convolution filters = 3, 5, 7
- The dropout rates of the Dropout layer = 0.3, 0.5
- The number of epochs of training = 3 and 5
- The batch size while training = 64, 128

The combination of above options lead to 72 different network variations

Task 4B: Hypothesis

- We expect that adding more convolution layers will improve accuracy but increase training time.
- We expect that using larger filter sizes will improve accuracy, but increase training time.
- We expect that adding another dropout layer after the fully connected layer will improve accuracy but increase training time.
- We expect that using more epochs will improve accuracy but increase training time.
- We expect that using a smaller batch size will increase accuracy but increase training time.

Task 4C: Execute your plan

After running the experiments, we found that our hypotheses were mostly supported. Adding more convolution layers improved accuracy but increased training time. Using larger filter sizes improved accuracy but increased training time. Adding another dropout layer after the fully connected layer improved accuracy but increased training time. Using more epochs improved accuracy but increased training time. Using a smaller batch size increased accuracy but increased training time.

Learning Outcomes:

1. Understanding the fundamentals of building and training deep neural networks for image recognition tasks using the MNIST dataset.
2. Ability to interpret the performance metrics of a trained network, including accuracy, loss, and confusion matrix, to evaluate the effectiveness of the model.
3. Knowledge of techniques for modifying and optimizing a neural network to improve its performance, including adjusting the architecture and hyperparameters, implementing regularization techniques, and using different activation functions.
4. Familiarity with deep learning tools and frameworks such as Pytorch, as well as proficiency in Python programming for implementing and analyzing deep learning models.

Acknowledgements

https://nextjournal.com/gkoehler/pytorch-mnist