CSCI 580 Final Project

Austin Kleinecke, Daniel Lopez, Andrew Roda

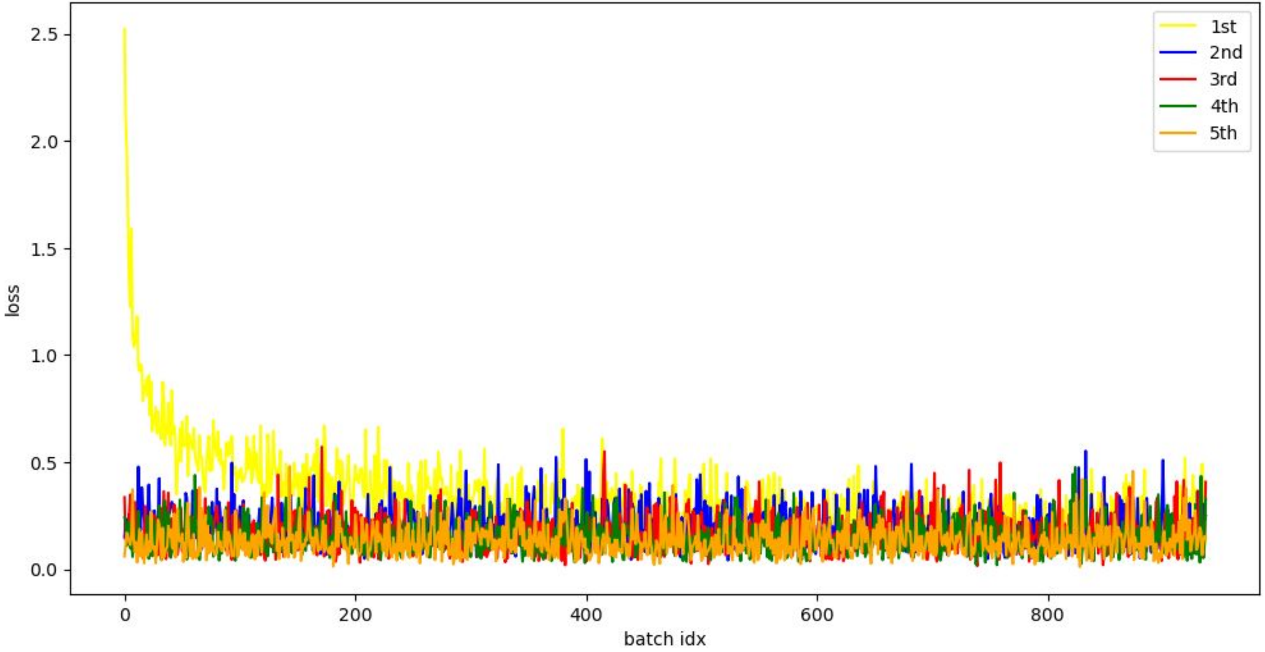
Preparing our model

```
model = nn.Sequential(nn.Linear(784, 128),   # Flattened MNIST image sizes (28x28) & Linear layer maps
                      nn.ELU(),              # Activation function
                      nn.BatchNorm1d(128),   # Normalize our batch for stability
                      nn.Dropout(0.2),       # Prevent overfitting
                      nn.Linear(128, 64),
                      nn.ELU(),
                      nn.BatchNorm1d(64),
                      nn.Dropout(0.2),
                      nn.Linear(64, 10),     # 10 output layers for each digit
                      nn.LogSoftmax(dim=1))  # this line is extra comparing to earlier nn.Sequential ca

# Check if we can run this on a GPU, otherwise use CPU
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
model = model.to(device)


## Loss function
criterion = nn.NLLLoss()
epochs = 5 # Number of training cycles
losses = []

## Training Loop
optimizer = optim.Adam(model.parameters(), lr=0.001)
```

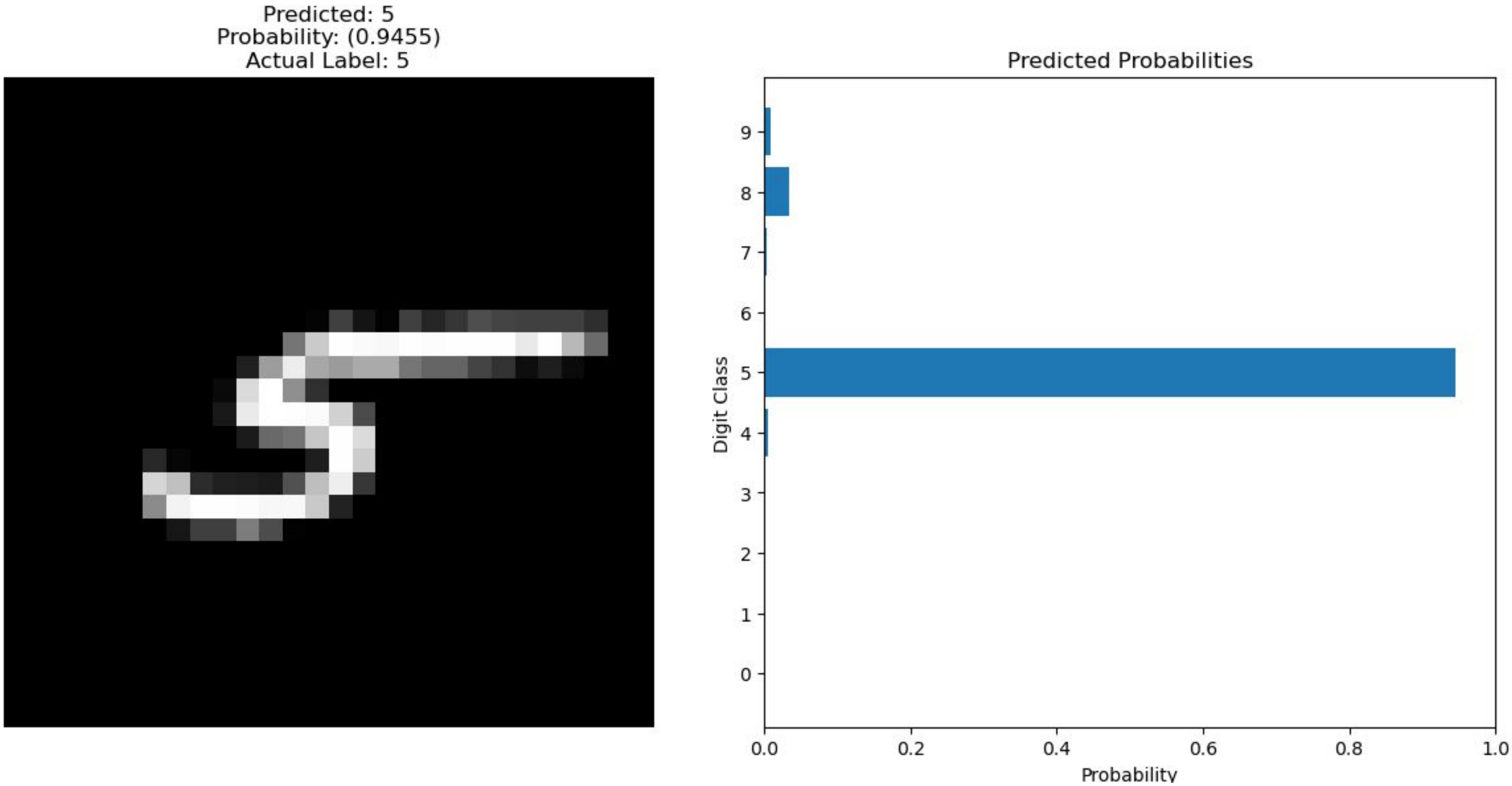
Training Loop

```
for epoch in range(epochs):
    model.train()
    running_loss = 0.0
    epoch_losses = []
    for images, labels in loader:
        # Zero out the gradients
        optimizer.zero_grad()
        # Forward pass
        output = model(images)
        # Calculate loss
        loss = criterion(output, labels)
        # Backward pass
        loss.backward()
        # Update weights
        optimizer.step()
        running_loss += loss.item()
        epoch_losses.append(loss.item())
    losses.append(epoch_losses)
```

MNIST Training Results



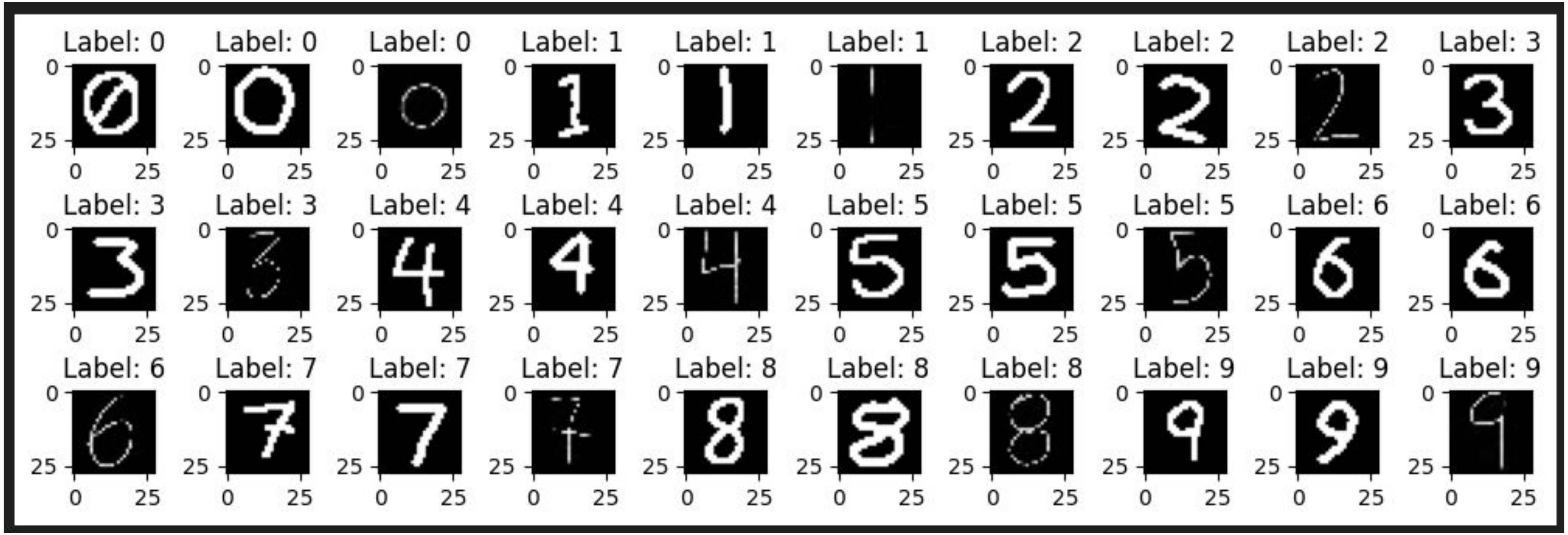

```
Epoch 1/5, Loss: 0.3355
Epoch 2/5, Loss: 0.1999
Epoch 3/5, Loss: 0.1659
Epoch 4/5, Loss: 0.1479
Epoch 5/5, Loss: 0.1354
```


Predicted: 5
Probability: (0.9455)
Actual Label: 5
Predicted Probabilities
Digit Class
Probability
0
1
2
3
4
5
6
7
8
9
0.0
0.2
0.4
0.6
0.8
1.0

Preparing the class dataset

- We loaded the images using the Pillow library
- Then, we converted the images into an array of numpy arrays and got the labels from the file name.
  - We also made sure to resize each image incase it wasn't 28x28
- We then saved the images and the labels into idx3 and idx1 respectively
  - same format used by the MNIST dataset
- We then loaded the arrays into a pytorch DataLoader to use with our model

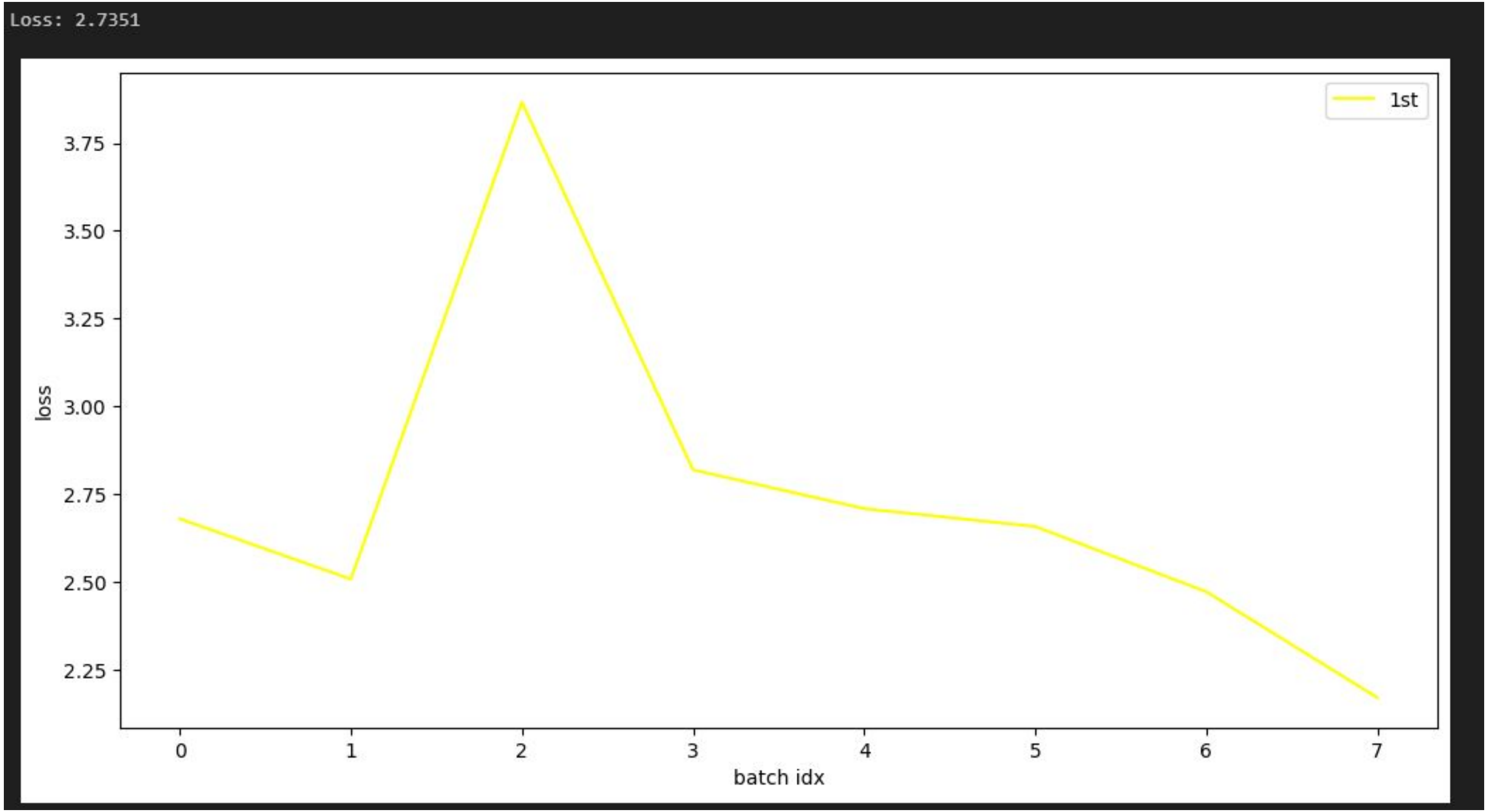
Images and Labels from IDX3/IDX1 in matplotlib



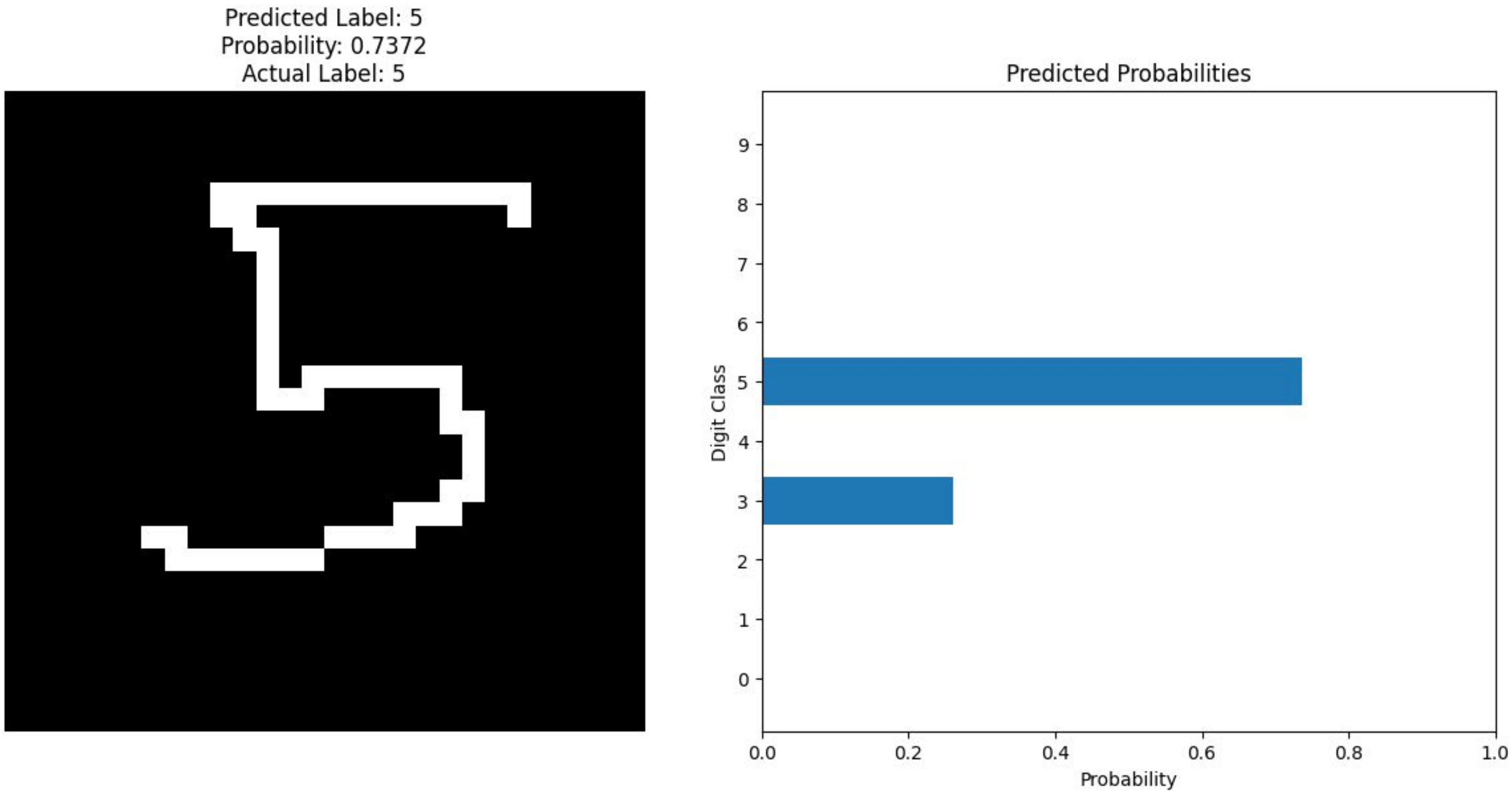
Running on the class dataset

- Using the model trained on the MNIST dataset we then ran a forward-only run on the class dataset to see how well the MNIST trained model would do on our class digit images.
- after getting the initial loss, we then further trained using the class dataset to see if we could improve the accuracy

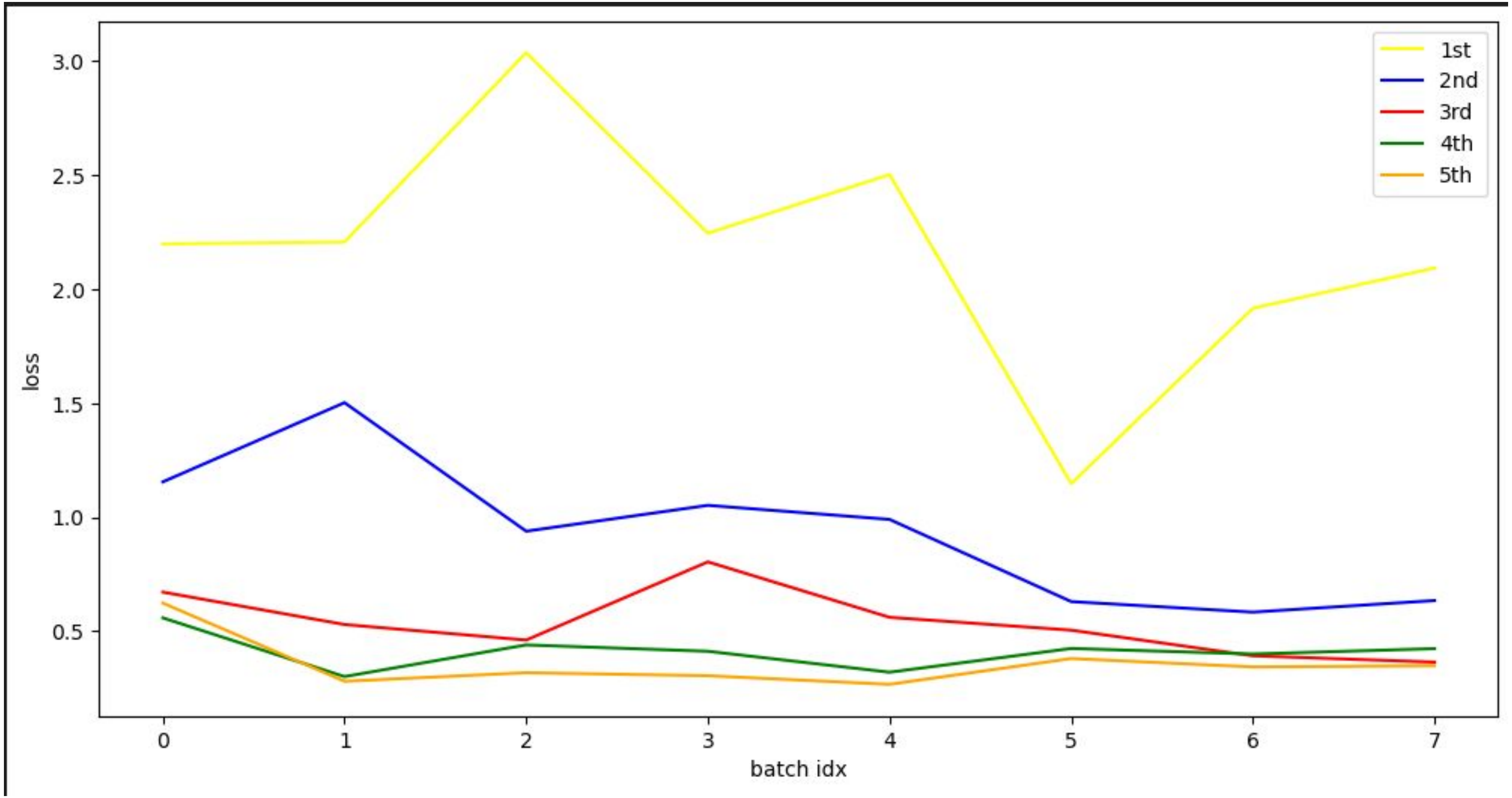
Initial Run (forward only, no training)



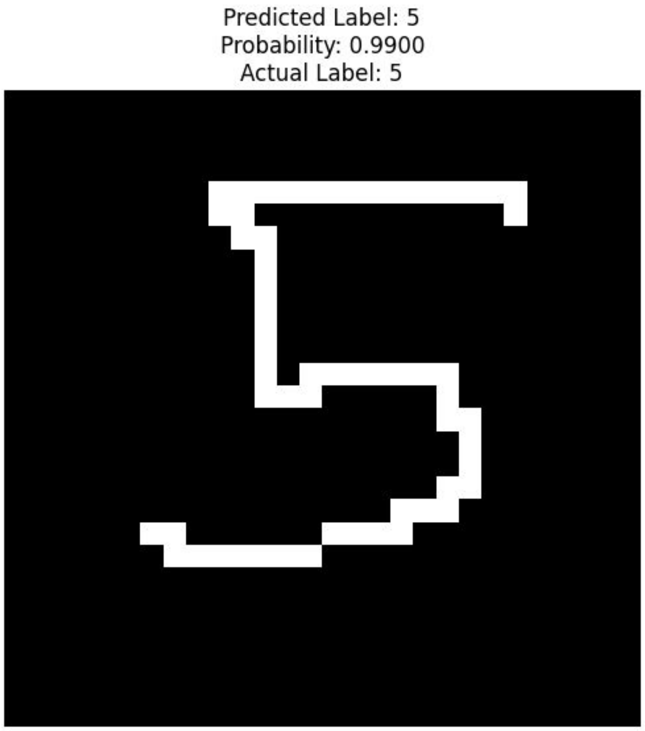
Initial Run continued

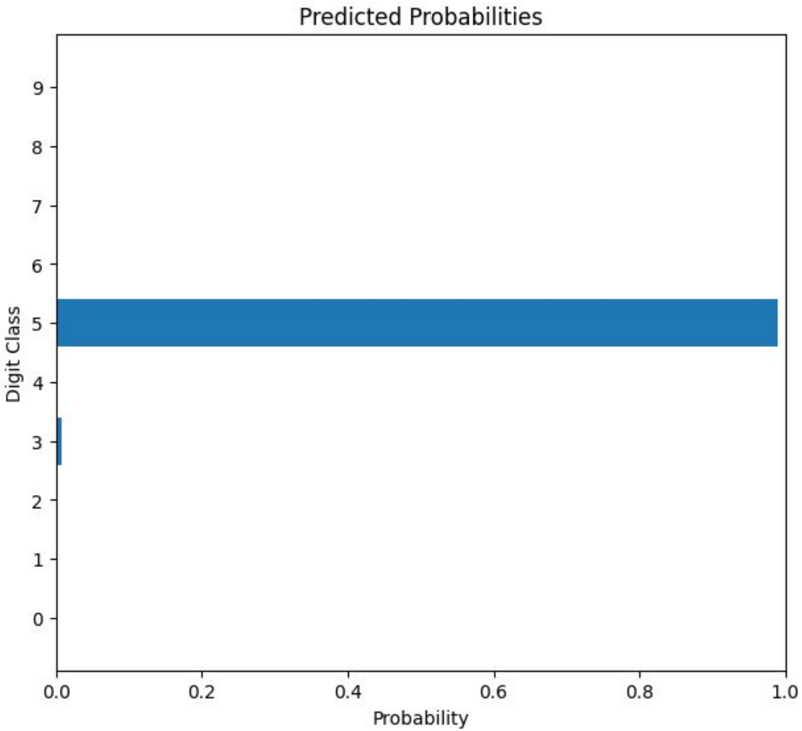


Train on class dataset



Trained Prediction

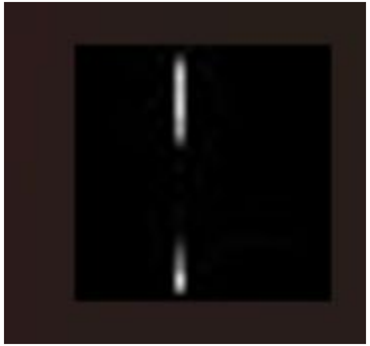




Conclusion

- Initial model trained on MNIST dataset was surprisingly inaccurate in its predictions on the test dataset.
- Further training on the test dataset improved accuracy to reasonable levels however.

Peak handwriting



we see you group 3

```
Image 0-3-4.png is not 28x28, it has shape (28, 26)
Group 3, shame on you!
Image 2-3-4.png is not 28x28, it has shape (28, 26)
Group 3, shame on you!
Image 5-3-4.png is not 28x28, it has shape (29, 28)
Group 3, shame on you!
Image 6-3-4.png is not 28x28, it has shape (29, 28)
Group 3, shame on you!
Image 8-3-4.png is not 28x28, it has shape (29, 28)
Group 3, shame on you!
```