Project 5: Recognition Using Deep Networks

Samara Holmes
Northeastern University

I. Introduction

This project will demonstrate how to build, train, and modify a deep neural network for image recognition tasks using the MNIST digit dataset. It will showcase the step-by-step process of creating a network, optimizing it for better performance, and analyzing its behavior. Additionally, it will highlight the effectiveness of deep learning models in recognizing patterns and solving classification problems, even with relatively simple datasets and basic hardware. Through this, the project will provide a clear and practical understanding of the core principles of deep learning.

II. Methodology

For this project, we have the following tasks:

1) Build and train a network to recognize digits
2) Examine the network
3) Transfer learning on Greek letters
4) Experiment with different dimensions
   i) The size of the convolution filters
   ii) The number of epochs of training
   iii) The batch size while training

Plan for experimentation:

1) Load the MNIST Fashion data set
2) Load the model
3) Train the model
4) Track the accuracy as the number of epochs increases (keeping filter size, batch size, and pool size constant)
5) Track the accuracy as the filter size increases (keeping epochs, batch size, and pool size constant)
6) Track the accuracy as the batch size increases (keeping epochs, filter size, and pool size constant)

Hypotheses for experimentation:

1) Increasing the size of the convolution filters will increase the accuracy
2) Increasing the number of epochs will increase the accuracy
3) Increasing the batch size will increase the accuracy

III. Results

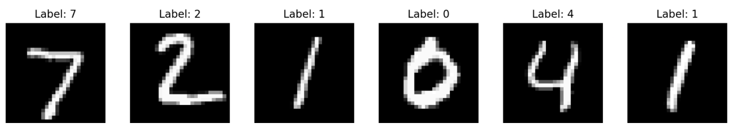


Figure 1. The first six digits from the test set

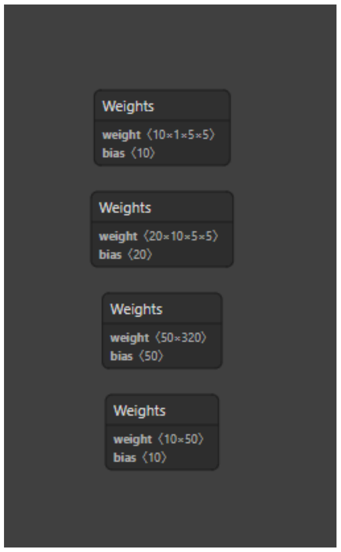


Figure 2. Network model diagram from Netron App

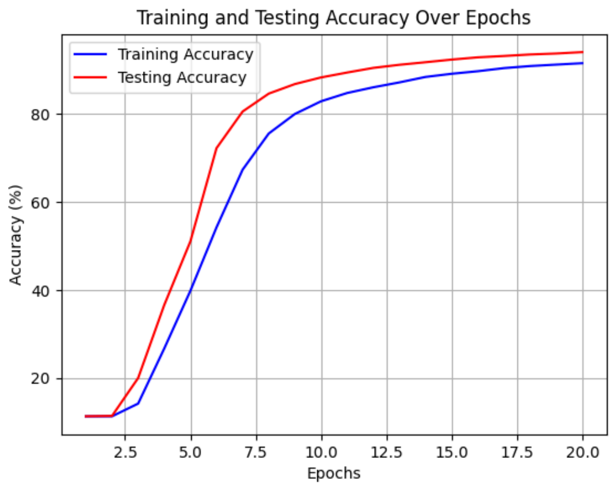
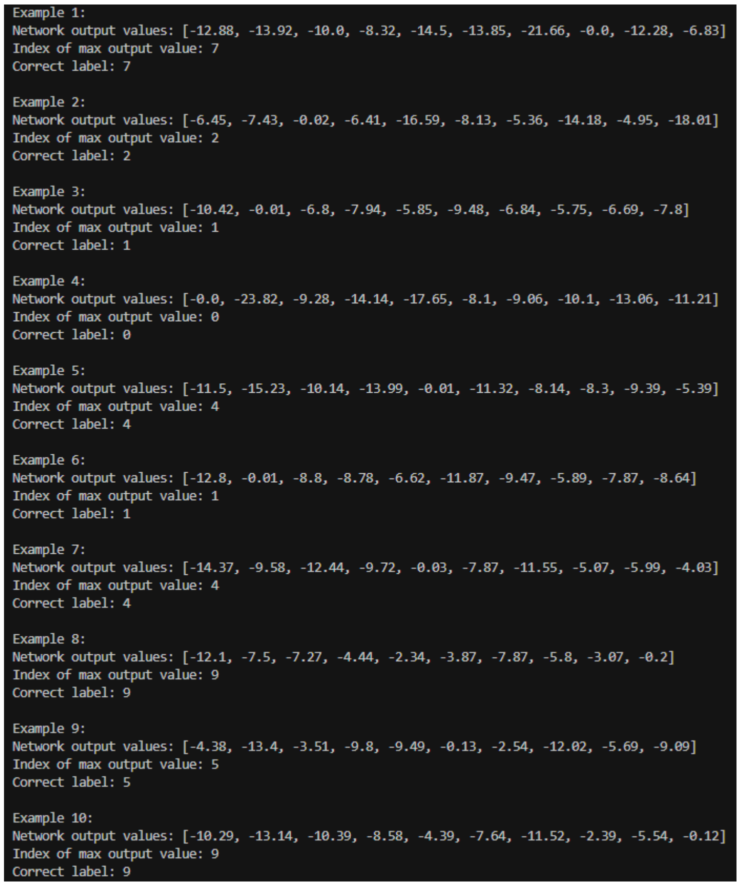


Figure 3. Training and test accuracy over epochs



```
Example 1:
Network output values: [-12.88, -13.92, -10.0, -8.32, -14.5, -13.85, -21.66, -0.0, -12.28, -6.83]
Index of max output value: 7
Correct label: 7

Example 2:
Network output values: [-6.45, -7.43, -0.02, -6.41, -16.59, -8.13, -5.36, -14.18, -4.95, -18.01]
Index of max output value: 2
Correct label: 2

Example 3:
Network output values: [-10.42, -0.01, -6.8, -7.94, -5.85, -9.48, -6.84, -5.75, -6.69, -7.8]
Index of max output value: 1
Correct label: 1

Example 4:
Network output values: [-0.0, -23.82, -9.28, -14.14, -17.65, -8.1, -9.06, -10.1, -13.06, -11.21]
Index of max output value: 0
Correct label: 0

Example 5:
Network output values: [-11.5, -15.23, -10.14, -13.99, -0.01, -11.32, -8.14, -8.3, -9.39, -5.39]
Index of max output value: 4
Correct label: 4

Example 6:
Network output values: [-12.8, -0.01, -8.8, -8.78, -6.62, -11.87, -9.47, -5.89, -7.87, -8.64]
Index of max output value: 1
Correct label: 1

Example 7:
Network output values: [-14.37, -9.58, -12.44, -9.72, -0.03, -7.87, -11.55, -5.07, -5.99, -4.03]
Index of max output value: 4
Correct label: 4

Example 8:
Network output values: [-12.1, -7.5, -7.27, -4.44, -2.34, -3.87, -7.87, -5.8, -3.07, -0.2]
Index of max output value: 9
Correct label: 9

Example 9:
Network output values: [-4.38, -13.4, -3.51, -9.8, -9.49, -0.13, -2.54, -12.02, -5.69, -9.09]
Index of max output value: 5
Correct label: 5

Example 10:
Network output values: [-10.29, -13.14, -10.39, -8.58, -4.39, -7.64, -11.52, -2.39, -5.54, -0.12]
Index of max output value: 9
Correct label: 9
```

Figure 4. Ten network output values, index of the max output value, correct label of the digit

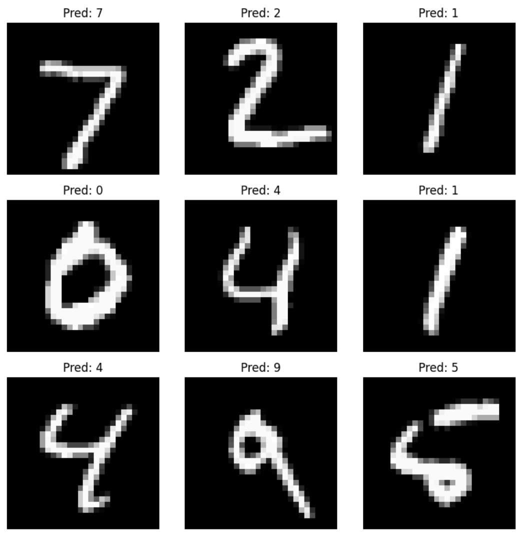


Figure 5. The first 9 digits after training the model

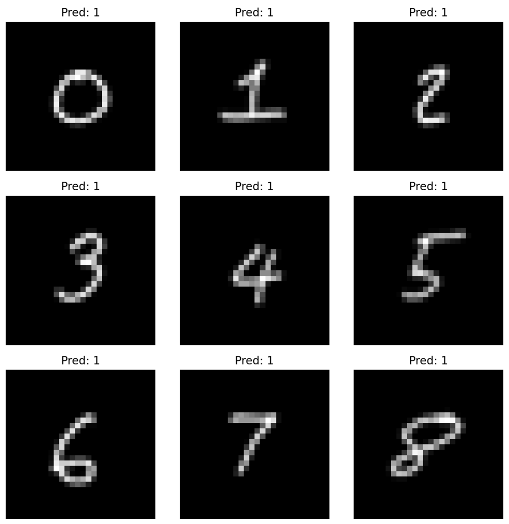


Figure 6a. Handwritten digits and their classified results (First attempt)

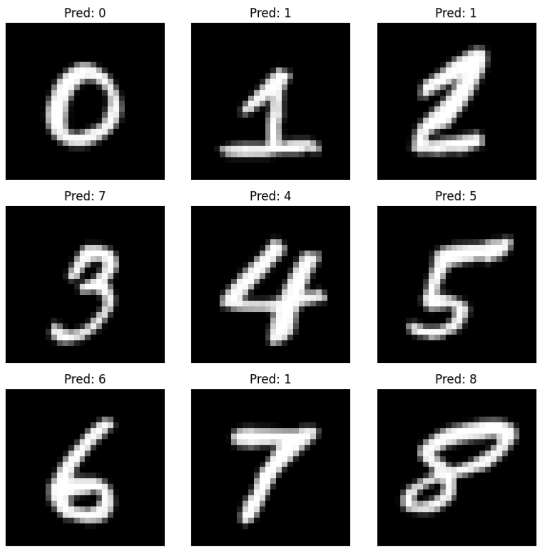


Figure 6b. Handwritten digits and their classified results (Second attempt)

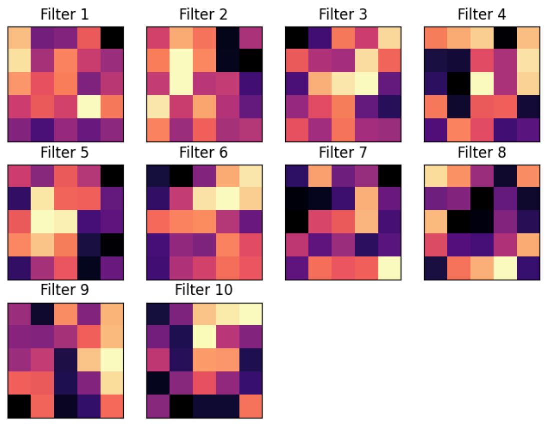


Figure 7. Filter visualization

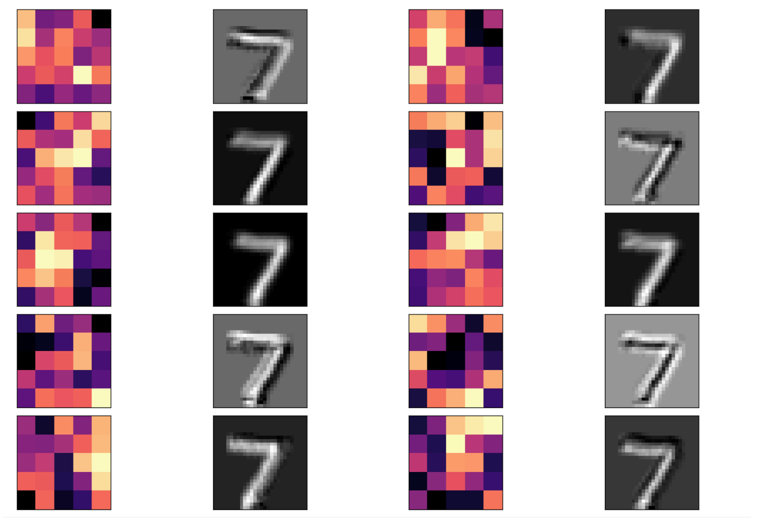


Figure 8. Filter effect visualization

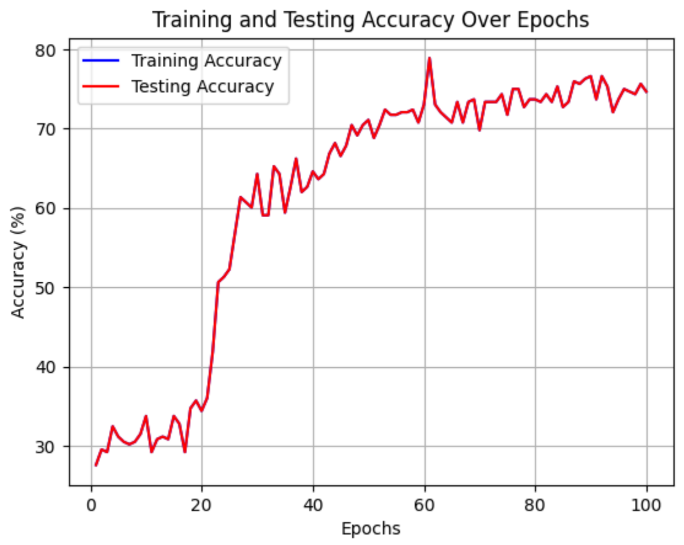
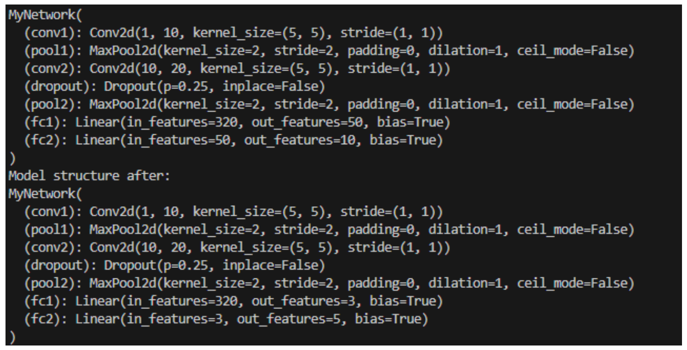
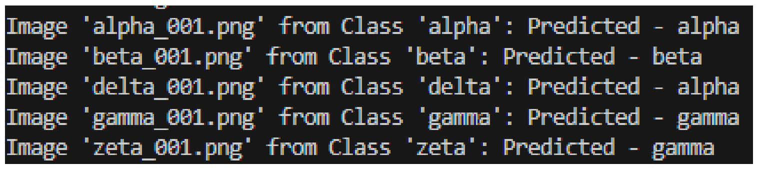


Figure 9. Training error (this graph isn't properly labeled since I reused a function)



```
MyNetwork(
  (conv1): Conv2d(1, 10, kernel_size=(5, 5), stride=(1, 1))
  (pool1): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (conv2): Conv2d(10, 20, kernel_size=(5, 5), stride=(1, 1))
  (dropout): Dropout(p=0.25, inplace=False)
  (pool2): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (fc1): Linear(in_features=320, out_features=50, bias=True)
  (fc2): Linear(in_features=50, out_features=10, bias=True)
)
Model structure after:
MyNetwork(
  (conv1): Conv2d(1, 10, kernel_size=(5, 5), stride=(1, 1))
  (pool1): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (conv2): Conv2d(10, 20, kernel_size=(5, 5), stride=(1, 1))
  (dropout): Dropout(p=0.25, inplace=False)
  (pool2): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (fc1): Linear(in_features=320, out_features=3, bias=True)
  (fc2): Linear(in_features=3, out_features=5, bias=True)
)
```

Figure 10. Modified network



```
Image 'alpha_001.png' from Class 'alpha': Predicted - alpha
Image 'beta_001.png' from Class 'beta': Predicted - beta
Image 'delta_001.png' from Class 'delta': Predicted - alpha
Image 'gamma_001.png' from Class 'gamma': Predicted - gamma
Image 'zeta_001.png' from Class 'zeta': Predicted - gamma
```

Figure 11.Images and their predicted labels

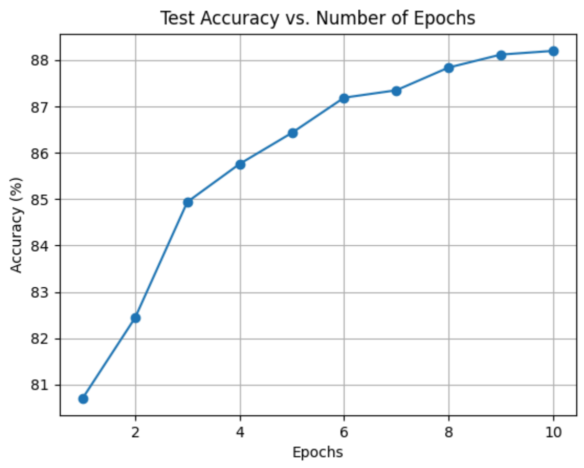


Figure 11. Number of epochs vs accuracy

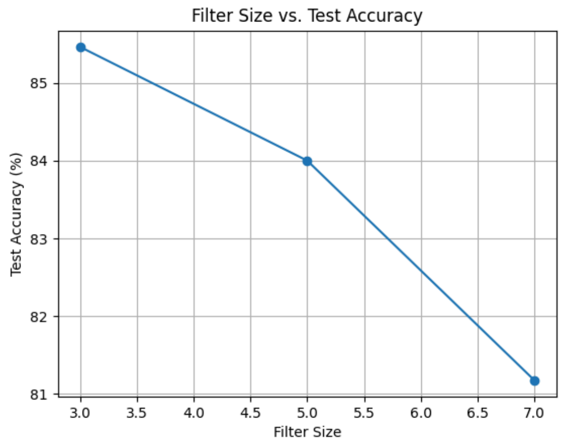


Figure 12. Filter size vs accuracy

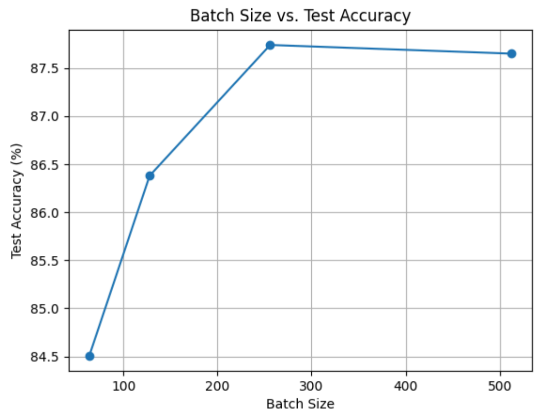


Figure 13. Batch size vs accuracy

IV. DISCUSSION

Figure 3 shows how the training and testing accuracy increases as the number of epochs increases.

Figure 5 shows the 9 digits and their classifications after I trained the model. What I noticed was that the first time I trained the model using 5 epochs, resulted in 3/9 of the digits being misclassified.

However, when I retrained the model with 20 epochs instead, it misclassified 0 times.

Figures 6a and 6b show the classification of my own handwriting. The first attempt was misclassified 8/9 times. The second attempt 3/9 times. This is because I went back and traced over my handwriting again, making the lines thicker.

Figure 7 shows the 5x5 filters that represent the learned features in my network’s first convolutional layer. Figure 8 shows these filters when applied to the first image in the test set. What these filters show is a very low-level representation of the image’s features such as edges.

Figure 9 shows the training error I was getting after running 100 epochs. As usual, we can see that the accuracy increases as the number of epochs increases. However, Figure 10, shows that only alpha, beta, and gamma we’re properly classified. This is likely due to the small dataset that I used for delta and zeta.

When running the number of epochs vs accuracy, I ran with a batch size of 128, filter size of 3, and pool size of 2. I then returned the accuracy after each epoch (for 10 epochs).

When running the batch size vs accuracy, I ran with a filter size of 3, pool size of 2, and 2 epochs just to reduce processing time. I observed that as the batch size increased, the accuracy increased until I reached a batch size of 512. Separately, I noticed that the processing time decreased as the batch size increased.

When running the filter size vs accuracy, I ran with a pool size of 2, 2 epochs, and a batch size of 64. What I observed was that the accuracy decreased as the filter size increased.

V. Reflection

Training the network demonstrated how adjusting epochs, batch sizes, and filter sizes impacts accuracy and efficiency. Increasing epochs improved digit classification, while larger batch sizes boosted accuracy and reduced processing time. Tuning these parameters is crucial for using datasets like handwritten digits or Greek letters for classification.

VI. References & Acknowledgments

https://pytorch.org/tutorials/beginner/basics/intro.html
https://nextjournal.com/gkoehler/pytorch-mnist
https://netron.app/