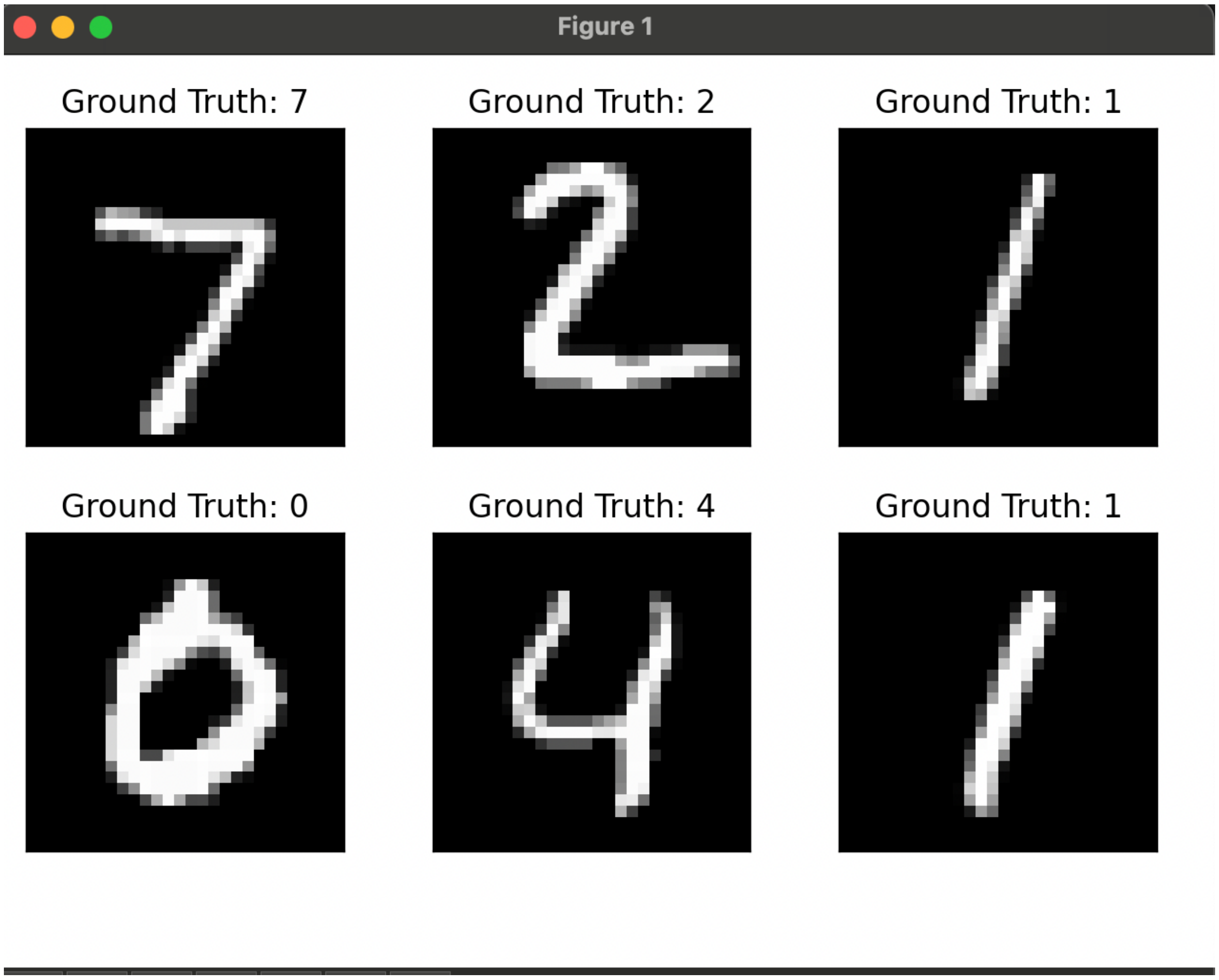
CS 5330: Project 5
Recognition Using Deep Networks

Description:

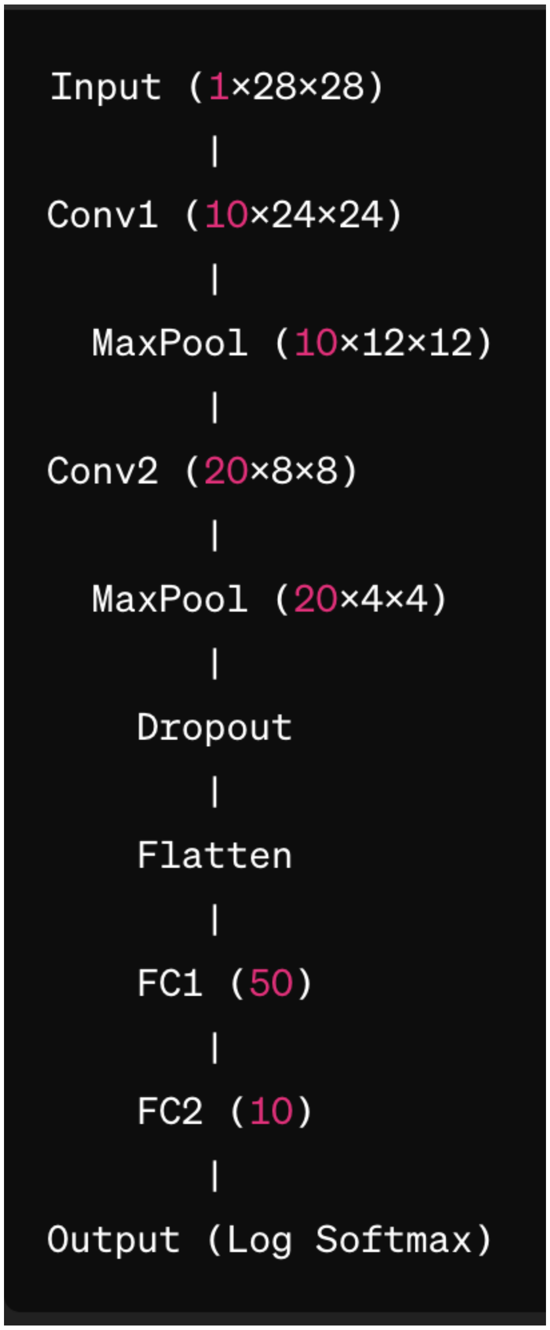
We started with the task of experimenting with deep neural networks for the MNIST dataset, aiming to evaluate the effects of different network architectures on performance and training time. We loaded the MNIST dataset using PyTorch's torchvision module and applied transformations to normalize the data. We defined a custom neural network architecture MyNetwork with configurable parameters such as the number of convolutional layers and the number of hidden nodes in the fully connected layers. We implemented functions to train the network (train_model) and evaluate its performance (evaluate_model) on the test set. These functions also track the training time for each epoch. We defined search parameters for the number of convolution layers, the number of hidden nodes, and the number of epochs. We evaluated different combinations of these parameters and recorded the accuracy and training time for each combination. We plotted the results for accuracy and training time against the number of epochs for each combination of network parameters. This visualization helps in understanding how different network architectures affect model performance and training efficiency.

Results:

First six example digits from the test set:

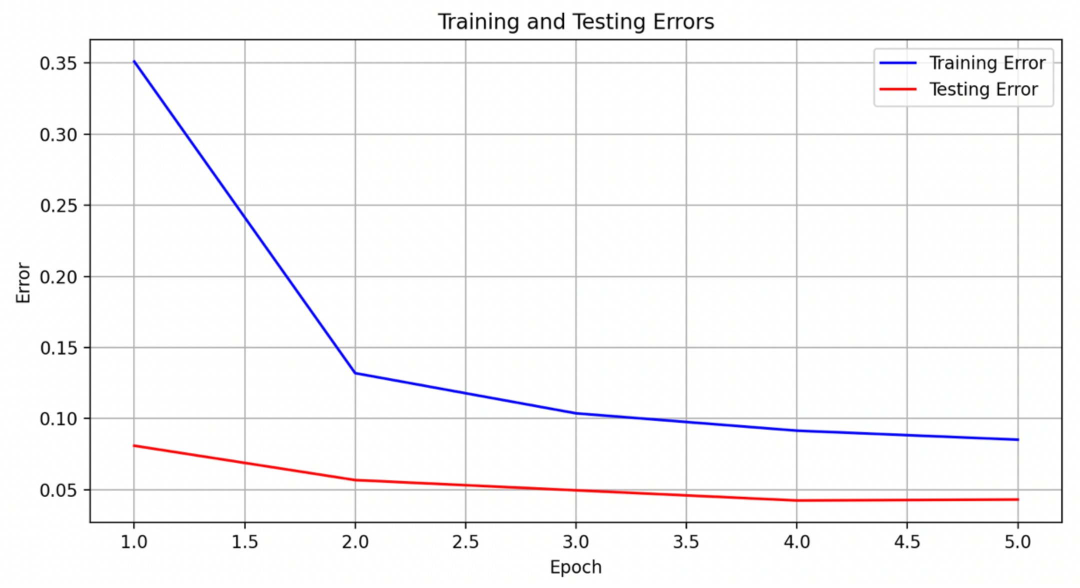


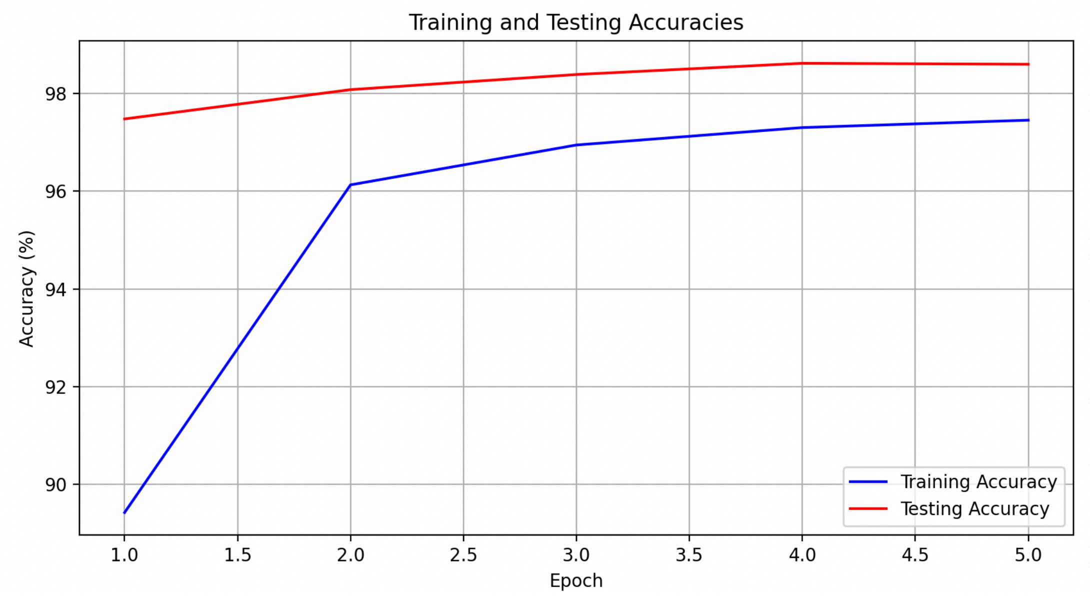
Network Diagram:

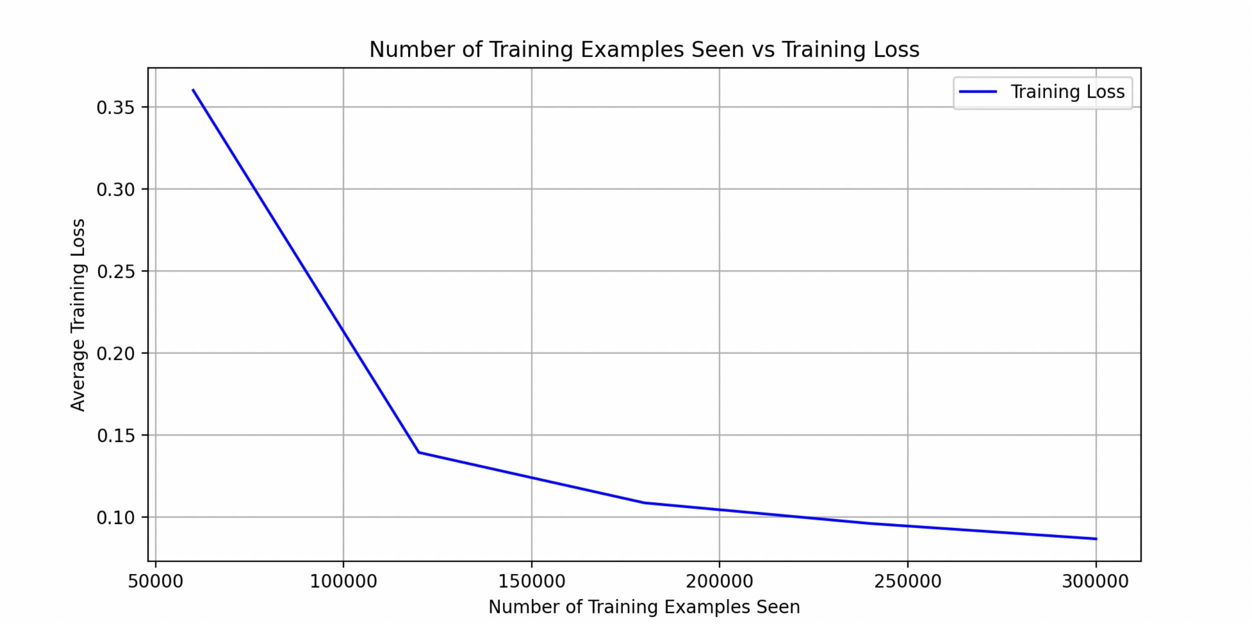


```
Input (1×28×28)
       |
Conv1 (10×24×24)
       |
  MaxPool (10×12×12)
       |
Conv2 (20×8×8)
       |
  MaxPool (20×4×4)
       |
    Dropout
       |
    Flatten
       |
    FC1 (50)
       |
    FC2 (10)
       |
Output (Log Softmax)
```

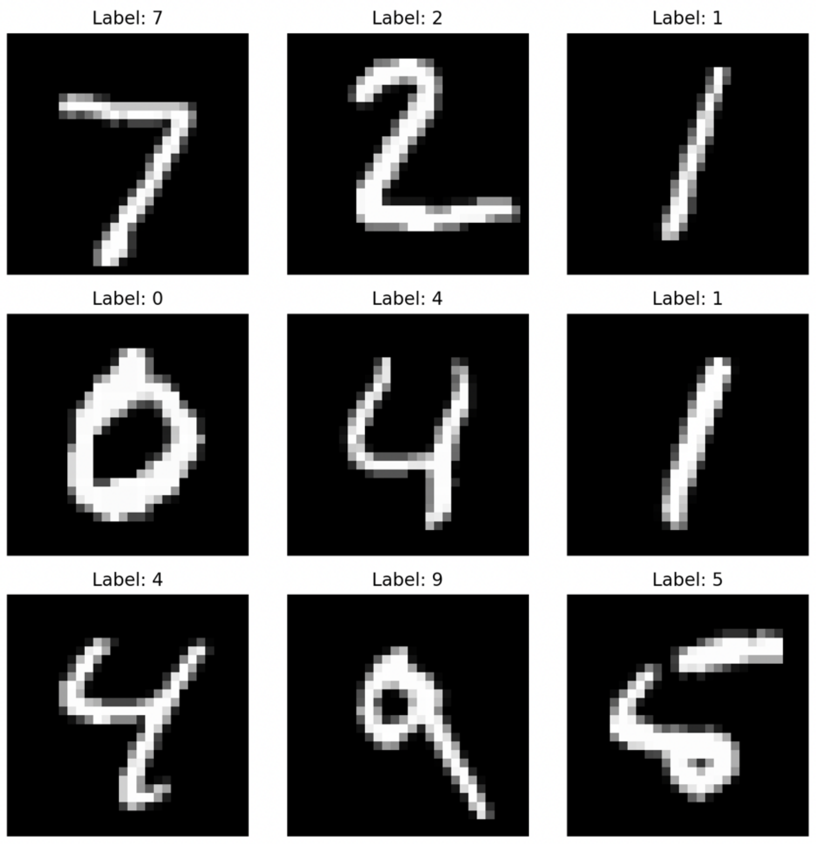
Metric Plots for the model:







Plot of the first 9 digits of the test set:

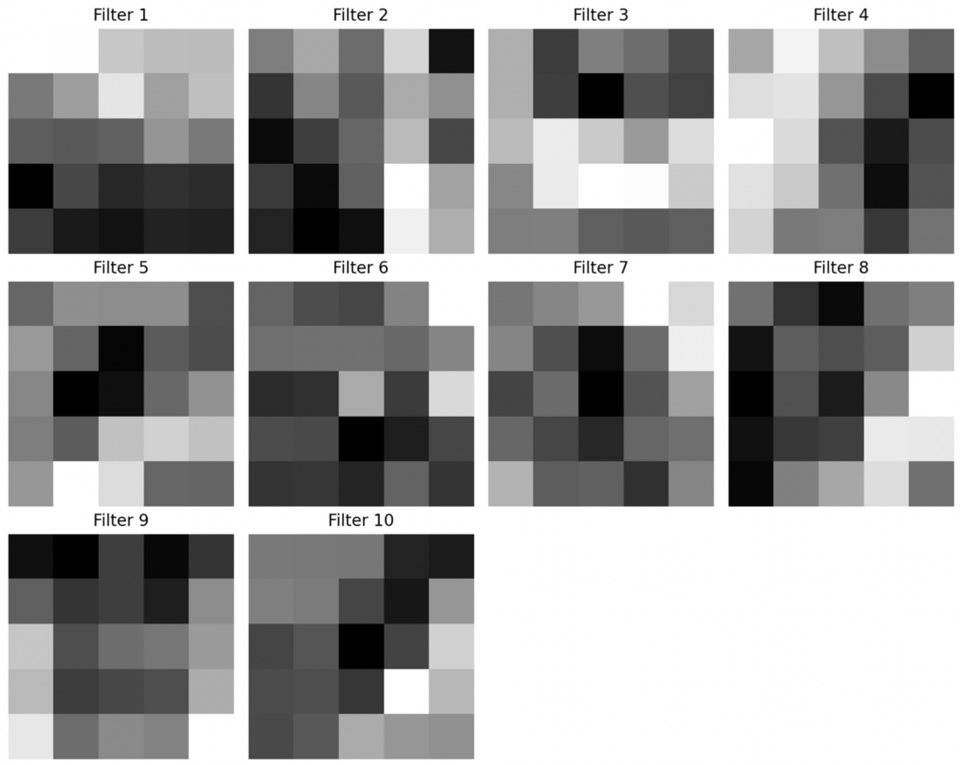


Predictions for the first 10 examples in the test set:

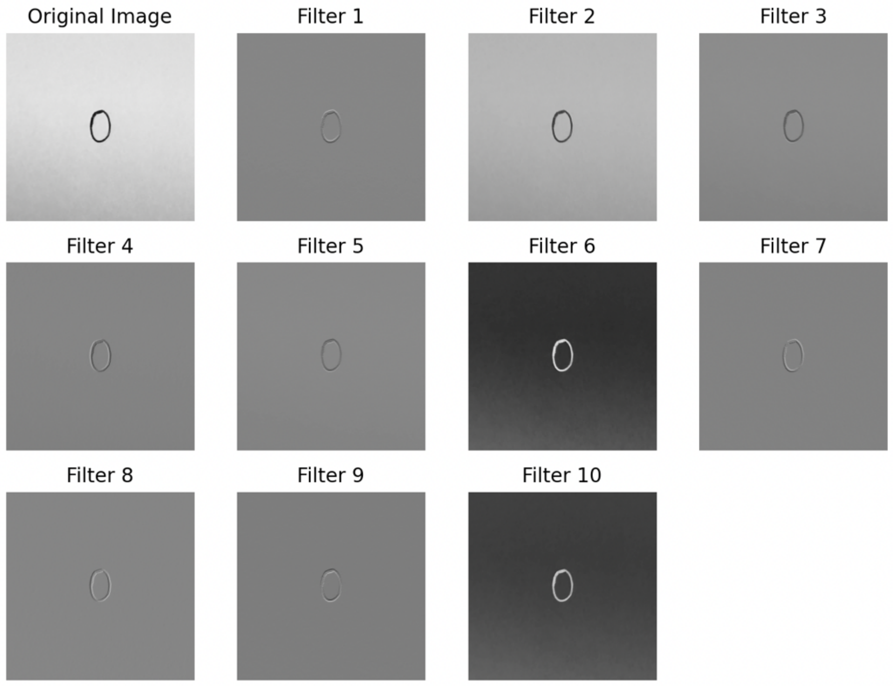
```
Example 1:

Network Output Values: ['0.00', '0.00', '0.00', '0.00', '0.00', '0.00', '0.00', '1.00', '0.00', '0.00']

Predicted Label Index: 7

Correct Label: 7

Example 2:

Network Output Values: ['0.00', '0.00', '1.00', '0.00', '0.00', '0.00', '0.00', '0.00', '0.00', '0.00']

Predicted Label Index: 2

Correct Label: 2

Example 3:

Network Output Values: ['0.00', '1.00', '0.00', '0.00', '0.00', '0.00', '0.00', '0.00', '0.00', '0.00']

Predicted Label Index: 1

Correct Label: 1

Example 4:

Network Output Values: ['1.00', '0.00', '0.00', '0.00', '0.00', '0.00', '0.00', '0.00', '0.00', '0.00']

Predicted Label Index: 0

Correct Label: 0

Example 5:

Network Output Values: ['0.00', '0.00', '0.00', '0.00', '1.00', '0.00', '0.00', '0.00', '0.00', '0.00']

Predicted Label Index: 4

Correct Label: 4

Example 6:

Network Output Values: ['0.00', '1.00', '0.00', '0.00', '0.00', '0.00', '0.00', '0.00', '0.00', '0.00']

Predicted Label Index: 1

Correct Label: 1
```

```
Example 7:

Network Output Values: ['0.00', '0.00', '0.00', '0.00', '1.00', '0.00', '0.00', '0.00', '0.00', '0.00']

Predicted Label Index: 4

Correct Label: 4

Example 8:

Network Output Values: ['0.00', '0.00', '0.00', '0.00', '0.00', '0.00', '0.00', '0.00', '0.00', '1.00']

Predicted Label Index: 9

Correct Label: 9

Example 9:

Network Output Values: ['0.00', '0.00', '0.00', '0.00', '0.00', '0.98', '0.01', '0.00', '0.00', '0.01']

Predicted Label Index: 5

Correct Label: 5

Example 10:

Network Output Values: ['0.00', '0.00', '0.00', '0.00', '0.00', '0.00', '0.00', '0.00', '0.00', '1.00']

Predicted Label Index: 9

Correct Label: 9
```

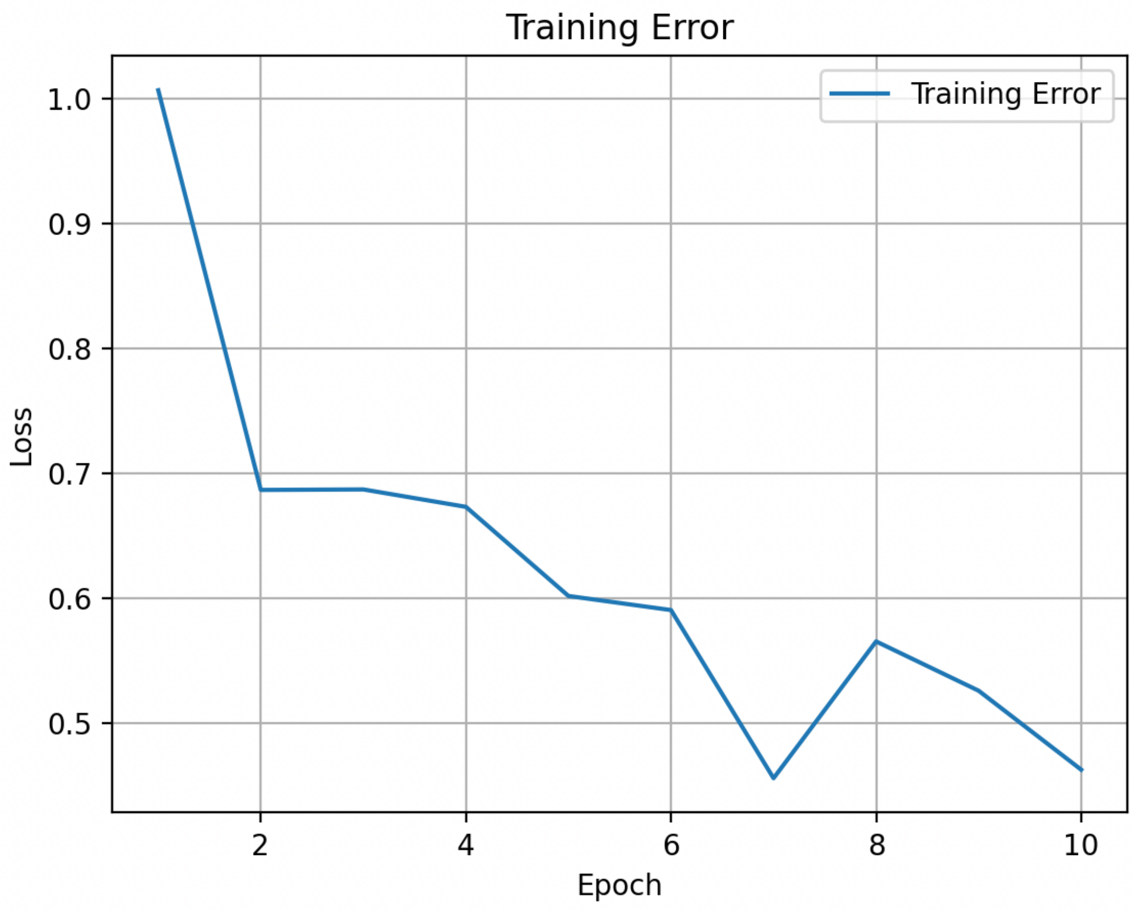
Visualization of 10 different filters:



Effects of filters on the image:



Training error graph for the Greek alphabet recognition model:



Accuracy and Loss data for Greek alphabet recognition:

Epoch 1, Loss: 1.0066594580809276

Epoch 2, Loss: 0.6864620844523112

Epoch 3, Loss: 0.6868575016657511

Epoch 4, Loss: 0.6729160149892172

Epoch 5, Loss: 0.6015137235323588

```
Epoch 6, Loss: 0.5902269780635834
Epoch 7, Loss: 0.4555145353078842
Epoch 8, Loss: 0.565109928448995
Epoch 9, Loss: 0.5256090958913168
Epoch 10, Loss: 0.46235277752081555
Accuracy on Greek dataset: 92.5925925925926%
```

Results obtained after training model on different parameters:

```
Num Conv Layers | Num Hidden Nodes | Num Epochs | Accuracy (%) | Training Time (s)
2          | 50          | 5       | 98.37       | 10.55
2          | 50          | 10       | 98.82       | 10.37
2          | 50          | 15       | 98.91       | 10.40
2          | 100          | 5       | 98.72       | 10.45
2          | 100          | 10       | 98.88       | 10.60
2          | 100          | 15       | 98.94       | 10.63
3          | 50          | 5       | 96.16       | 11.99
3          | 50          | 10       | 96.48       | 11.75
3          | 50          | 15       | 97.77       | 11.91
3          | 100          | 5       | 96.22       | 12.33
3          | 100          | 10       | 96.98       | 11.94
3          | 100          | 15       | 97.49       | 12.50
```

Reflection:

I gained insights into designing custom neural network architectures by defining the number of convolutional layers, hidden nodes, and other parameters. Acquired knowledge on training neural networks using backpropagation and optimizing techniques like Adam optimizer. Additionally, I learned how to evaluate model performance on test data and calculate metrics such as accuracy. Practiced analyzing experimental results through visualization techniques, enabling me to interpret trends and make informed decisions about network architecture selection. Gained experience in designing experiments to systematically explore the effects of different network configurations, fostering a deeper understanding of how architecture choices influence model behavior. Overall, this work provided a hands-on opportunity to apply fundamental concepts in deep learning, experiment with network architectures, and gain practical insights into building and optimizing neural networks for image classification tasks.

Acknowledgement:

I would like to thank Prof. Bruce Maxwell for helping me grasp these concepts with ease. The class lectures and notes were very useful in completing this project.