CS5330 Project 5
Yunyu Guo
March 26 2025

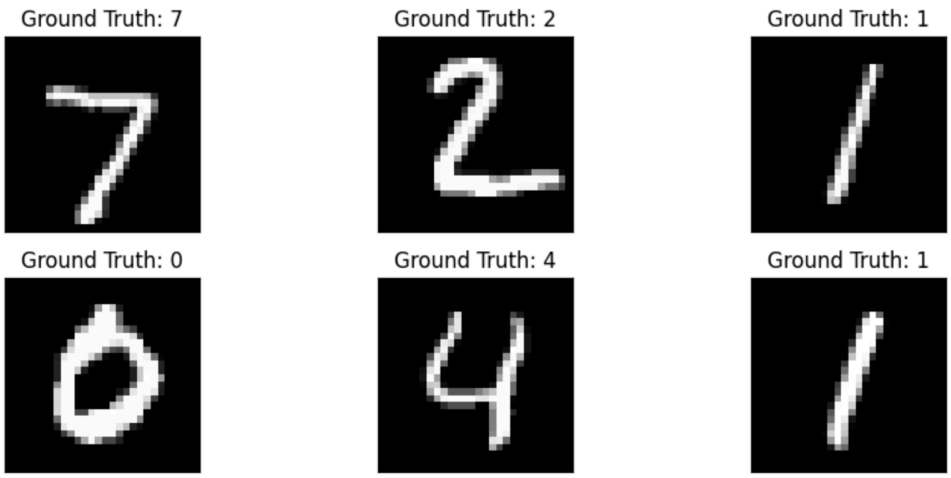
This project is to build, train, analyze, and modify a deep network for a recognition task. I use the MNIST digit recognition data set. All tasks were run on Colab. See colab.

Task 1 Build and train a network to recognize digits

A. Get the MNIST digit data set

Include a plot of the first six example digits from the test set in your report.

sample_digits.png ×



B. Build a network model

Put a diagram of your network in your report. A photo of a hand-drawn network is fine.

1. Input Layer

Takes 28x28 grayscale MNIST images (1 channel)

2. First Convolutional Block

```
self.conv1 = nn.Conv2d(1, 10, kernel_size=5)  # 28x28 -> 24x24 output = (input - kernel_size + 1))
self.relu = nn.ReLU()                    # Activation function
self.pool = nn.MaxPool2d(2)              # 24x24 -> 12x12
```

- Applies 10 5x5 filters, (creates 10 different feature maps)
- ReLU activation
- 2x2 max pooling reduces spatial dimensions,default stride is same as window size,
- output: 12×12 (dimensions halved)

3. Second Convolutional Block

```
self.conv2 = nn.Conv2d(10, 20, kernel_size=5)  # 12x12 -> 8x8
self.dropout = nn.Dropout(0.25)              # Prevents overfitting
self.pool = nn.MaxPool2d(2)                  # 8x8 -> 4x4
```

- Applies 20 5x5 filters
- 25% dropout for regularization
- 2x2 max pooling

4. Flatten Operation

```
x = x.view(-1, 320)  # 320 = 20 channels * 4 * 4
```

transforms the 4D tensor output from convolutional layers into a 2D tensor needed for fully connected layers

- Shape becomes 2D [batch_size, 320]
  -1: Automatically calculates the batch dimension
  320: Flattened features (20 * 4 * 4 = 320)
- Flattens the 4x4x20 feature maps into a 320-dimensional vector

5. Fully Connected Layers

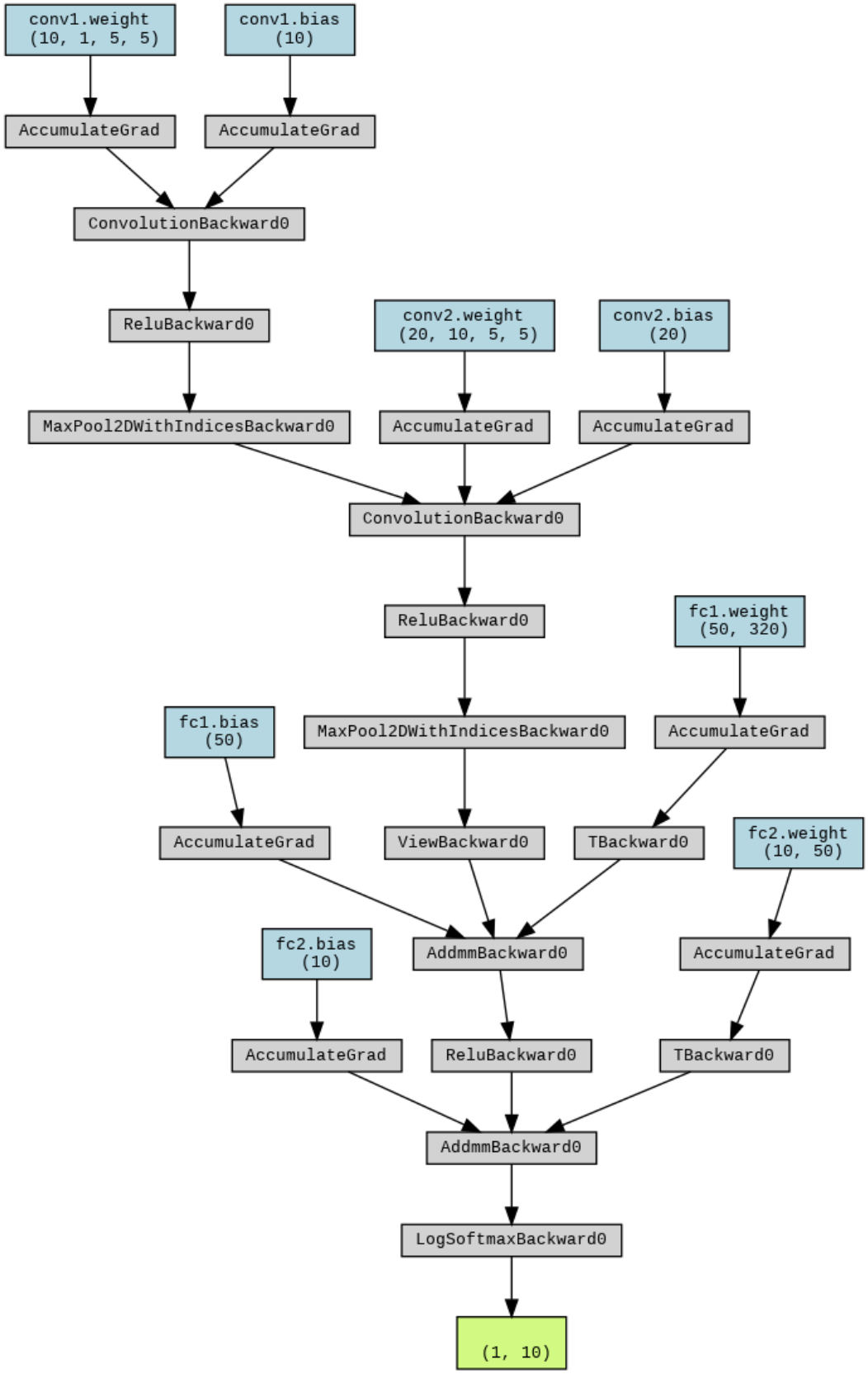
```
self.fc1 = nn.Linear(320, 50)  # First FC layer
self.fc2 = nn.Linear(50, 10)   # Output layer
```

- First FC layer: 320 → 50 nodes with ReLU
- Output layer: 50 → 10 nodes (one per digit)
- Log-softmax activation for classification

Data Flow Dimensions

- Input: 28x28x1
- After conv1: 24x24x10
- After first pooling: 12x12x10
- After conv2: 8x8x20
- After second pooling: 4x4x20
- Flattened: 320
- First FC: 50
- Output: 10 (probability distribution over digits 0-9)

Gemini MyNetwork_architecture.png ×



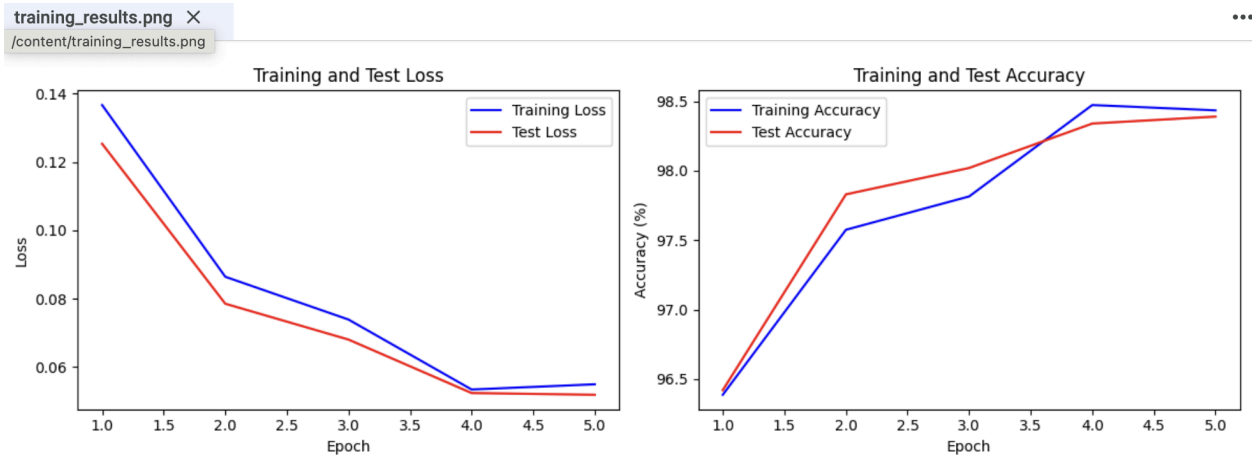
C. Train the model

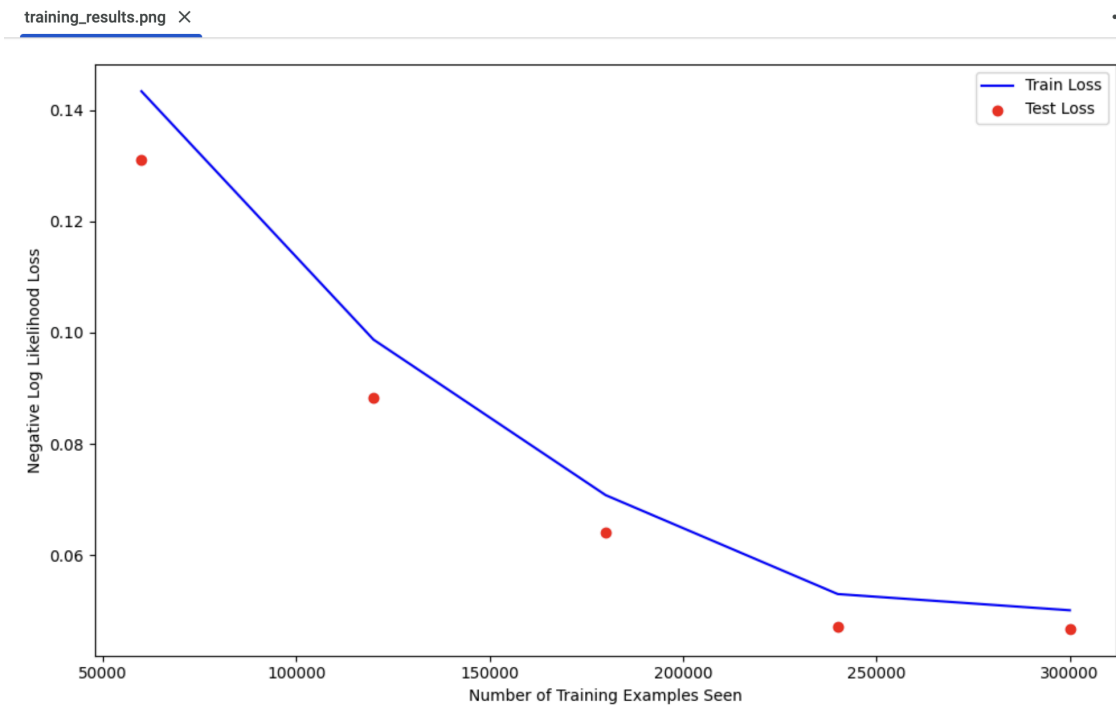
Make a plot of the training and testing error. Collect the accuracy scores and plot the training and testing accuracy in a graph. Try to use different colors. Include this plot in your report along with a discussion of the training behavior.

Training Loop Structure:

```
for batch_idx, (images, labels) in enumerate(train_loader):
  # 1. Zero the parameter gradients
  optimizer.zero_grad()
   # 2. Forward pass
  outputs = model(images)
  loss = criterion(outputs, labels)
   # 3. Backward pass
  loss.backward()
   # 4. Update weights
  optimizer.step()
```

```
Using cuda device
Starting training...
Epoch [1/5]:
Training Loss: 0.137, Training Accuracy: 96.39%
Test Loss: 0.125, Test Accuracy: 96.42%
Epoch [2/5]:
Training Loss: 0.086, Training Accuracy: 97.58%
Test Loss: 0.079, Test Accuracy: 97.83%
Epoch [3/5]:
Training Loss: 0.074, Training Accuracy: 97.81%
Test Loss: 0.068, Test Accuracy: 98.02%
Epoch [4/5]:
Training Loss: 0.053, Training Accuracy: 98.47%
Test Loss: 0.052, Test Accuracy: 98.34%
Epoch [5/5]:
Training Loss: 0.055, Training Accuracy: 98.44%
Test Loss: 0.052, Test Accuracy: 98.39%
Finished Training
Accuracy on test set: 98.39%
```





Training and Test Loss:

1. Decreasing Loss:

Both training and test losses decrease steadily over the epochs. This indicates that the model is learning and improving its predictions.

2. Convergence:

By the 4th or 5th epoch, the loss values for both training and test sets stabilize. This suggests the model has reached a point where further training may not significantly improve performance.

3. No Overfitting:

The training and test losses are close to each other throughout the training process. This indicates the model generalizes well and is not overfitting to the training data.

Training and Test Accuracy:

1. Increasing Accuracy:

Both training and test accuracies improve steadily over the epochs. This shows the model is becoming better at correctly classifying digits.

2. High Accuracy:

By the 5th epoch, the test accuracy is 98.39%, and the training accuracy is 98.44%. This is a strong performance for the MNIST dataset.

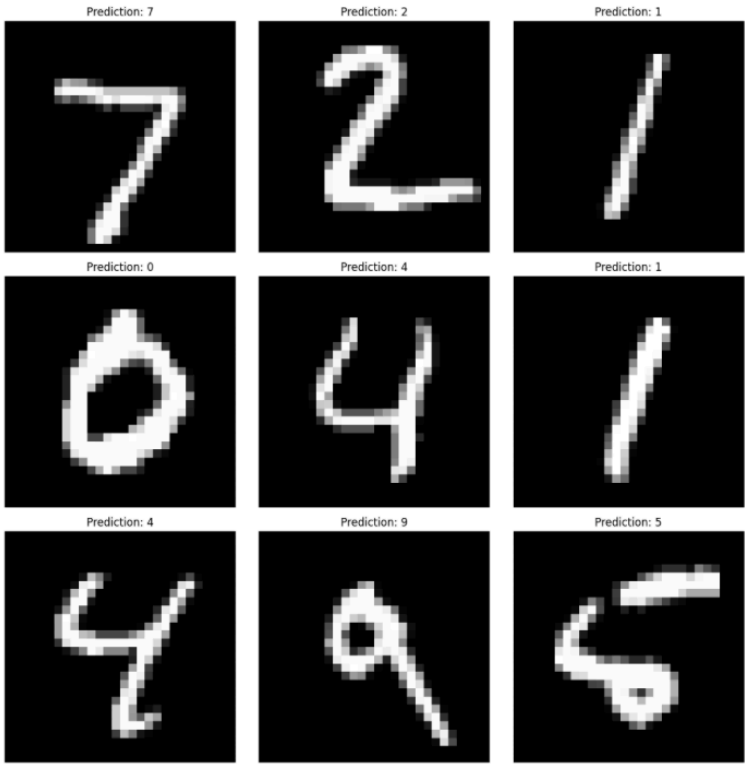
3. No Significant Gap:

The training and test accuracies are very close, indicating the model is not overfitting and performs well on unseen data.

E. Read the network and run it on the test set

Include a table (or screen shot) of your printed values and the plot of the first 9 digits of the test set in your report. Discuss the results.

test_predictions.png




```
Sample 1:
Output values: 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 1.00, 0.00, 0.00
Predicted: 7, Actual: 7
Correct

Sample 2:
Output values: 0.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00
Predicted: 2, Actual: 2
Correct

Sample 3:
Output values: 0.00, 0.99, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00
Predicted: 1, Actual: 1
Correct

Sample 4:
Output values: 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00
Predicted: 0, Actual: 0
Correct

Sample 5:
Output values: 0.00, 0.00, 0.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00
Predicted: 4, Actual: 4
Correct

Sample 6:
Output values: 0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00
Predicted: 1, Actual: 1
Correct

Sample 7:
Output values: 0.00, 0.00, 0.00, 0.00, 0.99, 0.00, 0.00, 0.00, 0.01, 0.00
Predicted: 4, Actual: 4
Correct

Sample 8:
Output values: 0.00, 0.00, 0.00, 0.02, 0.01, 0.00, 0.00, 0.00, 0.00, 0.96
Predicted: 9, Actual: 9
Correct

Sample 9:
Output values: 0.00, 0.00, 0.00, 0.00, 0.00, 0.98, 0.02, 0.00, 0.00, 0.00
Predicted: 5, Actual: 5
Correct

Sample 10:
Output values: 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.01, 0.00, 0.99
Predicted: 9, Actual: 9
Correct
```

The model correctly predicts all 10 samples.

F. Test the network on new inputs

Display how well the network performed on this new input in your report by showing the digits and their classified result. Discuss the results.

Most images are correctly predicted except digit 1, which does look like 7 and thus the model incorrectly predicts as 7.

Task 2 Examine your network

A. Analyze the first layer

Describe how you would interpret the filters at a high level. Do they look (roughly) like any of the standard filters?

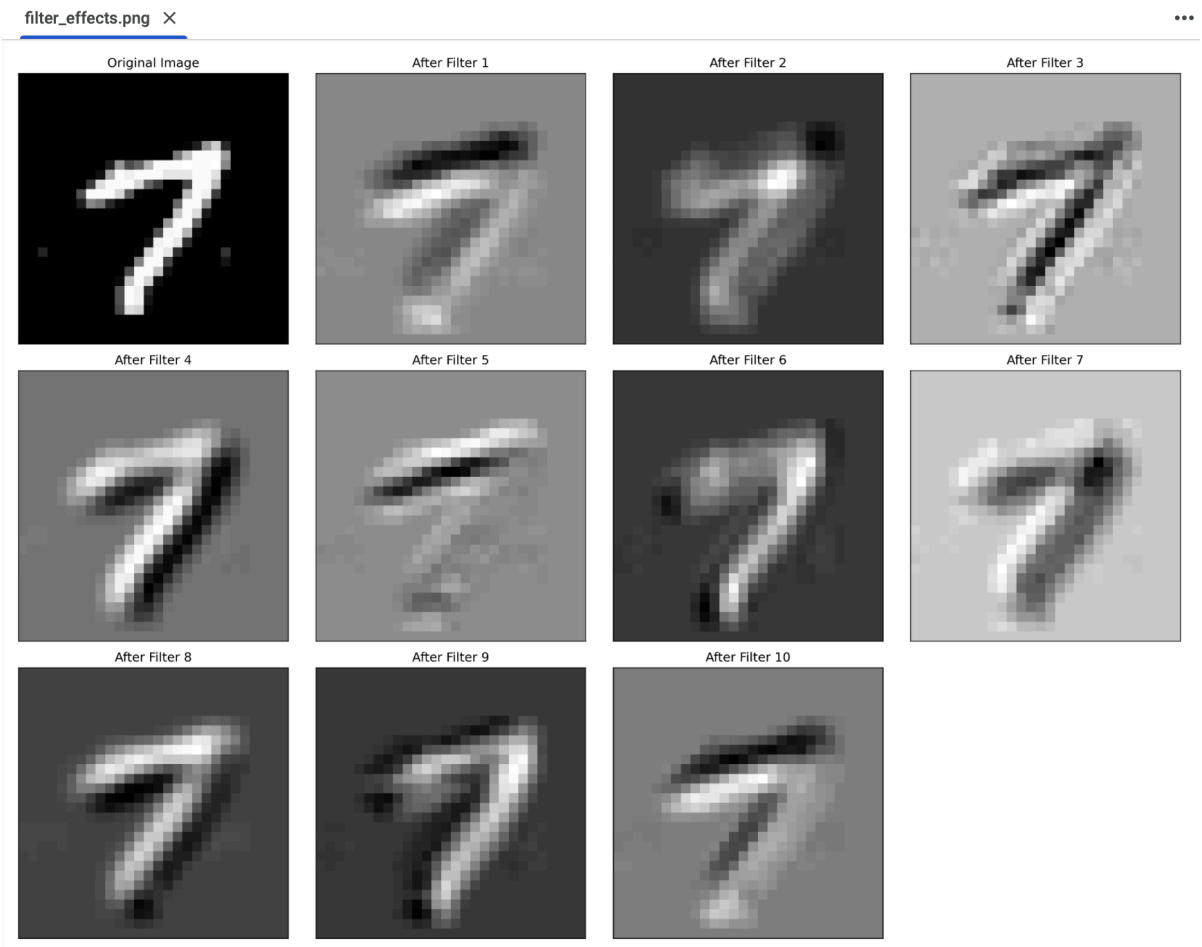
```
Network Structure:
MyNetwork(
  (conv1): Conv2d(1, 10, kernel_size=(5, 5), stride=(1, 1))
  (conv2): Conv2d(10, 20, kernel_size=(5, 5), stride=(1, 1))
  (dropout): Dropout(p=0.25, inplace=False)
  (pool): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (relu): ReLU()
  (fc1): Linear(in_features=320, out_features=50, bias=True)
  (fc2): Linear(in_features=50, out_features=10, bias=True)
)

First Layer Filter Shape: torch.Size([10, 1, 5, 5])

Filter Weights:

Filter 1:
[[-0.21304762 -0.24848029  0.09969722 -0.24896954  0.02483277]
 [ 0.16671564 -0.1414424   0.08002495 -0.03472392 -0.1502496 ]
 [ 0.38464788  0.06302445 -0.14176835 -0.22165877 -0.23069878]
 [ 0.04414254  0.05589436 -0.08290511  0.23051183  0.03202407]
 [ 0.28348127  0.2651888   0.40013027  0.3699833   0.42194027]]

Filter 2:
[[ 0.02578492  0.05089645 -0.15583207 -0.193287   -0.05961045]
 [-0.21908656  0.02799042 -0.24464442 -0.02972394 -0.11709998]
 [-0.03408781  0.03894707  0.03307539  0.02951729  0.3063681 ]
 [ 0.0350588   0.15464836  0.2514594   0.4104548   0.18243273]
 [ 0.22485183  0.22122589  0.11269278  0.0585833   0.0377887 ]]

Filter 3:
[[-0.02959627 -0.0984365  -0.0038854  -0.00346192  0.02771828]
 [ 0.26099697  0.1999997   0.27611694  0.260386    0.03640723]
 [ 0.02005132  0.25546712 -0.08004229 -0.00350432  0.00794827]
 [-0.14550772  0.04331685 -0.24910517  0.12486587 -0.07967188]
 [-0.19549689 -0.20323214 -0.03664631 -0.05249602 -0.05783603]]

Filter 4:
[[ 0.17335324  0.19344881  0.35513026  0.13764088 -0.24651249]
 [-0.2048651   0.04800268  0.41194427  0.25217578  0.00113184]
 [-0.14052944 -0.02406077  0.24816851  0.3560479  -0.1446131 ]
 [-0.21686971  0.24570404  0.2412739   0.04911781  0.10059284]
 [ 0.05540875  0.00655156  0.2679547   0.1790004   0.13298503]]
```

```
Filter 5:
[[-0.11163786 -0.050149    0.21306305  0.2595042   0.18360773]
 [-0.20555452 -0.2740096  -0.28102812 -0.13974003 -0.17829087]
 [ 0.1251172  -0.08638004 -0.3256074  -0.21072894 -0.22411497]
 [ 0.08624692  0.03629564  0.1116401  -0.04567947 -0.22765404]
 [ 0.3141452   0.1859108   0.31192532  0.14290018  0.25932893]]

Filter 6:
[[ 0.09890559  0.23182353  0.2883726  -0.04906081 -0.07583857]
 [ 0.13970433  0.20501679  0.09365705 -0.12571284 -0.3648219 ]
 [ 0.11516582  0.46301618  0.10985679 -0.22920851 -0.21009411]
 [ 0.22345224  0.3491329   0.01140848 -0.2830585  -0.20191893]
 [ 0.14085035  0.17084171  0.16734059 -0.04582665 -0.07948179]]

Filter 7:
[[-0.16877767  0.04168715 -0.27036792 -0.16435693 -0.04069125]
 [-0.2211138  -0.179613   -0.14846274 -0.15049896 -0.20829114]
 [-0.05727714 -0.12284907 -0.06249039 -0.22156075 -0.05675098]
 [ 0.08654697 -0.0818273  -0.02126748  0.04803213  0.2812653 ]
 [ 0.08479092  0.05264935 -0.05851885  0.23255125  0.26636615]]

Filter 8:
[[ 0.26903588 -0.16811366 -0.19525702  0.06434883 -0.2017284 ]
 [ 0.19702873  0.02139795 -0.06260137 -0.18958086 -0.07623367]
 [-0.13630338 -0.23057677 -0.17124082 -0.0463089   0.02408122]
 [-0.03304772  0.1051529  -0.2079997   0.14238453 -0.0074807 ]
 [-0.10964666  0.02045121 -0.1253862  -0.13959211  0.12370533]]

Filter 9:
[[ 0.264171    0.06587859  0.30820987  0.1377422   0.21543722]
 [ 0.32707706  0.22536117  0.16873908  0.09136549 -0.07300276]
 [-0.05623718 -0.06743828  0.00802987 -0.06046558 -0.15049545]
 [-0.18367241  0.03278014 -0.2396404  -0.09273607  0.0371022 ]
 [-0.1941125  -0.24271698 -0.13080344 -0.16050747 -0.00791843]]

Filter 10:
[[-0.07434348  0.2851969   0.34231967  0.00821447 -0.27385393]
 [ 0.04614712  0.25706866  0.34008864  0.03056925 -0.12852682]
 [-0.08261228  0.01110479  0.01842541  0.05640816 -0.05311442]
 [-0.20609006 -0.22129975 -0.23776749 -0.00409969  0.27449   ]
 [-0.09749012  0.08621728  0.02068515 -0.10651664 -0.06057091]]
```

- The colors indicate weights (dark regions = low values, bright regions = high values)
- Some filters (filter 6 and 9) have sharp contrasts between dark and bright regions, which may detect edges like Sobel-like Filters
- Some (filter 3) appear smoother, which could help detect textures or remove noise like Gaussian filter
- Filters 7 and 8 have striped or wavy patterns, they might be detecting orientations like Gabor Filter

B. Show the effect of the filters

In your report, include the plot and discuss whether the results make sense given the filters. How do the results match your interpretation of the filters?



3. Transfer Learning on Greek Letters

How many epochs does it take using the 27 examples in order to reasonably identify them?

```
Starting training...
Epoch [1/20], Loss: 1.1674, Accuracy: 48.15%
Epoch [2/20], Loss: 0.7010, Accuracy: 70.37%
Epoch [3/20], Loss: 0.2387, Accuracy: 96.30%
Epoch [4/20], Loss: 0.1281, Accuracy: 96.30%
Epoch [5/20], Loss: 0.0918, Accuracy: 96.30%
Epoch [6/20], Loss: 0.0903, Accuracy: 96.30%
High accuracy achieved at epoch 6. Early stopping.
Testing network on Greek letters...
Overall accuracy: 100.00%
Accuracy for alpha: 100.00%
Accuracy for beta: 100.00%
Accuracy for gamma: 100.00%
Model saved as 'greek_model.pth'
Training completed in 6 epochs
Final accuracy: 96.30%
```

Take a picture of several of examples of your own alpha, beta, and gamma symbols. Crop and rescale them to about 128x128 images to match the existing data and see if they are correctly classified by the trained network.

Test on 6 written images, 2 images for each letter.

The transfer learning on Greek letters model correctly identifies 3 written Greek letters out of 6.

4. Design your own experiment

The experiment was run on **Colab**. Dataset: MNIST

A. Develop a plan and predict the results

1. Convolutional Filters: Test different numbers of filters in first and second convolutional layers.
   First layer filters: [8, 16, 24, 32]. Starting from minimal (8) to substantial (32) to test if basic feature detection benefits from more filters
   Second layer filters: [16, 32, 48, 64]. Testing whether higher feature complexity (more filters) improves accuracy.
2. Hidden Units: Vary the size of the fully connected layer.

Hidden units: [30, 50, 70, 100]. Small FC Layer (30 units) may be faster training, less likely to overfit, and more efficient. It may lack capacity to represent complex decision boundaries. Medium FC Layer (50 units): balances capacity and efficiency.
Large FC Layer (70 units): Increased capacity to represent more complex patterns. May capture more subtle relationships between features. Very Large FC Layer (100 units): Maximum representational capacity. More prone to overfitting, slower training.

3. Dropout Configuration: Test different dropout rates
   Dropout rates: [0.1, 0.25, 0.4].
   0.1 (Light dropout): Only 10% of neurons are dropped during each training pass. Minimal regularization effect. Preserves most information flow
   0.25 (Moderate dropout): Standard dropout rate. Balances regularization and information preservation.
   0.4 (Heavy dropout): Aggressive regularization that removes 40% of neurons.
   Strongly encourages feature redundancy. May slow learning but potentially leads to better generalization

```
baseline_config = {
    "conv1_filters": 16,
    "conv2_filters": 32,
    "fc1_units": 50,
    "dropout_rate": 0.25,
    "fc_dropout": False
  }

# Parameter ranges to explore
  param_ranges = {
    "conv1_filters": [8, 16, 32],      # Remove 24
    "conv2_filters": [16, 48, 64],     # Remove 32
    "fc1_units": [30, 70],             # Remove 50, 100
    "dropout_rate": [0.1, 0.4],        # Remove 0.25
    "fc_dropout": [False]              # Only test without FC dropout
  }
```

```
Testing baseline configuration...
Epoch 1: Loss=1.8568, Accuracy=77.84%
Epoch 2: Loss=0.5490, Accuracy=88.75%
Epoch 3: Loss=0.3499, Accuracy=92.23%

Optimizing conv1_filters...
Epoch 1: Loss=1.5504, Accuracy=82.72%
Epoch 2: Loss=0.4629, Accuracy=87.91%
Epoch 3: Loss=0.3243, Accuracy=92.11%
Epoch 1: Loss=1.4196, Accuracy=81.09%
Epoch 2: Loss=0.4083, Accuracy=90.38%
Epoch 3: Loss=0.2879, Accuracy=92.59%
Best value for conv1_filters: 32

    Optimizing conv2_filters...
    Epoch 1: Loss=1.3052, Accuracy=86.84%
    Epoch 2: Loss=0.3989, Accuracy=91.24%
    Epoch 3: Loss=0.2802, Accuracy=93.28%
    Epoch 1: Loss=1.3091, Accuracy=88.56%
    Epoch 2: Loss=0.3688, Accuracy=91.88%
    Epoch 3: Loss=0.2682, Accuracy=93.36%
    Epoch 1: Loss=1.3831, Accuracy=85.03%
    Epoch 2: Loss=0.4237, Accuracy=90.46%
    Epoch 3: Loss=0.2931, Accuracy=90.75%
    Best value for conv2_filters: 48

    Optimizing fc1_units...
    Epoch 1: Loss=1.2204, Accuracy=88.44%
    Epoch 2: Loss=0.3801, Accuracy=92.64%
    Epoch 3: Loss=0.2764, Accuracy=94.14%
    Epoch 1: Loss=1.2806, Accuracy=85.02%
    Epoch 2: Loss=0.4056, Accuracy=90.78%
    Epoch 3: Loss=0.2855, Accuracy=92.67%
    Best value for fc1_units: 30


    Optimizing dropout_rate...
    Epoch 1: Loss=1.6771, Accuracy=77.41%
    Epoch 2: Loss=0.4665, Accuracy=90.14%
    Epoch 3: Loss=0.3053, Accuracy=92.28%
    Epoch 1: Loss=1.1661, Accuracy=88.16%
    Epoch 2: Loss=0.3775, Accuracy=92.67%
    Epoch 3: Loss=0.2606, Accuracy=94.09%
    Best value for dropout_rate: 0.4

    Optimizing fc_dropout...
    Best value for fc_dropout: False
```

Testing random combinations:

```
Top 5 Configurations:
1. Accuracy: 94.40%
   Conv1 filters: 32
   Conv2 filters: 64
   FC units: 70
   Dropout rate: 0.4
   FC dropout: False
   Training time: 64.74 seconds

2. Accuracy: 94.14%
   Conv1 filters: 32
   Conv2 filters: 48
   FC units: 30
   Dropout rate: 0.25
   FC dropout: False
   Training time: 60.65 seconds

3. Accuracy: 94.09%
   Conv1 filters: 32
   Conv2 filters: 48
   FC units: 30
   Dropout rate: 0.4
   FC dropout: False
   Training time: 59.88 seconds

4. Accuracy: 93.36%
   Conv1 filters: 32
   Conv2 filters: 48
   FC units: 50
   Dropout rate: 0.25
   FC dropout: False
   Training time: 60.52 seconds

5. Accuracy: 93.28%
   Conv1 filters: 32
   Conv2 filters: 16
   FC units: 50
   Dropout rate: 0.25
   FC dropout: False
   Training time: 49.09 seconds
```