Project 5 Recognition using Deep Networks

Name: Wenqing Fan, Hang Yin

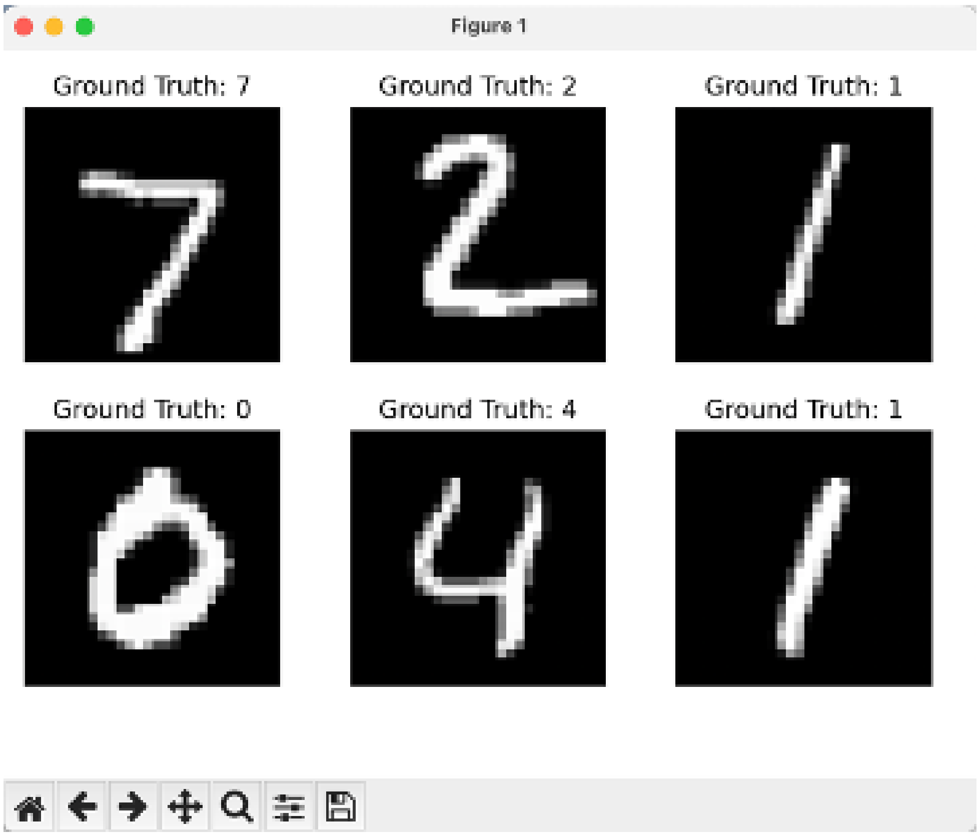
1. short description of the overall project.

In this project, we first learned the pyTorch tutorial using the MNIST data set. Then we used our own customized data to test the network. We also visualized the weights of the first layer and understood the effect it took. Besides, we learned how to load a pre-trained network and apply it to a different dataset. Finally, we run some experiments to check how to evaluate different dimensions of the network.

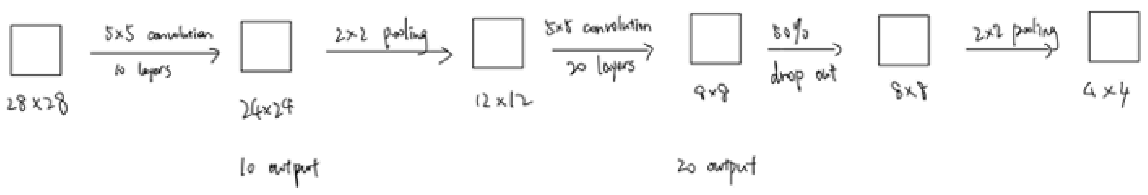
2. required images along with a short description of the meaning of the image.

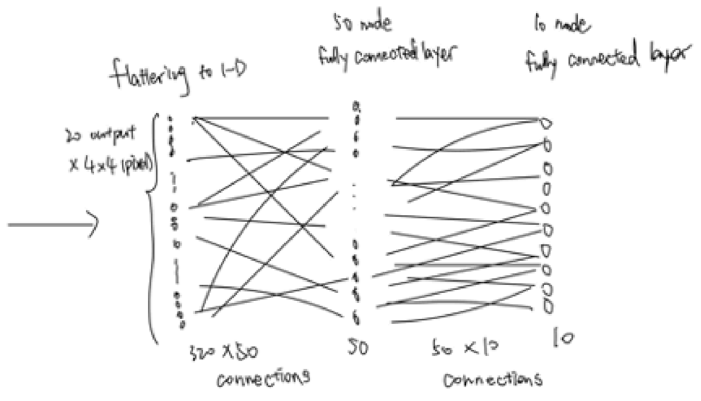
Task 1: Build and train a network to recognize digits

A: **Get the MNIST digit data set**

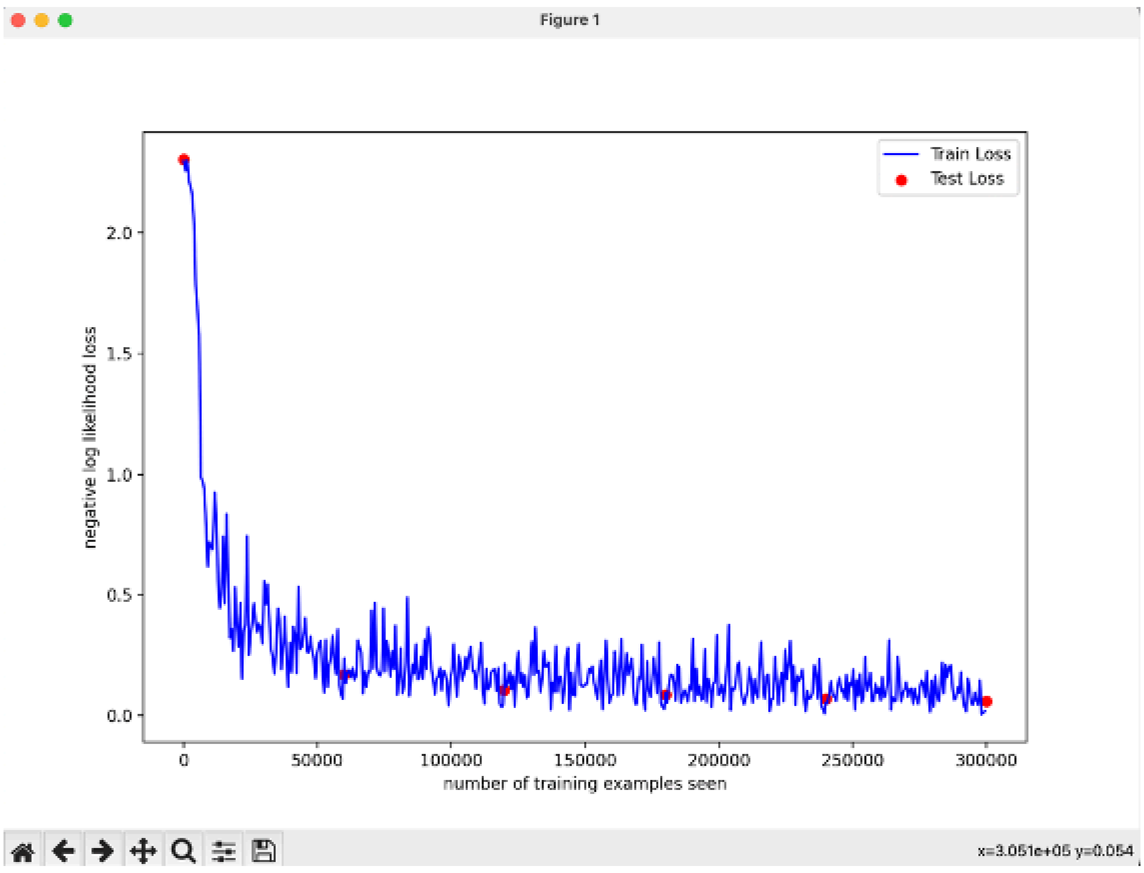


B: Build a network model

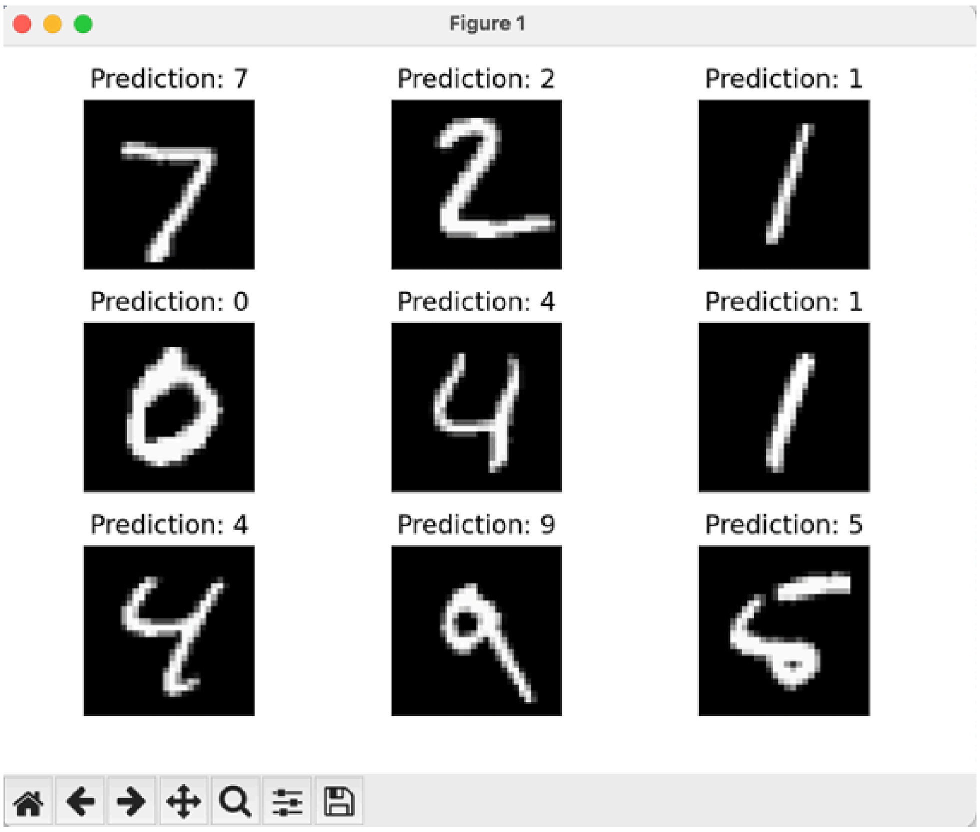




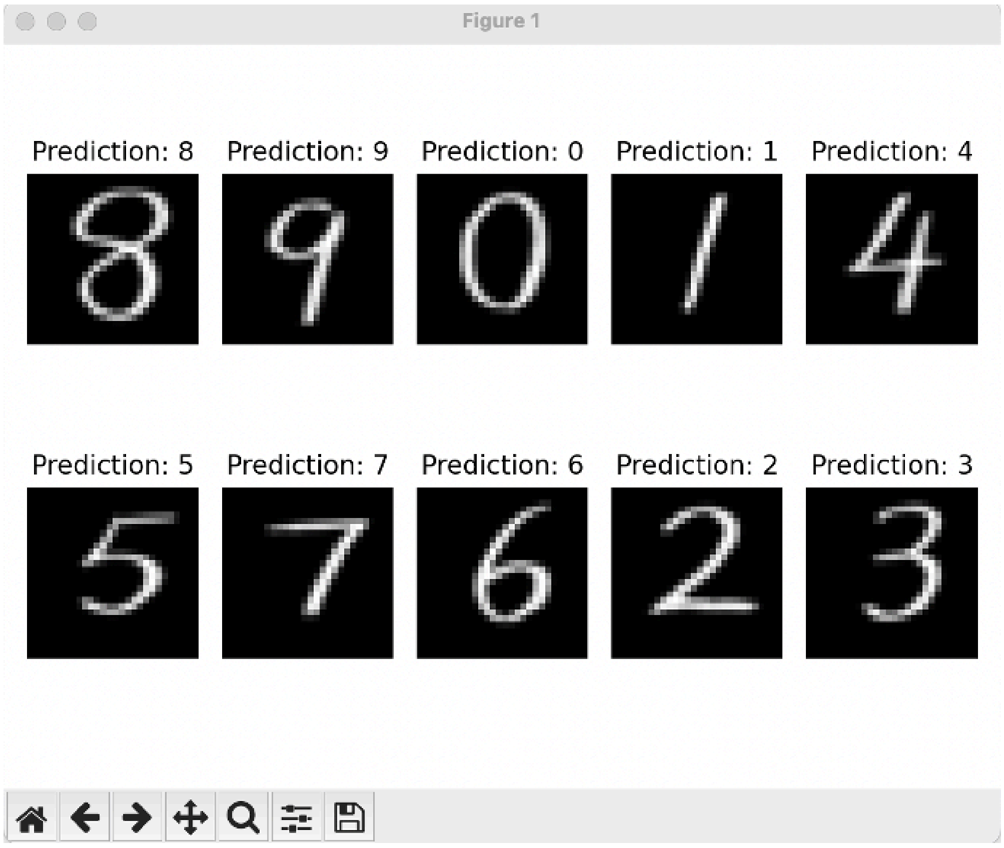
C: Train the model



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

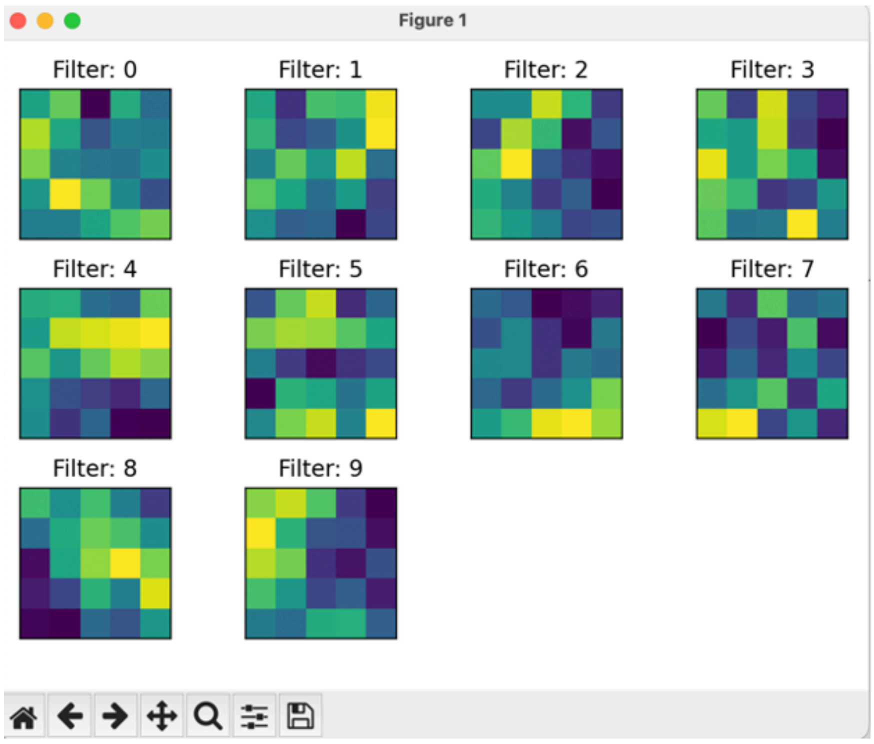


F: Test the network on new inputs



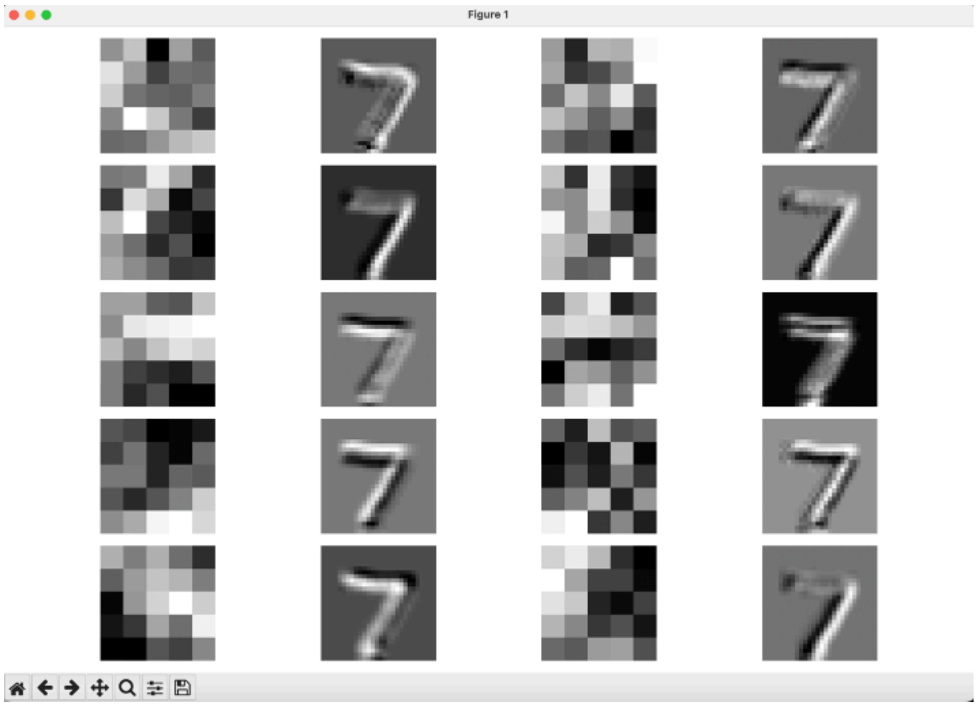
Task 2: Examine your network

A: **Analyze the first layer**



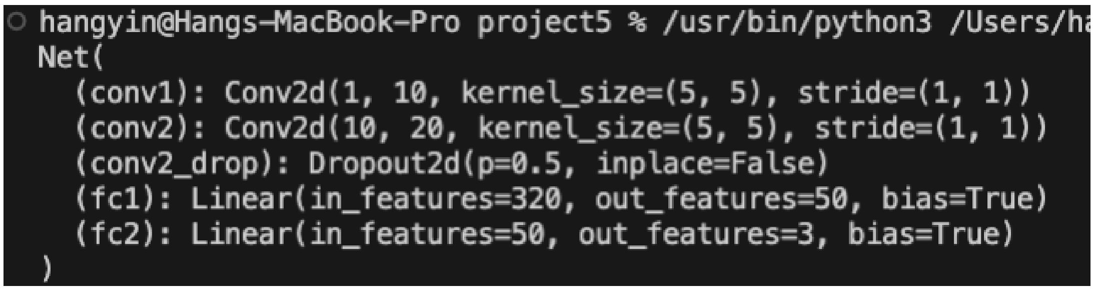
B: **Show the effect of the filters**

The filters make sense, for example, the top left filter has darker color to the right top corner, and lighter color to the left bottom corner. That is why the 7 in the effect has white border on the top of the number.



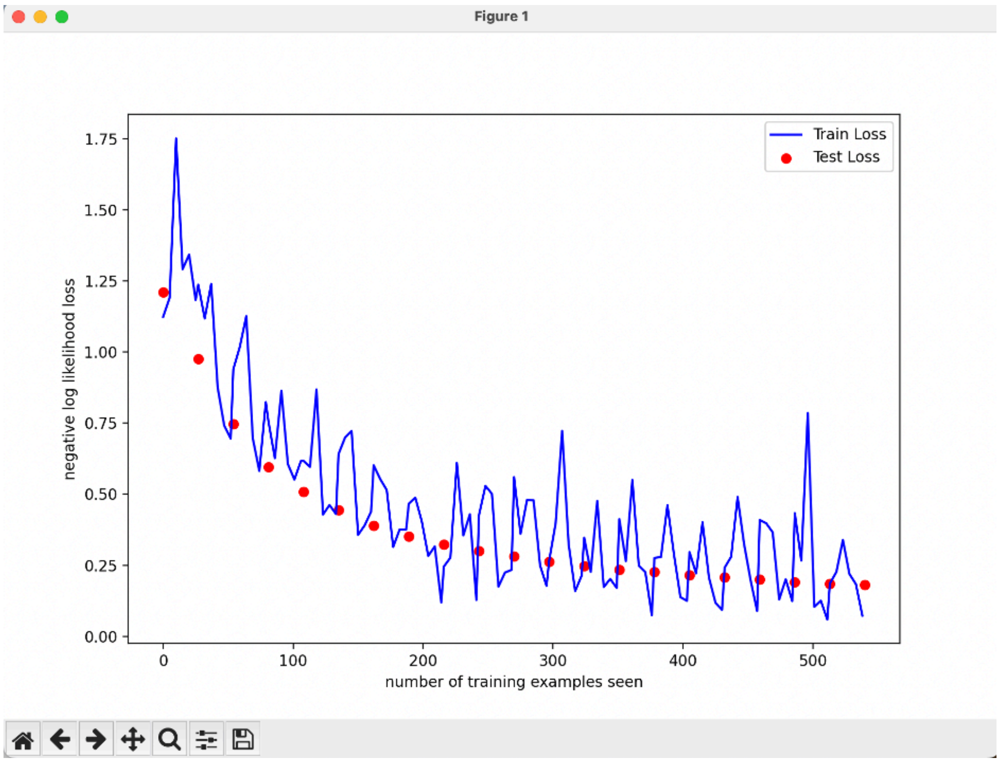
Task 3: Transfer Learning on Greek Letters

A print of the current network based on task one but changed the last layer to 3 nodes.

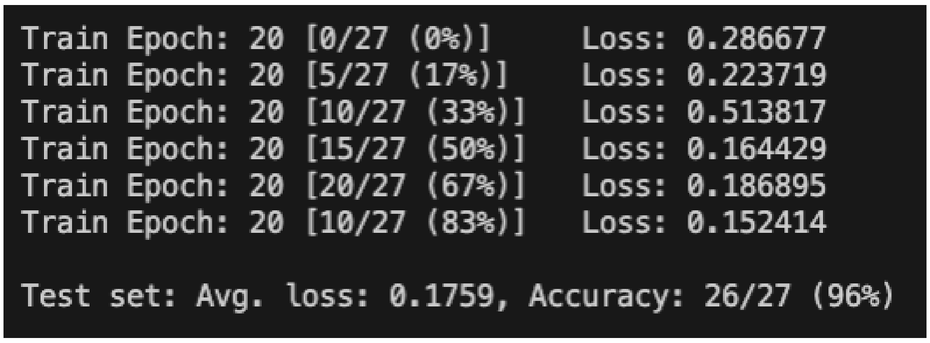


```
○ hangyin@Hangs-MacBook-Pro project5 % /usr/bin/python3 /Users/ha
  Net(
    (conv1): Conv2d(1, 10, kernel_size=(5, 5), stride=(1, 1))
    (conv2): Conv2d(10, 20, kernel_size=(5, 5), stride=(1, 1))
    (conv2_drop): Dropout2d(p=0.5, inplace=False)
    (fc1): Linear(in_features=320, out_features=50, bias=True)
    (fc2): Linear(in_features=50, out_features=3, bias=True)
  )
```

A plot of the train loss and test loss



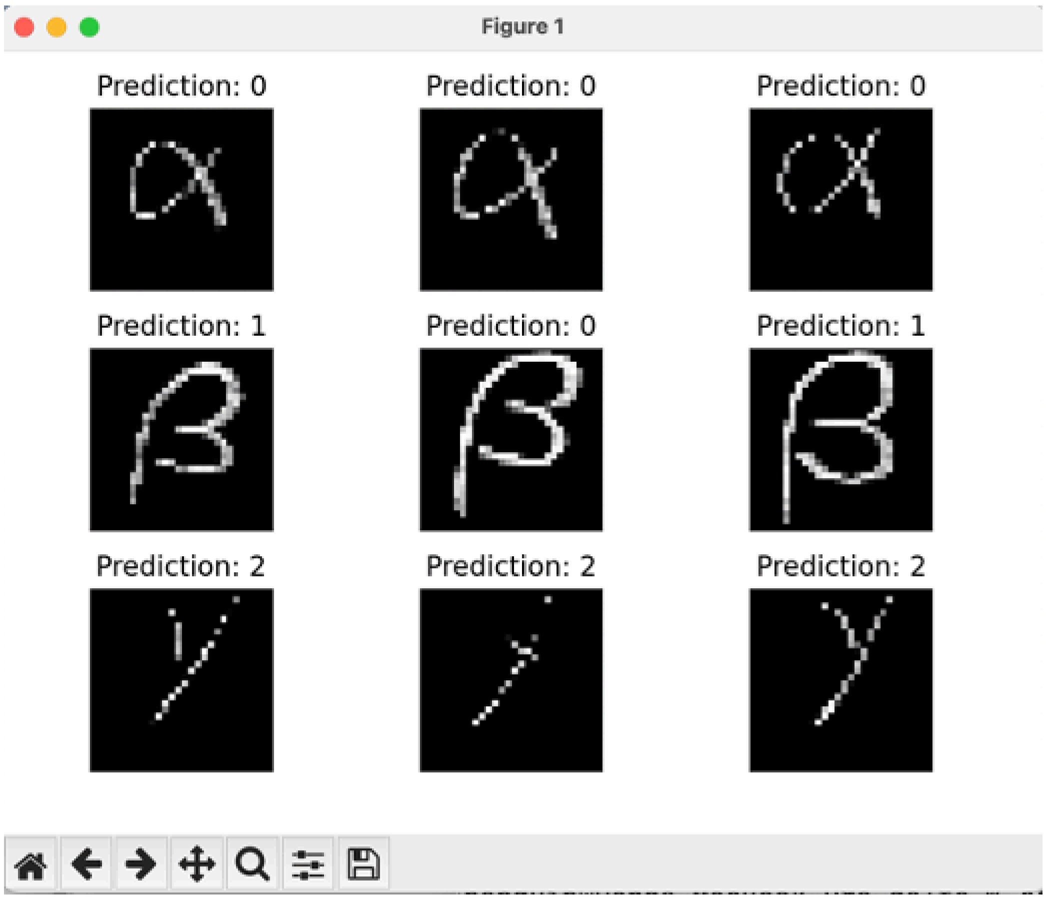
Running result of the accuracy. It takes about 20 epochs to train it to almost perfectly identify them. The final accuracy we can get is 96%.



```
Train Epoch: 20 [0/27 (0%)]      Loss: 0.286677
Train Epoch: 20 [5/27 (17%)]     Loss: 0.223719
Train Epoch: 20 [10/27 (33%)]    Loss: 0.513817
Train Epoch: 20 [15/27 (50%)]    Loss: 0.164429
Train Epoch: 20 [20/27 (67%)]    Loss: 0.186895
Train Epoch: 20 [10/27 (83%)]    Loss: 0.152414

Test set: Avg. loss: 0.1759, Accuracy: 26/27 (96%)
```

This is a plot of 9 hand written letters, and the accuracy is 89%, we got one beta classified as alpha and all other letters are classified correctly.



Task 4: Design your own experiment

Since we did most of the tasks with MNIST data, we decided to work with MNIST data first, and then we will test the FashionMNIST data. For all 4 groups of results below, the dimension of interest will be highlighted in blue and all the other dimensions will be kept the same for all networks.

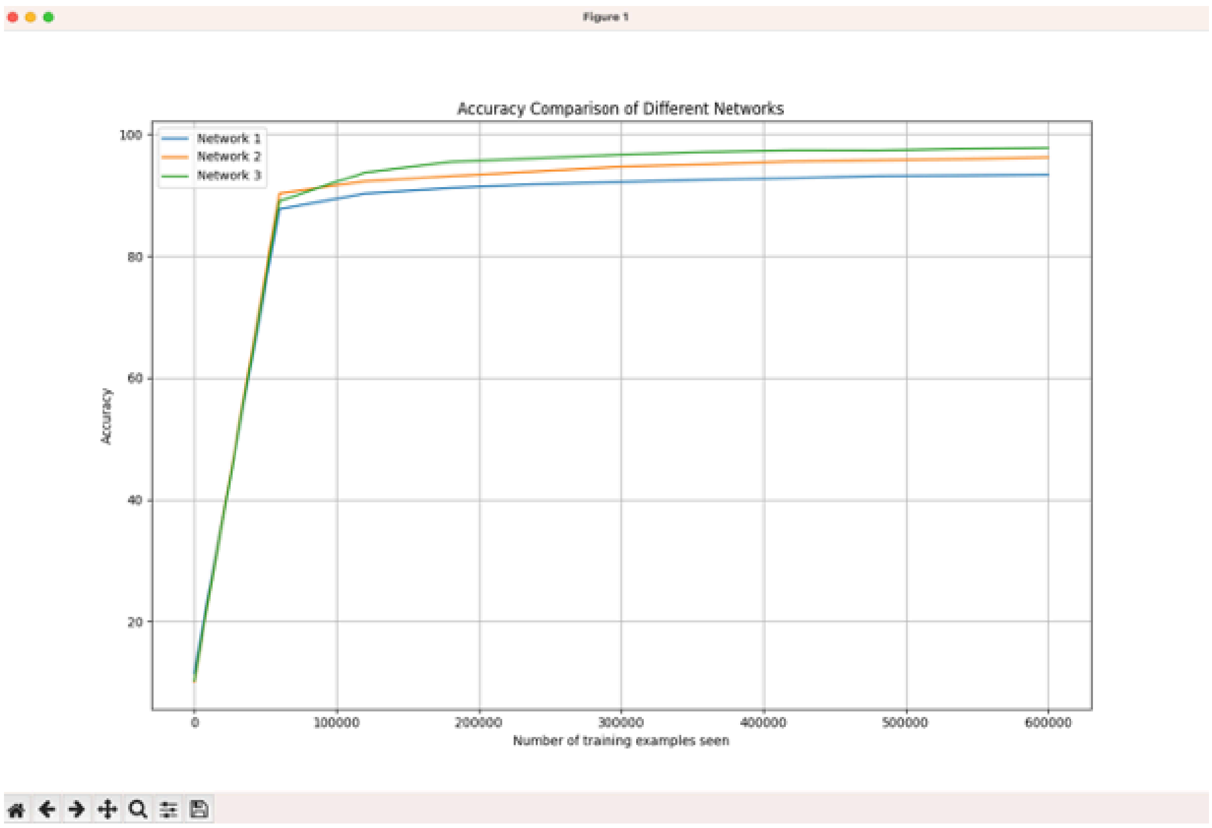
1). Plan: We try to see the network accuracy with numbers of convolution layers

(same num of filters and kernel size for each convolution layer)

Predict: the more convolution layers the more accuracy network

MNIST Data	Convolution layers	Number of filters in conv layer	Linear layers	Kernel size	Epoch

Network 1	0	10	2	None	10
Network 2	1	10	2	all 5x5	10
Network 3	2	10	2	all 5x5	10



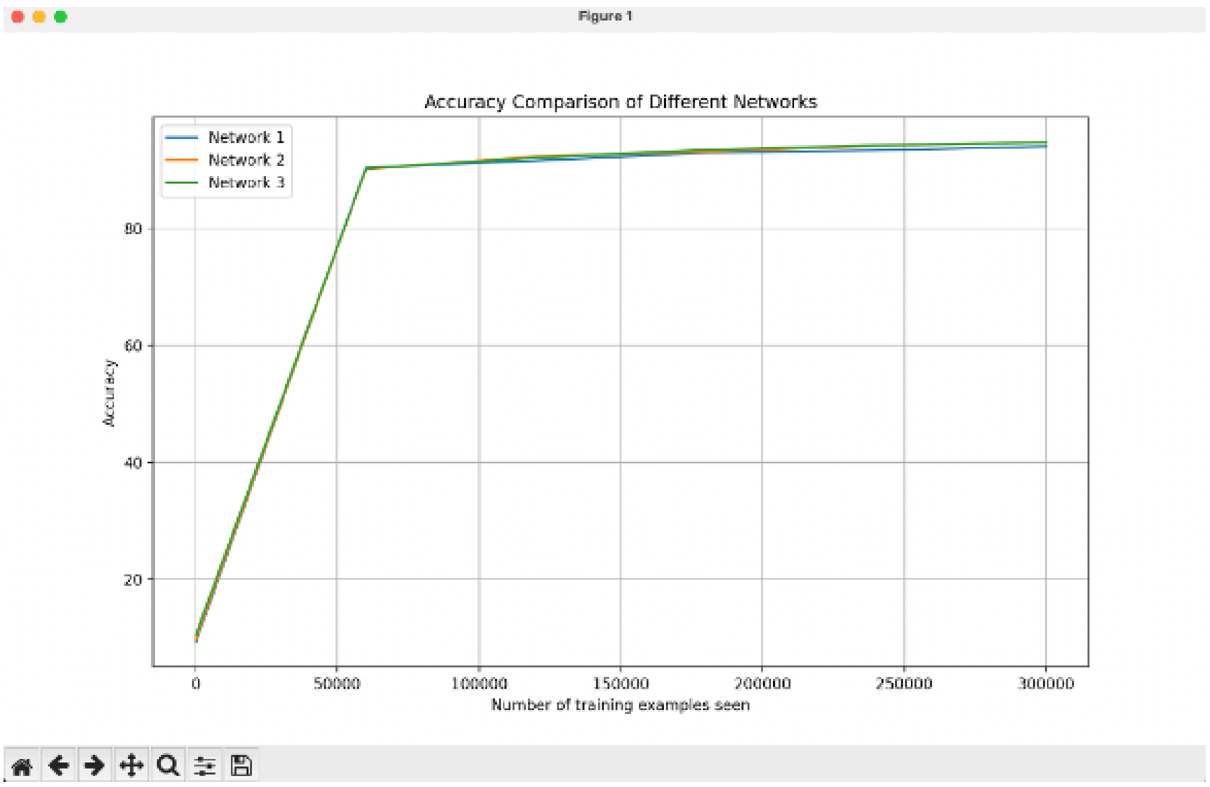
Result: By the result of the execution we can see, compared to changing kernel size and number of filters(plans will be mentioned below), different convolution layers have the biggest impact. And also we can see the more layers the better result(guessing correct)

2). Plan: We try to see the network accuracy with numbers of kernel size, all with one convolution layer.

(same amount of filters but different kernel size for the convolution layer)

Predict: the more kernel size the less accuracy(since the picture looks blurry with more convolution kernel.)

MNIST Data	Convolution layers	Number of filters in conv layer	Linear layers	Kernel size	Epoch
Network 1	1	10	2	3x3	5
Network 2	1	10	2	5x5	5
Network 3	1	10	2	7x7	5



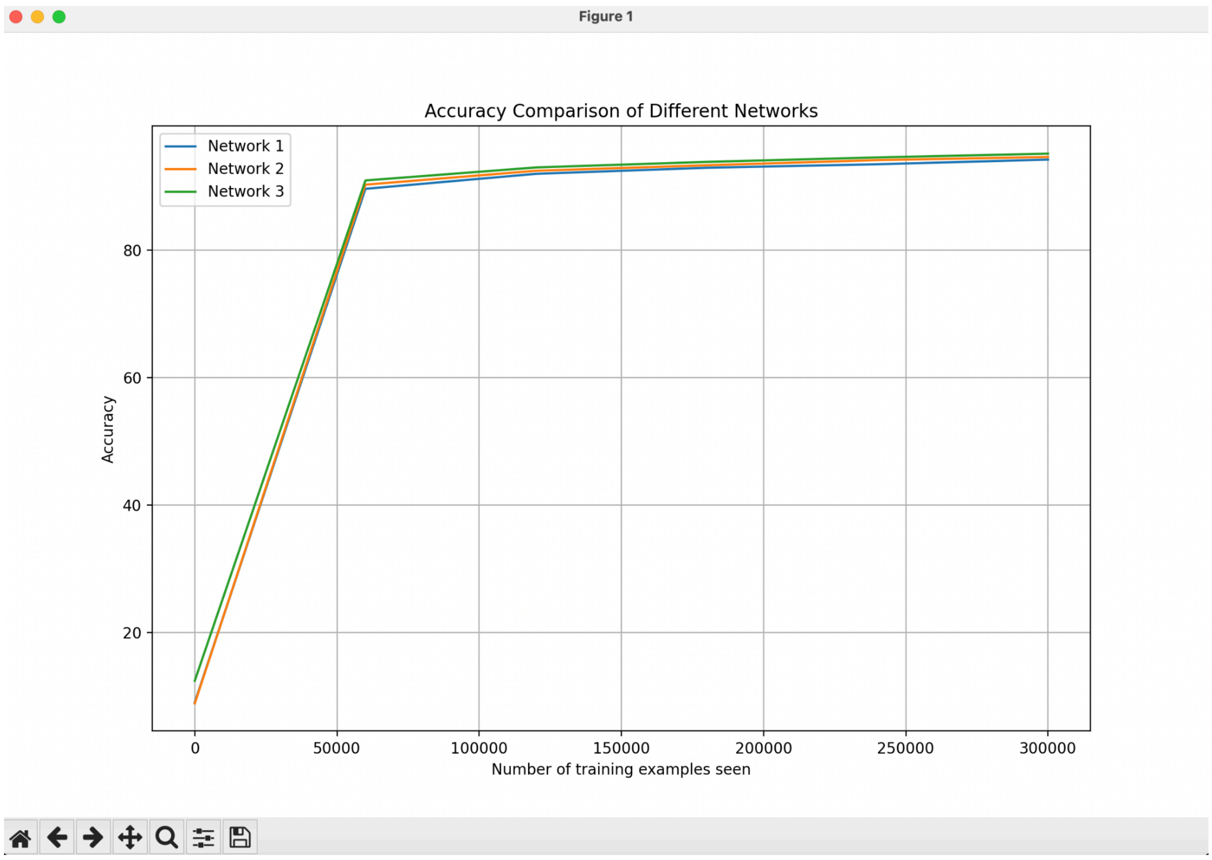
Result: By the result of the execution we can see, we can see the kernel size affects the network very little, the network1 have a 94% correct and the network2 have a 95% correct and the network 3 have a 95% correct after 5 epoch, even the difference is very little, the more kernel size the more accuracy (guessing wrong)

3). We try to see the network accuracy with numbers filters, all with one convolution layer.

(different nums of filters with same kernel size for the convolution layer)

guessing: the more filters the better results.

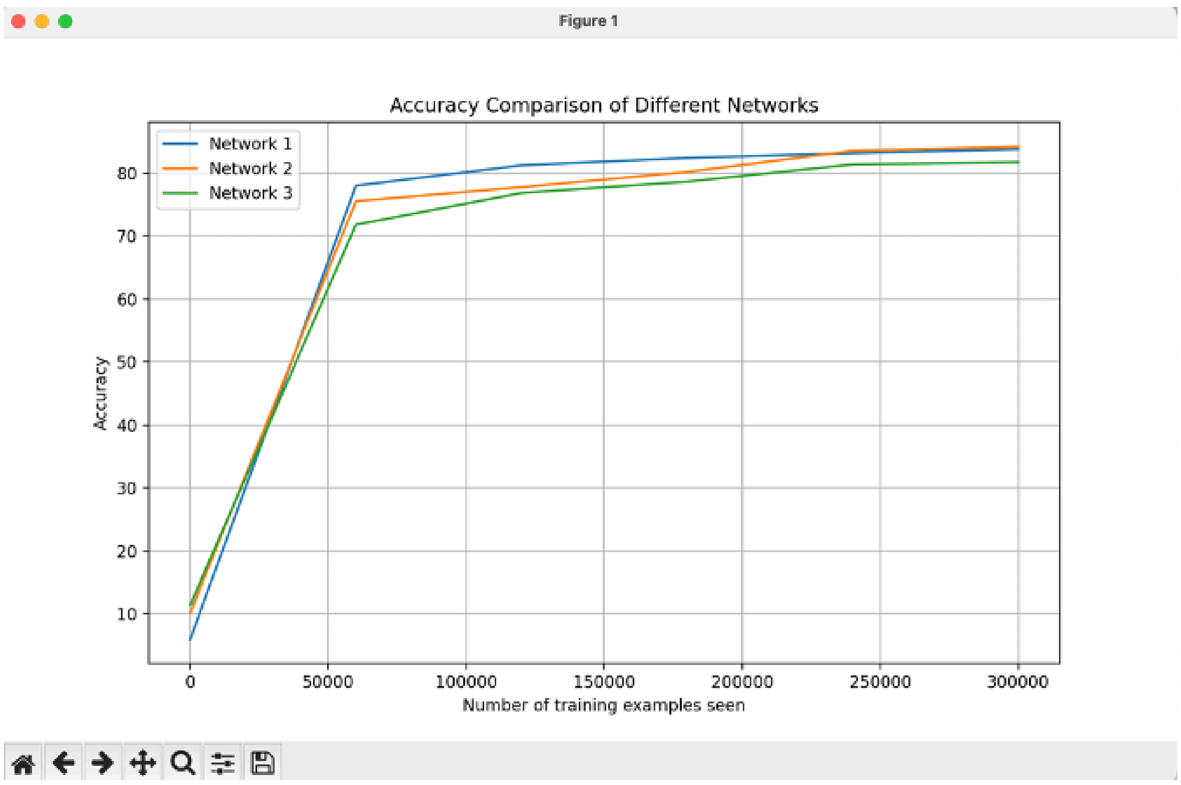
MNIST Data	Convolution layers	Number of filters in conv layer	Linear layers	Kernel size	Epoch
Network 1	1	5	2	5x5	5
Network 2	1	10	2	5x5	5
Network 3	1	20	2	5x5	5



Result: By the result of the execution we can see, we can see the numbers filters affects the network very little, the network1 have a 94% correct and the network2 have a 95% correct and the network 3 have a 95% correct after 5 epoch, even the difference is very little, the more filter the more accuracy (guessing correct)

4). different to test1, we try to use the same network to train fationMNIST data

Fashion Data	Convolution layers	Number of filters in conv layer	Linear layers	Kernel size	Epoch
Network 1	0	10	2	None	5
Network 2	1	10	2	all 5x5	5
Network 3	2	10	2	all 5x5	5



for MNIST data the accuracy ranking for the networks is:

network3>network2>network1

for FashionMNIST data the accuracy ranking for the networks is:

network2 >= network1>network3

After reading pytorch tutorial for fashionMINT, the tutorial flatten the pixel without convolution, instead have 2 connected Linear layers with 512 nodes and 1 with 10 nodes. The previous three experiments are all about changes in convolutional layers, which are not applicable to this FashionMNIST data. we get a conclusion that the network efficiency is different between different dataset.

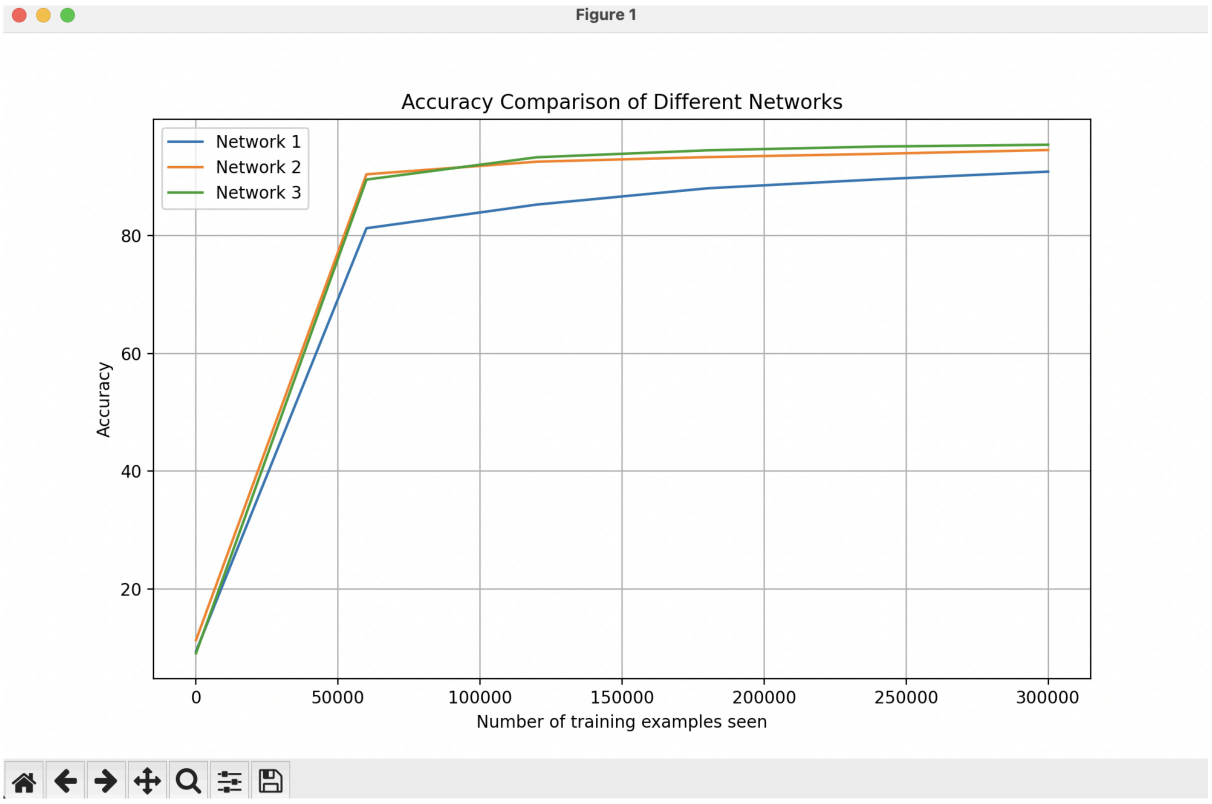
3. A description and example images of any extensions.

Extension 1: Tested one more dimension of the network.

Plan: Explore how the number of nodes in the linear layer affect the accuracy of the network.

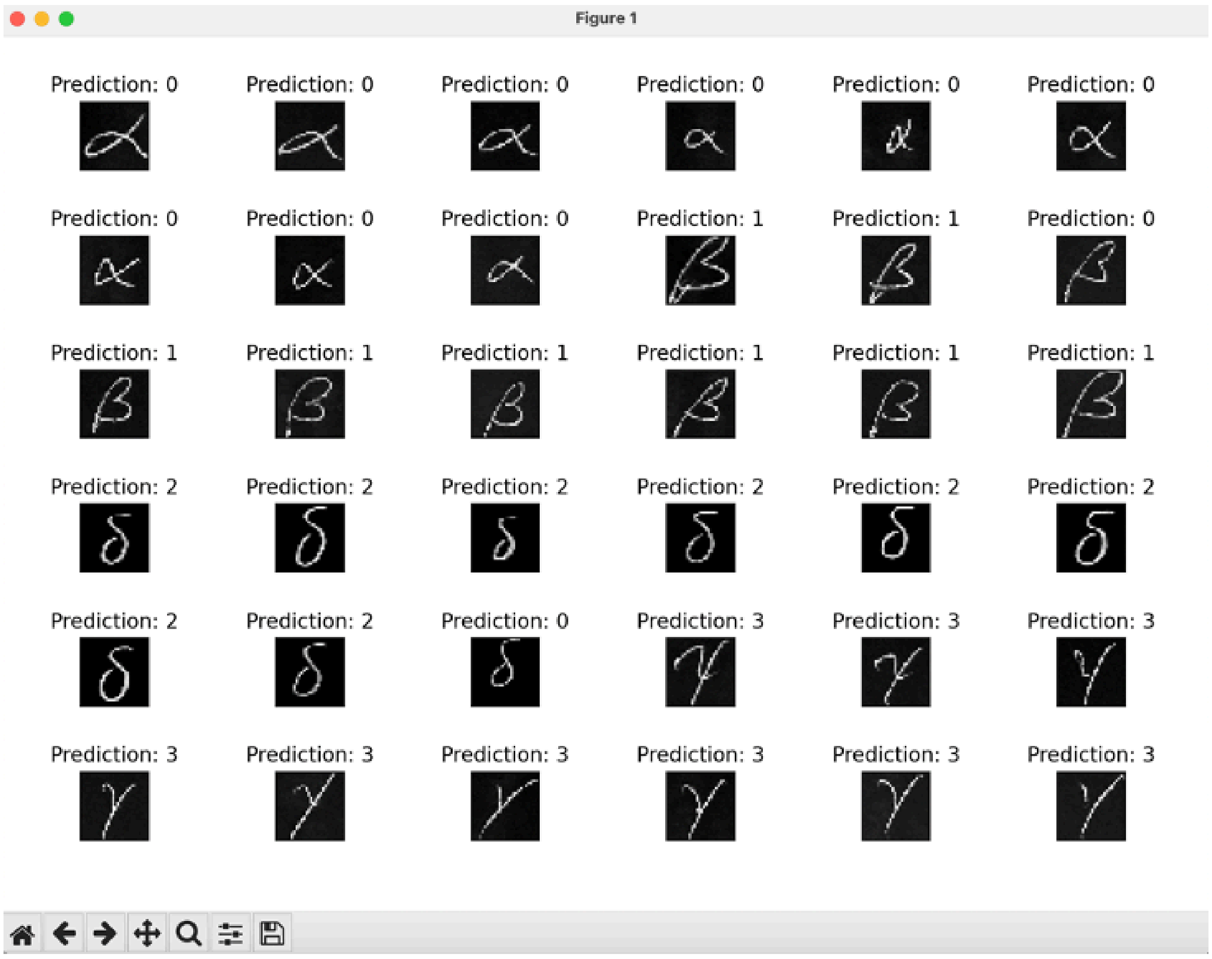
MNIST Data	Convolution layers	Number of filters in conv layer	Linear layers	Nodes in the first linear layer	Kernel size	Epoch
Network 1	1	10	2	10	5x5	5
Network 2	1	10	2	50	5x5	5
Network 3	1	10	2	100	5x5	5

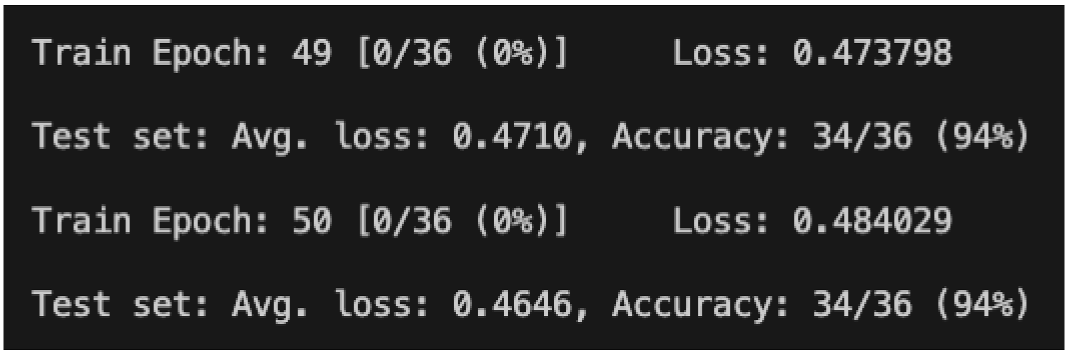
Prediction: We predict that the more the nodes in the linear layer, the more accurate the network will be since there will be more weights to be optimized.



Result: from the curve above, we can see that our prediction aligns with the result. But the room to optimize the network through the number of nodes in the linear layer is limited since the accuracy is close between 50 and 100 nodes.

Extension 2: We added one more Greek letter Delta in the training and tested the result.





```
Train Epoch: 49 [0/36 (0%)]	Loss: 0.473798

Test set: Avg. loss: 0.4710, Accuracy: 34/36 (94%)

Train Epoch: 50 [0/36 (0%)]	Loss: 0.484029

Test set: Avg. loss: 0.4646, Accuracy: 34/36 (94%)
```

After 50 epochs of training, the accuracy is about 94%.

4. A short reflection of what you learned.

   Form this project, we learn how to build, train and analyzing DNN for image classification, and we also how to use pytorch to visualize the filters in the convolutional layer, Moreover we know the network architectures (convolution layer, pooling layer, dropout layer, ReLU function,fully connected Linear layer), and how the parameters effect the network.(Number of filters in conv layer, Number of nodes in the linear layer, Kernel size, epoch size, batch size)

5. Acknowledgement of any materials or people you consulted for the assignment.

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

   https://pytorch.org/tutorials/beginner/basics/transforms_tutorial.html

   and thanks to the professor for the problem solving provided on piazza and office hours.

6. Whether you are using any time travel days and how many.

   Yes, we use 1 travel day.