PATTERN RECOGNITION & COMPUTER VISION

Project 5: Recognition using Deep Networks

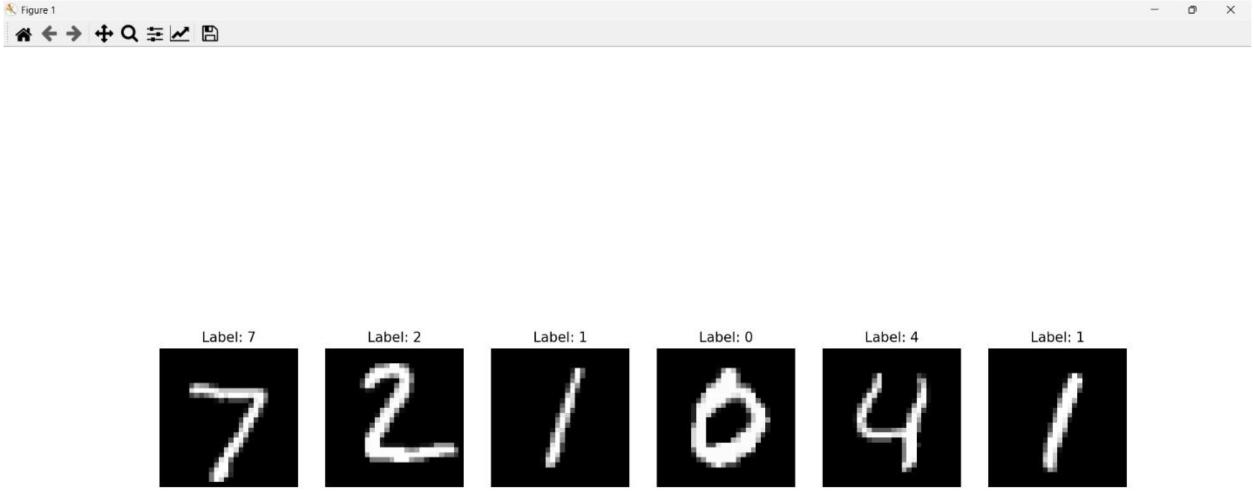
BASIL REJI & KEVIN SANI

Description:

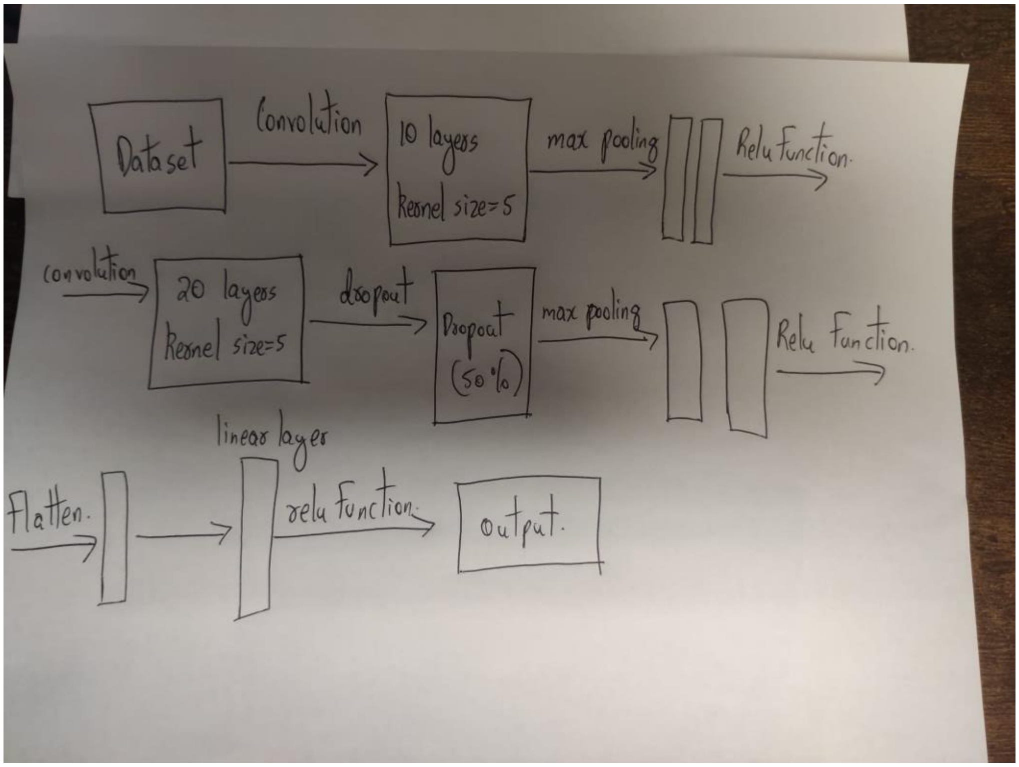
The project entails designing, implementing, training, and experimenting on a neural network to be fit for image recognition. First, the MNIST digits dataset was used. This is the chosen dataset, as it is well known for balancing simplicity and the level of challenge while designing and training a convolution neural network (CNN). The architecture of the model was initially designed; it kept in mind convolutional layers, pooling, and dropout for effective recognition of handwritten digits. To generalize this exploration further, that very network was adapted to recognize Greek letters. This was done by modifying the final layer to classify three different classes and retraining with a custom dataset. Afterwards, we varied the architecture of the network systematically along certain dimensions: the number of convolutional filters, the sizes of the kernels, and the rate of the dropouts. This research, in that respect, sought to run a set of experiments with a variety of configurations to optimize the network's performance and efficiency of the MNIST Fashion dataset. Very useful in grasping the behavior of the network in all setups, it clearly enabled us to understand how some architectural decisions affect both learning effectiveness and computational demands.

Task 1:Build and train a network to recognize digits

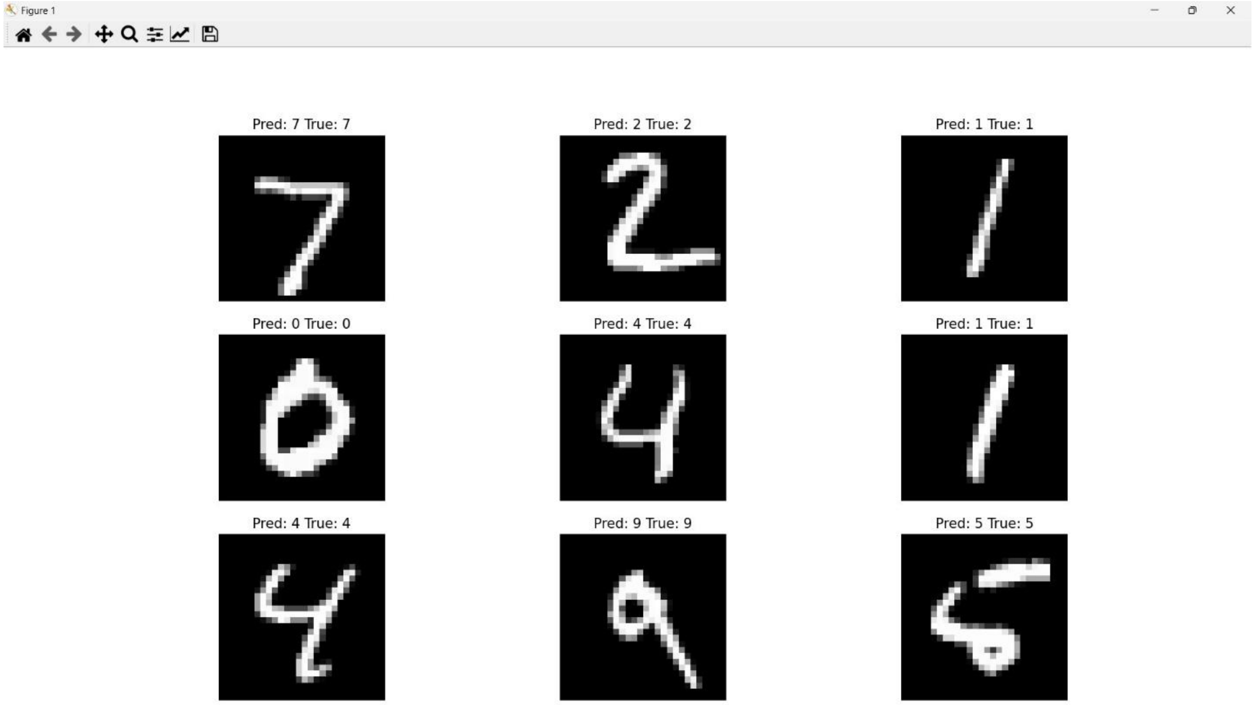
First six example digits from the test set



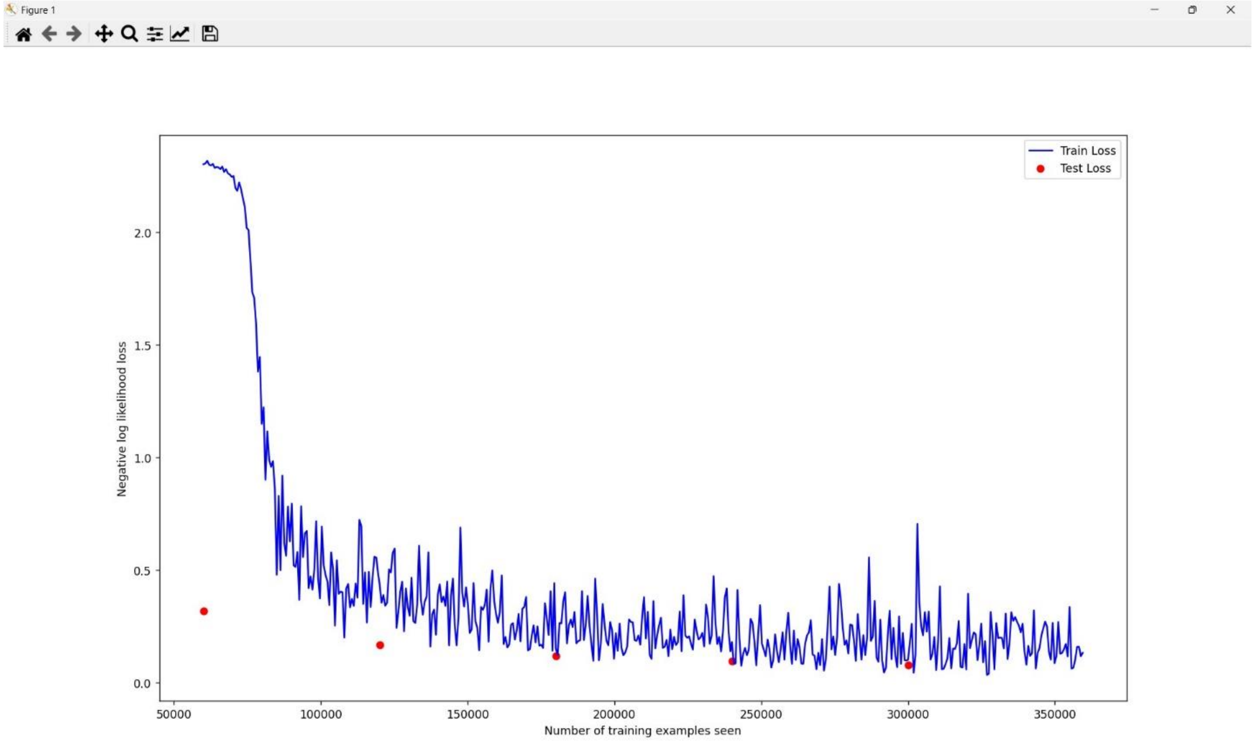
Network Model



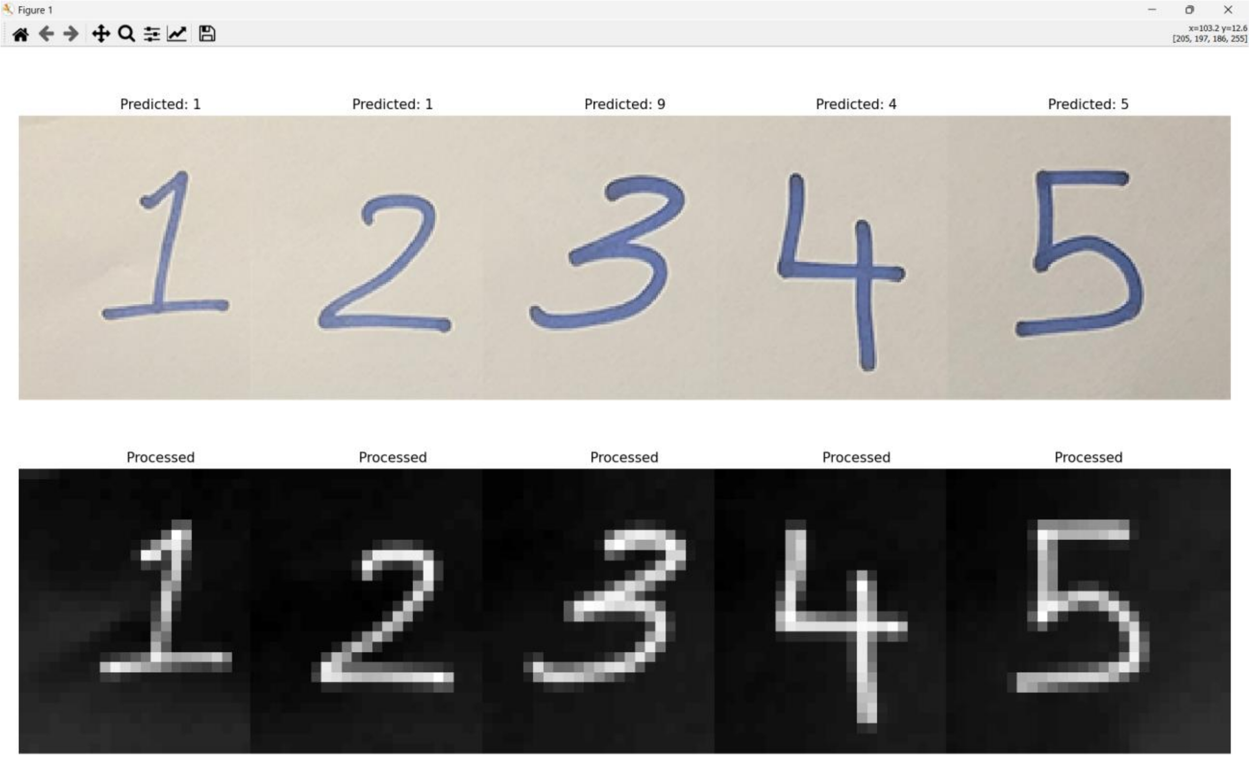
9 digits as a 3x3 grid with the prediction for each example above it



Training and testing accuracy

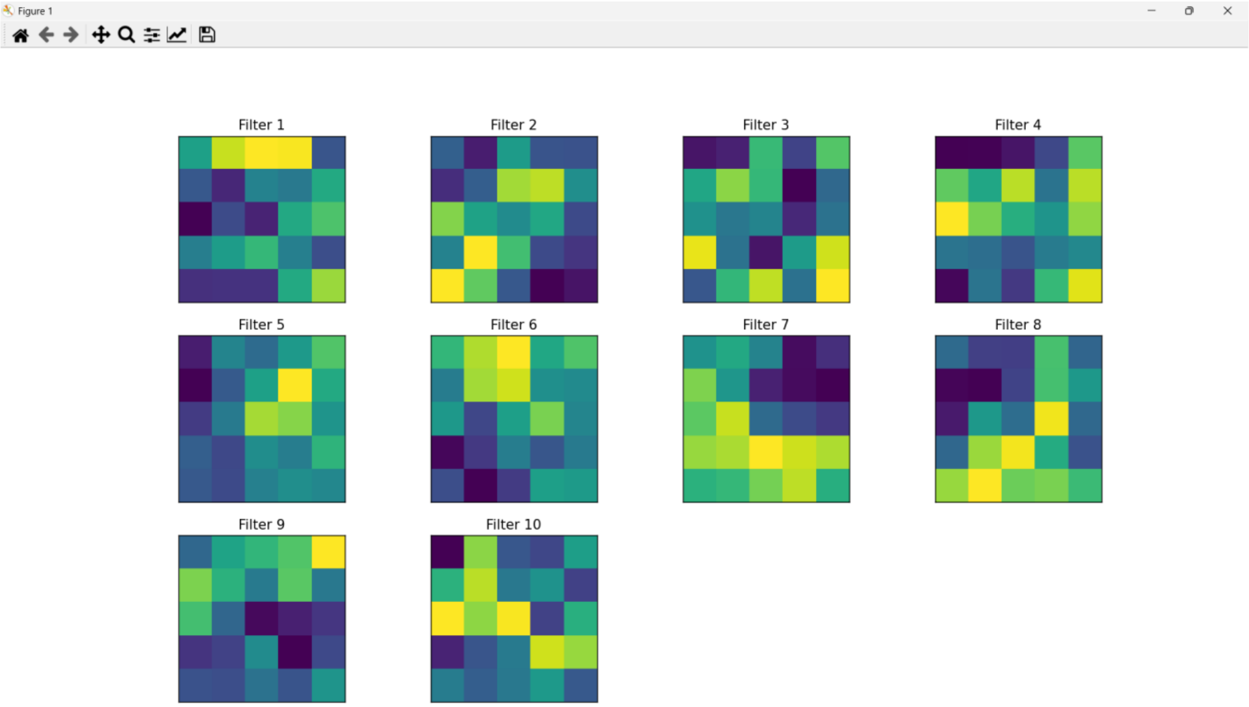


digits and their classified result for written dataset

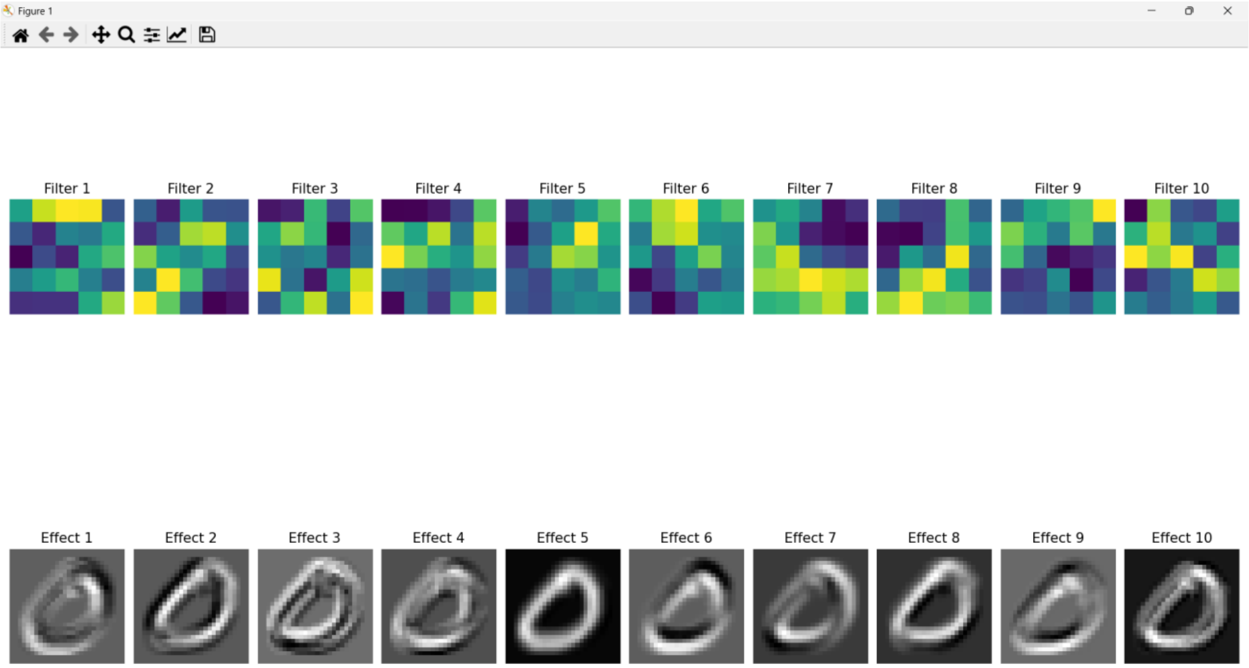


Task 2: Examine your network

Visualization of the filters

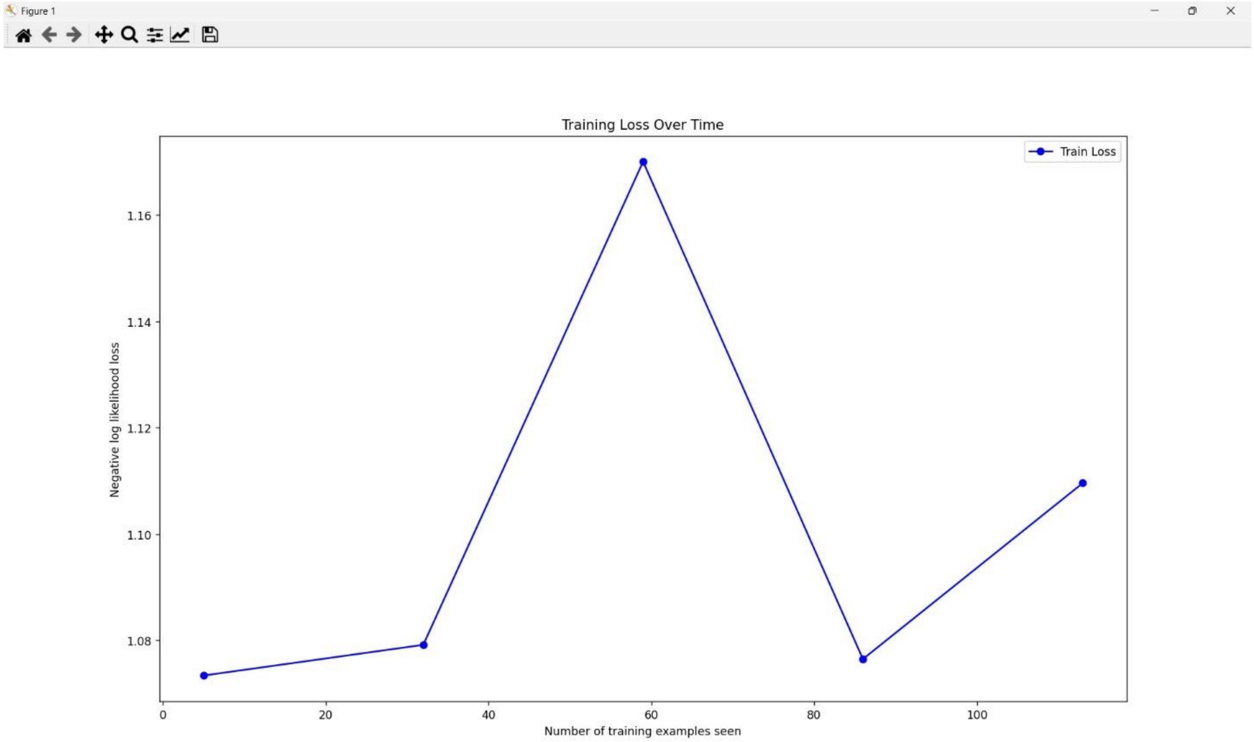


Effect of the corresponding filter on the corresponding data

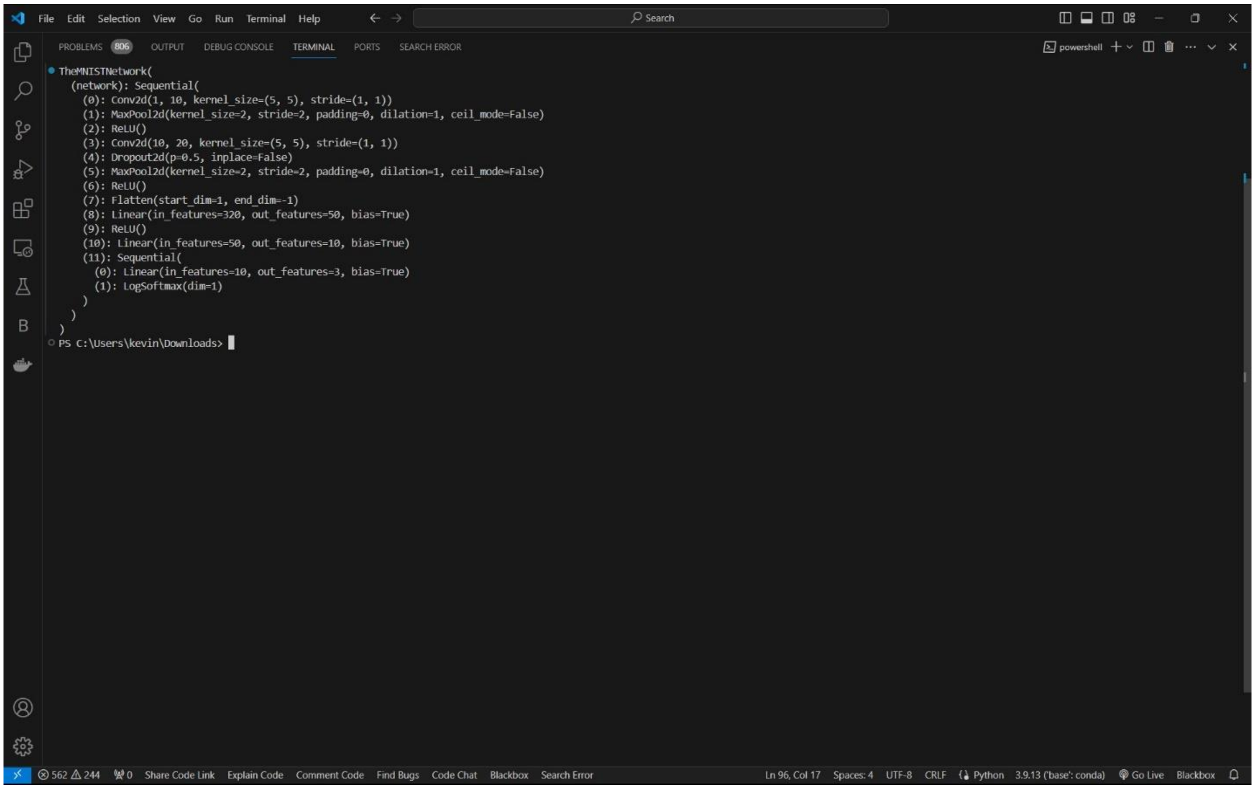


Task 3: Transfer Learning on Greek Letters

Training loss over the new network and the Greek Letter Dataset

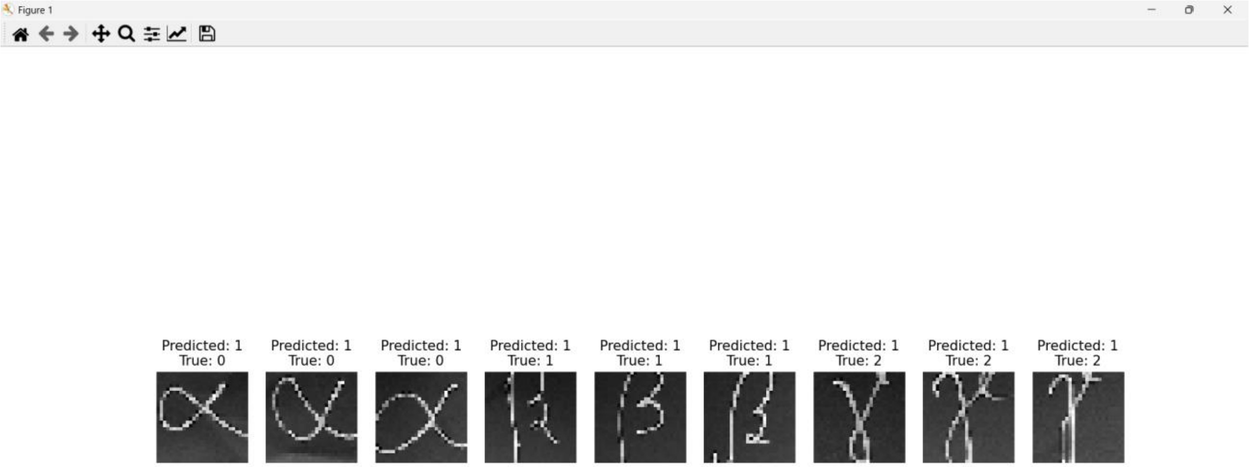


Specifications of the new Network Model



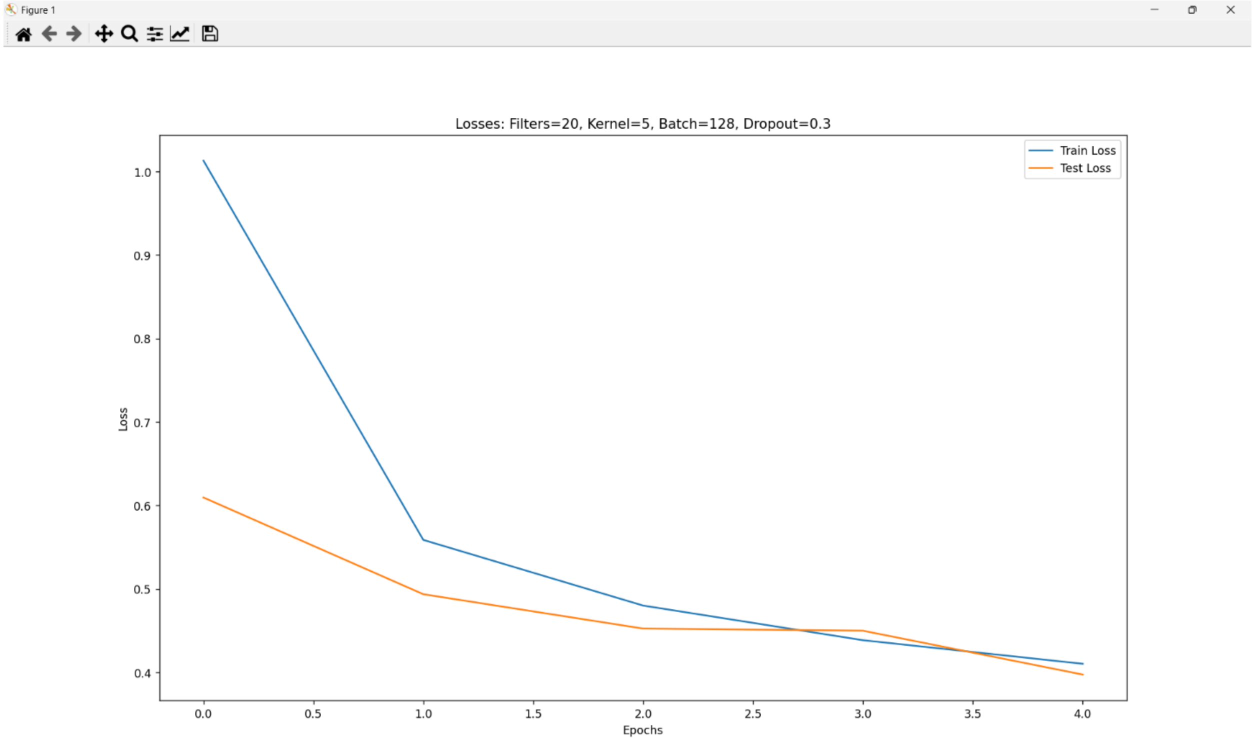
```
TheMNISTNetwork(
  (network): Sequential(
    (0): Conv2d(1, 10, kernel_size=(5, 5), stride=(1, 1))
    (1): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (2): ReLU()
    (3): Conv2d(10, 20, kernel_size=(5, 5), stride=(1, 1))
    (4): Dropout2d(p=0.5, inplace=False)
    (5): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (6): ReLU()
    (7): Flatten(start_dim=1, end_dim=-1)
    (8): Linear(in_features=320, out_features=50, bias=True)
    (9): ReLU()
    (10): Linear(in_features=50, out_features=10, bias=True)
    (11): Sequential(
      (0): Linear(in_features=10, out_features=3, bias=True)
      (1): LogSoftmax(dim=1)
    )
  )
)
PS C:\Users\kevin\Downloads>
```

classified by the trained network on written dataset



Task 4: Design your own experiment

Collection of images of training and testing error with different parameters run on the custom network the corresponding parameters used are displayed as titles for the images



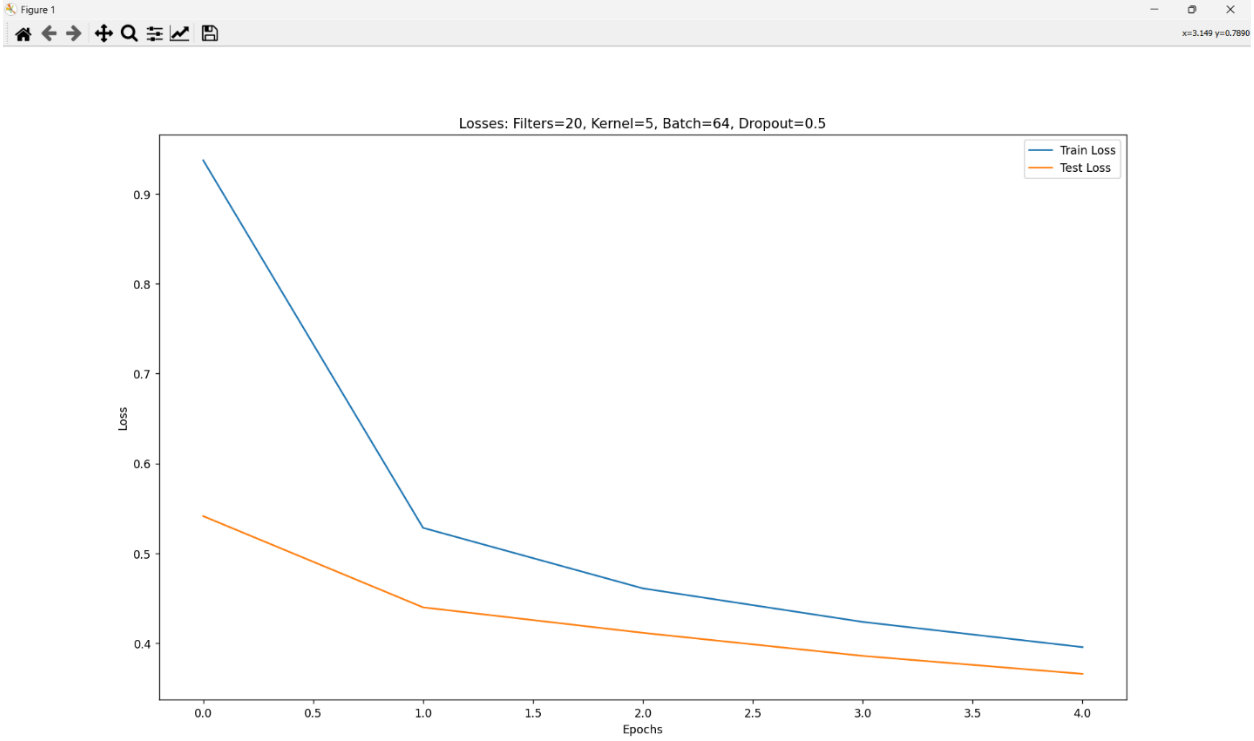


Figure 1
x=3.149 y=0.7890
Losses: Filters=20, Kernel=5, Batch=64, Dropout=0.5
Train Loss
Test Loss
Loss
0.9
0.8
0.7
0.6
0.5
0.4
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
Epochs

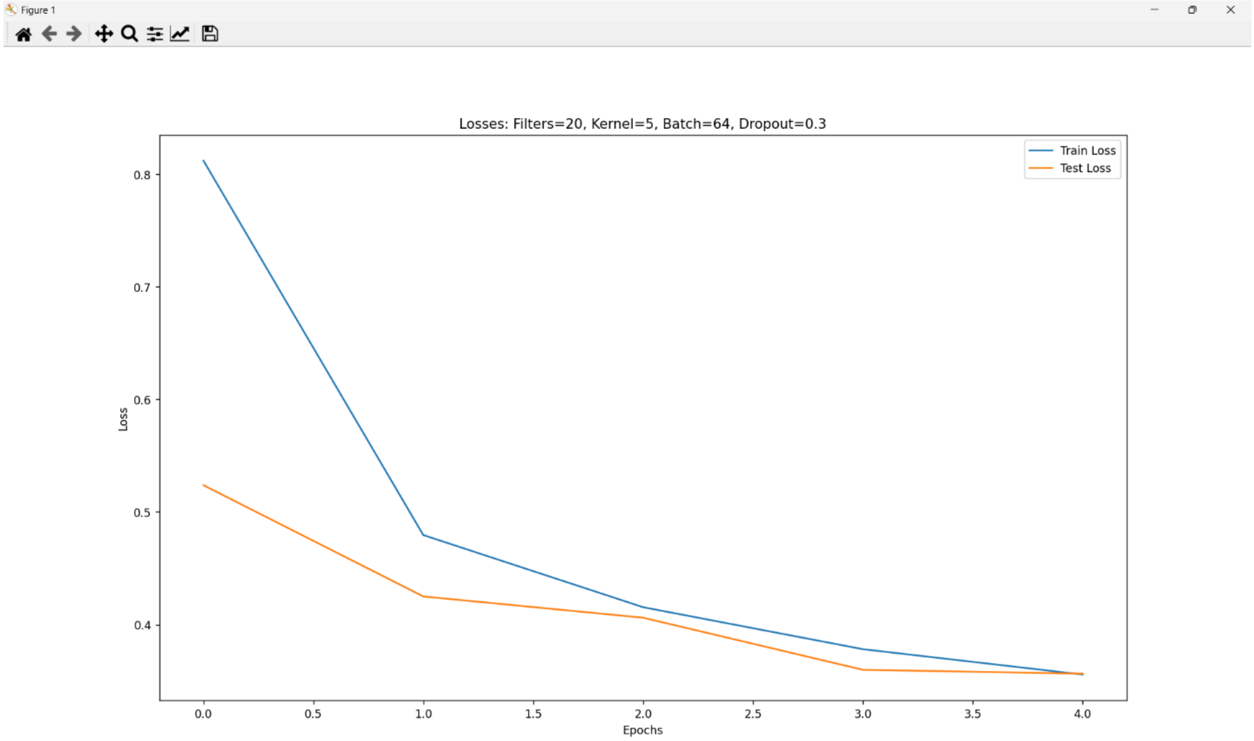


Figure 1
Losses: Filters=20, Kernel=5, Batch=64, Dropout=0.3
Train Loss
Test Loss
Loss
0.8
0.7
0.6
0.5
0.4
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
Epochs

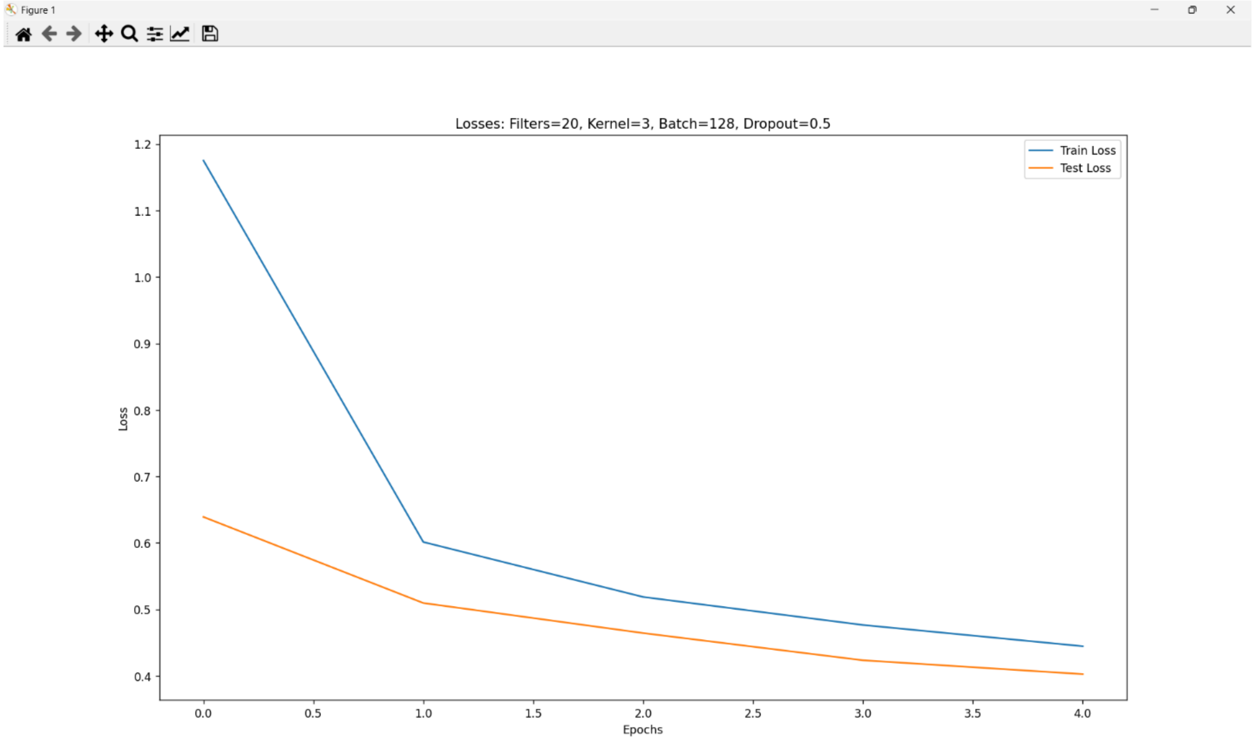


Figure 1
Losses: Filters=20, Kernel=3, Batch=128, Dropout=0.5
Train Loss
Test Loss
Loss
1.2
1.1
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
Epochs

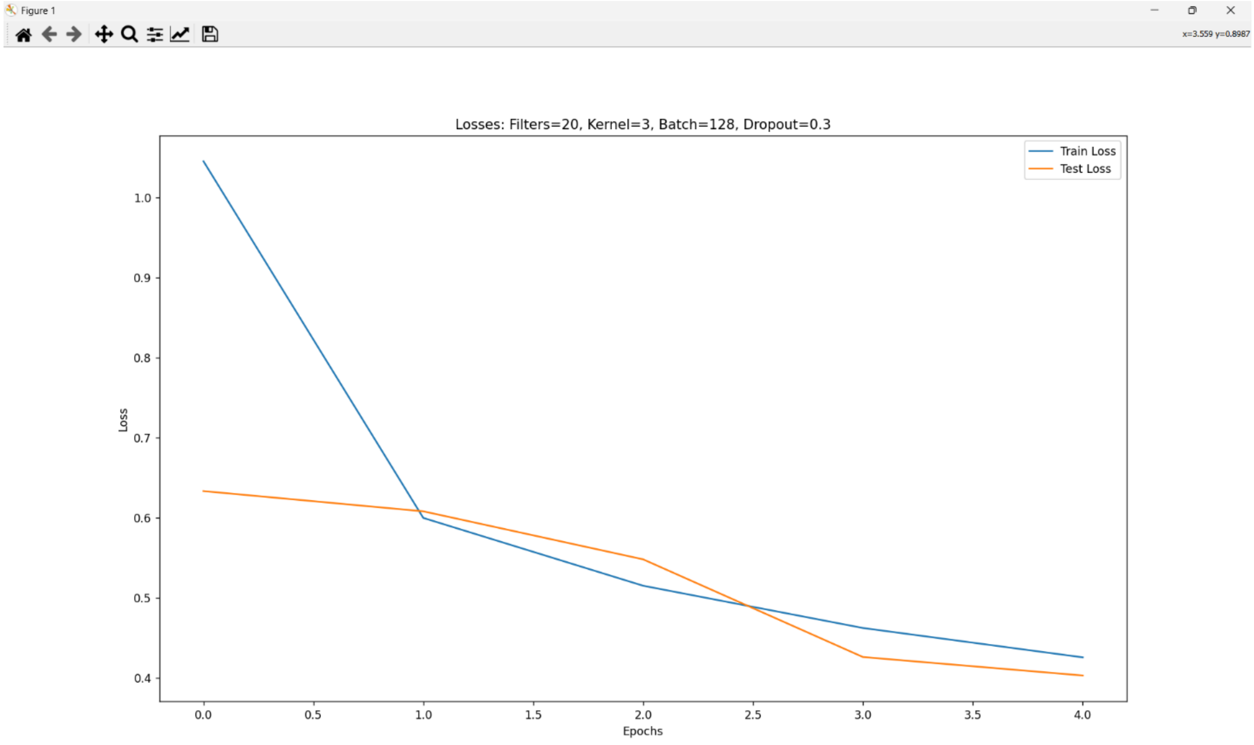


Figure 1
x=3.559 y=0.8987
Losses: Filters=20, Kernel=3, Batch=128, Dropout=0.3
Train Loss
Test Loss
Loss
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
Epochs

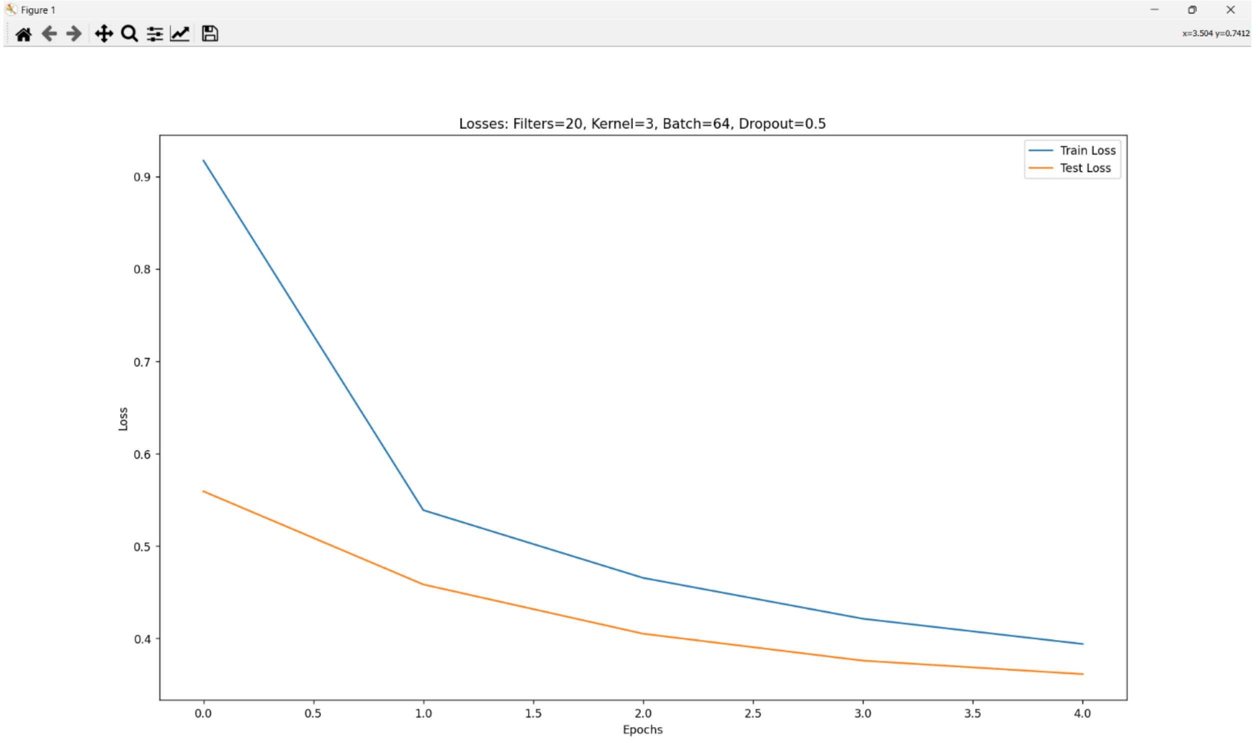


Figure 1
x=3.504 y=0.7412
Losses: Filters=20, Kernel=3, Batch=64, Dropout=0.5
Train Loss
Test Loss
Loss
0.9
0.8
0.7
0.6
0.5
0.4
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
Epochs

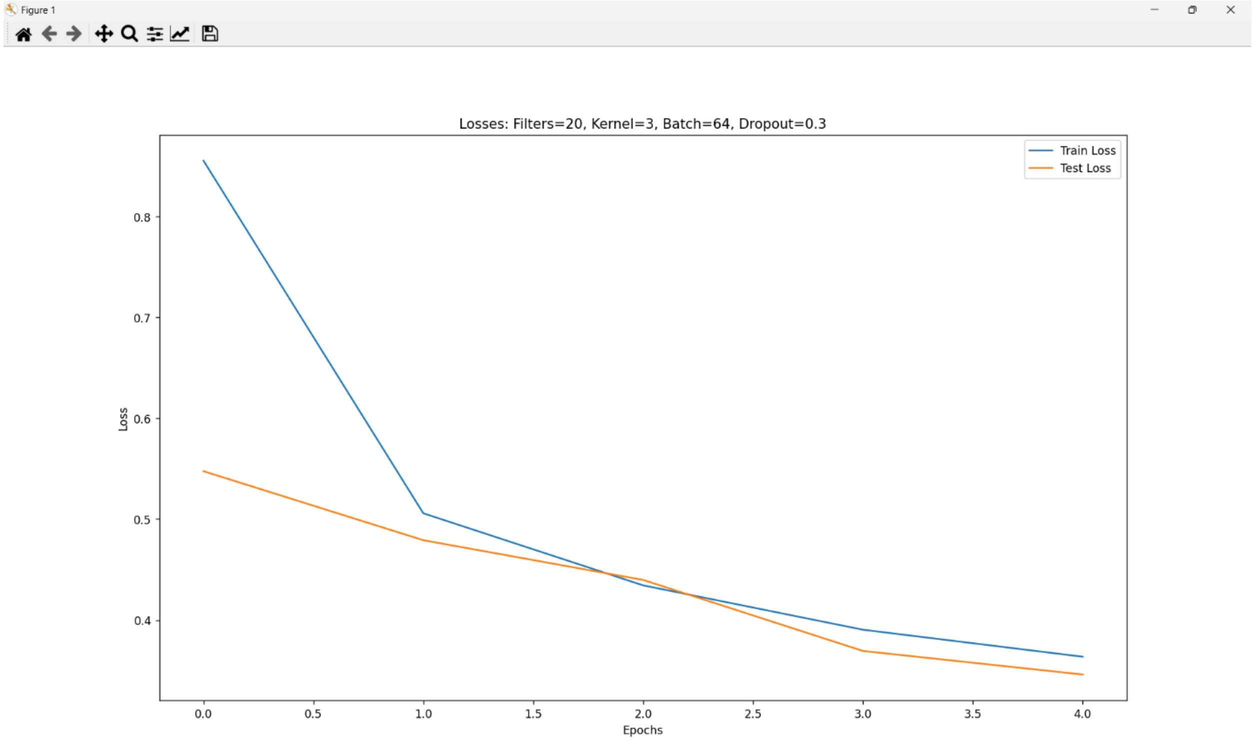


Figure 1
Losses: Filters=20, Kernel=3, Batch=64, Dropout=0.3
Train Loss
Test Loss
Loss
0.8
0.7
0.6
0.5
0.4
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
Epochs

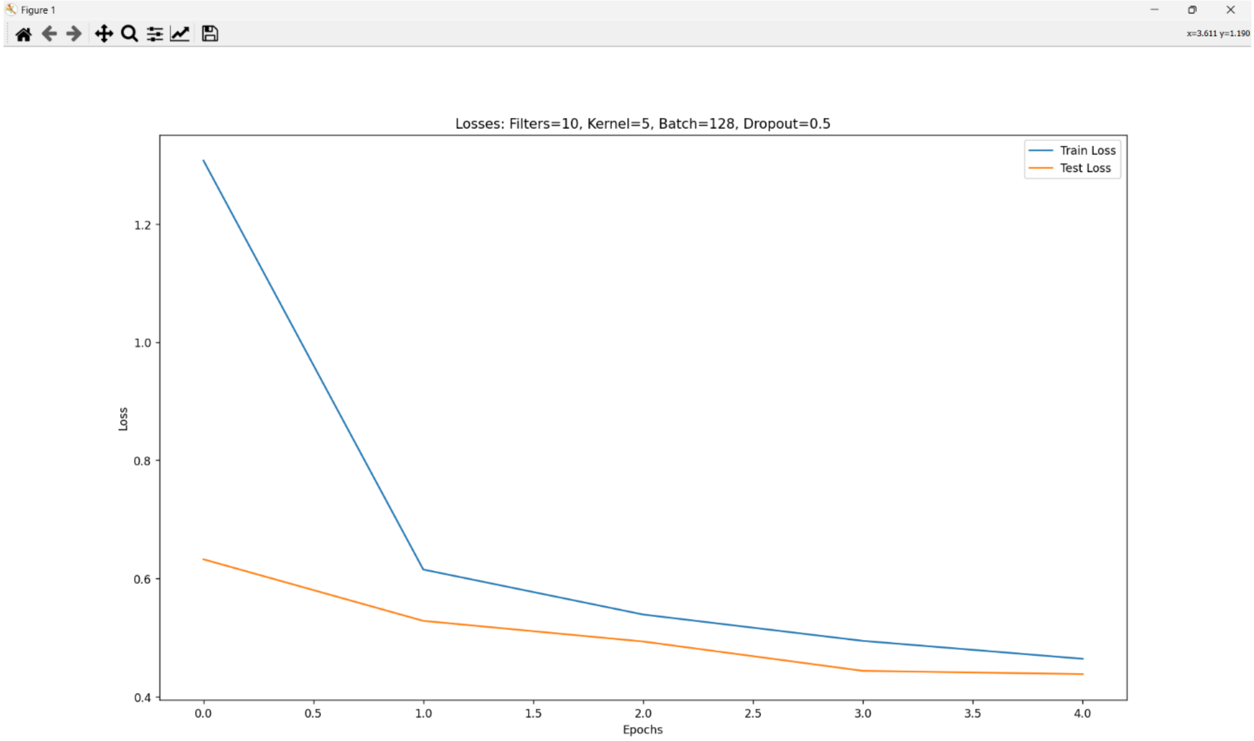


Figure 1
x=3.611 y=1.190
Losses: Filters=10, Kernel=5, Batch=128, Dropout=0.5
Train Loss
Test Loss
Loss
1.2
1.0
0.8
0.6
0.4
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
Epochs

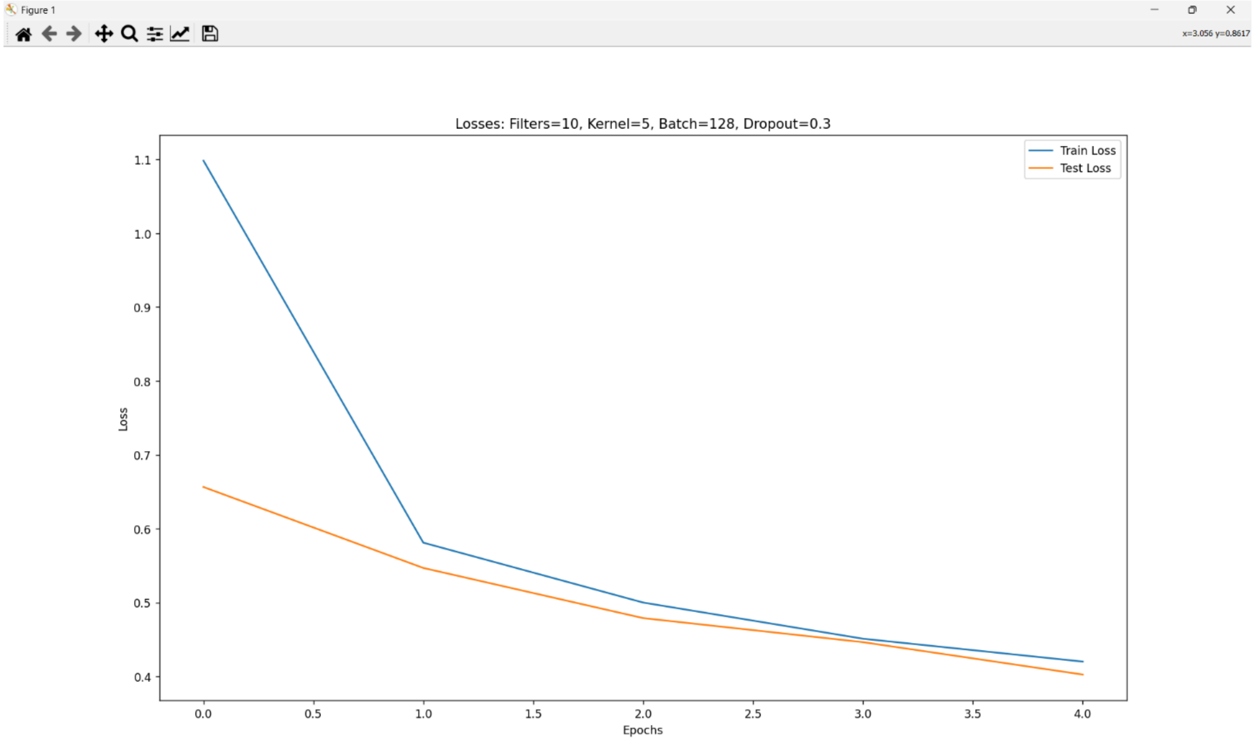


Figure 1
x=3.056 y=0.8617
Losses: Filters=10, Kernel=5, Batch=128, Dropout=0.3
Train Loss
Test Loss
Loss
1.1
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
Epochs

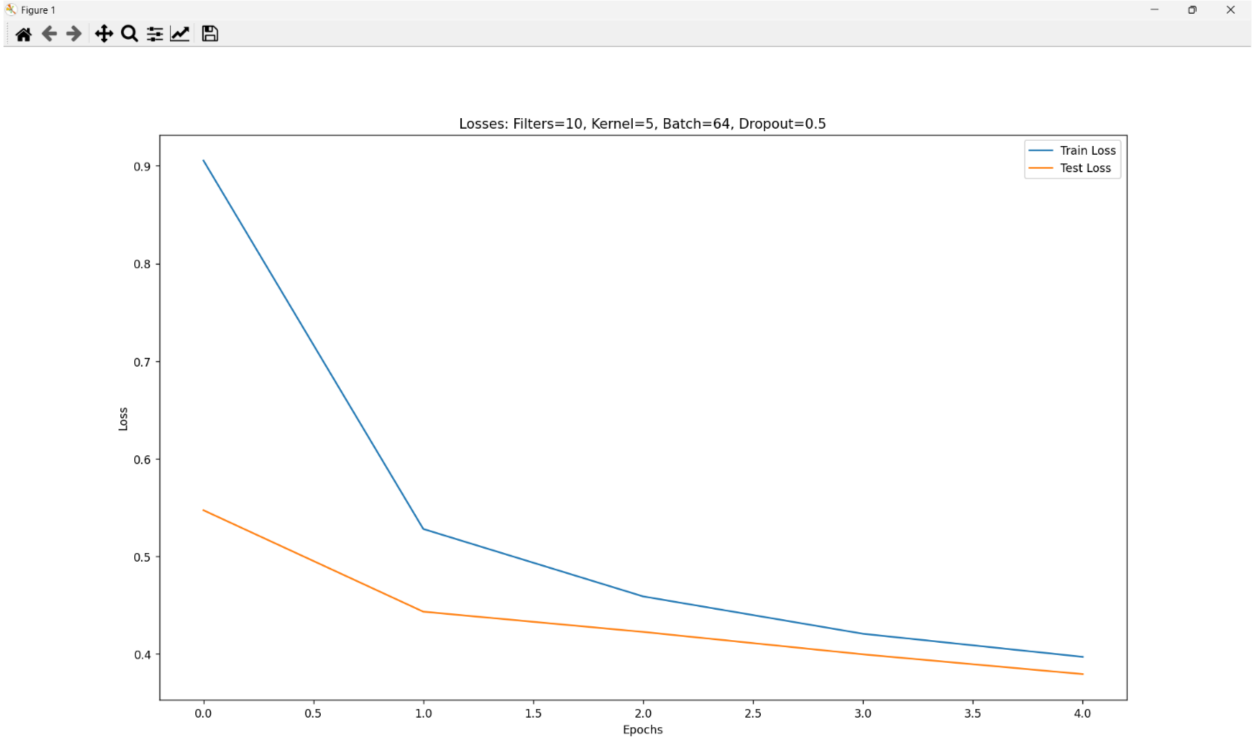


Figure 1
Losses: Filters=10, Kernel=5, Batch=64, Dropout=0.5
Train Loss
Test Loss
Loss
0.9
0.8
0.7
0.6
0.5
0.4
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
Epochs

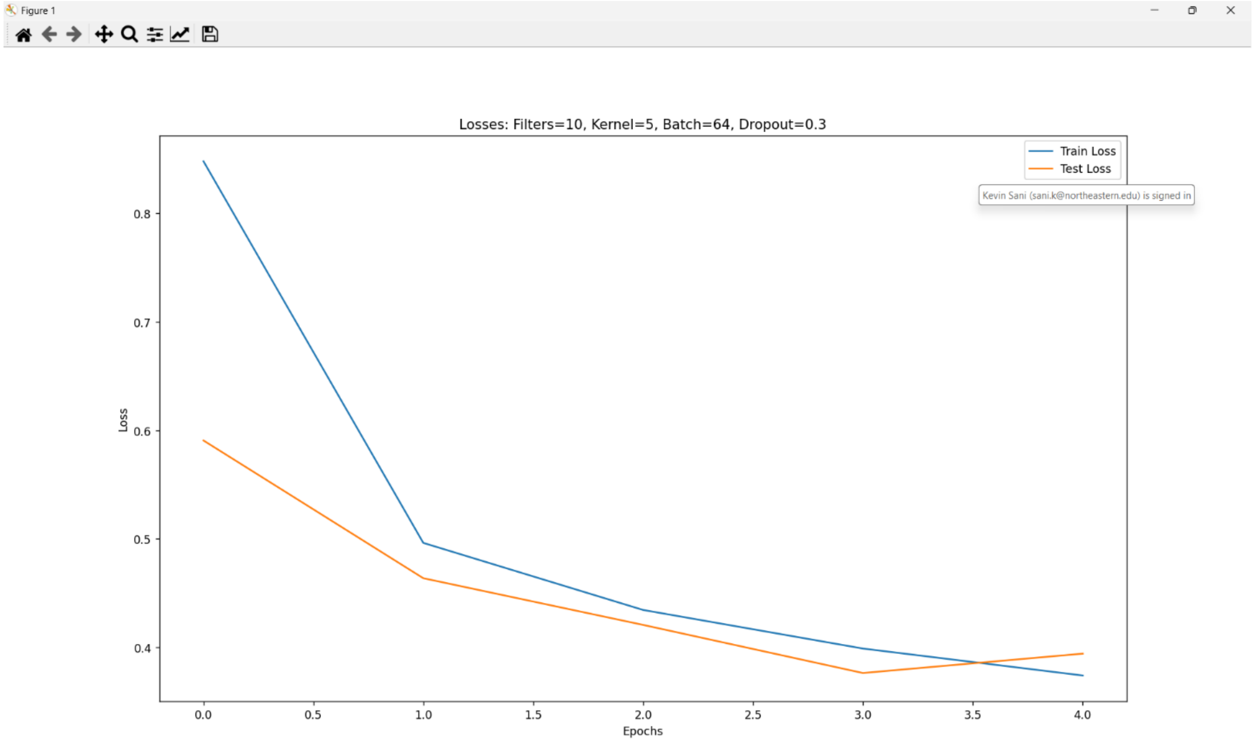


Figure 1
Losses: Filters=10, Kernel=5, Batch=64, Dropout=0.3
Train Loss
Test Loss
Kevin Sani (sani.k@northeastern.edu) is signed in
Loss
0.8
0.7
0.6
0.5
0.4
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
Epochs

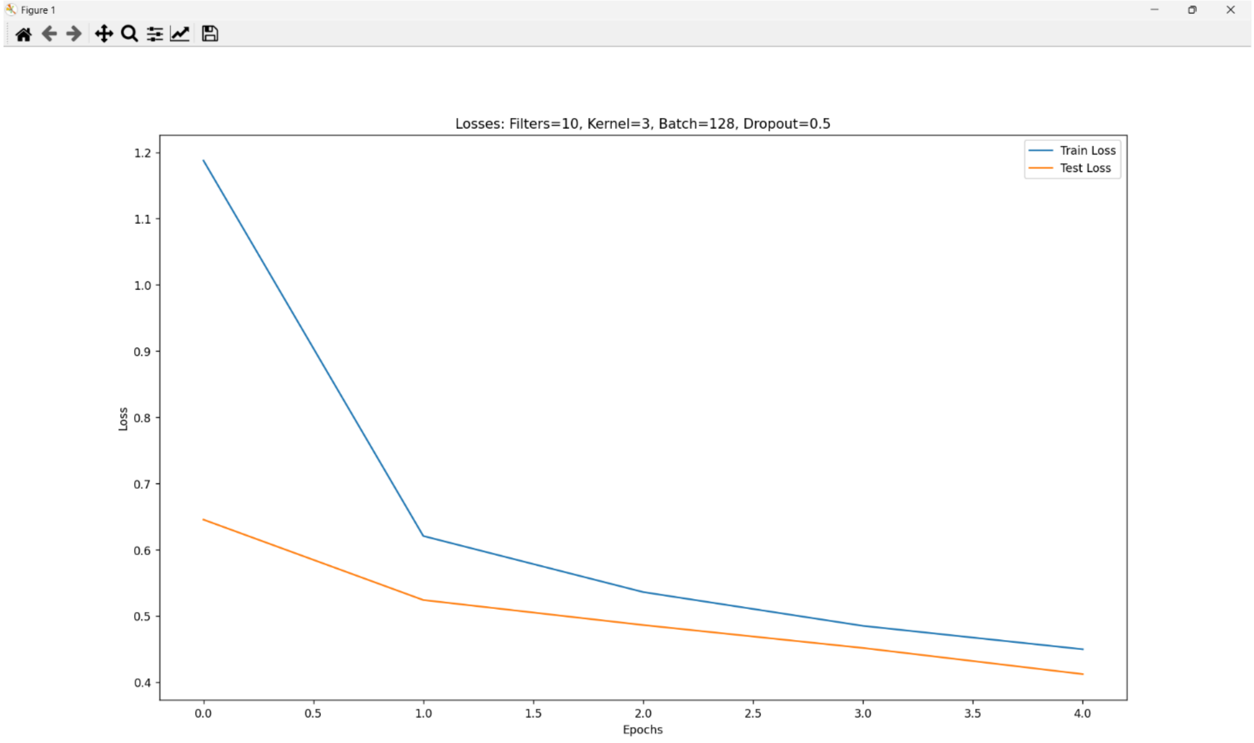


Figure 1
Losses: Filters=10, Kernel=3, Batch=128, Dropout=0.5
Train Loss
Test Loss
Loss
1.2
1.1
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
Epochs

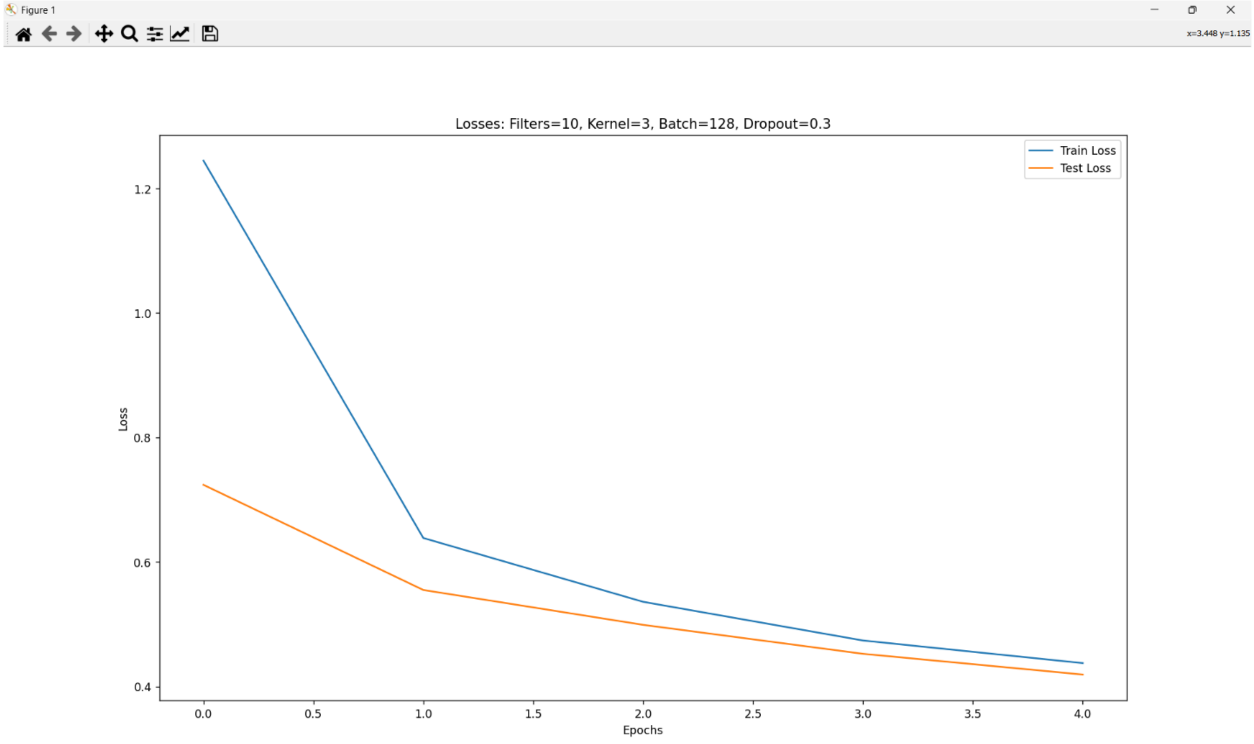


Figure 1
x=3.448 y=1.135
Losses: Filters=10, Kernel=3, Batch=128, Dropout=0.3
Train Loss
Test Loss
Loss
1.2
1.0
0.8
0.6
0.4
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
Epochs

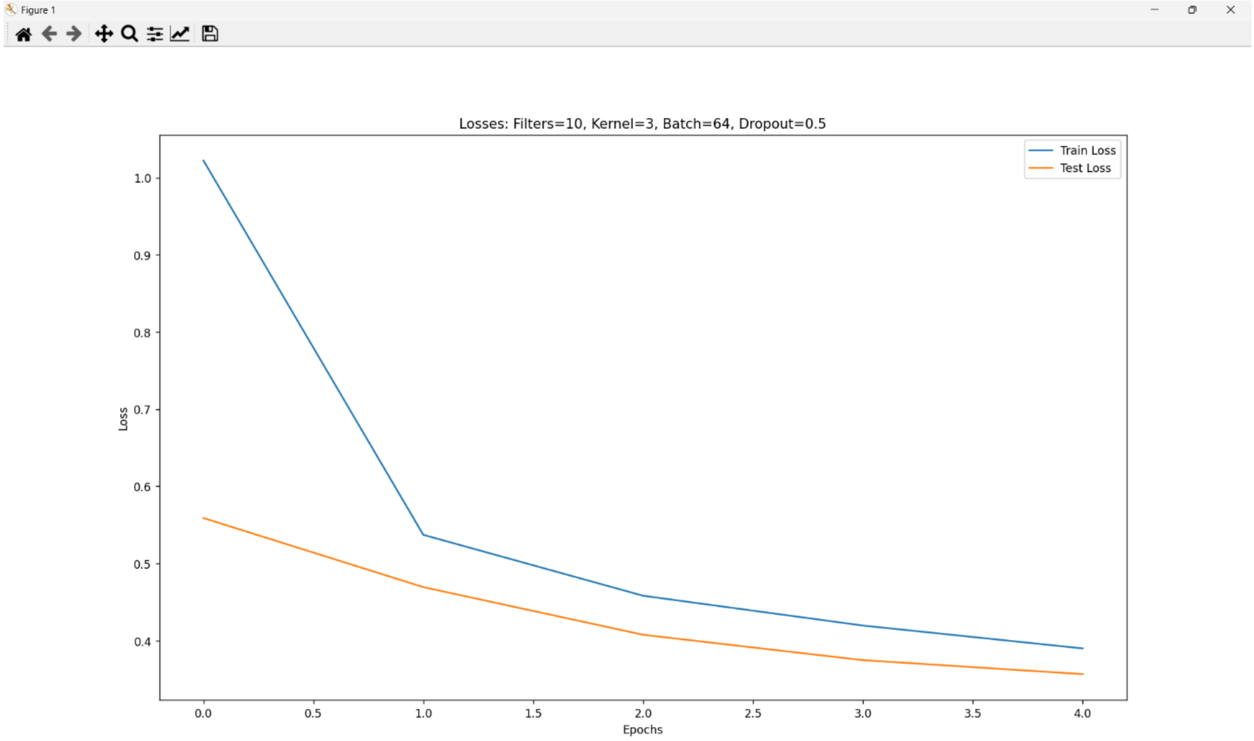


Figure 1
Losses: Filters=10, Kernel=3, Batch=64, Dropout=0.5
Train Loss
Test Loss
Loss
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
Epochs

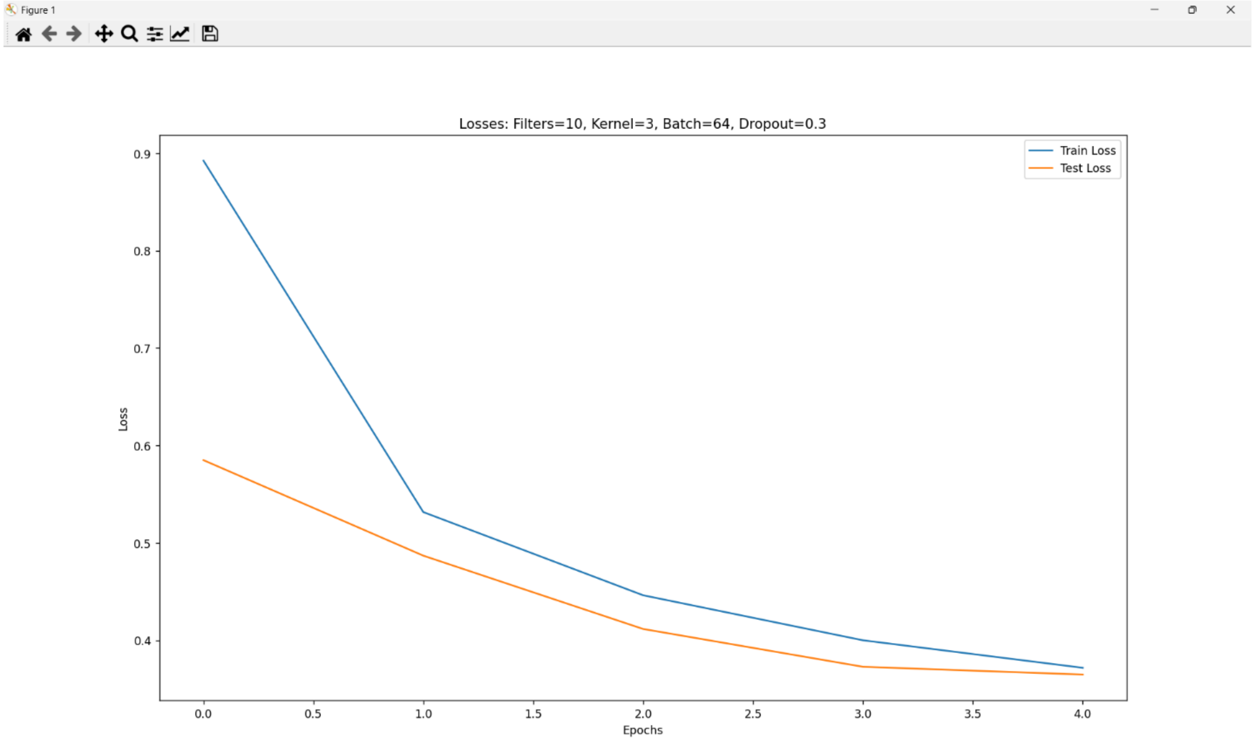
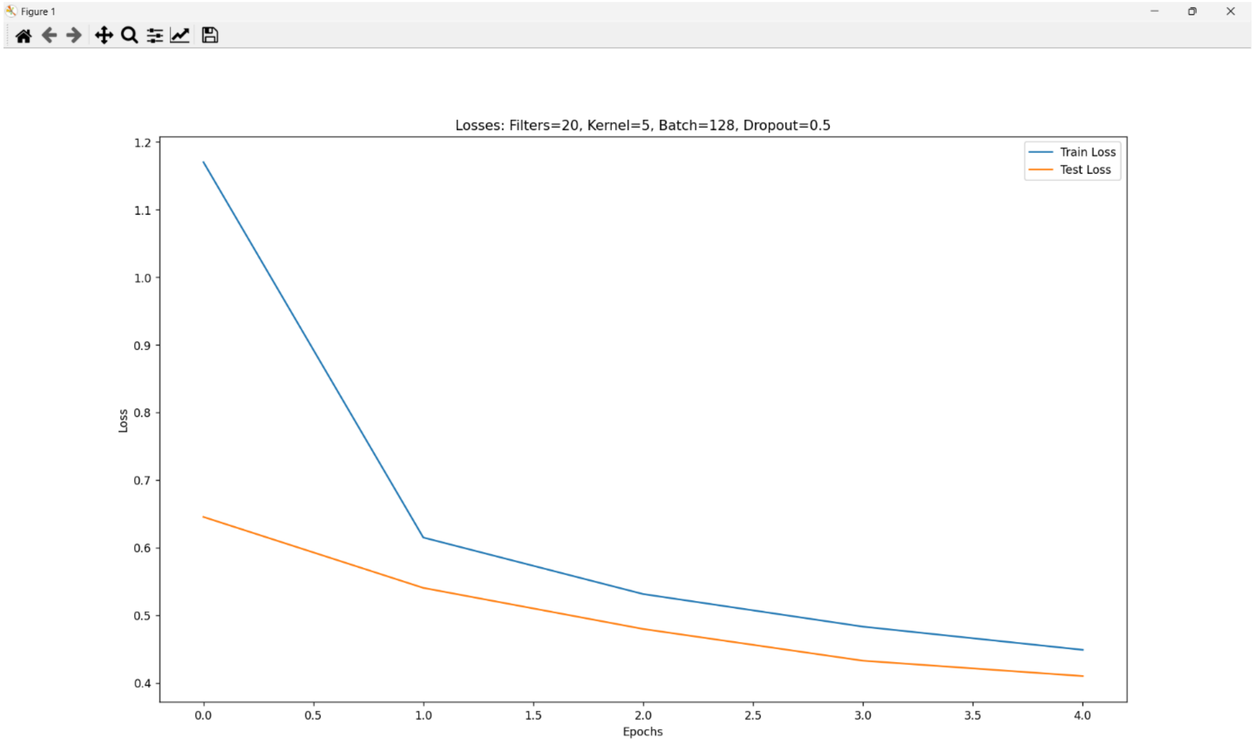
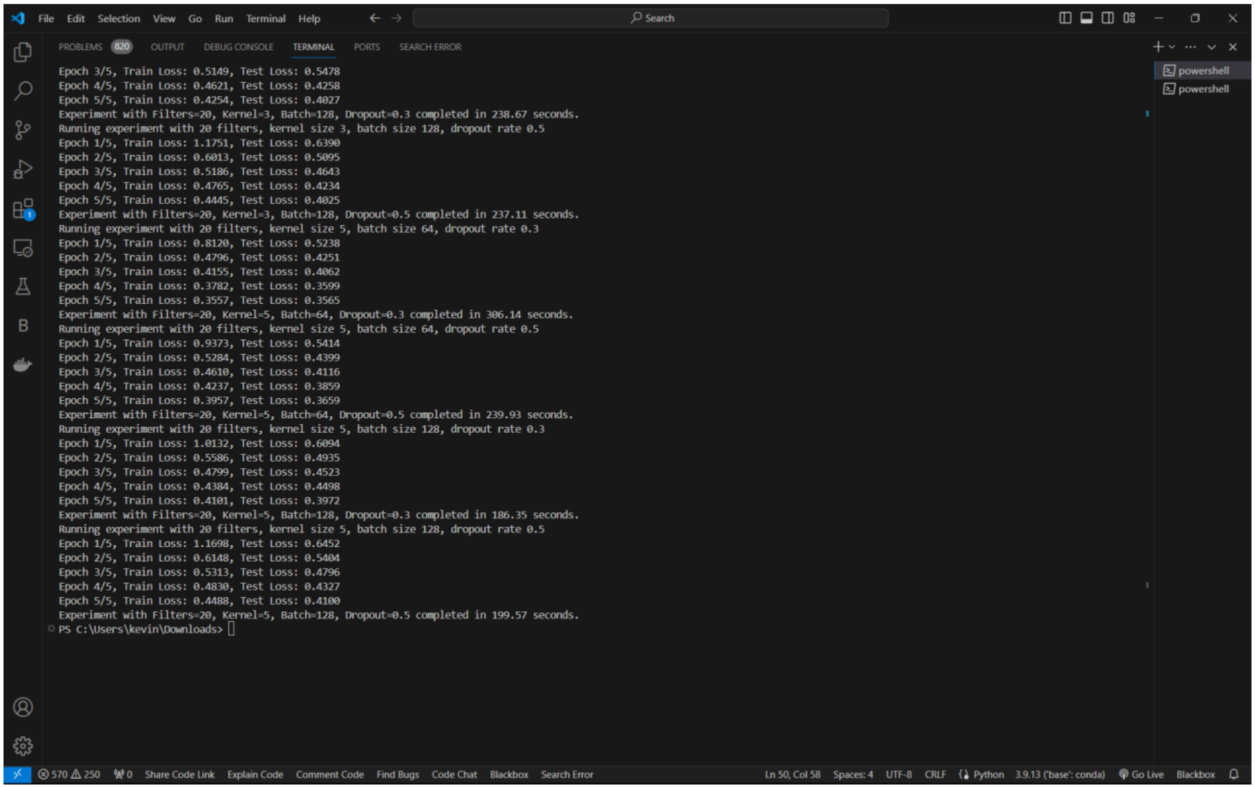


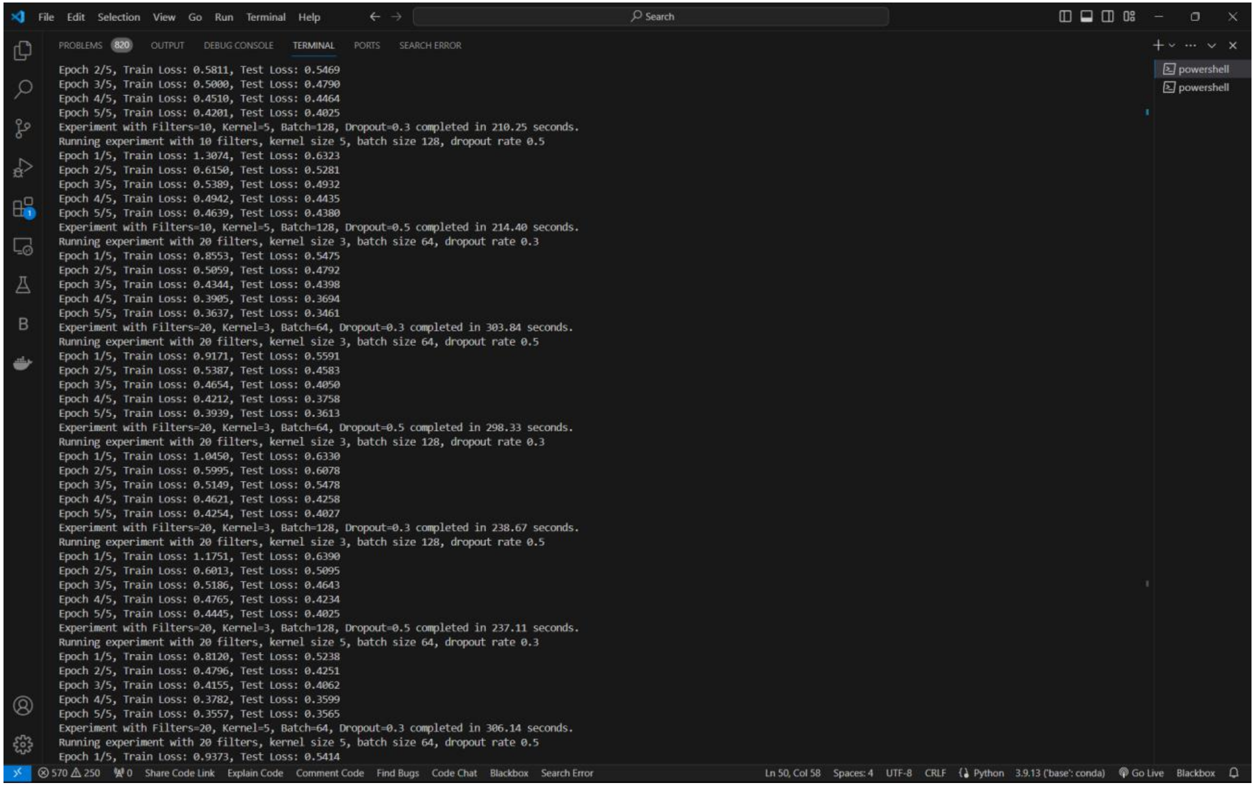
Figure 1
Losses: Filters=10, Kernel=3, Batch=64, Dropout=0.3
Train Loss
Test Loss
Loss
0.9
0.8
0.7
0.6
0.5
0.4
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
Epochs



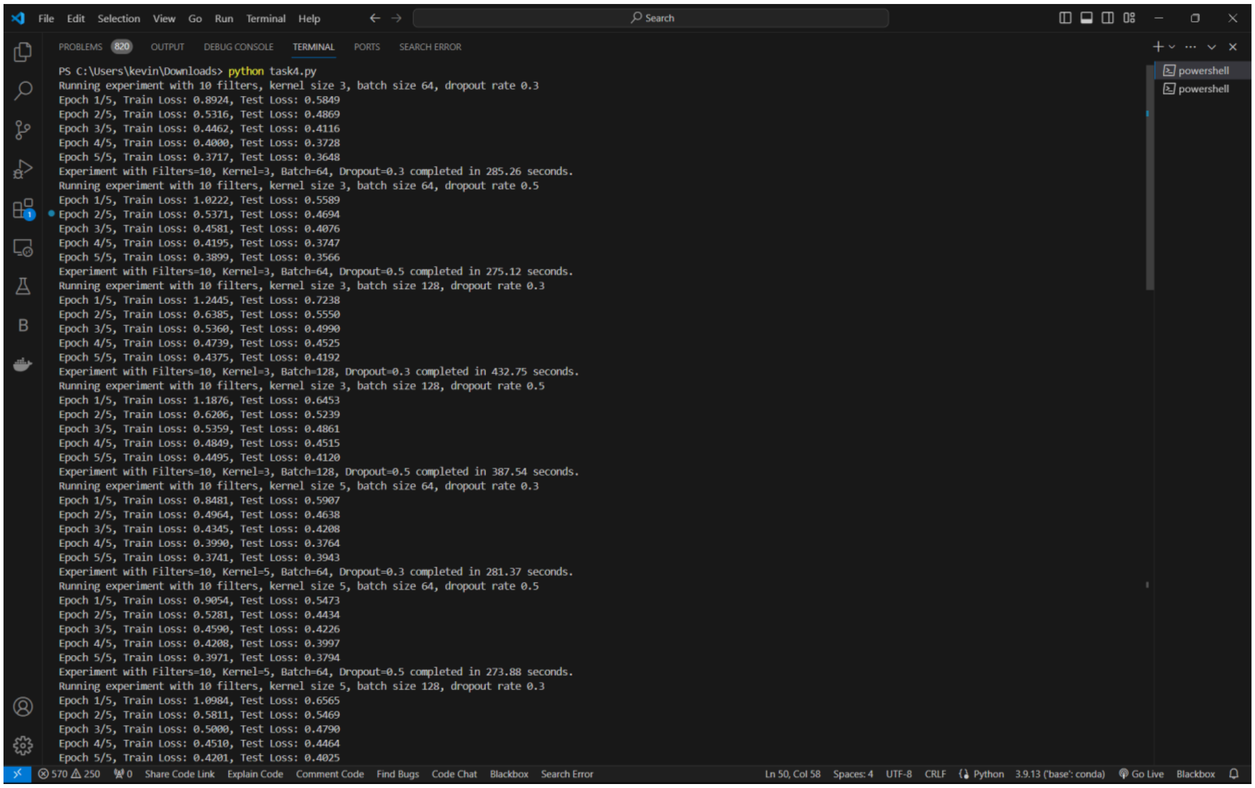
The corresponding time taken for the network to complete its execution the order format is the same as the order of the images



```
Epoch 3/5, Train Loss: 0.5149, Test Loss: 0.5478
Epoch 4/5, Train Loss: 0.4621, Test Loss: 0.4258
Epoch 5/5, Train Loss: 0.4254, Test Loss: 0.4027
Experiment with Filters=20, Kernel=3, Batch=128, Dropout=0.3 completed in 238.67 seconds.
Running experiment with 20 filters, kernel size 3, batch size 128, dropout rate 0.5
Epoch 1/5, Train Loss: 1.1751, Test Loss: 0.6390
Epoch 2/5, Train Loss: 0.6013, Test Loss: 0.5095
Epoch 3/5, Train Loss: 0.5186, Test Loss: 0.4643
Epoch 4/5, Train Loss: 0.4765, Test Loss: 0.4234
Epoch 5/5, Train Loss: 0.4445, Test Loss: 0.4025
Experiment with Filters=20, Kernel=3, Batch=128, Dropout=0.5 completed in 237.11 seconds.
Running experiment with 20 filters, kernel size 5, batch size 64, dropout rate 0.3
Epoch 1/5, Train Loss: 0.8120, Test Loss: 0.5238
Epoch 2/5, Train Loss: 0.4796, Test Loss: 0.4251
Epoch 3/5, Train Loss: 0.4155, Test Loss: 0.4062
Epoch 4/5, Train Loss: 0.3782, Test Loss: 0.3599
Epoch 5/5, Train Loss: 0.3557, Test Loss: 0.3565
Experiment with Filters=20, Kernel=5, Batch=64, Dropout=0.3 completed in 306.14 seconds.
Running experiment with 20 filters, kernel size 5, batch size 64, dropout rate 0.5
Epoch 1/5, Train Loss: 0.9373, Test Loss: 0.5414
Epoch 2/5, Train Loss: 0.5284, Test Loss: 0.4399
Epoch 3/5, Train Loss: 0.4610, Test Loss: 0.4116
Epoch 4/5, Train Loss: 0.4237, Test Loss: 0.3859
Epoch 5/5, Train Loss: 0.3957, Test Loss: 0.3659
Experiment with Filters=20, Kernel=5, Batch=64, Dropout=0.5 completed in 239.93 seconds.
Running experiment with 20 filters, kernel size 5, batch size 128, dropout rate 0.3
Epoch 1/5, Train Loss: 1.0132, Test Loss: 0.6094
Epoch 2/5, Train Loss: 0.5586, Test Loss: 0.4935
Epoch 3/5, Train Loss: 0.4799, Test Loss: 0.4523
Epoch 4/5, Train Loss: 0.4384, Test Loss: 0.4498
Epoch 5/5, Train Loss: 0.4101, Test Loss: 0.3972
Experiment with Filters=20, Kernel=5, Batch=128, Dropout=0.3 completed in 186.35 seconds.
Running experiment with 20 filters, kernel size 5, batch size 128, dropout rate 0.5
Epoch 1/5, Train Loss: 1.1698, Test Loss: 0.6452
Epoch 2/5, Train Loss: 0.6148, Test Loss: 0.5404
Epoch 3/5, Train Loss: 0.5313, Test Loss: 0.4796
Epoch 4/5, Train Loss: 0.4830, Test Loss: 0.4327
Epoch 5/5, Train Loss: 0.4488, Test Loss: 0.4100
Experiment with Filters=20, Kernel=5, Batch=128, Dropout=0.5 completed in 199.57 seconds.
PS C:\Users\kevin\Downloads> 
```



```
Epoch 2/5, Train Loss: 0.5811, Test Loss: 0.5469
Epoch 3/5, Train Loss: 0.5000, Test Loss: 0.4790
Epoch 4/5, Train Loss: 0.4510, Test Loss: 0.4464
Epoch 5/5, Train Loss: 0.4201, Test Loss: 0.4025
Experiment with Filters=10, Kernel=5, Batch=128, Dropout=0.3 completed in 210.25 seconds.
Running experiment with 10 filters, kernel size 5, batch size 128, dropout rate 0.5
Epoch 1/5, Train Loss: 1.3074, Test Loss: 0.6323
Epoch 2/5, Train Loss: 0.6150, Test Loss: 0.5281
Epoch 3/5, Train Loss: 0.5389, Test Loss: 0.4932
Epoch 4/5, Train Loss: 0.4942, Test Loss: 0.4435
Epoch 5/5, Train Loss: 0.4639, Test Loss: 0.4380
Experiment with Filters=10, Kernel=5, Batch=128, Dropout=0.5 completed in 214.40 seconds.
Running experiment with 20 filters, kernel size 3, batch size 64, dropout rate 0.3
Epoch 1/5, Train Loss: 0.8553, Test Loss: 0.5475
Epoch 2/5, Train Loss: 0.5059, Test Loss: 0.4792
Epoch 3/5, Train Loss: 0.4344, Test Loss: 0.4398
Epoch 4/5, Train Loss: 0.3905, Test Loss: 0.3694
Epoch 5/5, Train Loss: 0.3637, Test Loss: 0.3461
Experiment with Filters=20, Kernel=3, Batch=64, Dropout=0.3 completed in 303.84 seconds.
Running experiment with 20 filters, kernel size 3, batch size 64, dropout rate 0.5
Epoch 1/5, Train Loss: 0.9171, Test Loss: 0.5591
Epoch 2/5, Train Loss: 0.5387, Test Loss: 0.4583
Epoch 3/5, Train Loss: 0.4654, Test Loss: 0.4050
Epoch 4/5, Train Loss: 0.4212, Test Loss: 0.3758
Epoch 5/5, Train Loss: 0.3939, Test Loss: 0.3613
Experiment with Filters=20, Kernel=3, Batch=64, Dropout=0.5 completed in 298.33 seconds.
Running experiment with 20 filters, kernel size 3, batch size 128, dropout rate 0.3
Epoch 1/5, Train Loss: 1.0450, Test Loss: 0.6330
Epoch 2/5, Train Loss: 0.5995, Test Loss: 0.6078
Epoch 3/5, Train Loss: 0.5149, Test Loss: 0.5478
Epoch 4/5, Train Loss: 0.4621, Test Loss: 0.4258
Epoch 5/5, Train Loss: 0.4254, Test Loss: 0.4027
Experiment with Filters=20, Kernel=3, Batch=128, Dropout=0.3 completed in 238.67 seconds.
Running experiment with 20 filters, kernel size 3, batch size 128, dropout rate 0.5
Epoch 1/5, Train Loss: 1.1751, Test Loss: 0.6390
Epoch 2/5, Train Loss: 0.6013, Test Loss: 0.5095
Epoch 3/5, Train Loss: 0.5186, Test Loss: 0.4643
Epoch 4/5, Train Loss: 0.4765, Test Loss: 0.4234
Epoch 5/5, Train Loss: 0.4445, Test Loss: 0.4025
Experiment with Filters=20, Kernel=3, Batch=128, Dropout=0.5 completed in 237.11 seconds.
Running experiment with 20 filters, kernel size 5, batch size 64, dropout rate 0.3
Epoch 1/5, Train Loss: 0.8120, Test Loss: 0.5238
Epoch 2/5, Train Loss: 0.4796, Test Loss: 0.4251
Epoch 3/5, Train Loss: 0.4155, Test Loss: 0.4062
Epoch 4/5, Train Loss: 0.3782, Test Loss: 0.3599
Epoch 5/5, Train Loss: 0.3557, Test Loss: 0.3565
Experiment with Filters=20, Kernel=5, Batch=64, Dropout=0.3 completed in 306.14 seconds.
Running experiment with 20 filters, kernel size 5, batch size 64, dropout rate 0.5
Epoch 1/5, Train Loss: 0.9373, Test Loss: 0.5414
```



```
PS C:\Users\kevin\Downloads> python task4.py
Running experiment with 10 filters, kernel size 3, batch size 64, dropout rate 0.3
Epoch 1/5, Train Loss: 0.8924, Test Loss: 0.5849
Epoch 2/5, Train Loss: 0.5316, Test Loss: 0.4869
Epoch 3/5, Train Loss: 0.4462, Test Loss: 0.4116
Epoch 4/5, Train Loss: 0.4000, Test Loss: 0.3728
Epoch 5/5, Train Loss: 0.3717, Test Loss: 0.3648
Experiment with Filters=10, Kernel=3, Batch=64, Dropout=0.3 completed in 285.26 seconds.
Running experiment with 10 filters, kernel size 3, batch size 64, dropout rate 0.5
Epoch 1/5, Train Loss: 1.0222, Test Loss: 0.5589
Epoch 2/5, Train Loss: 0.5371, Test Loss: 0.4694
Epoch 3/5, Train Loss: 0.4581, Test Loss: 0.4076
Epoch 4/5, Train Loss: 0.4195, Test Loss: 0.3747
Epoch 5/5, Train Loss: 0.3899, Test Loss: 0.3566
Experiment with Filters=10, Kernel=3, Batch=64, Dropout=0.5 completed in 275.12 seconds.
Running experiment with 10 filters, kernel size 3, batch size 128, dropout rate 0.3
Epoch 1/5, Train Loss: 1.2445, Test Loss: 0.7238
Epoch 2/5, Train Loss: 0.6385, Test Loss: 0.5550
Epoch 3/5, Train Loss: 0.5360, Test Loss: 0.4990
Epoch 4/5, Train Loss: 0.4739, Test Loss: 0.4525
Epoch 5/5, Train Loss: 0.4375, Test Loss: 0.4192
Experiment with Filters=10, Kernel=3, Batch=128, Dropout=0.3 completed in 432.75 seconds.
Running experiment with 10 filters, kernel size 3, batch size 128, dropout rate 0.5
Epoch 1/5, Train Loss: 1.1876, Test Loss: 0.6453
Epoch 2/5, Train Loss: 0.6206, Test Loss: 0.5239
Epoch 3/5, Train Loss: 0.5359, Test Loss: 0.4861
Epoch 4/5, Train Loss: 0.4849, Test Loss: 0.4515
Epoch 5/5, Train Loss: 0.4495, Test Loss: 0.4120
Experiment with Filters=10, Kernel=3, Batch=128, Dropout=0.5 completed in 387.54 seconds.
Running experiment with 10 filters, kernel size 5, batch size 64, dropout rate 0.3
Epoch 1/5, Train Loss: 0.8481, Test Loss: 0.5907
Epoch 2/5, Train Loss: 0.4964, Test Loss: 0.4638
Epoch 3/5, Train Loss: 0.4345, Test Loss: 0.4208
Epoch 4/5, Train Loss: 0.3990, Test Loss: 0.3764
Epoch 5/5, Train Loss: 0.3741, Test Loss: 0.3943
Experiment with Filters=10, Kernel=5, Batch=64, Dropout=0.3 completed in 281.37 seconds.
Running experiment with 10 filters, kernel size 5, batch size 64, dropout rate 0.5
Epoch 1/5, Train Loss: 0.9054, Test Loss: 0.5473
Epoch 2/5, Train Loss: 0.5281, Test Loss: 0.4434
Epoch 3/5, Train Loss: 0.4590, Test Loss: 0.4226
Epoch 4/5, Train Loss: 0.4208, Test Loss: 0.3997
Epoch 5/5, Train Loss: 0.3971, Test Loss: 0.3794
Experiment with Filters=10, Kernel=5, Batch=64, Dropout=0.5 completed in 273.88 seconds.
Running experiment with 10 filters, kernel size 5, batch size 128, dropout rate 0.3
Epoch 1/5, Train Loss: 1.0984, Test Loss: 0.6565
Epoch 2/5, Train Loss: 0.5811, Test Loss: 0.5469
Epoch 3/5, Train Loss: 0.5000, Test Loss: 0.4790
Epoch 4/5, Train Loss: 0.4510, Test Loss: 0.4464
Epoch 5/5, Train Loss: 0.4201, Test Loss: 0.4025
```

Reflection:

The project gave me an opportunity to delve into the specifics of how to build and optimize convolutional neural networks (CNNs) for image recognition exercises with the MNIST and Fashion MNIST datasets. I then took the digit recognition model from the foundational model I had built earlier and adapted it for a completely different, more complex task: recognizing handwritten Greek letters. It has been journeying through important aspects of designing a neural network, e.g., how to choose and adjust layers, filters, and other hyperparameters so that, in turn, they can increase model accuracy and efficiency. Experimentation over several such configurations of networks shed light on the fine balance between model complexity and computational demand, hence elaborating on the significance of thoughtful architecture design. I have automated lots of systematic experiments with the network that evaluates many possible architectures and have learned much more about architectural choices and their effect on performance than if I had to do all of that by hand. This project further polished my knowledge of the principles and practices of computer vision and how important the pytorch library is important in both python and computer vision.

Acknowledgements & References:

- Computer Vision: Algorithms and Applications, 2nd ed.
- Pytorch Documentation
- Object Detection Made Easy: Nicolai Nielson, YouTube Channel