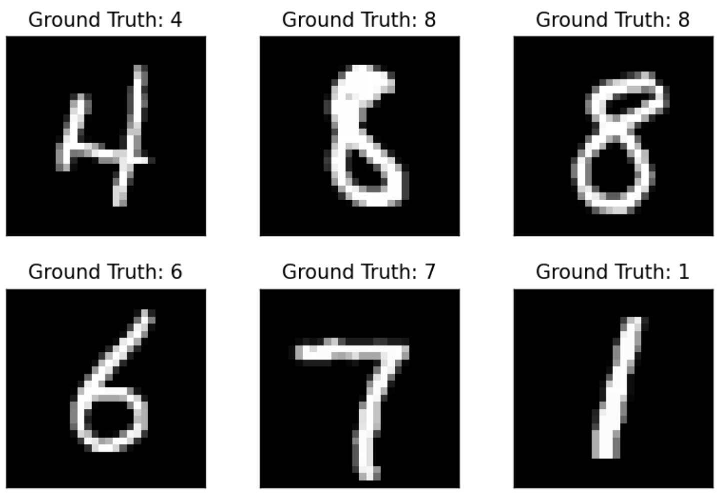
Project 5: Recognition using Deep Networks

By Avnish Patel

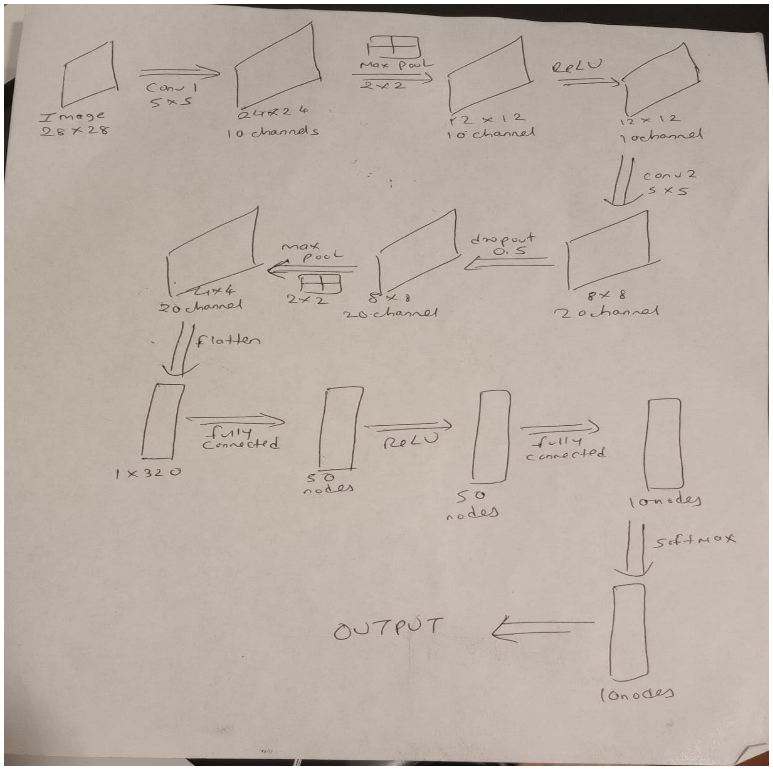
Summary

The aim of the project is to build, train, analyze, and modify a deep network for a recognition task. We will be working on different datasets like MNIST and MNIST fashion to train the model to recognize digits and fashion sets. Hyperparameter tuning along with transfer learning is also done to see various effects they have on accuracy of classifying.

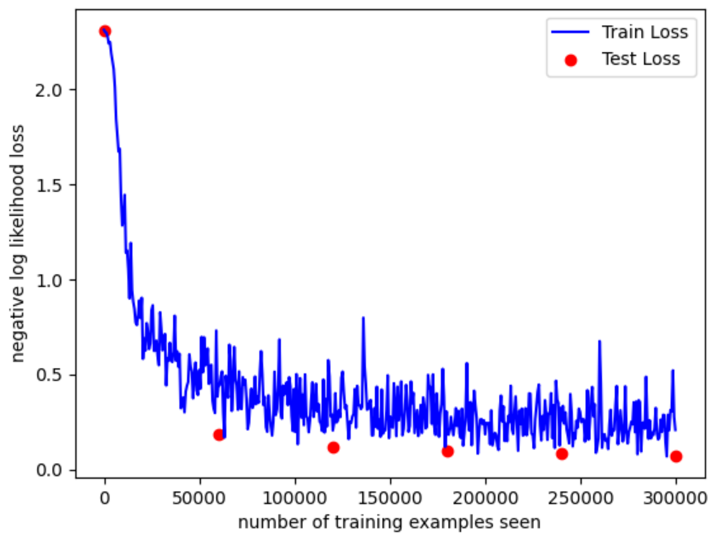
Required Image 1A – **Build and train a network to recognize digits**



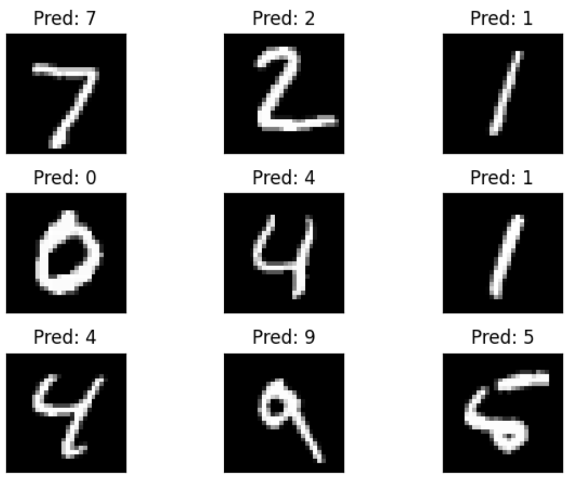
Required Image 1C – **Build a network model**



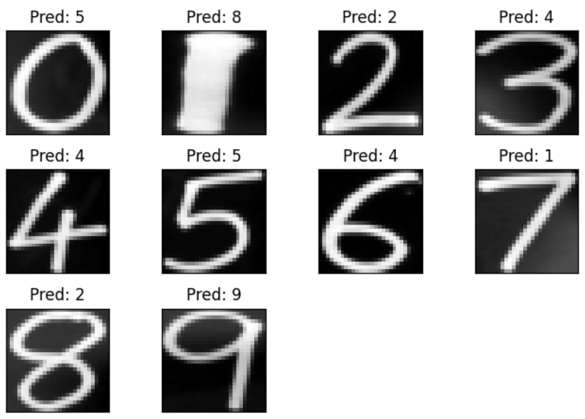
Required Image 1D - Train the model



Required Image 1F – Read the network and run it on the test set

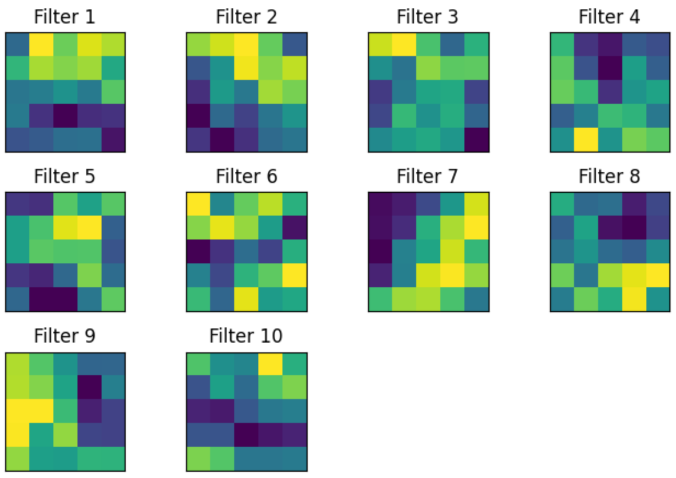


Required Image 1G – Test the network on new inputs

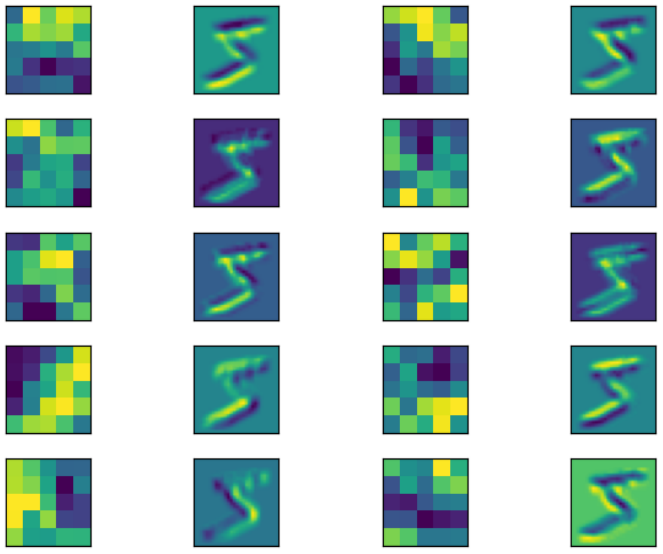


The network unexpectedly did not perform well maybe due to less training data making it difficult to recognize digits

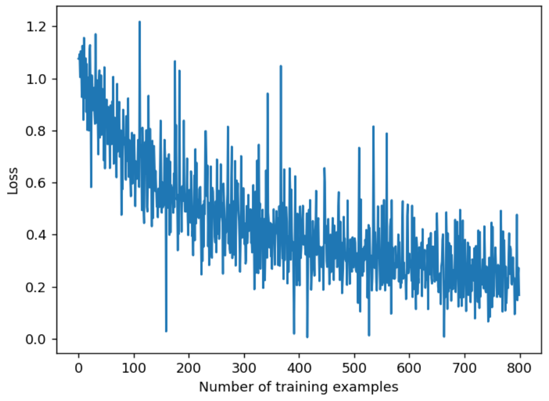
Required Image 2A – Analyze the first layer

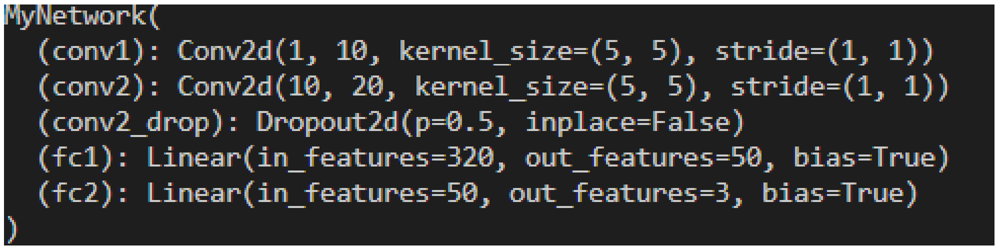


Required Image 2B – Show the effect of the filters.



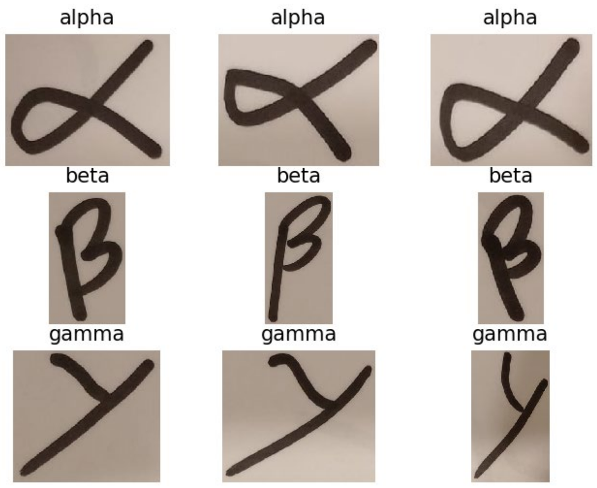
Required Image 3 – Transfer Learning on Greek Letters





```
MyNetwork(
  (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)
)
```

For classifying the 27 digits of alpha, beta, gamma it took around 100 epochs for 100% accuracy



The model was able to classify the custom digits accurately.

Task 4 – **Experiment with the Deep Networks**

For experimenting we will try a new dataset FashionMNIST which includes different categories for fashion wear like jeans, T-shirts, sandals etc. This dataset gives us more insight into the effects of the different parameters than the normal MNIST dataset.

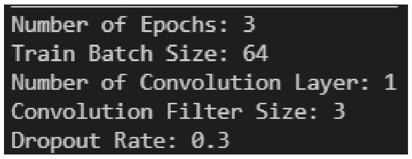
We will be testing our deep network on several different parameters like:

a) epoch sizes :3,5
b) training batch size: 64,128
c) number of Conv layer: 1,2,3
d) convolution layer filter size: 3,5,7
e) dropout rate: 0.3,0.5

My Hypothesis:

1) On increasing the epoch size the accuracy is also increasing as the weights and the layers will train more and the problem of overfitting will not occur in this dataset
2) When we increase the batch size, the accuracy also increases as more train data is given
3) When we increase the convolution layers, the accuracy also increases as more features are analyzed by the network
4) For a convolution layer filter the size 5 works the best as the size 3 and 7 should decrease the accuracy
5) Dropout rates help us reduce overfitting. Both works pretty well.

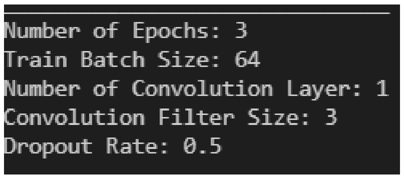
First Set



```
Number of Epochs: 3
Train Batch Size: 64
Number of Convolution Layer: 1
Convolution Filter Size: 3
Dropout Rate: 0.3
```

We get an accuracy of 84%

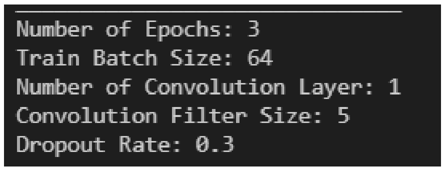
Second Set



```
Number of Epochs: 3
Train Batch Size: 64
Number of Convolution Layer: 1
Convolution Filter Size: 3
Dropout Rate: 0.5
```

We get accuracy of 83%. So, there is not much change in the dropout rates.

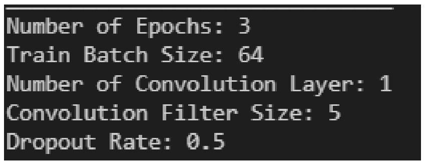
Third Set



```
Number of Epochs: 3
Train Batch Size: 64
Number of Convolution Layer: 1
Convolution Filter Size: 5
Dropout Rate: 0.3
```

We get accuracy of 85%. On increasing the filter size and comparing with the first set accuracy is similar.

Fourth Set



```
Number of Epochs: 3
Train Batch Size: 64
Number of Convolution Layer: 1
Convolution Filter Size: 5
Dropout Rate: 0.5
```

We get accuracy of 84%. On increasing the dropout rate and comparing with the third set, accuracy is similar

Therefore, we can conclude that the results may support the hypothesis. To be more conclusive we may have to train on many values for same parameter.

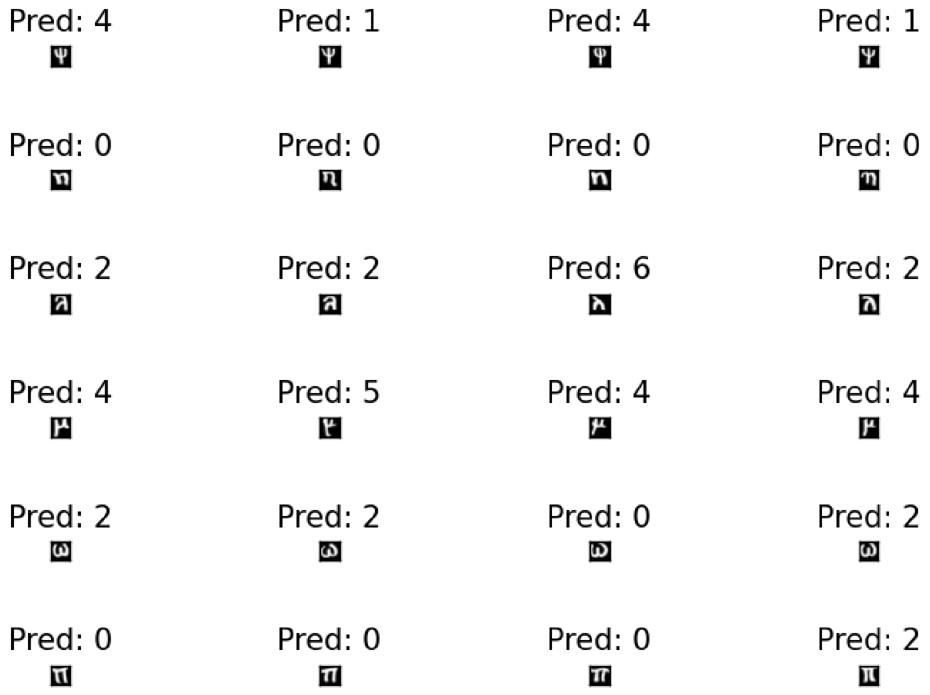
Extensions

1) Evaluate more dimensions on task 4

I have explained above that I have worked with 5 parameters

2) Try more Greek letters than alpha, beta, and gamma.

Using the Kaggle dataset of Greek letters, I have trained on different Greek letters like theta, pi, lambda etc.



The above image shows the predicted labels for different Greek characters. We can observe that for most of the characters, correct identification is taking place. For example, for the Greek letter ‘pi’, most of them are labeled 0 except 1 which has been labeled 2 which is for lambda

Reflection:

After learning the powerful effects of transfer learning, it makes sense that is widely used as we do not have to train the whole network again as it has already been trained by somebody else thus saving compute and time. This proved to be extremely effective in classifying data. I got to learn various hyperparameter tuning methods like Grid Search and Randomized Search which can become effective in improving the classification. All these concepts pave a new way in the field of Computer vision, and I hope to   learn more about them in the future.

Acknowledgements:

Pytorch Home page:
https://northeastern.instructure.com/courses/136693/assignments/pytorch.org

Deep learning blog: https://nextjournal.com/gkoehler/pytorch-mnist