CS5330: Pattern Recognition and Computer Vision

Project 5: Recognition using Deep Networks

Team Members: Krishna Prasath Senthil Kumaran & Gopisainath Mamindlapalli

Email: senthilkumaran.k@northeastern.edu & mamindlapalli.g@northeastern.edu

Aim of the Project:

We are using the MNIST digit recognition dataset and PyTorch to build, train and modify a deep network.

We are provided with four tasks. The first task guides us to build and train a deep network that recognizes digits. This task also helps us to get familiar with different types of layers such as the ***"convolution layer"***, "***max pooling layer"***, "***dropout layer"***, and "***fully connected layer"*** and along with this, some functions that are applied to these layers such as ***"ReLU function"*** and ***"log_softmax function"***. Then in the second task, we train the network and analyze it. In task three, we reuse the network that we trained to recognize digits to recognize three Greek letters: ***"alpha", "beta" & "gamma"***. Then at last in task 4, we choose a particular set of dimensions from the provided options, and we use the Fashion MNIST dataset as recommended. We also analyzed the least loss from the overall network variations.

Task 1: Build and train a network to recognize digits

In task 1, we built the deep network according to the dimensions provided. MNIST dataset has been used to train the network.

A) First six example digits:

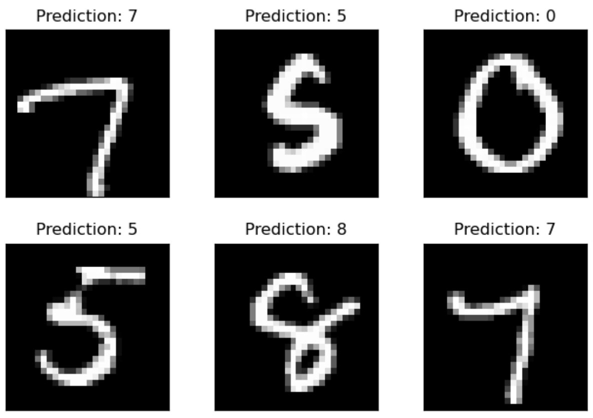


Fig.1 First six example digits.

B) Diagram of our network:

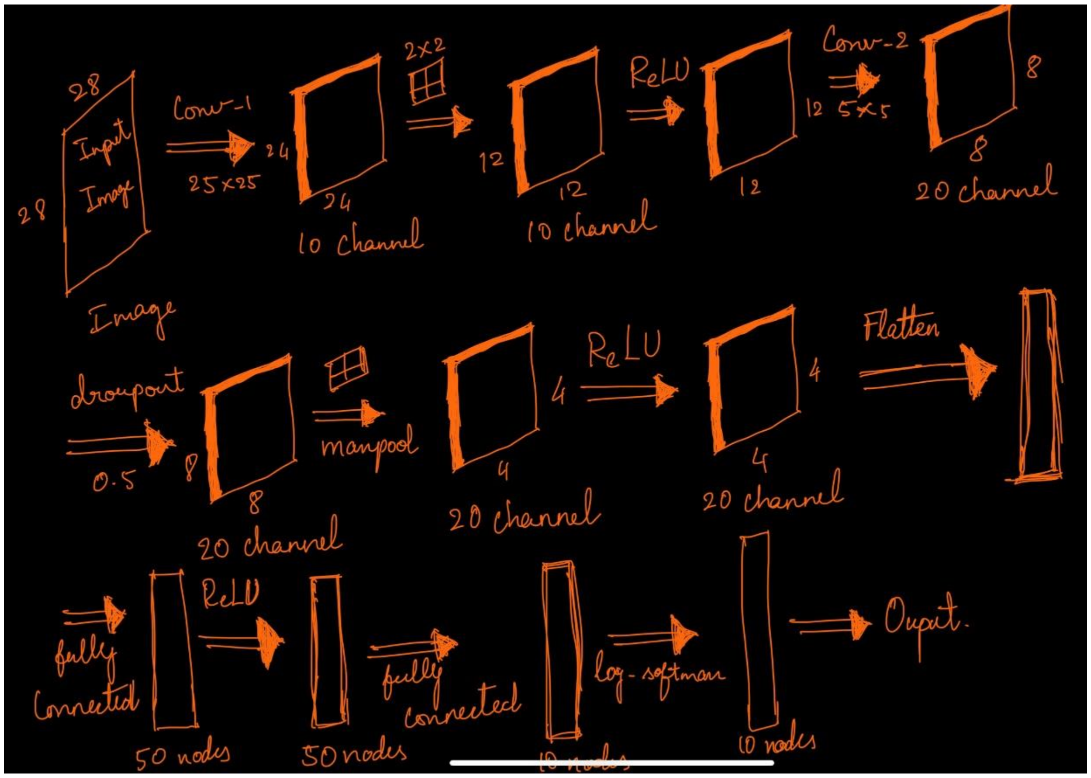


Fig.2 Deep Network sketch.

D) Training and testing accuracy scores graph.

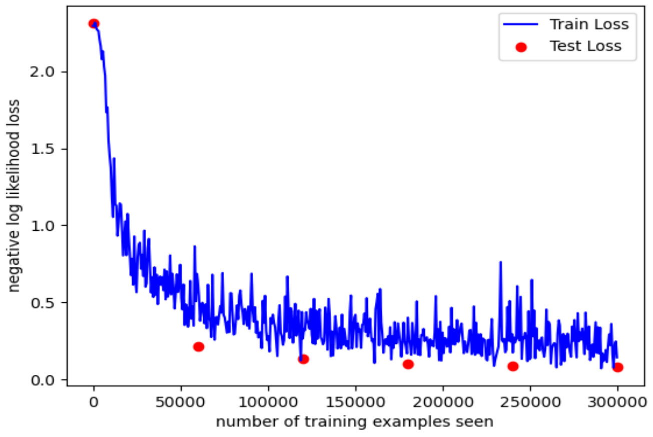


Fig.3 Training and test accuracy plot.

F) Read the network and run it on the test set.

The network gave a good prediction of digits, except for 4 and 6 where it was predicting number 4 as 6 and number 6 as 8 in a few cases.

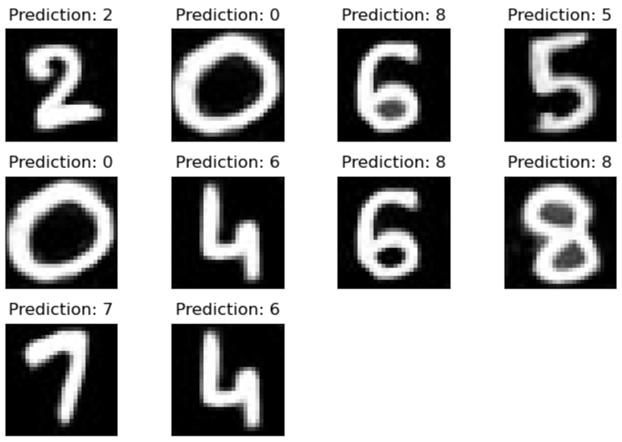


Fig.4 Prediction of first 9 digits.

G) Test the network on new inputs.

Based on the table plotted above the overall accuracy of the network is around 70%. The reduction in accuracy is due to differences in intensities.

Task 2: Examine your network

A) Analyze the first layer.

Visualization of ten filters using Pyplot.

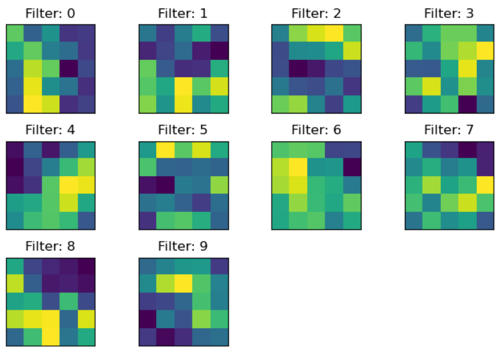


Fig.5 First 10 filters.

B) **Show the effect of the filter.**
Applying the 10 filters to the first example set of images without calculating the gradient.

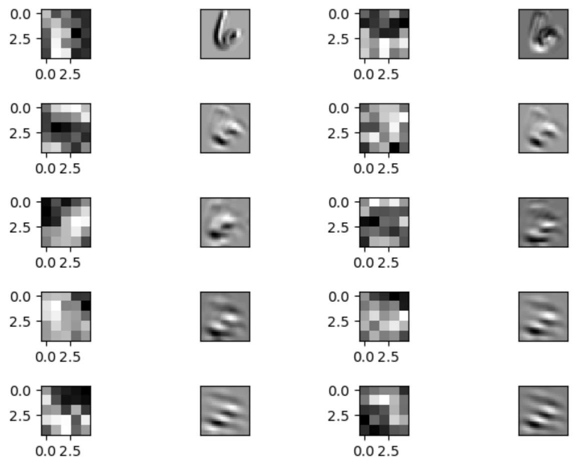


Fig.6 Applying filters on the example set of images.

Task 3: Transfer Learning on Greek Letters

In task 3, we freeze the weights gained from the network trained using MNIST and we change the last layer of the network to three nodes. We have used 500 epochs to train the network. The reason for using these many epochs is that the size of the dataset is very less for training.

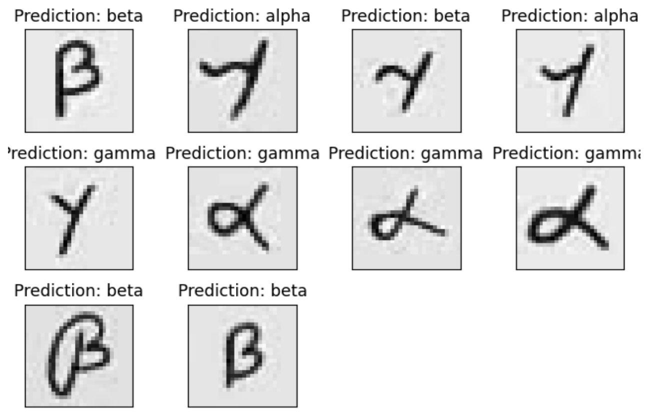


Fig.7 Hand-written Greek letters.

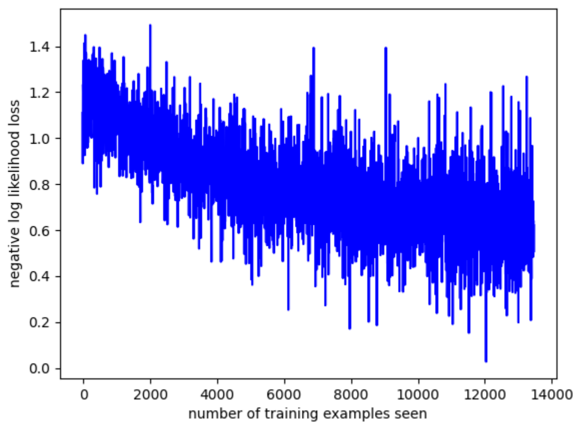
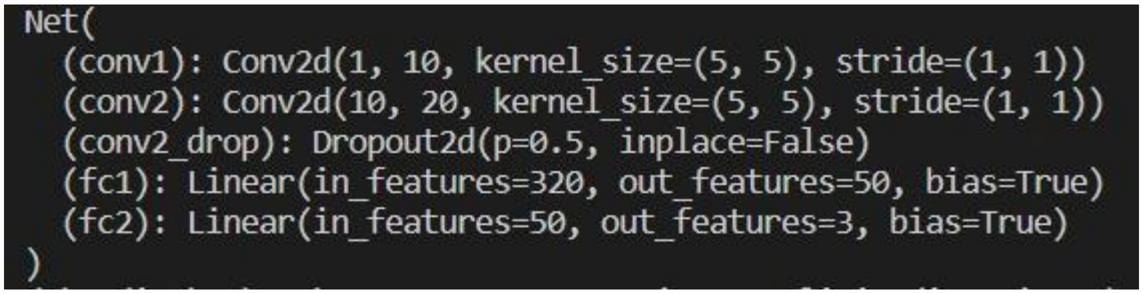


Fig.8 Calculated the training error for Greek letters dataset.



```
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)
)
```

Fig.9 Dimensions of the modified network.

Task 4: Design your own experiment

We have used 5 dimensions for this experimentation. We are evaluating 96 different network variations in this task. The dimension of the network are:

No of convolution layers:1,2;

Size of convolution filters: 2,3,4,5;

Activation function: ReLU, GeLU, Sigmoid;

Batch size: 64, 128;

No of epochs:3,5;

Hypothesis: The overall assumption for our experimentation is to try different variations of networks. We have developed the network to increase the overall network accuracy and reduce its loss. We know that when we increase the epoch size the accuracy of the network also increases as the weights get adjusted each time. We might overfit the model when we try to increase the number of epochs for the dataset and eventually the training error of the network should reduce if we have enough data in the dataset. We are not increasing the batch size for this model drastically, but we tried doubling the size, which should produce higher accuracy.

We have also increased the number of convolution layers and the kernel size to increase accuracy. ReLU activation function should give us accurate results in comparison to GeLU and Sigmoid.

Result: We were able to satisfy the stated hypothesis and matched our expectations. As the MNIST database is small we are not able to see significant differences.

(Note: The link to access all the plots is: https://drive.google.com/drive/folders/1cHgkme1BOblJKf2cGvaWPuhSppVOyHUM?usp=sharing)

Extension 1: Evaluation of more dimensions for Task 4

In this extension, we have evaluated more than 3 dimensions. We have combined this extension along with Task 4. Please verify task 4 for more details.

Extension 2: Evaluation of more dimensions for Task 4

In extension 2, we have reused the code from the 2D object detection project and have created a bounding box around the digits as displayed in the figure. We trained the bounding region using the ***"Resnet_18 model"***. We trained the model using the MNIST dataset which helped us identify the digits in real-time.

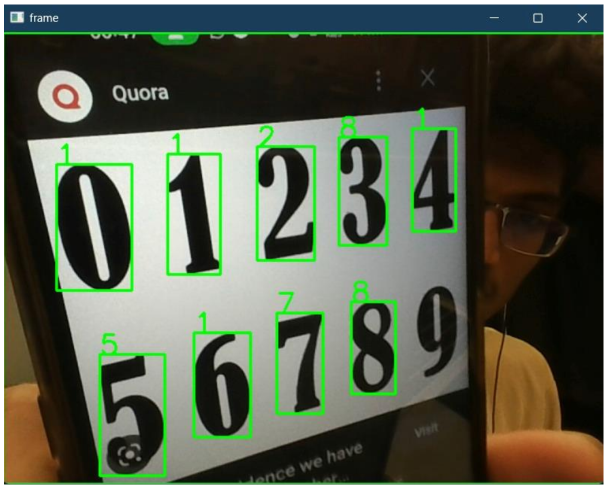


Fig.10 Training of Resnet model to recognize digits

Acknowledgement of resources or people consulted for this project:

- OpenCV Tutorials: https://docs.opencv.org/4.5.1/index.html
- MNIST tutorial: https://nextjournal.com/gkoehler/pytorch-mnist
- Camera calibration functions: https://docs.opencv.org/3.4/d9/d0c/group__calib3d.html#ga549c2075fac14829ff4a58bc931c033d