Project 5: Recognition using Deep Networks

by Yixiang Xie for CS5330 CVPR

Project Description

This is the fifth project for the class CS5300 Pattern Recognition & Computer Vision. The project is about how to build, train, analyze, and modify a deep network for a recognition task.

The project can be divided into four tasks.

In task one, we need to build and train a network to do digit recognition using the MNIST dataset, and save the trained network to a file. We then look at the network 10 output values of each label for the first 10 test dataset examples. Finally, we test the network on our own hand written digits.

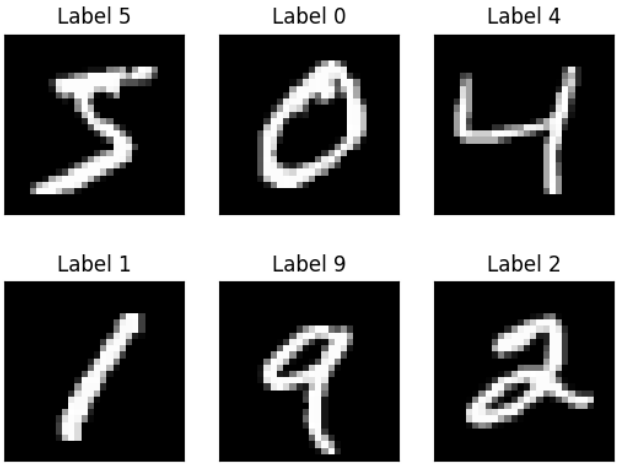
In task two, we need to visualize the 10 filters from the first layer from the previously built network and apply those filters to an example image to see the effect of each of the filters.

In task three, we need to re-use the previously built network to recognize greek letters by freezing the network weights, replacing the last classification layer and retraining this layer with greek letter data.

In task four, we need to design an experiment to evaluate the effect of changing different aspects of the network, such as number of convolution layers and size of the filters.

Required Images

1. Plot of the first six example digits from MNIST



Plot of the first six images from MNIST train dataset

2. Diagram of the network

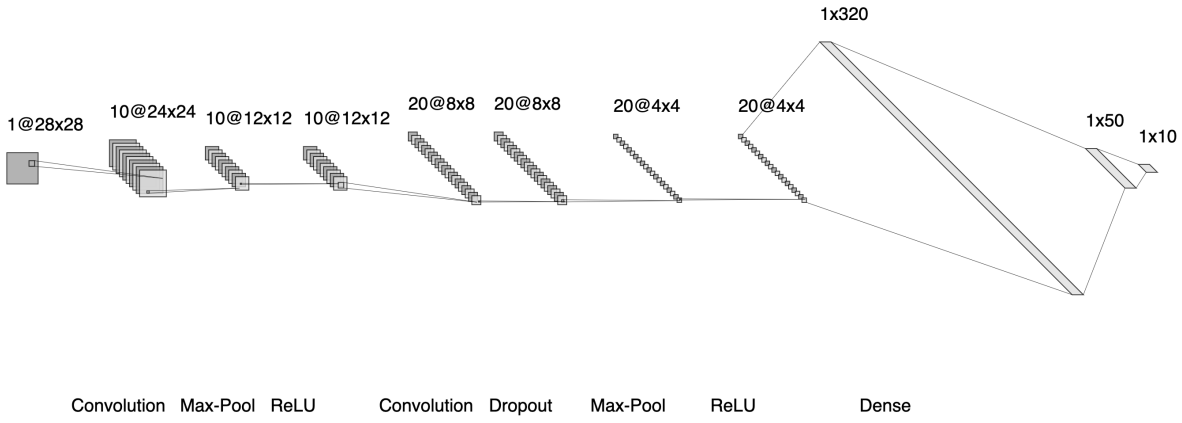
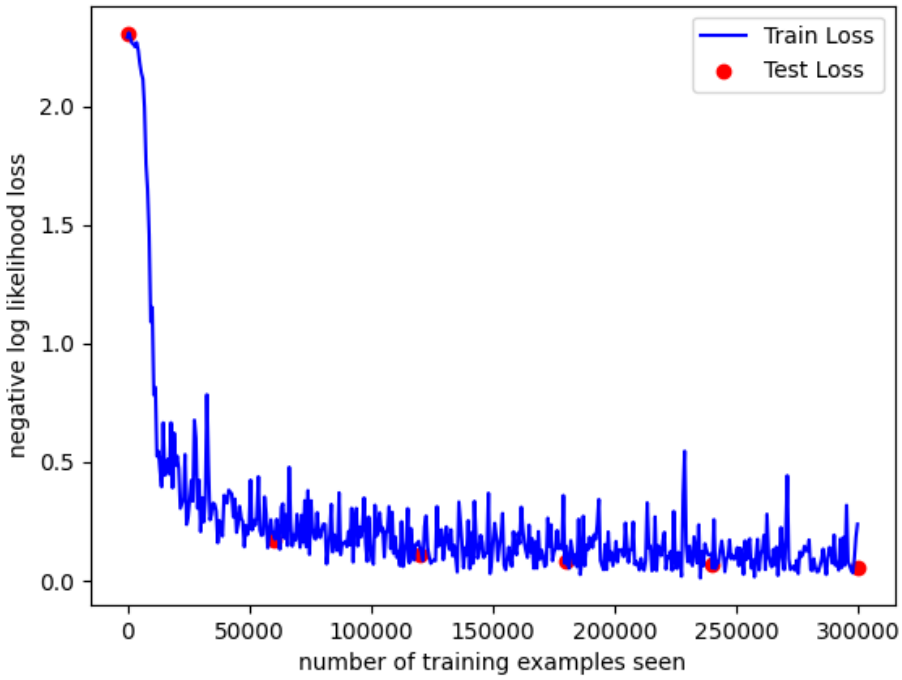


Diagram of the CNN

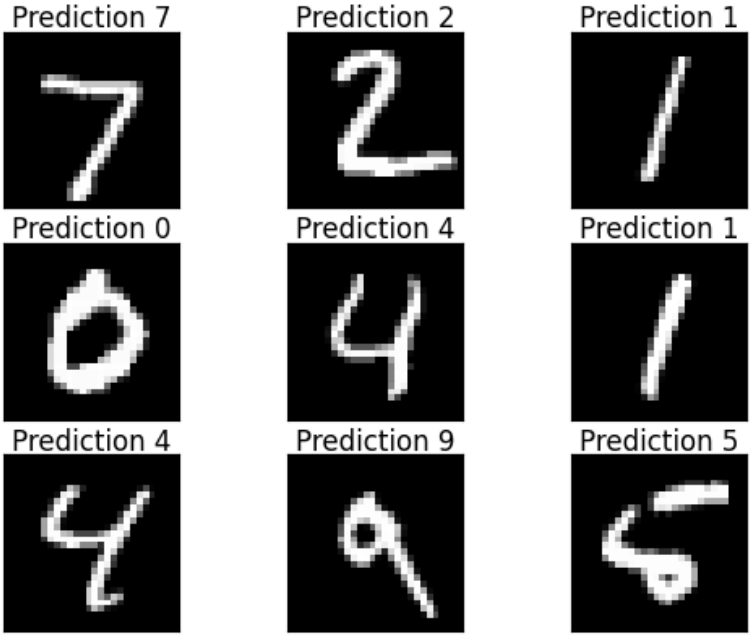
3. Plot of the train and test error



Plot of the train (shown in blue line) and test (shown in red dots) error for five epochs

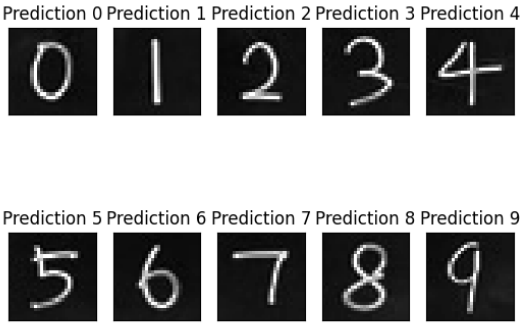
4. Test model on first ten examples from MNIST test set

		Network output values for each label; highlighted value has prediction index									
		0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
Actual labels	**7**	-20.51	-23.16	-10.82	-11.6	-25.29	-21.53	-37.75	0.0	-20.13	-14.17
	2	-10.6	-12.34	0.0	-16.82	-19.57	-18.05	-14.05	-23.01	-13.65	-26.42
	1	-13.02	0.0	-6.95	-9.51	-8.81	-11.64	-11.41	-8.15	-7.48	-11.66
	0	0.0	-24.5	-11.35	-15.96	-24.01	-12.31	-10.7	-16.31	-13.9	-12.77
	4	-13.87	-21.31	-12.45	-13.57	0.0	-11.96	-14.18	-13.16	-12.64	-5.45
	1	-15.48	0.0	-8.9	-10.31	-9.89	-15.4	-15.41	-8.21	-9.36	-12.66
	4	-25.67	-16.19	-16.45	-14.48	0.0	-11.39	-26.92	-10.37	-9.07	-5.67
	9	-20.25	-16.05	-11.27	-8.24	-4.82	-8.69	-22.71	-8.14	-7.8	-0.01
	5	-14.88	-26.86	-15.0	-17.97	-17.55	0.0	-8.79	-17.76	-8.45	-9.86
	9	-20.8	-26.62	-19.57	-12.11	-10.69	-11.71	-29.53	-6.38	-11.44	0.0



Plot of the first 9 digits from MNIST test dataset with their predictions

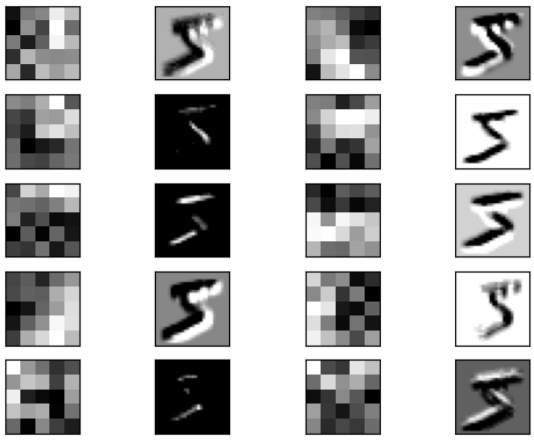
5. Test model on ten new handwritten examples



Plot of the 10 new handwritten example with their predictions; accuracy is 10/10

6. Effect of 10 filters from the first layer

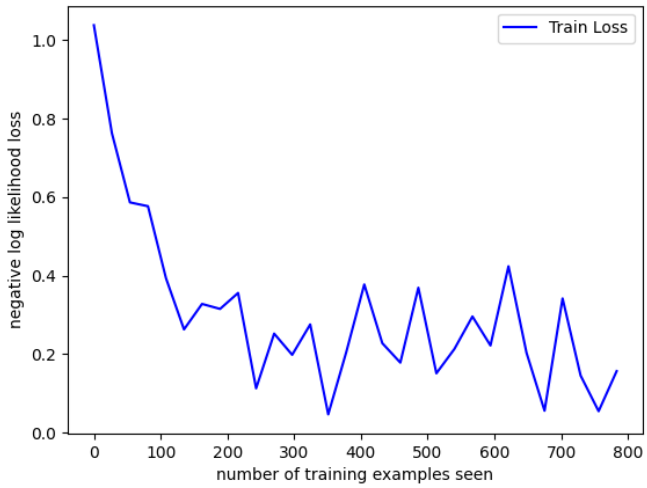
Applying the 10 filters from the first layer to the first training example image yields the following result. The result makes sense. For example, filter 1 is dark on left side and bright on right side. This looks like and has the effect of a Sobel X filter. For the corresponding image, we can see that the edge features are captured in that left side of vertical edges are dark and right side are bright. The similar result is observed in filter 8, where the result is also like Sobel X and just horizontally flipped. Filter 6 and 10 are like Sobel Y, but vertically flipped. Image applied filter 6 has horizontal edges dark on the top side and bright on the bottom side. Applying filter 10 gets the opposite result. Filter 2 and 7 are like Sobel filters, but rotated 45 degrees. The effect reveals diagonal edges (for example, result of filter 2 is dark in top right side of the diagonal edges and bright in bottom left side).



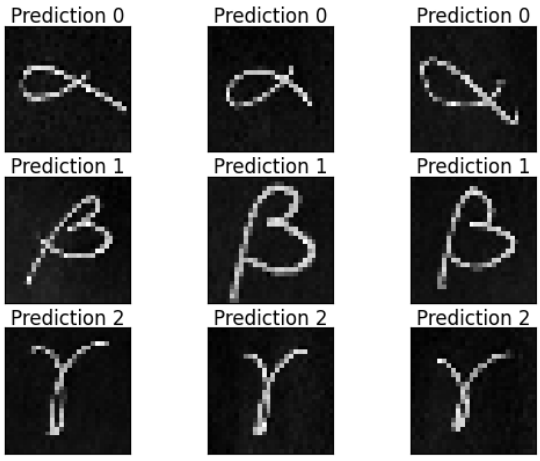
Plot of the 10 filters from the first layer and their effect on an example image

7. Transfer the previous network to Greek letters

The updated network has the following structure:
Net(
  (conv1): Conv2d(1, 10, kernel_size=(5, 5), stride=(1, 1))
  (conv2): Conv2d(10, 20, kernel_size=(5, 5), stride=(1, 1))
  (dropout): 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)
).
It takes at least 20 epochs to almost perfectly identify them. The curve gets flat after 20 epochs.



Plot of the training error for the greek data over 30 epochs



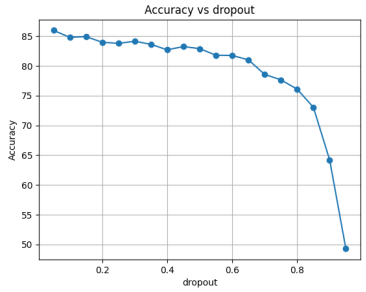
Plot of testing the network on the additional handwritten data; accuracy is 9/9

8. Evaluate the effect of changing aspects of the network

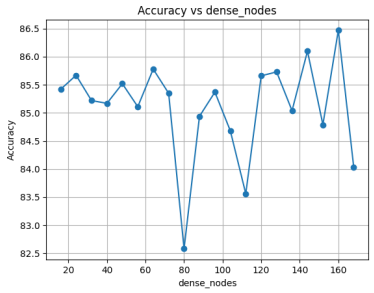
The metrics to explored are dropout rate for the dropout layer after the second convolution layer, the number of dense nodes for the first fully connected linear layer and the training batch size. The plan is to hold two parameters constant and optimize the third, and then switch the parameters up. Initially, the dropout rate is set as 0.5; number of dense nodes is set as 50; batch size is set as 64. These initial values are set based on experience. The range for dropout rate is from 0.05 to 1, step of 0.05, 20 values in total. The range for number of dense nodes is from 16 to 176, step of 8, 20 values in total. The range for dropout rate is from 4 to 80, step of 4, 20 values in total. The linear search will result in 60 network variations in total. The experiment is conducted in automated manner.

I expected the accuracy to go up and reach peak and then go down as each parameter changes from low to high. This hypothesis is based on that the tutorial recommended these values for each of these parameters and they are sort of in the middle of the range. Another hypothesis is the behavior follows Gaussian distribution. The result from the experiment doesn't support these hypothesis. The result is as the following.

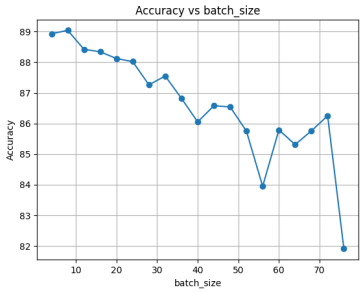
As dropout rate goes up, the accuracy goes down. The peak accuracy is obtained when dropout rate is set as 0.05. As dropout number of dense nodes goes up, the accuracy goes down and then goes up. The peak accuracy is obtained when number is set as 160. As batch size goes up, the accuracy goes down, and then goes up for a bit, and eventually goes to the bottom. The peak accuracy is obtained when batch size is set as 8. The relationship between the parameters and the accuracy is plotted as the following.



Plot of the accuracy when dropout rate changes



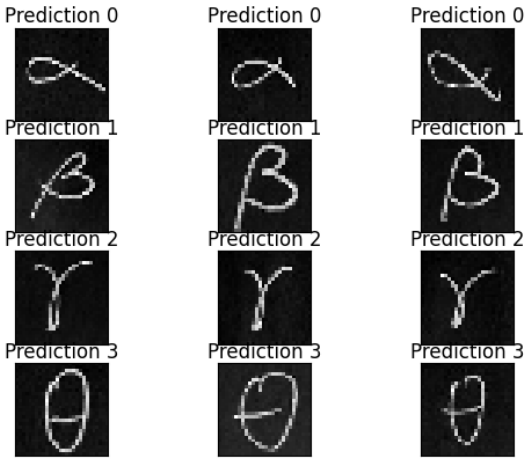
Plot of the accuracy when number of dense nodes changes



Plot of the accuracy when training batch size changes

Extensions

1. More greek letters



Plot of testing the network on more greek letters

Reflections

In this project, I learned about how to build, train, analyze, and modify a deep network for a recognition task. I learned about how to use PyTorch and related libraries for building a deep network. I got to visualize the filters in the network and this helps me to understand how CNN works. I also tried transfer learning. I also get to experiment how parameters would affect the performance of a network.

Acknowledgments

Professor Bruce Maxwell, for the the starter code.
PyTorch tutorial for building a network. Link: https://pytorch.org/tutorials/beginner/basics/intro.html
Gregor Koehler for MNIST network tutorial. Link: https://nextjournal.com/gkoehler/pytorch-mnist