ECE 5332 – Machine Learning

Dr. Hamed Sari-Sarraf

Project 6

Digits Recognition using Convolutional Neural Network

By

An Tran

Bo Pei

Texas Tech University

May 14th, 2018

a) Network architecture and complexity (in terms of number of learnable parameters)

```
% Define CNN architecture
layers = [...
    imageInputLayer([28 28 1])

    convolution2dLayer(5,8)             % Filter 1: 8 filters size 5x5
    batchNormalizationLayer
    reluLayer

    maxPooling2dLayer(2,'Stride',2)     % Pooling layer: 2x2 with Stride 2

    convolution2dLayer(5,10)            % Filter 2: 16 filters size 5x5
    batchNormalizationLayer
    reluLayer

    maxPooling2dLayer(2,'Stride',2)     % Pooling layer: 2x2 with Stride 2

    fullyConnectedLayer(10)             % Fully connected network layer
    softmaxLayer
    classificationLayer];               % Softmax layer with 10 classes
```

Figure 1. Network architecture and layers

#	Name	Size	Learnable parameters
1	Input	28x28x1	**0**
2	Conv2d1 (Filter 1)	$\frac{28-5}{1}+1=24 \rightarrow$ **24x24x8**	(5x5)x8 + 8 = **208**
3	Maxpool1	$\frac{24-2}{2}+1=12 \rightarrow$ **12x12x8**	**0**
4	Conv2d2 (Filter 2)	$\frac{12-5}{1}+1=8 \rightarrow$ **8x8x10**	(5x5x8)x10 + 10 = **2010**
5	Maxpool2	$\frac{8-2}{2}+1=4 \rightarrow$ **4x4x10**	**0**
6	Fully Connected	4x4x10 = **160**	(160x10)+10 = **1610**

- Total learnable parameters: 208 + 2010 + 1610 = **3828** parameters

b) Training time

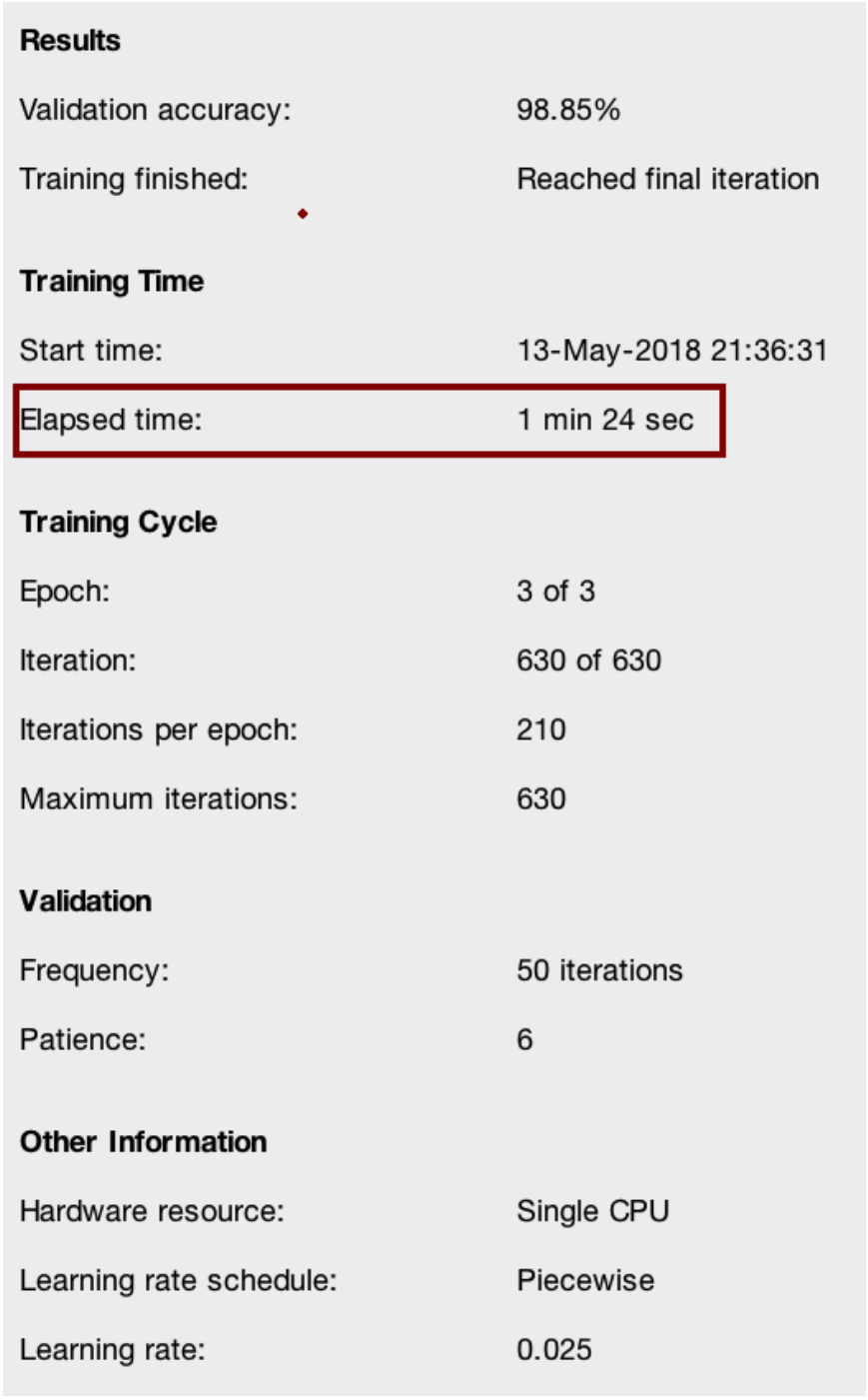

Figure 2. Training times and Validation accuracy

c) Training and testing accuracies

Epoch	Iteration	Time Elapsed (hh:mm:ss)	Mini-batch Accuracy	Validation Accuracy	Mini-batch Loss	Validation Loss	Base Learning Rate
1	1	00:00:01	1.56%	9.40%	2.3425	2.2113	0.1000
1	50	00:00:10	96.09%	95.34%	0.1261	0.1557	0.1000
1	100	00:00:16	96.09%	97.24%	0.1383	0.0961	0.1000
1	150	00:00:23	96.48%	97.72%	0.0885	0.0774	0.1000
1	200	00:00:30	97.27%	97.69%	0.0937	0.0782	0.1000
2	250	00:00:36	98.44%	98.29%	0.0713	0.0579	0.0500
2	300	00:00:42	99.61%	98.27%	0.0224	0.0554	0.0500
2	350	00:00:48	98.05%	98.52%	0.0513	0.0508	0.0500
2	400	00:00:53	97.66%	98.47%	0.0692	0.0503	0.0500
3	450	00:00:59	100.00%	98.59%	0.0190	0.0466	0.0250
3	500	00:01:05	98.44%	98.73%	0.0353	0.0436	0.0250
3	550	00:01:11	98.83%	98.78%	0.0475	0.0409	0.0250
3	600	00:01:17	99.22%	98.82%	0.0382	0.0408	0.0250
3	630	00:01:20	97.66%	98.70%	0.0944	0.0427	0.0250

```
test_accuracy =

   98.7100
```

Figure 3. Testing and training accuracies

d) Brief description of live demo

- Image of each digits are taken using camera and saved as ‘jpg’ file and put in the same folder as the m files. Read the image into console, and using classify() function that was trained on the network to classify the actual hand written digits.
- There is a function named ‘process.m’ associated with the program. It is capable of reading in an image and process it so that the original image has the same format as the MNIST dataset which the model was trained on.

- First, the original image is converted to gray scale, and then the number is cropped from the redundant background using ‘largest connected component’, box that region surrounding the image and then resize it to 28x28 just like the MNIST dataset.
- From here, the model can classify the digits.

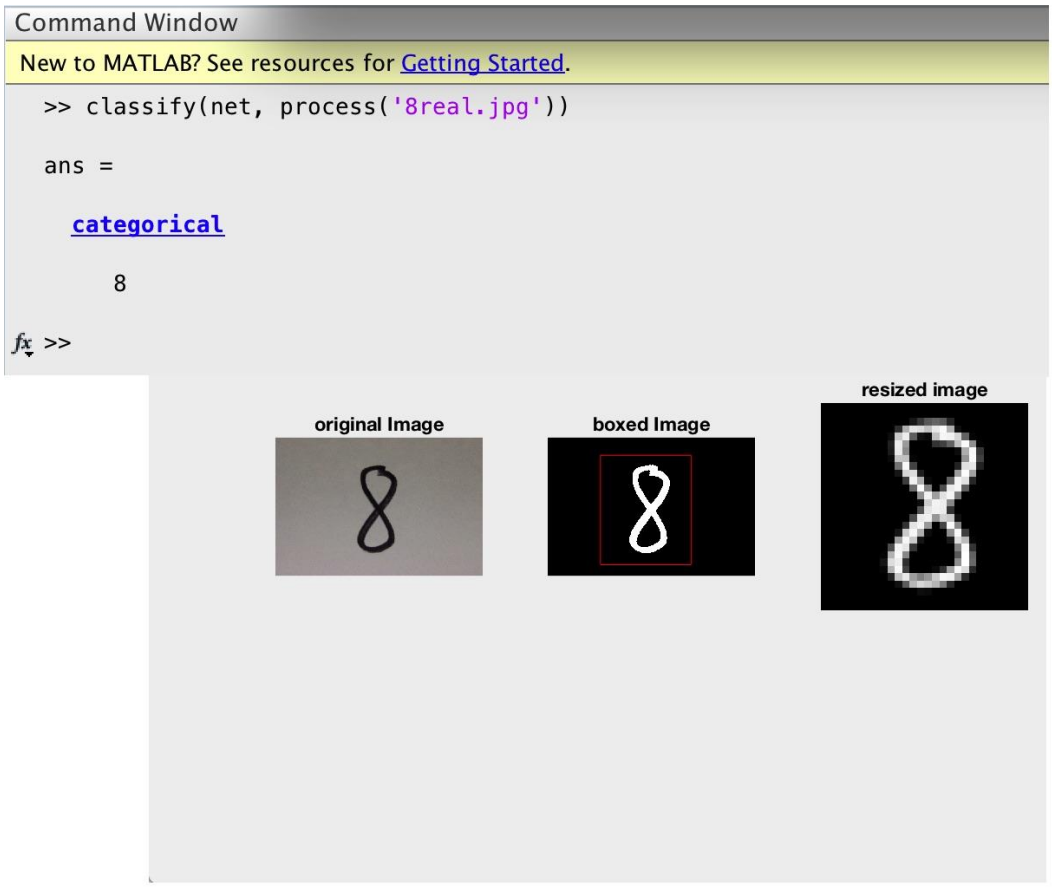

Figure 4. Demo