USC Viterbi
School of Engineering

Characters and Digits Recognition Using Neural Networks

Yiyue Zhang

USC Viterbi
School of Engineering

Abstract

- **Motivation**: Using Neural Networks to perform digits and characters recognition on DSP Board.

- **Goal**: Allowing user to write characters and digits on paper and display the printed version of them on a monitor in a relative short time.

- **Application**: Conversion from handwritten characters to printed forms.

USC Viterbi School of Engineering

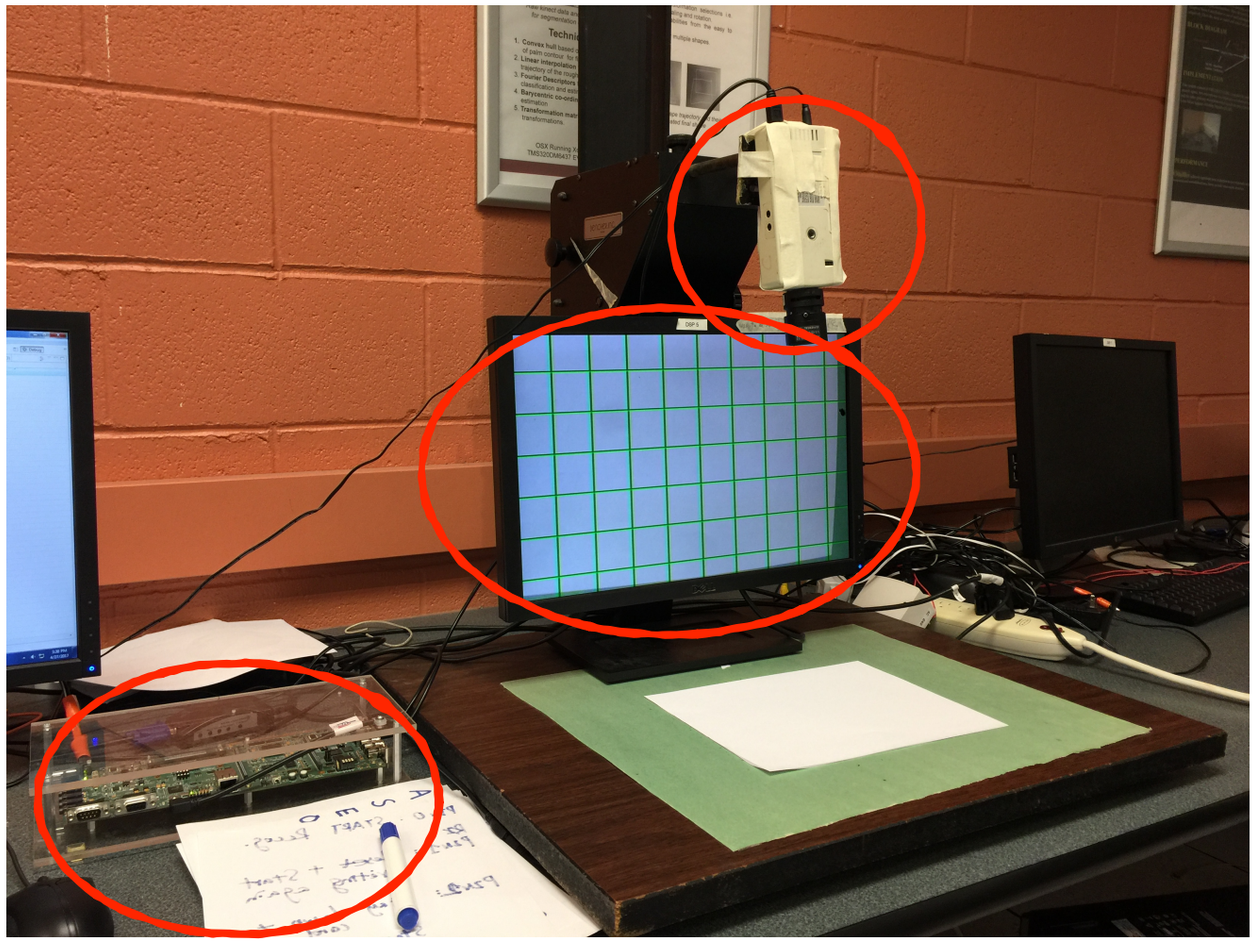
Description of System

- **System consists of:**

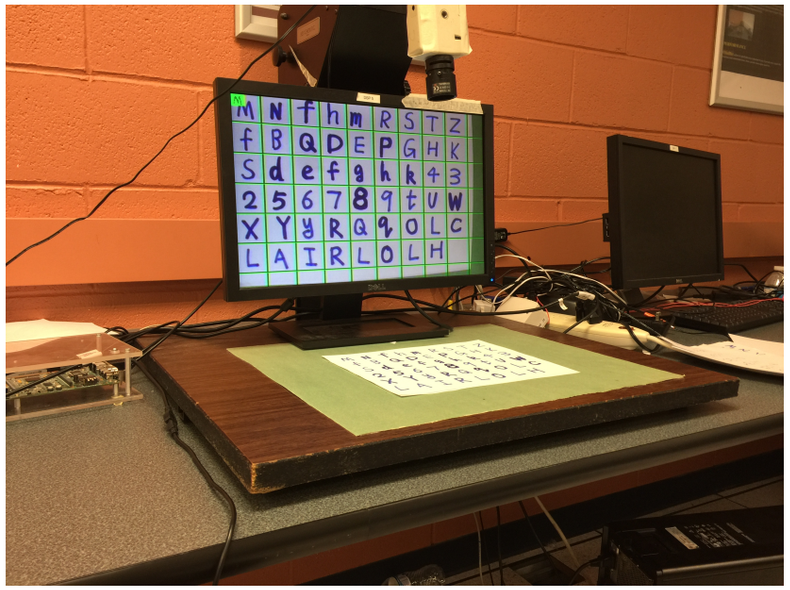
  Camera

  DSP Board

  Monitor



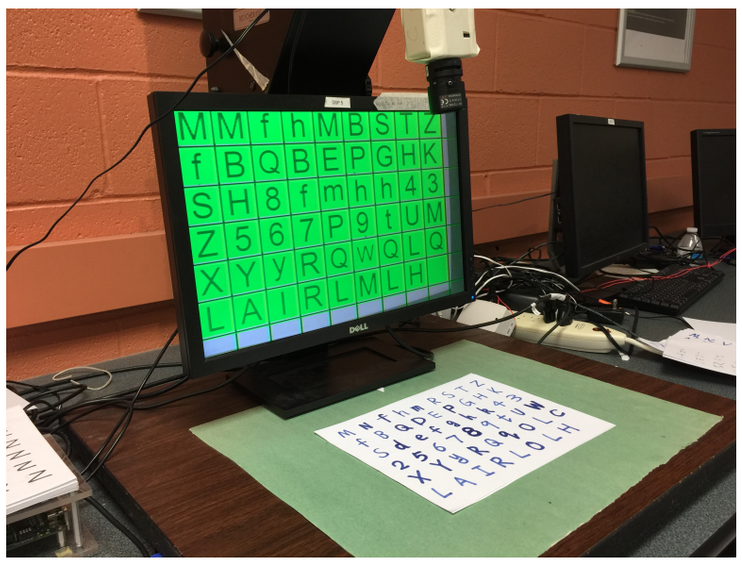
Final Test



Write inside grids on Monitor

PIN1: Start the Recognition Process
PIN2: Reset the System
PIN3: Start/Pause the Recognition Process
PIN2+3: Start Writing Process

Accuracy: 39/54



Modeling Tasks

Char74K Dataset	Data Collection
• Hand-written characters • 55 samples per class • 62 classes(a-z,A-Z,0-9)	• Data Collected from camera • 54 samples per class • 63 classes (a-z,A-Z,0-9 and White)

USC Viterbi School of Engineering

Data Processing in final network

Step1: RGB to Gray Scale

Step2.0: Gray Scale to Black-White Scale

Step2.1: Using Balanced histogram thresholding Method

Step 2.2: Process images with threshold percentages 16%,17%,18%,20%

(55+54)*63*4 = 27468 samples (for newest CNN)

USC Viterbi School of Engineering

What I did

What I did	Tranning Samples	Test Samples	Problems	Accuracy
Step1: Training with Softmax	All samples Char74k	10 samples per class	Overlapping Train and Test	Above 90%
Step2: Training with 1 Convolutional Layer	All samples Char74k	10 samples per class	Overlapping Train and Test	Above 80%
Step3: Separate Training and Testing samples	45 samples per class	10 samples per class	Low accuracy	Below 60%
Step4: Using 2 layers of Conv, MaxPool, and Fully connected, and 1 layer softmax	45 samples per class	10 samples per class		Around 70%
Step5: Converting TensorFlow functions to C function (Conv2d, Maxpool, FC, and Softmax)	45 samples per class	10 samples per class	Inaccurate on DSP	

USC Viterbi School of Engineering

What I did

What I did	Samples	Problems	Accuracy
Step6: Data Collection	No preprocessing	Depends on environment Low Accuracy	
Step7: Balanced histogram thresholding Method	Char74k and collected data With fixed threshold percentage 20%	Better than before	72% TF
Step8: Train the network in Step 4 again	Char74k and collected data With fixed threshold 16%,17%,18%, 20%	Weights and Bias saving	88% TF
Step9: DSP testing	Direct Camera Input	Ramdom Output	Low Accuracy on DSP
Step10: Debug network on C	Direct Camera Input	Improved	Lower than expected

USC Viterbi School of Engineering

What I did

What I did	Samples	Problems	Accuracy
Step10: Compare network in C with TF	Char74k and Collected Data Sample performance in C and TF	Cannot reach 88% accuracy	
Step11: Focusing on Final Test	Direct Camera Input And Pins (1-3) Produce character displays on monitor Voting System(Slow Down Recognition Speed)		
Step12: Save weights in Numpy formate	Failed on demo day		

Overall: The performance on DSP can get around 60% accuracy. Some classes with nearly 90% accuracy. Some with less than 10% accuracy.

Trouble Shooting

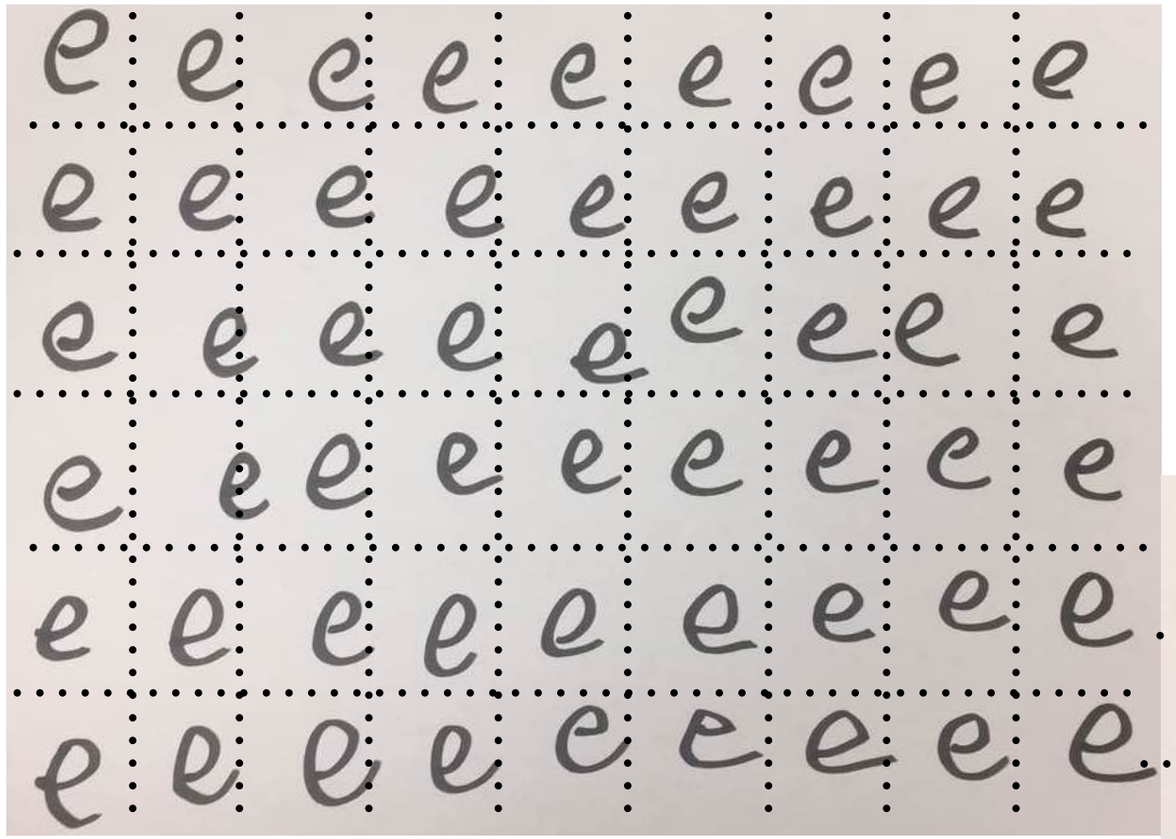
Tensorflow Model—Training + Testing	88%
Tensorflow Model—Testing + saved weights and biases	68%
C model—Testing + saved weights and biases	68%

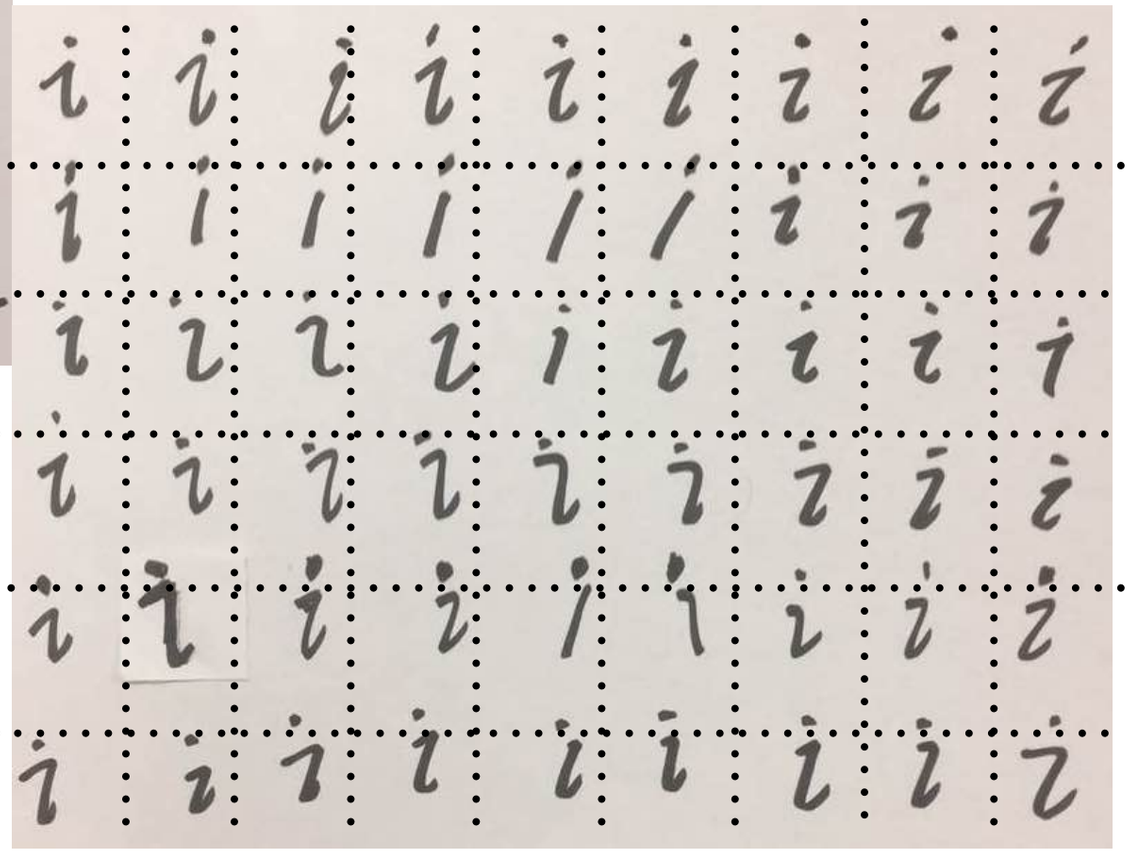
Observation:
The C model I constructed is indeed right
Problem may come from weights and biases saving and loading
Tried to debug, failed

Trouble Shooting

Classes: X, V, M, N, W, E, F, R, T, P, Y, H, K, k, S, A, Q, Z, L	High Accuracy
Classes: D, C, B, i, j, o, r, 1, 2, 8, 0, I, l, e	Low Accuracy
Other Classes	Moderate Accuracy

Observation:
Problem might come from collecting data segmentation





Resize collected data to (9*42)*(6*64) = 576*384
And segment evenly to 64*64 squares
Wrong segmentations for some classes

USC Viterbi School of Engineering

Summary

Milestones	Details
TensorFlow Network (Training + Testing)	88%
Converting TensorFlow Model to C	68%, but right code
Recognition on DSP	50%—65%
Recognition Speed without Voting System	About 3 seconds per Character
Recognition Speed with Voting System	About 9 seconds per Character

USC Viterbi
School of Engineering

Human Factors

- The size of characters that user writes
- Inside gridlines or Outside gridlines
- Care vs Poor Handwriting

USC Viterbi
School of Engineering

What can be done to Improve the performance

- Collecting more data (increase varieties)
- Put bounding box around each characters(remove size dependency)
- Using binary instead of float (increase speed)
- Saving weights and biases in a more accuracy way (increase performance)
- Using more complex network (Increase performance)