Stanislav Listopad
09/04/2017
CS 216

Final Project Report

MNIST Digit Recognizer & Classifier (Zip Code Reader)

1) Summary of the Problem

Transform an image of a handwritten five digit zip code into a five digit zip code.

Sub-problems:

a) Detect each handwritten digit within the image.

b) Classify each handwritten digit within the image.

2) Technical Approach

a) Used Histogram of Gradient Descents (HoG) sliding-window multi-scale detection algorithm with ten templates (one for each digit 0-9) to detect the handwritten digits in the image. Built the templates using the MNIST training images (~600 training images per each digit template 0-9). Used two techniques to filter out false positives. 1) Selected highest scoring detections out of groups of overlapping detections. 2) Removed any detection whose scale was more than twice the scale of an average detection or less than half the scale of an average detection.

Template Size Used: 64 by 96

Initial Scale: 4.0

Rescale Factor: 0.95

Detections limit per digit template (ndet): 5

Detections limit per digit template per single scale (ndet_perscale): 5

b) Transformed all detected handwritten digit images into MNIST format (white on black background, 28 x 28 pixels). Classified each detected digit image using kNN algorithm. Compared this with the labels found during stage a).

Training Samples Used: 30,000

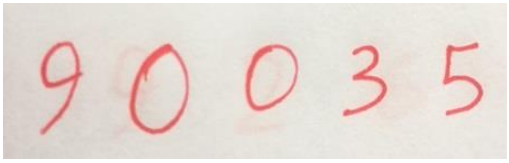
K = 5

3) Results

Used five images of handwritten five-digit zip codes to test the zip code reader program. All zip codes were written using a red pen on white paper. The digits were evenly spaced out, written on the same line, and of approximately the same size.

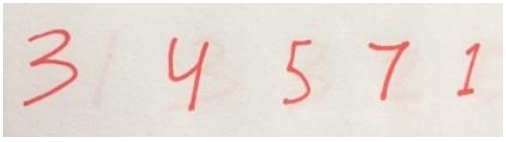
Stanislav Listopad
09/04/2017
CS 216

Test Image 1:

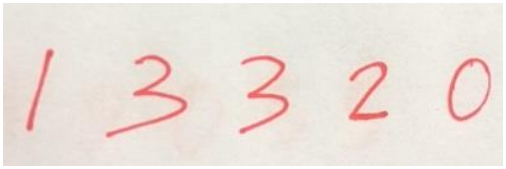


Test Image 2:

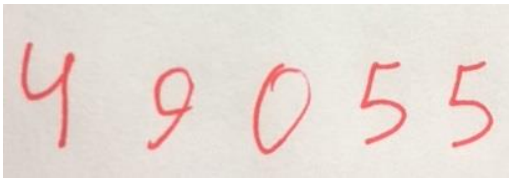
Test Image 3:



Test Image 4:



Test Image 5:



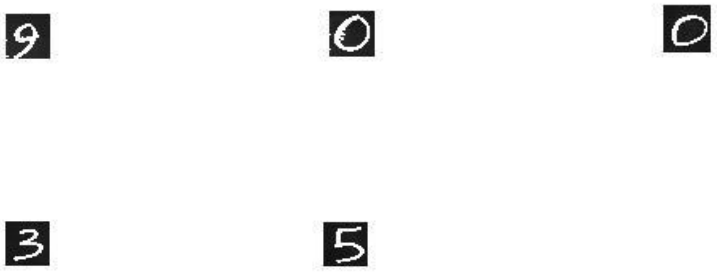
Each image was first analyzed using HoG sliding-window algorithm with ten templates (one for each digit 0-9). All the handwritten digits were correctly detected in all 5 test images. Digit classification, however, performed poorly in some test images. While all handwritten digits were correctly detected, they were not all detected **as correct digits**. For example, in test image 1 the last digit (5) was detected using template for digit 3.

Test Image 1 Results:

Detected as Labels: 9 0 0 3 **3**

I have attempted to improve upon this by employing kNN classification on the extracted detected digit images. However, first, I've transformed these images to be more similar to the ones in MNIST dataset.

MNIST Formatted Detections:



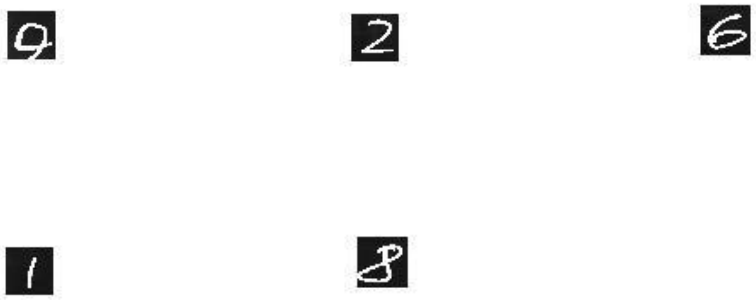
Classified Labels (kNN): 9 0 0 3 5 (CORRECT)

The digits were correctly detected in all 5 test images. However, they were only correctly labeled (via kNN classification) in test images 1 & 4.

Test Image 2 Results:

Detected as Labels: **0** 2 **0** 1 8

MNIST Formatted Detections:



Classified Labels (kNN): **0** 2 **0** 1 8

I did not include the images of detected digits or MNIST formatted detections for test images 3-5 in order to save time. These can be simply inferred by looking at the original test images since all handwritten digits were correctly detected.

Test Image 3 Results:

Detected as Labels: 3 **0** 5 7 **0**

Classified Labels (kNN): **2** **0** 5 7 **5**

Test Image 4 Results:

Detected as Labels: 1 3 3 **8** 0

Classified Labels (kNN): 1 3 3 2 0 (CORRECT)

Test Image 5 Results:

Detected as Labels: 4 9 0 **6** **6**

Classified Labels (kNN): 4 9 0 **6** **3**

4) Discussion

Digit detection performed 100% correctly in all 5 test images. Digit classification (via detected labels and/or via kNN classification) performed poorly in some images.

There are at least two obvious strategies to improve the classification results.

1) Improve the quality of detected labels. Find a way to improve the quality of HoG multi-scale template detection in terms of detecting each digit with the appropriate template (detect as correct digit). This would make follow-up classification unnecessary.

2) Improve the quality of extracted digit images classification. Pre-process the images differently? Use a more powerful machine learning algorithm (Neural Networks) on extracted digit images?

5) References

I have used code from homeworks 1 & 4 within the project. Some of the code utilized was provided by the professor, the rest was written by me. I have also utilized the loadMNISTImages.m & loadMNITLabels.m code from Stanford Wiki in order to load the MNIST dataset into Matlab.