A Practical Model For Handwritten Digit Segmentation and Recognition

Chenxing Ouyang, Jiali Xie
University of California, San Diego
COGS 118 B
jlxie@eng.ucsd.edu
c2ouyang@ucsd.edu

Abstract— **Handwriting recognition have been an active research area over the past few decades because of its many real world applications in deciphering historical documents and validating check signatures. In this paper, we introduce a practical model for recognizing handwritten digits consisting of two modules, namely segmentation module and recognition module. In the segmentation module, digits are identified and cropped out from the original image using a combination of image processing methods including adaptive thresholding, binary thresholding and canny edge detection. In the second module, digit classification is achieved through the use of multilayer perceptron trained on the MNIST dataset using deep learning framework Theano. The system has been tested on the Street View House Number Dataset(SVHN) as well as handwritings captured in real time [1]. Though our proposed model for digit segmentation is proven to be error prone due to edge detection issues, digit recognition is shown to be effective in recognizing digits in images with high saliency.**

I. INTRODUCTION

The application of our proposed system for real time digit recognition through webcam is part of the challenge in the 2016 AUVSI Seafarer Chapter's 14th Annual Student UAS Competition. The goal of this competition is to program an autonomous aircraft to accurately recognize target characters on the ground. The hope is to use our recognition system to periodic analyze for pictures taken by the camera on the drone to locate and identify the characters.

The goal of this project is to propose a system that is capable of analyzing handwriting digits by integrating segmentation algorithms with recognition algorithms. The first step is the digit segmentation module, which involves a cascades of image preprocessing techniques in hope to separate the digits in the image from the background using edge detection then extract them. The segmentation module should result in a list of isolated characters cropped from the original image and processed into suitable binary, which then reshaped into a list of 28 by 28 pixel binary images that is of the MNIST data format. The results of the first step is then fed into the recognition module individually for recognition. The second step is to train a neural network, in this case, a multilayer perceptron to be able to detect and recognize cropped digit in real time streaming via webcam.

II. SEGMENTATION MODULE

The purpose of the segmentation module is to crop out the digit in an image by segmenting the image and processing each resulting image predefined, which means it does not allow us to obtain an accurate number of digits on the image into 28 by 28 MNIST data image format. The image frame for analysis is obtained from random samples in the SVHN Dataset and images frames OpenCV's video stream since the goal is to be able to capture and analyze frames from live video streams. The segmentation module is consisted of two primarily procedures, image preprocessing and digit segmentation.

A. IMAGE PREPROCESSING

A number of methods are attempted to preprocess the image for better segmentation results. Each of which is a combination of multiple image processing techniques.

First Attempt: K-means Clustering on RGB Values First Then on Positions of Binary Pixels.

This method attempts to separates the background from the digits by clustering the digits into one color and the background into another color as seen in Figure 1. So essentially, we use the K-mean clustering on RGB values with k equals 2 as a binary filter.

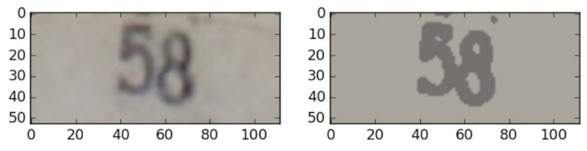

Fig. 1. (a) is the original image from the SVHN Dataset. (b) is the binary filtered result using K-means on RGB values.

After the binary filtering, the digits' pixel positions are obtained and are then subjected to K-means on position vectors. The idea is since digits are usually isolated from one another, we can then use K-means to group digits into their own clusters based their positions. The number of k-centers are selected based on the number of digits presented on the image. This is inflexible because the number of k-centers has to be frame. Furthermore, since K-means algorithm converges to local minimum easily, if the digits are represented compactly as shown below in figure 1, then the digits won't be segmented ideally as expected as shown in Figure 2.

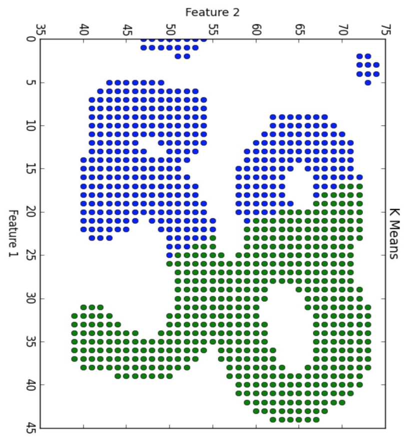

Fig.2. Incorrect Segmentation On Digits As A Result of K-means on Position Values of Digits.

Such compactness could be due to the limitation of K-means on RGB values. Depending on the saliency of the digits, using K-means on RGB values could result in thickened digit edges, which results in even more compactness. This method is shown likely to get stuck in a local minimum which results in failure to separate the digits.

Second Attempt: Adaptive Thresholding and K-means on Positional Values

Our first attempt didn't go well because we tried to use K-means for thresholding and that produces very thick edges and results in compact letter spacings. After trying out various filters on our second attempt, we settled on Otsu's method for adaptive thresholding. Adaptive thresholding is a form of thresholding that takes into account spatial variations in illumination. In particular, Otsu's method is able to classify pixels as either "dark" or "light" by exhaustively search for the threshold that minimizes the intra class variance, which provides high contrast result even with low saliency [2]. However, a drawback is that it might fail to filter out the noise in the background when the variation in the background pixels is very high as shown in Figure 3.

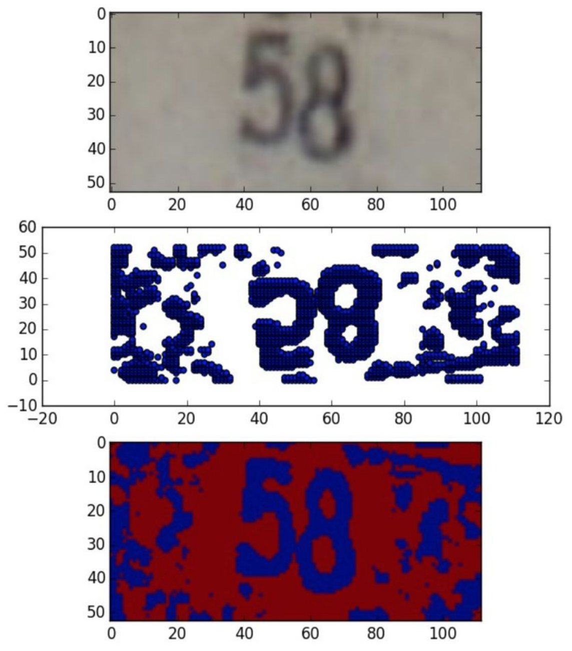

Fig.3 The top image is the original image from SVHN Dataset, the middle and bottom images are the result after using Otsu's method. As suggested, high contrast provides more saliency for background noises.

After running adaptive thresholding, K-means on positional values of the digits is then used again to separate out the thresholded digits in the Cartesian coordinate. However, using K-mean for segmentation once again showed no promises because of the noisy background and poor initialization of k-centers as suggested in Figure 4.

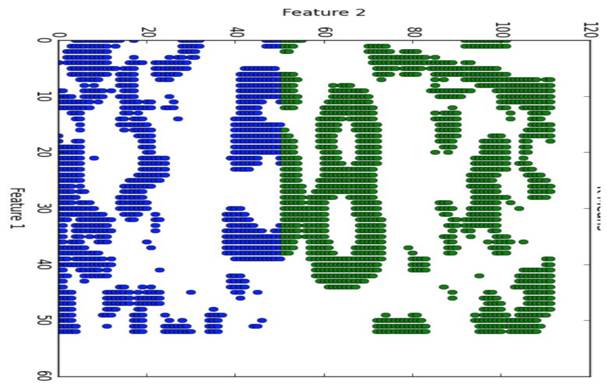

Fig.4 K means on the positional values in Cartesian coordinate converges at a local minimum.

Third Attempt: Canny-edge Detection and Binary+Otsu Thresholding, Then Use Find Contour To Extract Connected Clusters

By now we have realized that Adaptive thresholding itself isn't enough to clear out the noisy background because it creates contracts between pixels of little differences, which exacerbate noise generation, and K-means is not suitable for segmentation because it converges at a local minimum easily and requires prior knowledge of the number of clusters. To solve problems in the second attempt, it is proposed to first blur the image to reduce noise in the background and use canny edge detection which is able to produce thinner edges. A contour finding algorithm is used next on the edges found to extract clusters of pixels that are connected in groups. Binary and Otsu thresholding is then used to make the digits more salient after the digits are being thinned out by the canny-edge detection algorithm as shown in Figure 5.

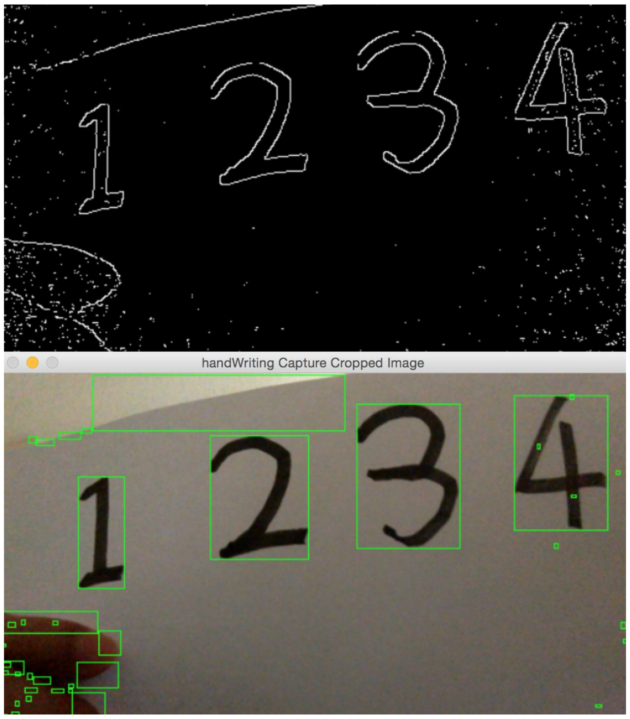

Fig.5 Result from canny-edge detection, adaptive thresholding+binary thresholding, then boxed using findContour algorithm in OpenCV.

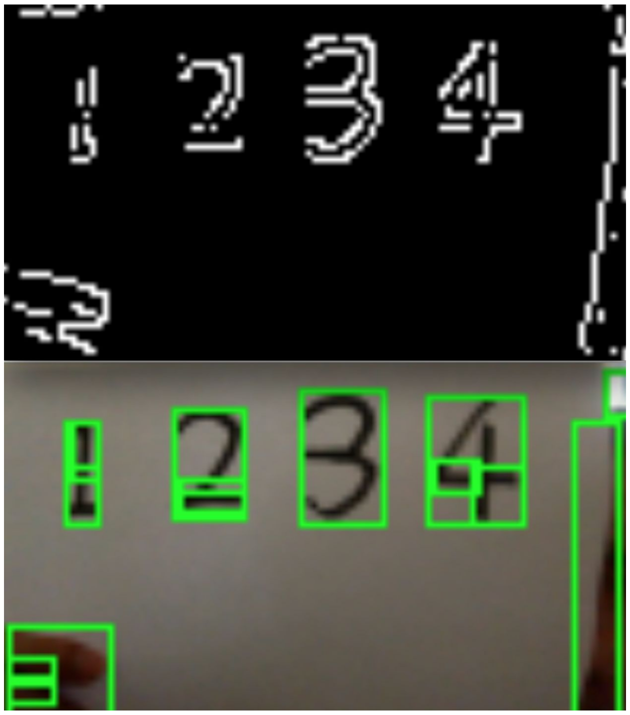

Fig. 6. This shows the scaled downed image with 25 times less pixels.

The pitfall of Gaussian blur is that it might blur out the digits so much that if the digits are not so salient, they might be cut into pieces then grouped incorrectly into contours after Canny Edge detection. As an alternative to reduce background noises and increasing computational efficiency, we simply scaled down the image then do the preprocessing on the scaled down image. To show the effect of scaling down, Figure 7 shows the same image scaled down with 25 times less pixels. As we can see, the number of garbage contours are significantly less yet MLP recognition rate is not affected because the image will be scaled down once again to match the MNIST Dataset format.

Once the digits are cropped out, they are then once again edged detected and binary thresholded individually to match the format of MNIST dataset. The third attempt is proven to be relatively successfully in comparisons to other two attempts.

III. RECOGNITION MODULE

The goal of the recognition module is to classify digits based on the extracted images outputted from the segmentation module using an artificial neural network with the help of the popular deep learning framework Theano.

A. The Model

A multilayer perceptron (mlp) can be used as a logistic regression classifier where the input data is projected onto a set of hyperplanes corresponding to the respective class [6]. A mlp with two hidden layers can be represented as follow in Figure 7.

:

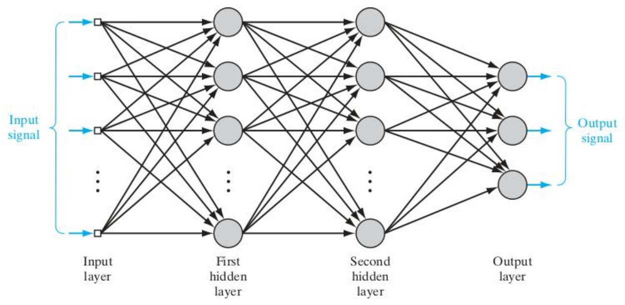

Fig.1: showing a sample mlp with two hidden layers.

In this case, the input layer consists of 1 by 28 by 28 pixel images, which are then reshaped into a 784-dimensional vector when stacking a hidden layer [4]. Each hidden layer contains 800 nodes and there will be a 20% dropout in the input layer and 50% dropout in the hidden layers [4].The main reason for dropping out some of the nodes is that it prevents overfitting of the data and encourages each node in the hidden layer to learn a useful feature rather than correcting its mistakes based on other nodes in the hidden layer.[3] It is also easier to implement than Bayesian model averaging which could be computationally expensive.

To find the weights for each layer, Stochastic Gradient Descent is used through the backpropagation algorithm provided in Theano. The output layer consists of 10 nodes each representing the probability that the image belongs to the respective class. That being said, the probability that an input vector x belongs to class i can be written as:

$$P(Y = i|x, W, b) = softmax_i(Wx + b)$$
$$= \frac{e^{W_i x + b_i}}{\sum_j e^{W_j x + b_j}}$$

[5]. Then the model's prediction will be the class whose probability is maximal:

$$y_{pred} = \text{argmax}_i P(Y = i|x, W, b)$$

[5], where Y is the stochastic variable and W,b are the weights and biases respective.

B. Training the model

Subsequently, the mlp is trained with 50000 images in the MNIST dataset, each labeled with the class it belongs. To do that, a minibatch approach is used to train all data at once. For each batch, 500 images are fed into the mlp and the progress is monitored through computing the training loss, validation loss and classification accuracy.

A cost function is used to update the weights through Stochastic Gradient Descent with Nesterov momentum. The learning rate used is 0.01 and the momentum is 0.9. The resulting accuracy for each epoch is as follows:

epoch	validation accuracy%
1	88.29
2	90.97
3	92.33
4	93.16
5	93.71
6	94.33
7	94.76
8	95.22
9	95.32
10	95.64

C. Results

Although the validation accuracy for training set looks very promising, the mlp functions poorly for the SVHN Dataset images. In the beginning, the problem was attributed to the error-prone nature of segmentation. However, even in some seemingly well-cropped images, a ‘7’ can still be mistaken to a ‘2’ and ‘6’ to a ‘5’.

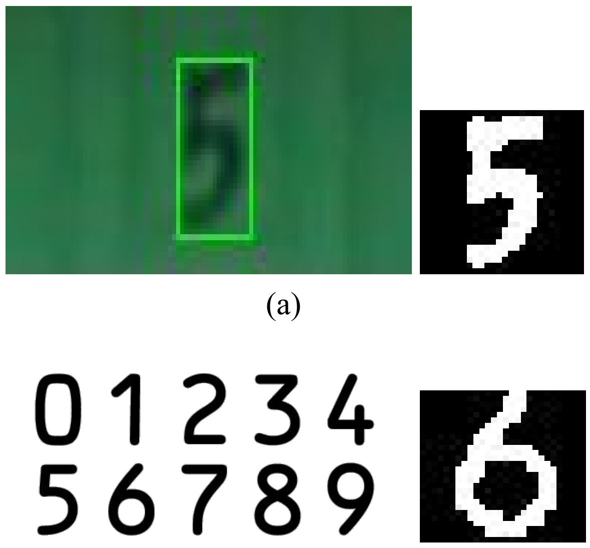

Fig.2: Left is the original image and right is the processed image. (a) The extracted digit ‘5’ is correctly classified as a ‘5’ whereas (b) the extracted digit ‘6’ is classified as a ‘5’ instead of a ‘6’.

It is also found that the mlp works better for handwritten digits. It is possible that since the mlp is trained on handwritten digits, it is better able to detect features on handwritten digits than the others such as pretty prints.

IV. EXPERIMENTATION RESULT

The overall system has been tested both on the SVHN Dataset as well as handwritings captured in real time through OpenCV. Though our proposed model for digit segmentation is proven to be error prone due to edge detection issues, digit recognition is shown to be effective in recognizing digits in images with high saliency and no background noises.

However, the biggest problem in the recognition module is actually the result of a segmentation issue, that is the garbage images extracted by the segmentation algorithm. Even though we have minimized the risk of extracting garbage digits by scaling the original image frame down and giving constraints for finding the minimize size of the an isolated group of pixels, it is still hard to distinguish between a group of pixel that forms a digit and a group of pixel that forms garbage. Because of the segmentation issue, our trained multilayer perceptron is unable to differentiate between meaningful images and garbage correctly as it is only trained to differentiate between meaningful digits but not recognizing garbage. Since the result from the segmentation module is usually full of high proportions of garbage to digit ratio due to low saliency of digits in SVHN Dataset, the segmentation module is proven to be error prone (Figure 7). Thus the segmented result on all segmented images from the SVHN Dataset is rather disappointing. However, if no background noises are present in the present of handwritten digits with high saliency, the segmentation module usually results in significantly less portion of garbage images, and in those with less background noises, digit recognition is shown to be effective in recognizing digits in images with high saliency.

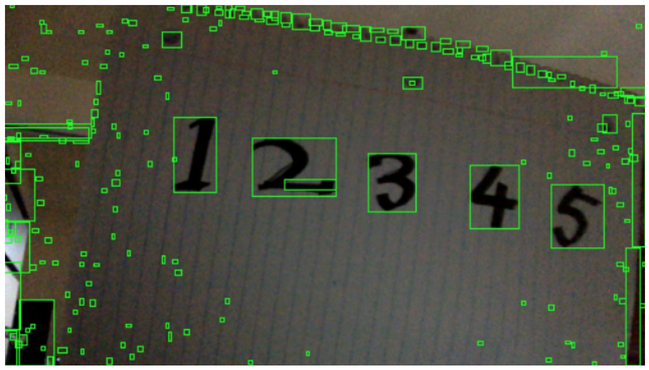

Fig. 7 Segmented Image With High Proportion of Garbage to digit ratio.

V. FUTURE IMPROVEMENTS

Despite of the setbacks discusses four digit recognition is shown to be effective in recognizing digits in images with high saliency and our neural network indeed have a high accuracy for identifying meaningful digits. Alternative algorithms should be considered to implement the segmentation module for more accurate segmentation. One possible improvement is to skip segmentation all together, use feature mapping in a Convolutional Neural Network to classify digits in the image directly.

As an alternative to neural network, garbage images resulted from segmentation can be avoided by training the mlp with more representative dataset with the incorporation of labeled garbage values, printed digits, as well as rotated digits for more diverse and accurate classification.

REFERENCIAS

[1] Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., & Ng, A. (2011). The Street View House Numbers (SVHN) Dataset. Retrieved December 11, 2015, from http://ufldl.stanford.edu/housenumbers/

[2] Bradley, D., & Roth, G. (n.d.). Adaptive Thresholding using the Integral Image. *Journal of Graphics, GPU, and Game Tools,* 13-21.

[3] G. E. Hinton∗ , N. Srivastava, A. Krizhevsky, I. Sutskever and R. R. Salakhutdinov, *Improving neural networks by preventing co-adaptation of feature detectors.* 3 July 2012.

[4]Tutorial¶. (n.d.). Retrieved December 11, 2015, from http://lasagne.readthedocs.org/en/latest/user/tutorial.html

[5] Classifying MNIST digits using Logistic Regression¶. (n.d.). Retrieved December 11, 2015, from http://deeplearning.net/tutorial/logreg.html

[6] Multilayer Perceptron¶. (n.d.). Retrieved December 11, 2015, from http://deeplearning.net/tutorial/mlp.html

[7]Ian J. Goodfellow, Yaroslav Bulatov, Julian Ibarz, Sacha Arnoud, Vinay Shet, *Multi-digit Number Recognition from Street View Imagery using Deep Convolutional Neural Networks* https://themenwhostareatcodes.files.wordpress.com/2014/03/figure-08.png