Handwriting Recognition with Artificial Neural Networks and OpenCV

Tristan Wright
CS488 - Senior Capstone 2012

December 12, 2012

1 Abstract

Two brief arguments are made for why handwriting recognition is important. Existing approaches and my approach are summarized along with the stated goals of my project. The full handwriting recognition pipeline of scanning, segmenting, and recognition is examined and described thoroughly in order. Feed forward Artificial Neural Networks are explained on a low level along with their weaknesses and how to go about character recognition with one. The outcome of the project is summarized, two of the three project goals are marked as incomplete. Speculation on future work on the project is given should the project's problems get fixed.

2 Introduction

For sixty years handwriting and computers have coexisted, yet a machine's ability for recognizing human handwriting is still imperfect. Fluidly free writing notes and ideas on paper will never be phased out by digital input methods. Even so handwriting may drastically change its style over time. The handwriting on the original United States Constitution for example cannot be so fluently read today. So what will handwriting look like in another 225 years? It may be so different that we will need methods for recognizing and reading handwriting of yore. With the advent of mobile devices, equipped with cameras many times more accurate and higher resolution they had ten or even five years ago, the ability to scan text with a camera for later is a commonplace. If an image is of written notes it would be impossible to search for it, and transcribing notes from images can be arduous, why not let a machine do it for you? It is for these reasons above that the problems in handwriting recognition are opportunities to be improved upon.

2.1 Other Approaches

There is high commercial potential for handwriting recognition applications, thus a relatively little amount of literature has been published on it and even fewer open source software options are available. Typically handwriting recognition works as a pipeline, segmenting lines, words, characters and then trying to recognize the result word by word or letter by letter. Either approach involves analyzing the extracted characters or words through a recognition model such as a Hidden Markov model or an Artificial Neural Network (ANN) both of which need to have extensive training with a text samples before hand. In the case of analyzing a document word by word (as opposed to letter by letter) a lexicon and language model is also needed. [3]

2.2 My Approach

Segmenting text features is a relatively trivial step provided the text is level and the scan is of an acceptable quality. Character recognition will involve extracting a character's attributes as input to be recognized in an Artificial Neural Network. Provided the ANN is trained well, this method of recognition would allow a broad set of varying characters to be recognized. The ANN could also be trained on typed characters or any arbitrary linear script so the application itself is not necessarily limited to handwriting or even English. However the initial focus will be recognizing basic ASCII characters.

2.3 My Project

There are three initial goals of the project.

1. Accurately recognize visible ACSII characters in images of handwriting that are less than one megapixel. While higher resolution mobile and point and shoot cameras are becoming more available, the ability to process and recognize handwriting elements in low resolution images will not restrict those who only have basic camera access from using the software.

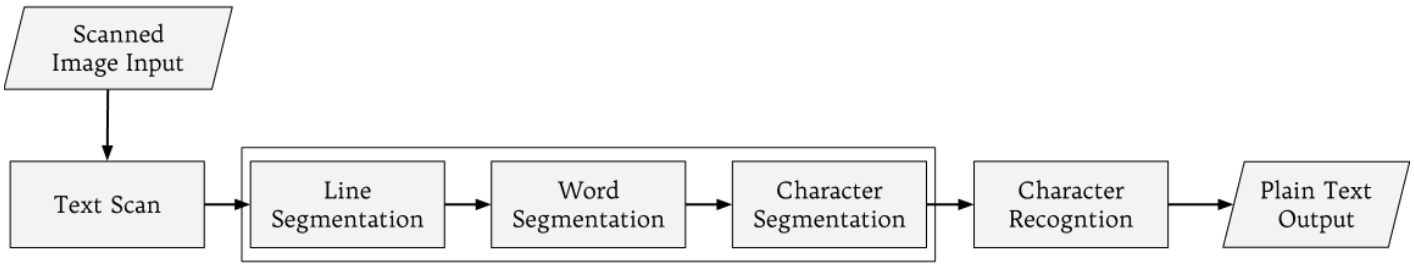

Figure 1: The recognition pipeline, segmentation steps have been grouped together. Slanted boxes signify some input or output.

2. The software is trained to recognize general handwriting, not specifically mine or someone else's. This simply requires a wide variety of readily available handwriting images to train the software on. Rather than writing out pages and scanning them, or searching for handwriting training sets; a combination of actual handwriting scans and images with computer text using handwriting fonts will be generated for the software to train with. (Unachieved)
3. Compiled source can run on multiple platforms provided the libraries are on that platform. Primarily to serve this goal, OpenCV was chosen as the supporting image processing library. The library also gave support for types of machine learning, one of those being an ANN type. The library is supported on every major desktop operating system such as Linux, Mac OS, and Microsoft Windows. Furthermore Google Android and Apple iOS, the two major mobile operating systems, are also supported. [2]

3 Offline Handwriting Recognition

There are two fundamental types of handwriting recognition, online and offline. Online recognition deals with text that has been input to a machine through a digitizer. Offline deals with writing that is on a physical sheet of paper and scanned into a computer for analysis. Here we are dealing with the latter. There are several discrete and ordered steps that go into recognizing offline handwriting: scanning, segmentation and recognition. These steps are visibly laid out in figure 1 with the sub steps of the segmentation process outlined as well.

3.1 Text Scanning

There are three steps here, preparing the image for processing, finding the text on the page, and if necessary normalizing it. It's easiest for the software to recognize higher contrast areas, a image threshold operation makes this easier by turning the entire image into just black and white pixels. The most basic threshold operation by OpenCV scans through an image's pixels and applies the operation:

$$dst(x,y) = \begin{cases} maxvalue & \text{if } src(x,y) > thresholdvalue \\ 0 & \text{otherwise} \end{cases}$$

[1] Where *maxvalue* is usually 255 for white. Determining the threshold value can be a little more challenging. Manually I've found that threshold values of 110 ± 10 work well. However we cannot manually adjust every image every time, adaptive thresholds solve this problem. By taking a mean of the area around the pixel as *thresholdvalue* above and then comparing the pixel's value ($src(x,y)$), a more optimal threshold value can be determined. However if the area to get the threshold value is not large enough, the image as a result of an adaptive threshold can be loaded with features we don't want brought out rendering it useless for feature extraction. [1] Unlike printed text, it's easy for a writer to accidentally start writing at an angle. If the scan of the paper is crooked or the text area is not level, the whole text areas and the text lines can be normalized or made straight without too much processing with a linear rotation transformation. [10]

3.2 Feature Segmentation

3.2.1 Line Segmentation

For segmenting handwritten lines a good approach is to assume that the writer is following imaginary baselines. These baselines can be found with a clustering technique which given lines can find a left most point where the lines start and the straight direction they continue in. [9] A more simple approach is to scan the text from top to bottom and segment on lines where there is no ink or features.

3.2.2 Word Segmentation

Components of the line can are then extracted word by word. The approach is very similar to line segmentation however instead of scanning for gaps top to bottom, we scan side to side. Larger gaps can easily be distinguished as spaces. At this point there are two paths an application can go down once a word has been segmented. The holistic approach recognizes the word without looking at the characters closely. This approach uses a lexicon of possible words and can use a template based or shape recognition approach in matching the image of the word, to the actual word. However the use of a lexicon recognition system can be limiting because not every word may be entered. The analytic approach recognizes and extracts the characters from the word and tries to output the exact word letter by letter. [4] Higher accuracy in recognizing words can be accomplished by combining the holistic and analytical approaches. [9] For the rest of this paper we're only going to look at the analytical approach due to its dynamic nature.

3.2.3 Character Segmentation

For each word image extracted, there is a variety of ways to recognize which textual characters it contains. Additional difficulty can come from mixed text of upper and lowercase letters. [9] The algorithms that go behind word segmentation, by weighing space between components, can be applied here assuming letters are reasonably spaced in a word. [9] Issues arise when letters or components in a word are connected. It's connected letters that make character segmentation less trivial than line or word segmentation. One method for segmenting numbers in zip codes works by looking at successive high and low points and assuming that these belong to new letters. Cursive characters can also be searched for a letter's landmarks, ascending or descending strokes. These landmarks are compared to a pool of stroke templates which are known to be characteristics of certain letters, thus a separation of characters can be found by looking which landmarks occur on the left or right side of the character. However this does not work well for every character like n' and m' which have no landmarks. [4]

4 Character Recognition

Each character segmented has some real letter it represents, it's now the machine's job to figure out what it is. To do this we need some sort of recognition model. The two primary models used to recognize a character are Hidden Markov models and Artificial Neural Networks. Hidden Markov models are a statistical model that makes predictions based on a set of observations and states. [3] In the domain of optical character recognition, its applications are more useful in recognizing uniform characters. In comparison ANNs are a more flexible model, made up of layers of neurons with adjustable weights linking them together. These weights effect the overall output of the network and are adjusted through training. Provided input is uniform enough and the training set is large enough, an ANN can be trained to recognize a character that varies shape. Due to their flexible nature I have chosen to recognize characters with an ANN.

4.1 Artificial Neural Networks

The earliest Artificial Neural Network models were studied in the early 1940s and presented as models of biological neural networks. It wasn't until the notion of error correction and back propagation algorithms that the ANN became prevalent in computing primarily for its learning capability. There are a few different types of ANNs, we'll be looking at the most used and basic type: a feed forward ANN. A single layer feed forward ANN consists input neurons, hidden neurons, and output neurons each in their own layer. Each hidden layer neuron has an activation function and returns some function which takes parameters from input neurons. On the connections between neurons there are weights which alter the input in some way. Suppose the function for y in 2 is $w_1x_1 + w_2x_2$. and the activation function for y is a piecewise function

$$f(s) = \begin{cases} 1 & \text{if } s \text{ is even} \\ -1 & \text{otherwise} \end{cases}$$

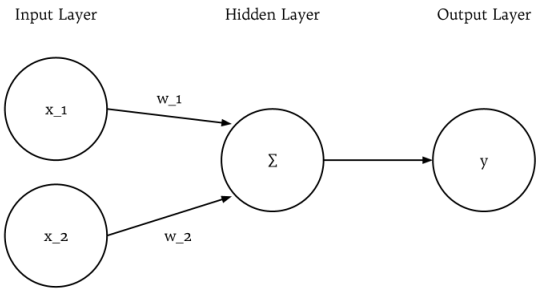

Figure 2: A single layer artificial neural network with a single neuron in the hidden layer.

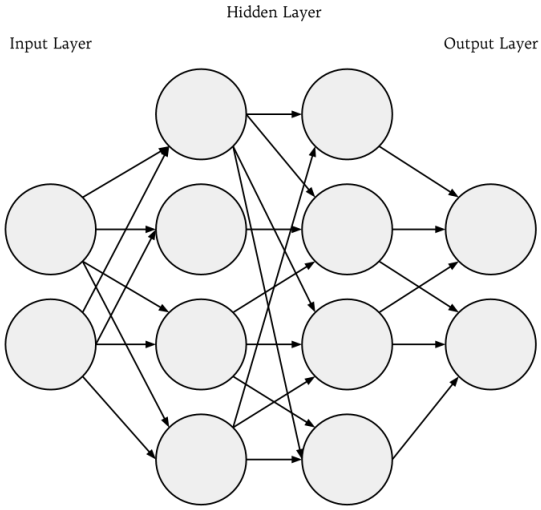

Figure 3: A multi layer Artificial Neural Network with two hidden layers, four neurons each.

The output of this network will either be a 1 or -1.
However depending on the weights between the input and output (w_1 and w_2), the NN can output a wrong answer. To correct this we can train it. The most basic way to train a NN is with the perceptron learning rule. [6]

1. Suppose that the weights are at first random. Input and output nodes are as in (figure 1)
2. Given some training value z.
3. If $y \neq f(z)$, modify the weights by $\Delta w_i = f(z)x_i$.
4. Go to 2.

After some amount of training the NN's weights will be adjusted such that the output of the NN is almost always correct, that is it has converged with the correct solution.

4.1.1 Multi-Layer Artificial Neural Networks and Back Propagation

Described above is a single layer Artificial Neural Network. More complex calculations, such as analyzing the shape of a letter and determining what letter it is, can be performed by adding another layer of neurons to the hidden layer. The Artificial Neural Network is now a multi-layer Artificial Neural Network (figure 3). With a new topology, we can no longer use the perceptron learning rule to update weights, instead we use a method called back propagation which works similarly to it. Connections between two neurons j and k in the hidden layer have weights too and is annotated as w_{jk}.

1. Initialize all weights random.
2. Given some training values z for output nodes y.

3. If $y \neq z$, update each of the weights for the output and hidden layers in the network so that $\Delta w_{ij} = \gamma \delta_k y_j$.

[6]This is called the generalized delta rule, δ is an error rate from the training value and γ a learning rate. The error rate changes each time depending on the layer where the weights are being changed. The whole network gets affected this way, and weights are not changed as drastically as they are in a perceptron learning model because there is some gradient descent for the error as the goes back through the network. [8]

4.1.2 Deficiencies of Artificial Neural Networks

There is a fine balance in using ANNs. If there is too much training of the network, the NN will only give accurate output for values similar to the training values, this is aptly called overtraining or overfitting. In a handwriting recognition context, if I were to train a NN on my handwriting alone, it would only be able to accurately recognize my handwriting. We prevent overtraining by finding out an optimal training set size by incrementing now many training sets there are and then running a validation set on the NN, the lower the error on the validation set the more precise we can be finding n. [8] If input or output values are too large, network paralysis can occur due to the weights being too large for back propagation to have a proper effect. [6] A careful design of the network's input neurons and output neurons can aid against this.

4.2 Using Artificial Neural Networks to Recognize Characters

Recognizing a character is most dependent on the quality of the scanning and segmentation steps. If character segmentation incorrectly slices a character in half, and if the network is trained with that value, it will not be able to recognize full versions of that character or it might incorrectly categorize other characters. Even with good segmentation the input for an ANN must be something we can quantify about a character. Likewise its output must be something we can get a character from. There is a wide variety of data we use to analyze an individual character from as simple as ratio of pixels on the top half to the bottom half, to analyzing histograms of the image and contour analysis of the binary image. [11] The more information you can get about a character the better, and more input neurons you'll have. Remember not to have too many input values across a wide range or else network paralysis can occur. The same goes for output values, an expected large output value from a bunch of small inputs can easily lead to a case of paralysis because the input values will not change as proportional to the output.

5 Results and Discussion

Of the three goals set in section 2.3, one could truly be considered complete.

Accurately Recognize ASCII Characters from low resolution images The most work in the project was put into figuring out reliable segmentation methods instead of recognition. Segmentation is reliable enough such that characters are successfully segmented given samples of legible handwriting. While Spaces and new lines are recognized, accurate character recognition across a wide range of image resolutions was not accomplished.

General recognition with a general training set The ANN was trained with set of about 250 characters. However recognition of input characters was not successful. I believe this stems from how the ANN was designed in the first place. Five input neurons take the width to height ratio of a character and black pixel counts from the top, bottom, left, and right halves of a character. One output neuron gives an ASCII character code. I think the input is not descriptive enough for a single character and the expected output is too wide a range given the input.

Portable binary The use of the OpenCV library was a success. Should the software want to be ported to a mobile device, some small changes in the code. Changes in the language and implementation of GUI views to start would have to be made, however the calls to OpenCV would stay more or less the same.

5.1 Further Work

Should the software get to a proper working state, possible additions include recognition of multiple character sets and non-linear scripts. Recognizing a character in ASCII is no different from recognizing it in UNICODE, the difference being there is much much more searching involved in UNICODE recognition given its size and the amount of characters that have been implemented. Recognizing non-linear scripts or writing styles that aren't based on discrete 26 characters like English will be a challenge to undertake, Korean Hangul and Egyptian Hieroglyphs come to mind respectively.

6 References

References

[1] Opencv documentaton. http://docs.opencv.org/.

[2] About opencv. http://opencv.org/about.html, December 2012.

[3] Homayoon S.M. Beigi. An overview of handwriting recognition. In *Proceedings of the 1 st Annual Conference on Technological Advancements in Developing Countries*, pages 30–46, 1993.

[4] Richard G. Casey and Eric Lecolinet. A survey of methods and strategies in character segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(7):690–706, 1996.

[5] M.-Y. Chen, A. Kundu, and J. Zhou. Off-line handwritten word recognition (hwr) using a single contextual hidden markov model. In *Computer Vision and Pattern Recognition, 1992. Proceedings CVPR '92., 1992 IEEE Computer Society Conference on*, pages 669 –672, jun 1992.

[6] Ben Kröse and Patrick van der Smagt. *An Introduction to Neural Networks*. University of Amsterdam, 1996.

[7] U.-V. Marti and H. Bunke. Text line segmentation and word recognition in a system for general writer independent handwriting recognition. In *Document Analysis and Recognition, 2001. Proceedings. Sixth International Conference on*, pages 159 –163, 2001.

[8] Tom M. Mitchell. Artificial neural networks. http://www.cs.cmu.edu/~epxing/Class/10701-10s/Lecture/lecture7.pdf, February 2010.

[9] R. Ecole Plamondon. Online and off-line handwriting recognition: a comprehensive survey. *This paper appears in: Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(1):63–84, Jan 2000.

[10] Thomas Plötz and Gernot Fink. Markov models for offline handwriting recognition: a survey. *International Journal on Document Analysis and Recognition*, 12:269–298, 2009. 10.1007/s10032-009-0098-4.

[11] Øivind Due Trier, Anil K Jain, and Torfinn Taxt. Feature extration methods for character recognition. Jul 1995.