Applying Neural Networks to Character Recognition

Eric W. Brown
feneric@ccs.neu.edu

Abstract

This paper explores the application of neural networks to the problem of identifying machine printed characters in an automated manner. In particular, a backpropagation net is trained on an eighty-four character font and tested on two other fonts. Its results are compared against results obtained on the same data by a more traditional approach.

Keywords

Text recognition, Optical character recognition (OCR), Neural networks, backpropagation, Connectionist nets, Feature extraction, Feature point, Off-line character recognition, machine printed character recognition.

The need for some form of automated or semi-automated *optical character recognition (OCR)* has been recognized for decades. Today there are numerous algorithms that perform this task, each with its own strengths and weaknesses. This paper explores the differences between two different OCR algorithms: a feature extraction method using traditional artificial intelligence techniques for classification, and a neural network approach with virtually no preprocessing.

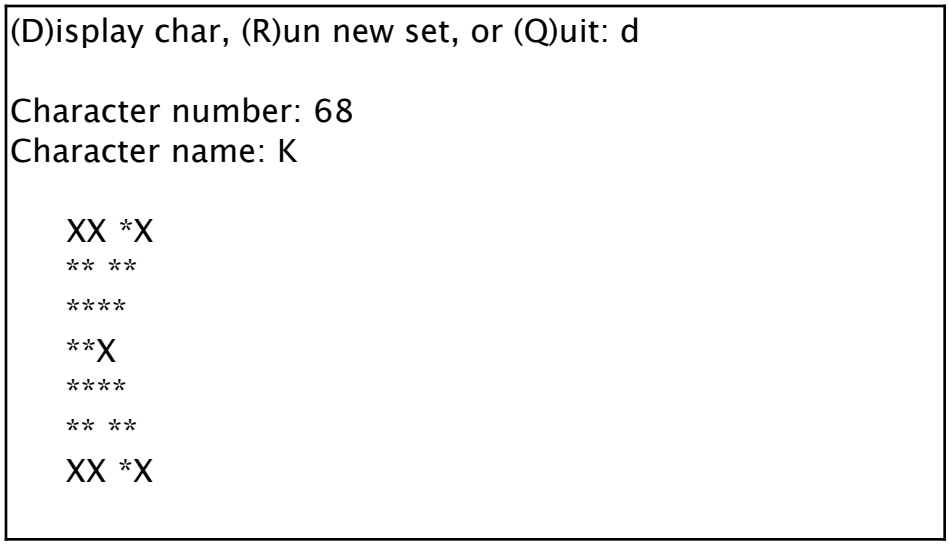
The basic idea of the experiment was essentially to run the identical data through the two different algorithms and note the differences in each run along the way. The feature extraction method was actually performed by the author at an earlier date[1] while the neural net method was implemented specifically for this paper utilizing the exact same character data and following a similar procedure. Thus comparisons between the two methods were not influenced by unfair differences in the data or unbalanced procedures that would favor one method over the other.

The data used consists of three eighty-four character fonts. The character resolution for all three fonts is 8X8. The sets contain both upper and lower case letters as well as numbers and a handful of miscellaneous punctuation marks. The sets are all used as actual display characters on an old CBM C128 computer[2] so they represent real-world data; they were not designed for this experiment. In particular, the sets include a few examples of allowable alternate forms of symbols (i.e. the zero with the slash versus the zero without the slash, the open four versus the closed four, etc.) and no real attempt was made to statistically deal with this fact. Thus the overall results are somewhat lower than they should be; a fair treatment would be to eliminate these alternate forms as they would probably have to be implemented as special cases in a commercial OCR program.

It is not the purpose of this paper to cover the original feature extraction algorithm in great detail. A general summary of its design and operation is included here for convenience. The idea of the feature point extraction algorithm is to identify characters based on features that are somewhat similar to the features humans use to identify characters. The rationale is that when the algorithm does misclassify a character (as every algorithm does) it should pick a character that a human would consider to be a reasonable guess, because it is easier for humans to correct mistakes that are typical of humans (i.e. it is easier to get "Save" out of "5ave" than "Mave"). To implement this notion, the algorithm would scan through the entire 8X8 character matrix and analyze each non-empty pixel. The immediate neighborhood of the pixel would be examined and pixels that seemed to be worthy of notice were marked as such (see Figure 1, "Character Display"). The original C128 ROM character set was processed in this manner and used as a gold standard; characters could then be identified by comparison against the different entries in this *"dictionary"*. Comparisons were executed by computing the sum of the minimum distances between the feature points of the character to be identified and the feature points of the dictionary character. The guess was the dictionary character with the smallest sum within a certain threshold. Note that this algorithm made no attempt to consider different types of feature points, and although it penalized missing or extra feature points, it did not do so highly. It was never optimized to provide the best possible results; it was merely an attempt to test the general concept of the algorithm. Since the neural net was also not optimized, the overall comparison between the two methods of OCR is still on even ground. The results of the feature point

extraction algorithm are outlined in Table 2, "Character Recognition Results -- Feature Extraction Method".

Figure 1: Character Display

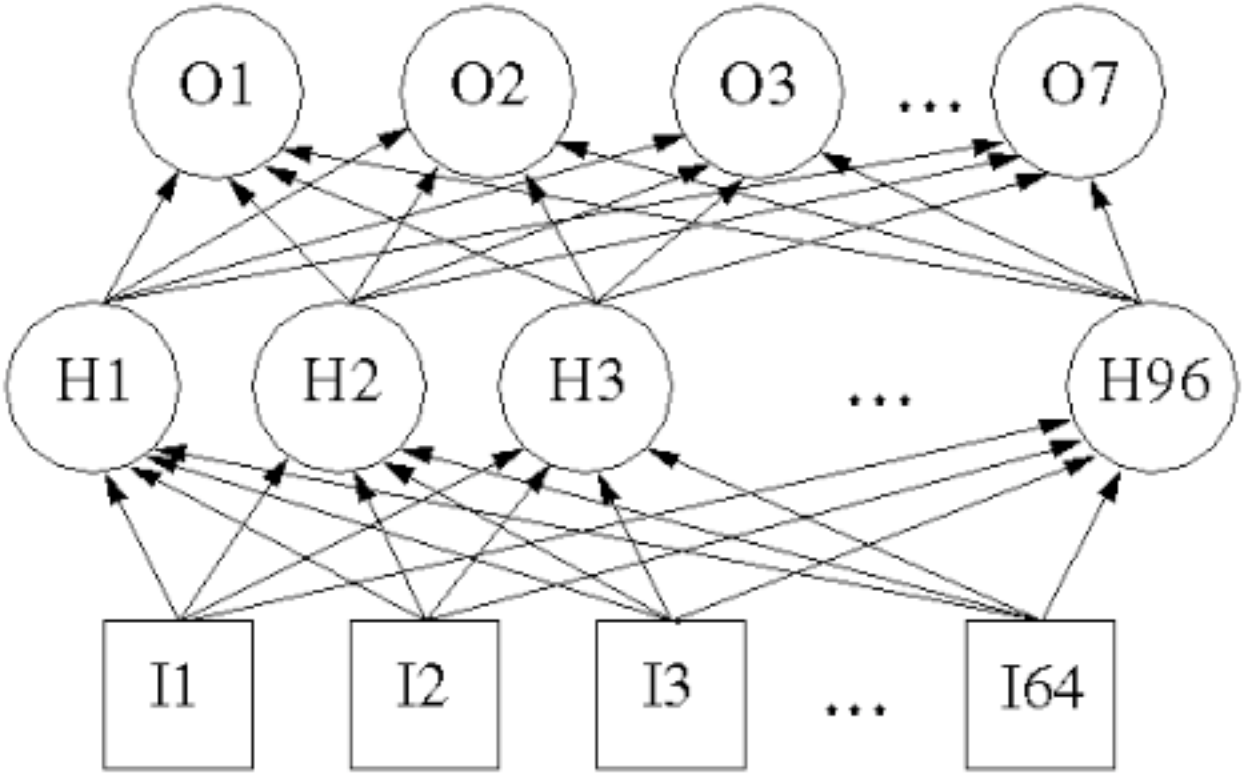


The neural net approach utilized three separate steps. The first step simply translated the binary character data into a friendlier form. The second step took the output of the first and trained a backpropagation network on it, outputting all the resulting weights and general network information. The third step took the output of the second and created a network. It then ran a full character set through the network and output identification information for all the characters the set contained. The reasons for implementing theh neural net OCR as three programs were all practical. By keeping the first step separate, the preprocessing code from the feature extraction OCR program could be used, eliminating this one area of difference between the two algorithms. The second step was separated just because learning was such a slow process. Several machines could thus be dedicated to nothing but learning while a different machine was used to analyze the results.

The network consisted of sixty-four inputs, ninety-six hidden nodes, and seven outputs. It was essentially a flat feedforward network that was fully connected without self-inputs or biases (see Figure 2, "Network Topography"). It was made to train on the same character set that the feature extraction algorithm had used as its dictionary (see Figure 3, "The CBM Character Set & Extracted Feature Points"). Learning was achieved through backpropagation without momentum[3, 4].

Each of the sixty-four inputs was wired to one of the pixels in the 8X8 character. An input was taken to be zero if the pixel was empty, or a one otherwise. The seven outputs were simply used to make a seven bit numerical label (unique labels for eighty-four characters require seven bits) that coincided with the ordering of the character set. The labels ran from zero to eighty-three.

Figure 2: Network Topography



Training the net was troublesome and fraught with difficulties. Small changes in the step size used in the update of the weights resulted in large changes in the behavior of learning overall. In particular, most step sizes resulted in a complete failure to converge within a reasonable number of sweeps through the data set (the learning program was set to consider the number 100,000 to be reasonable). There seemed to be no was *a priori* to determine a good step size (or the number of hidden nodes to use) form the outset. The learning program required several hundred computer hours on a dedicated Sun SPARCstation to run, making it practically impossible to really try and obtain optimal results. It is almost guaranteed that ninety-six is not the optimal number of hidden nodes; this number was produced by multiplying the number of inputs by 1.5. Quite possibly the use of momentum[3] or Newton's method[5] could partially alleviate the learning problems. Currently this learning problem stands as an undesirable property that the feature extraction method lacks; it makes it extremely difficult to optimize the neural net classifier within any sort of time constraints.

With all this difficulty stated, the network was made to learn properly by using a step size of 1.05. Not only did it manage to converge with this particular step size, it converged fairly quickly and only required one-hundred thirty-six sweeps through the data set.

Once a properly trained net had been obtained, it produced reasonable results for an unoptimized

algorithm (see Table 1, "Character Recognition Results -- Neural Net Method"). It also produced these decidedly faster than the feature extraction method did. This difference was partially due to the fact that while the feature extraction method had to carry around its dictionary and do a character by character comparison for each symbol to be identified at runtime, the neural net method learned everything it needed to know about the gold standard character set during its training phase and could get results for a symbol with a single feedforward pass.

Table 1: Character Recognition Results -- Neural Net Method

Results for the 'Alt' Character Set		
Total Correct	66	79%
Total Correct (without counting identical characters)	15	18%
Total Unknowns	2	2%
Total Wrong Guesses	16	19%
Total Wrong Or Unknown	18	21%
Results for the 'Ult' Character Set		
Total Correct	33	39%
Total Correct (without counting identical characters)	12	14%
Total Unknowns	3	4%
Total Wrong Guesses	48	57%
Total Wrong Or Unknown	51	61%
Total Combined Results		
Total Correct	99	54%
Total Correct (without counting identical characters)	27	16%
Total Unknowns	5	3%
Total Wrong Guesses	64	38%
Total Wrong Or Unknown	69	41%

Its accuracy is significantly worse than that of the feature extraction method (see Table 2, "Character Recognition Results -- Feature Extraction Method"), and it did not do as good of a job at making typical human guesses. Still, its speed improvement makes it an algorithm that should not be immediately dismissed.

Table 2: Character Recognition Results -- Feature Extraction Method

Results for the 'Alt' Character Set		
Total Correct	72	86%
Total Correct (without counting identical characters)	21	25%
Total Unknowns	5	6%
Total Wrong Guesses	7	8%
Total Wrong Or Unknown	12	14%
Results for the 'Ult' Character Set		
Total Correct	49	58%
Total Correct (without counting identical characters)	28	33%
Total Unknowns	12	14%
Total Wrong Guesses	23	27%
Total Wrong Or Unknown	35	42%
Total Combined Results		
Total Correct	121	72%
Total Correct (without counting identical characters)	49	29%
Total Unknowns	17	10%
Total Wrong Guesses	30	18%
Total Wrong Or Unknown	47	28%

Perhaps the best method would be something that combined both approaches. Although the current network was trained on the full character data, it would be relatively easy to train a set on just the extracted feature data. The modification has been made, but after several thousand minutes of computer time on numerous different machines no such net has yet been made to settle. It is expected that such a combined approach would maintain the desirable feature of the feature extraction method of generally making human-like mistakes while keeping the neural net method's identification speed. Careful adjustments to both the network's topography and the feature extraction process could radically improve the accuracy.

Overall the results of the experiment were promising but far from good in their present form. If one method had to be chosen over the other, the results here would indicate that the feature extraction method using standard AI techniques for classification would be the better choice. The possiblility of a hybrid algorithm makes other options possible, and quite possibly the most potential lies down this path.

References

1. E. W. Brown, "Character Recognition by Feature Point Extraction", Northeastern University internal paper, 1992

2. The Commodore 128 Computer, Commodore Business Machines, 1984

3. Stephen I. Gallant, Chapters 11 & 12; *Neural Network Learning and Expert Systems*, 1993, The MIT Press, Cambridge, MA

4. D. S. Levine, pp. 58-62; *Introduction to Neural & Cognitive Modeling*, 1991, Lawrence Erlbaum Associates, Publishers; Hillsdale, NJ

5. Patrick K. Simpson, pp. 112-123, *Artificial Neural Systems*, 1990, Pergamon Press, New York, NY

Figure 3: The CBM Character Set & Extracted Feature Points

Figure 4: The 'Ult' Character Set & Extracted Feature Points

Figure 5: The 'Alt' Character Set & Extracted Feature Points