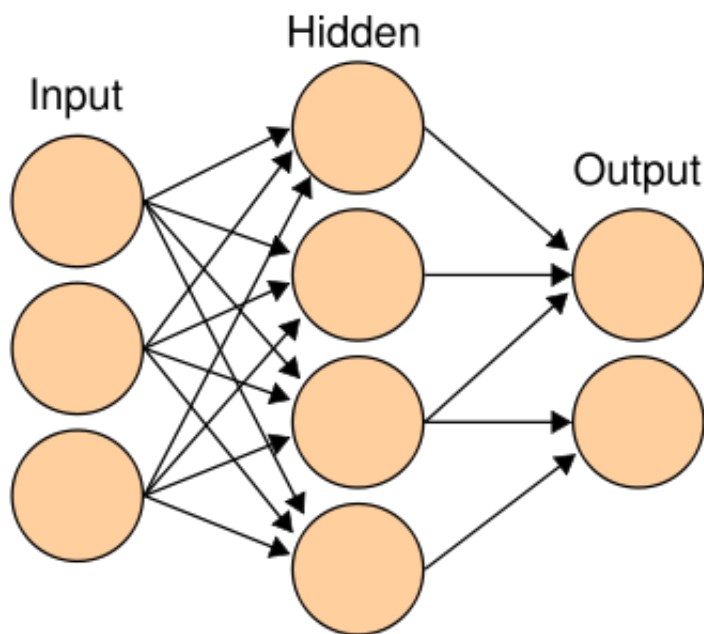


4005-750-01

Introduction to Artificial Intelligence OCR using Artificial Neural Networks

Kurt Alfred Kluever (kurt@klover.com)
Department of Computer Science
Golisano College of Computing and Information Sciences
Rochester Institute of Technology

February 18, 2008



Abstract

Optical character recognition refers to the process of translating images of hand-written, typewritten, or printed text into a format understood by machines for the purpose of editing, indexing/searching, and a reduction in storage size. The OCR process is complicated by noisy inputs, image distortion, and differences between typefaces, sizes, and fonts. Artificial neural networks are commonly used to perform character recognition due to their high noise tolerance.

1 Optical Character Recognition

1.1 Definition

Optical Character Recognition, or OCR, is the process of translating images of handwritten, typewritten, or printed text into a format understood by machines for the purpose of editing, indexing/searching, and a reduction in storage size. There is a great need to accurately OCR printed materials:

Much of the worlds information is held captive in hard-copy documents. OCR systems liberate this information by converting the text on paper into electronic form. [5]

While artificial neural networks are an excellent solution to the classification stage of OCR, the more important step in OCR is the selection of inputs to the artificial neural network. Most OCR systems decompose the process into several stages:

1. Format Analysis (Document Image \Rightarrow Character String Image)
2. Character Segmentation (Character String Image \Rightarrow Character Image)
3. Feature Extraction (Character Image \Rightarrow Character Properties)
4. Classification (Character Properties \Rightarrow Character ID)

It is not within the scope of this report to evaluate segmentation methods. An in-depth survey of this field can be found in [2].

1.2 Feature Extraction

Feature extraction is the process of transforming the input data into a reduced representation. It is commonly employed when there is too much input data to efficiently process or if the input data is redundant (lots of data but not much information). This simplification of the input data provides an accurate description of a larger set of data. The set of application-dependent features are typically chosen by domain experts. If no such experts exist, other dimensionality reductions, such as principle component analysis, can still be performed. In this summary, the feature vectors will be computed using the COMPUTE-FEATURE-VECTOR(img, n, m) function, where img is the binarized input image, n is the number of rows, and m is the number of columns.

1.3 Classification

Character classification is performed based on the feature vectors that were computed in the previous step. Classification can be done by nearest neighbor techniques, template matching, neural networks, etc. The surveyed papers feed the feature vectors into an artificial neural network for character classification.

2 Vertical/Horizontal Projections [4]

In 1997, a group of researchers from the Monash University in Australia developed simple feed-forward artificial neural network using back propagation, and then trained with noisy data. The system reduces image processing time while maintaining efficiency and versatility. In order to produce a font size independent system, the inputs are first scale normalized to 32×32 pixels. The selection of features that represent the separate character classes is crucial to the performance of the resulting system.

2.1 Feature Extraction

The feature vector used for this system consists of the concatenation of 32 elements from the vertical feature vector V_v and 32 elements from the horizontal feature vector V_h . The vertical/horizontal feature vectors are computed as the count of the number of true pixels across a given projection. The vectors can be easily computed in $O(n \times m)$ polynomial time:

```
COMPUTE-FEATURE-VECTOR(img, n, m)
1   $V_v \leftarrow \emptyset$ 
2   $V_h \leftarrow \emptyset$ 
3  for row  $\leftarrow 1$  to n
4      do for col  $\leftarrow 1$  to m
5          do if  $i[\textit{row}][\textit{col}]$ 
6              then  $V_v[\textit{col}] \leftarrow V_v[\textit{col}] + 1$ 
7                   $V_h[\textit{row}] \leftarrow V_h[\textit{row}] + 1$ 
8  return  $V_v + V_h$ 
```

2.2 Neural Network Structure

The network contains 64 input neurons that are connected to a single hidden layer. The hidden layer is then connected to an output layer, which presumably has a neuron for each output class. There is a unique output class for each character to be recognized.

2.3 Results

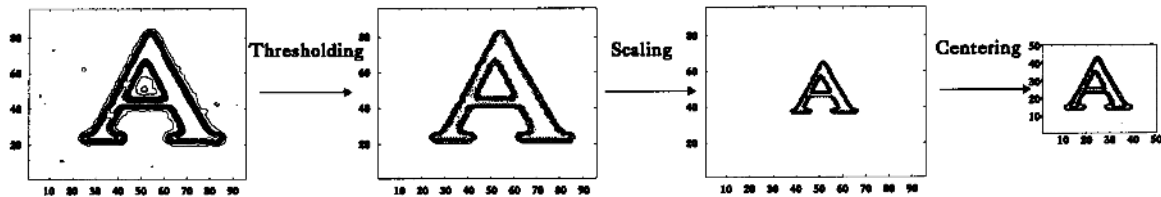
Training using two different learning rates was performed until the error tolerance dropped to $E_t = 0.0001$. With a learning rate of $\eta = 0.90$, it took 1255 iterations to drop the error tolerance below E_t . A slightly slower learning rate yielded faster convergence. A learning rate of $\eta = 0.45$, it took 1053 iterations to drop the error tolerance below E_t . Noisy input yields a recognition rate of only 70%, while clean input yields recognition rates over 99%. This system performs well given the limited variability in the input (single font). However, the significant drop in recognition rate when noisy data is introduced shows the system is not very fault tolerant.

3 Character Matrix as Feature Vector [1]

In 1995, a joint group of researchers from Stanford University and Canon Research Center America developed fully-connected, feed-forward artificial neural network using back propagation, and then trained with generated data. The system can successfully recognize 12 common font faces with sizes between 8 and 32 point. The high accuracy is achieved through a centroid dithering process that creates many training samples from a single rendered character.

3.1 Feature Extraction

As explained above, the system does not rely on a subset of features of the characters. Instead, the entire character matrix is used as an input to the neural network. The image is first pre-processed by thresholding the image to remove noise, centering the character on the characters centroid, and normalization via scaling.



Next, the feature vector is trivially computed in $O(n \times m)$ polynomial time, where img is the pre-processed input image, n is the number of rows, and m is the number of columns:

```
COMPUTE-FEATURE-VECTOR( $img, n, m$ )
1  $V \leftarrow \emptyset$ 
2  $counter \leftarrow 1$ 
3 for  $row \leftarrow 1$  to  $n$ 
4     do for  $col \leftarrow 1$  to  $m$ 
5         do  $V[counter] \leftarrow img[row][col]$ 
6          $counter \leftarrow counter + 1$ 
7 return  $V$ 
```

3.2 Neural Network Structure

Instead of training a single network to recognize multiple fonts, the network could have been implemented as a bank of single-font networks. However, this approach was not chosen because the individual networks would not be able to benefit from associating the “correct” character of a different font with the “correct” character of their font (and similarly for wrong characters). Creating a single network that can successfully recognize any of the fonts increases redundancy, durability, and complexity of the network.

The input layer contains an astronomical 2500 neurons. Since the input images consist of a $n \times m$ matrix of pixels (50×50), the feature vector consists of a 2500 element vector. This

vector is fed directly into the neural network. The input layer then connects to a hidden layer consisting of 100 neurons. The hidden layer then connects to an output layer consisting of 94 neurons, each of which corresponds to a given character class.

3.3 Training via Centroid Dithering

The training database contains 1,128 (94×12) images, one image for each character in each font. The network is trained by applying a centroid dithering technique to each of the training images.

The inputs for the training data, as explained above, are template images for each of the fonts. Centroid dithering is applied to the image inputs before training. This creates many different images from a single image by displacing the input image in both the horizontal and vertical direction over the range of $[-2, +2]$ pixels. This step allows the network to tolerate slight variations in stroke width which could occur from printer settings, toner levels, or bold faced fonts.

The network is trained using the centroid dithered images to 8,650,000 iterations, to reach a mean squared error of 2.0×10^{-6} .

3.4 Results

The test database contains 347,712 characters (28,976 samples for each of the 12 fonts) , with an even mixture of font sizes, scanned at 400 dpi in 8-bit gray scale resolution.

The testing program recorded all discrepancies, excluding spaces, tabs, and carriage returns. In general, a discrepancy between the expected output and the actual output constitute an error. However, two special cases of discrepancies were not counted towards the recognition rates. The first are cases where the input image is unrecognizable due to segmentation error, scanner error, or paper residue. The second are cases where the input character is ambiguous without the surrounding context (i.e., ‘I’ vs. ‘l’ vs. ‘—’). Since the system is able to accept multiple fonts, these characters become ambiguous when viewed cross-font. Therefore, these errors are not counted towards the recognition rates.

The network correctly recognized 100.0% of the 347,712 characters in the testing database. The network was also tested on another data set consisting of 1,072,452 characters in a single font and made only 1 incorrect classification. The overall accuracy of the system is incredibly impressive. The limited problem domain (i.e., limit font selection, perfect pre-segmentation, etc.) may have artificially inflated the accuracies, but nevertheless, it is still impressive.

Laplace’s Special Rule of Succession is employed to answer the following question: if n independent trials of an experiment have resulted in success, what is the probability that the next trial will result in success? The estimated probability of success for $n = 347,712$ is given as:

$$p = \frac{n + 1}{n + 2} = \frac{347,713}{347,714} = 99.99971\%$$

Hence, it is claimed that this method provides “five nines” worth of accuracy. This technique, albeit relatively simple, is indeed a very robust method for optical character recognition.

4 “Fundamental” Features [6]

In 1997, a group of researchers at the University of Sao Paulo used 17 character features as inputs to an artificial neural network to detect isolated capital letters from 16 different fonts. The recognition of the characters was performed by a back-propagation artificial neural network.

4.1 Feature Extraction

The set of 17 features included observations based on curvature, line slope, space and line interconnection, relative distance between two lines and other topological and geometrical features. These features were chosen by human experts. The features contain 14 “general” features and 3 “special case” features used to differentiate between certain characters. The following is a complete list of features extracted from the input character:

1. The number of crossings of the horizontal cut line ($V1$) at 30% of the character height.
2. The number of crossings of the horizontal cut line ($V2$) at 50% of the character height.
3. The number of crossings of the horizontal cut line ($V3$) at 80% of the character height.
4. The number of crossings of the horizontal cut line ($V4$) at 40% of the character height.
5. The number of crossings of the horizontal cut line ($V5$) at 70% of the character height.
6. The number of crossings of the vertical cut line ($VV1$) at 30% of the character width.
7. The number of crossings of the vertical cut line ($VV2$) at 50% of the character width.
8. The number of crossings of the vertical cut line ($VV3$) at 80% of the character width.
9. The number of crossings of the vertical cut line ($VV4$) at 40% of the character width.
10. The number of crossings of the vertical cut line ($VV5$) at 70% of the character width.
11. The number of crossings of the vertical cut line ($VV6$) at 90% of the character width.
12. The character line slope, computed from the crossings of $V1$ and $V2$.
13. The upper “open” or “closed” area, computed by $VV1$ and the top coordinate YT .
14. The lower “open” or “closed” area, computed by $VV1$ and the base coordinate YB .
15. The “distance” used to distinguish between ‘I’ and ‘T’, computed by $VV2$.
16. The “arc” or “straight” used to distinguish between ‘O’ and ‘D’, computed by XT and XA coordinates.
17. The “stroke length” used to distinguish between ‘F’ and ‘P’, computed by $V2$.

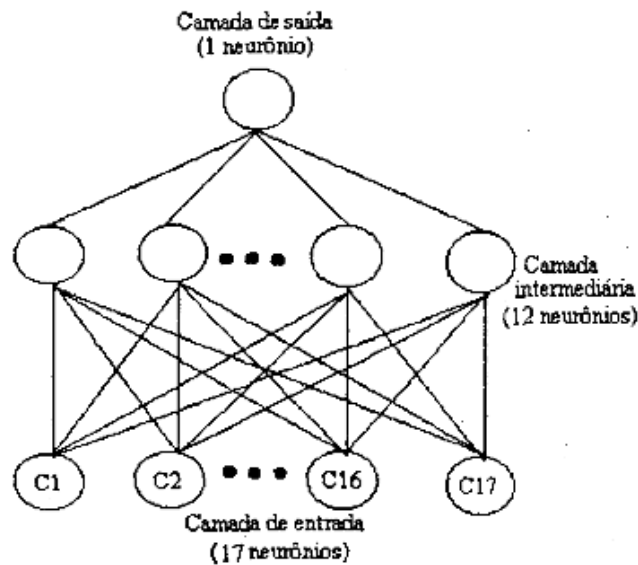
The example below illustrates how the $VV1$ vertical cut line can differentiate between character classes. If $VV1 = 1$, the character is J, I, L, T, etc. If $VV1 = 2$, the character is A, C, F, etc. If $VV1 = 3$, the character is E, F, S, Z, etc.



The other cut lines provide similar distinguishing characteristics of the input images. By combining all of these features, a unique fingerprint for each character is developed.

4.2 Neural Network Structure

The artificial neural network was a 3 layer, back-propagation network. The input layer has 17 nodes (each of which corresponded to a feature in the feature vector described above). The hidden layer has 12 nodes which are connected to the single output node.



4.3 Training

The network was trained using the Levenberg-Marquardt algorithm (LMA) because it yielded better results than the standard descent gradient method. After 100 iterations, the square medium error was reduced to 0.082.

4.4 Results

The trained network performed reasonably well. The 16 different fonts that the network were trained on were also used in the testing stage. The network made only 3 misclassifications. 13 of the 16 fonts yielded perfect recognition rate (100%) while the other 3 yielded a recognition rate of 96.15%. The testing of 416 unique characters from the 16 different fonts yielded an overall recognition rate of 99.3%:

$$\frac{n - e}{n} = \frac{416 - 3}{416} = \frac{413}{416} = 0.992788 \approx 99.3\%$$

The network was also evaluated using 5 new fonts that the network was not trained on. The network yielded perfect recognition rates on 3 of the 5 new fonts. The other 2 new fonts had rather poor recognition rates of 76.92% and 69.2%. This demonstrates the flexibility in the network, as it can do a decent job at recognizing untrained fonts.

5 Structure Adapting Network [7]

Based on T.C. Lee's PhD dissertation on structure level adaptation of neural networks [3], a group of researchers from the University of the Americas in Puebla, Mexico applied a similar approach to an artificial neural network designed to perform optical character recognition. The network was designed to recognize a single font of multiple sizes.

5.1 Feature Extraction

The features used for this network were simple run-length codings of the characters. Run-length encoding (RLE) is a simple, loss-less, method of feature extraction where runs of data (i.e., sequences of consecutive values) are stored as a single data value and count. RLE works best on data where n is very small, such as binary images of characters. Consider the following example where the input length is 50 and the RLE is only 6 (value, count) pairs.

$$\begin{aligned} S &= 0000111100111111000011111111111111111111111111111111 \\ RLE(S) &= \{(0, 4), (1, 4), (0, 2), (1, 6), (0, 4), (1, 30)\} \end{aligned}$$

An input string S which has n changes from 0 to 1 or 1 to 0 requires $(n + 1)$ bits to store the values and $\log(S)$ bits to store the counts. The RLE of the input image can be computed in $O(n \times m)$ polynomial time:

```
COMPUTE-FEATURE-VECTOR(img, n, m)
1  x ← ∅
2  y ← ∅
3  for row ← 1 to n
4      do for col ← 1 to m
5          do val ← img[row, col]
6              count ← 1
7              col ← col + 1
8              while img[row, col] = val
9                  do count ← count + 1
10                 col ← col + 1
11                 y[row] ← y[row] + (val, count)
12 for col ← 1 to m
13     do for row ← 1 to n
14         do val ← img[row, col]
15             count ← 1
16             row ← row + 1
17             while img[row, col] = val
18                 do count ← count + 1
19                 row ← row + 1
20                 x[col] ← x[col] + (val, count)
21 return x, y
```

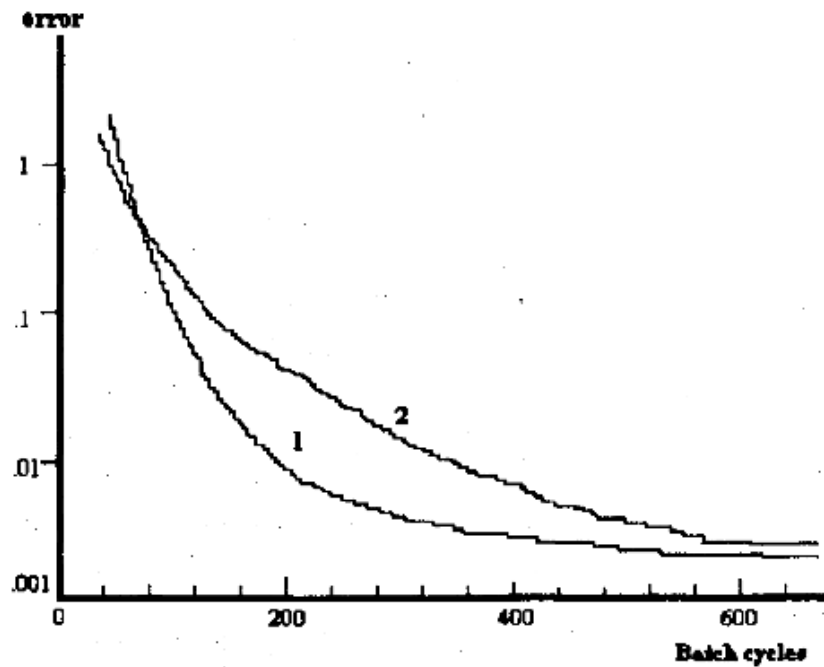
5.2 Neural Network Structure

As noted above, the structure of the neural network was modified during the training.

5.3 Training

5.4 Results

Application of the structure adapting technique proved to produce worse results than a conventional, static structure network. In the graph below, the conventional, static structure network is represented by curve 1 and the structure adapting network is represented by curve 2. Although the structure adapting network out performs the static structure network at first, after approximately 80 iterations the static structure network yields better results.



6 Conclusion

Artificial neural networks are commonly used to perform character recognition due to their high noise tolerance. As shown above, the systems have the ability to yield excellent results.

The feature extraction step of optical character recognition is the most important. A poorly chosen set of features will yield poor classification rates by any neural network. Features must be chosen such that they are loss-less or still accurately represent the character. However, loss-less feature extraction does not guarantee good results. Choices for feature extraction include:

- Use of the entire input image as the feature vector. However this requires a huge network that must be trained to millions of iterations. [1]
- Computing the vertical and horizontal projections of the characters. This is a lossy reduction. [4]
- Computing the run lengths of the character, providing a loss-less reduction in information. [7]
- Use hand selected features chosen by human experts to classify characters. This is a lossy reduction. [6]

These features vectors are fed into feed-forward, back-propagation artificial neural networks. The output of the network determines the correct character class. Trivial systems work only a a single font [7] [4], while more complex systems can recognize many fonts using the same network [1] [6]. Most systems can accept a character printed in any size font by performing scaling and normalizing before computing the feature vectors. Below is a comparison of the 4 different techniques discussed in the papers surveyed:

Paper	Feature Extraction	Lossy/Loss-less	Font Size	# Fonts	Accuracy
[7]	RLE	Loss-less	Any	1	Unknown
[4]	Projections	Lossy	Any	1	99%
[6]	Chosen by Experts	Lossy	Any	16	99.3%
[1]	Image Matrix	Loss-less	Any	12	100.0%

References

- [1] H.I. Avi-Itzhak, T.A. Diep, and H. Garland. High accuracy optical character recognition using neural networks with centroid dithering. *Transactions on Pattern Analysis and Machine Intelligence*, 17(2):218–224, Feb 1995.
- [2] 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, July 1996.
- [3] Tsu-Chang Lee. *Structure level adaptation for artificial neural networks: theory, applications, and implementations*. PhD thesis, Stanford University, Stanford, CA, USA, 1990. Adviser-Allen M. Peterson.
- [4] N. Mani and B. Srinivasan. Application of artificial neural network model for optical character recognition. *Systems, Man, and Cybernetics, 1997. 'Computational Cybernetics and Simulation', 1997 IEEE International Conference on*, 3:2517–2520, October 1997.
- [5] George L. Nagy, Stephen V. Rice, and Thomas A. Nartker. *Optical Character Recognition: An Illustrated Guide to the Frontier*. Kluwer Academic Publishers, Norwell, Massachusetts, USA, 1999.
- [6] E.M. de A. Neves, A. Gonzaga, and A.F.F. Slaets. A multi-font character recognition based on its fundamental features by artificial neural networks. *Cybernetic Vision, 1996. Proceedings., Second Workshop on*, pages 196–201, December 1996.
- [7] J.M. Ramirez, P. Gomez-Gil, and D. Baez-Lopez. On structural adaptability of neural networks in character recognition. *Signal Processing, 1996., 3rd International Conference on*, 2:1481–1483, October 1996.