Proceedings of Students/Faculty Research Day, CSIS, Pace University, May 4th, 2007

Application of Neural Networks to Character Recognition

Dong Xiao Ni
Seidenberg School of CSIS, Pace University, White Plains, NY

-

Abstract

This paper describes the basic biological neuron and the artificial computation model; outlines network architectures and learning processes; and presents multilayer feed-forward networks. It presents two OCR demo applications, one with VB.Net, one with C#.Net. It concludes with a real-world application -- Advanced Facer Canceller System-Optical Character Reader from United State Postal Service.

Introduction

An Artificial Neural Network (ANN) is an information processing paradigm that is inspired by the way biological nervous systems, such as the brain, process information. The key element of the ANN paradigm is the novel structure of the information processing system. It is composed of a large number of highly interconnected processing elements (neurons) working in unison to solve specific problems [1]. ANNs, like people, learn by example. An ANN is configured for a specific application, such as pattern recognition or data classification, through a learning process. Learning in biological systems involves adjustments to the synaptic connections that exist between the neurons. This is true of ANNs as well.

A neural network is a powerful data modeling tool that is able to capture and represent complex input/output relationships. The motivation for the development of neural network technology stemmed from the desire to develop an artificial system that could perform "intelligent" tasks similar to those performed by the human brain. Neural networks resemble the human brain in the following two ways: they acquire knowledge through learning, and the knowledge is stored within inter-neuron connection strengths known as synaptic weights [1, 12]. The true power and advantage of neural networks lies in their ability to represent both linear and non-linear relationships and in their ability to learn these relationships directly from the data being modeled. Traditional linear models are simply inadequate when it comes to modeling data that contains non-linear characteristics.

The most common neural network model is known as a supervised network because it requires a desired output in order to learn. The goal of this network type is to create a model that maps the input to the output using historical data so that the model can then be used to produce the output when the desired output is unknown. A graphical representation of a Multi-Layer Perceptron (MLP) [1, 13] is shown in Figure 1.

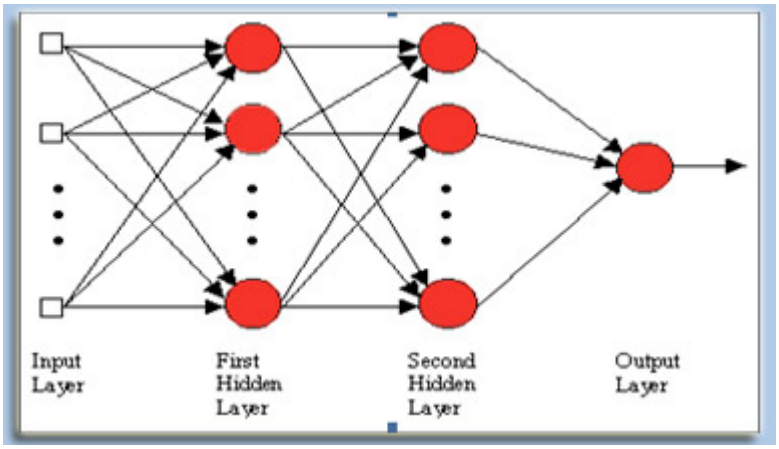


Figure 1: Diagram of 4-layer Perceptron with two hidden layers

The inputs are fed into the input layer and get multiplied by interconnection weights as they are passed from the input layer to the first hidden layer. Within the first hidden layer, they get summed then processed by a nonlinear function (usually the hyperbolic tangent). As the processed data leaves the first hidden layer, again it gets multiplied by interconnection weights, then summed and processed by the second hidden layer. Finally the data is multiplied by interconnection weights then processed one last time within the output layer to produce the neural network output.

Neural networks, with their remarkable ability to derive meaning from complicated or imprecise data, can be used to extract patterns and detect trends that are too complex to be noticed by either humans or other computer techniques. A trained neural network can be thought of as an "expert" in the category of information it has been given to analyze. This expert can then be used to provide projections given new situations of interest and answer "what if" questions. Other advantages include:

- Adaptive learning: An ability to learn how to do tasks based on the data given for training or initial experience.
- Self-Organization: An ANN can create its own organization or representation of the information it receives during learning time.
- Real Time Operation: ANN computations may be carried out in parallel, and special hardware devices are being designed and manufactured which take advantage of this capability.
- Fault Tolerance via Redundant Information Coding: Partial destruction of a network leads to the corresponding degradation of performance. However, some network capabilities may be retained even with major network damage.

Character Recognition by Neural Networks

Optical Character Recognition (OCR) programs are capable of reading printed text. This could be text that was scanned in from a document, or hand written text that was drawn to a hand-held device, such as a Personal Digital Assistant (PDA). OCR programs are used widely in many industries. Many of today's document scanners for the PC come with OCR software that allows you to scan in a printed document and then convert the scanned image into an electronic text format such as a Word document, enabling you to manipulate the text. In order to perform this conversion the software must analyze each group of pixels (0's and 1's) that form a letter and produce a value that corresponds to that letter. Some of the OCR software on the market uses a neural network as the classification engine.

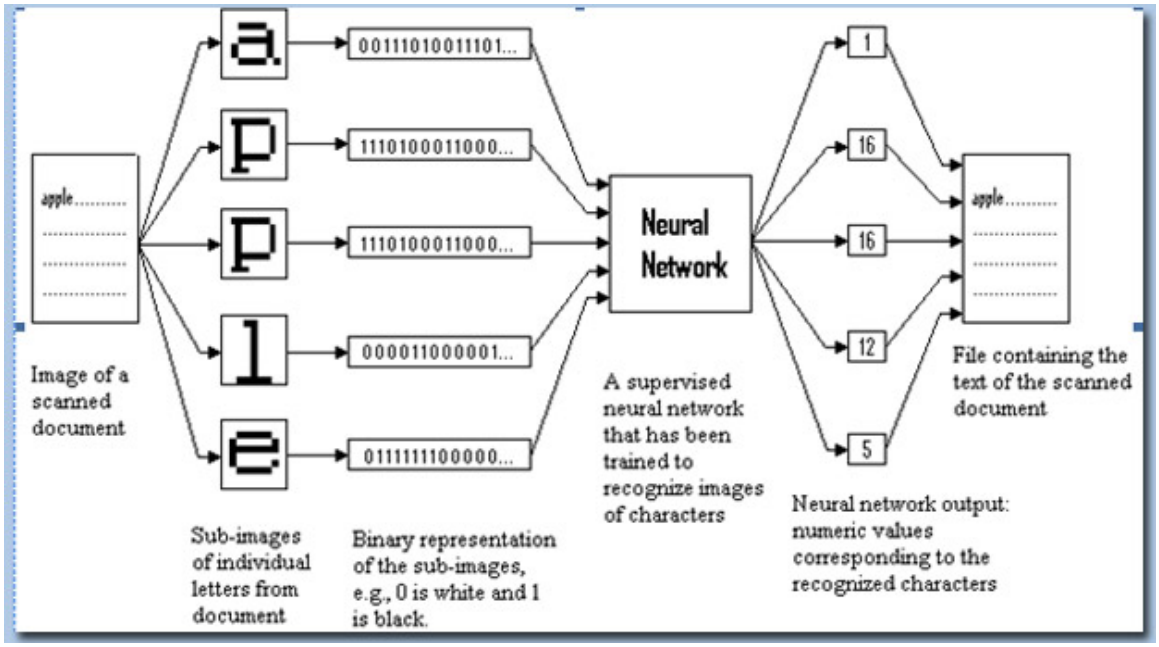


Figure 2: Example of a neural network for OCR.

Figure 2 demonstrates of a neural network used within an optical character recognition (OCR) application [1, 12]. The original document is scanned into the computer and saved as an image. The OCR software breaks the image into sub-images, each containing a single character. The sub-images are then translated from an image format into a binary format, where each 0 and 1 represents an individual pixel of the sub-image. The binary data is then fed into a neural network that has been trained to make the association between the character image data and a numeric value that corresponds to the character. The output from the neural network is then translated into ASCII text and saved as a file.

A simple Demo for OCR

A character recognition demo [10] in VB Visual Studio.Net [2, 14] illustrates a neural network (Figure 3).

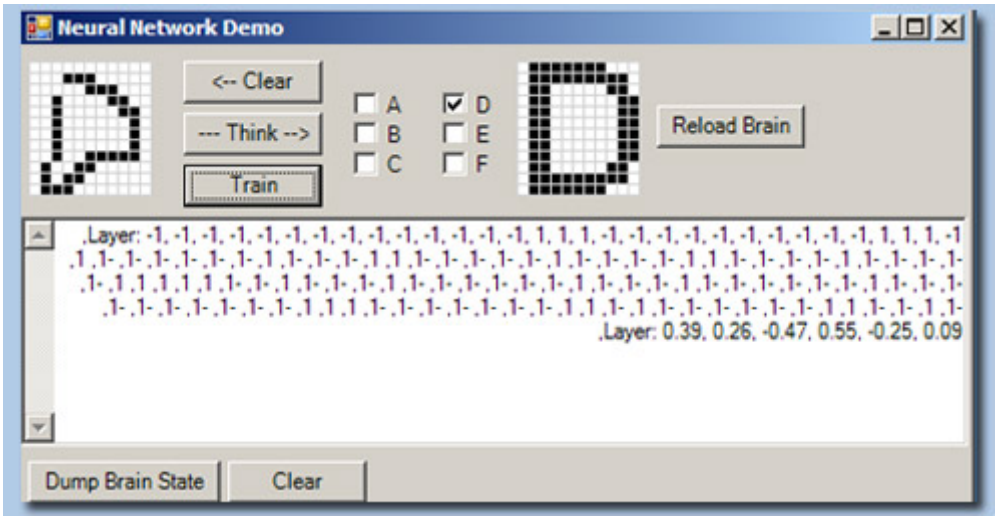


Figure 3: Character recognition demo.

The goal of this window is to demonstrate basic character recognition. The left side in an input you can draw on and the six check boxes labeled "A" through "F" are the outputs.

The inputs are represented by a paint-style grid [10]. You can click and drag on this grid to paint. Starting from the cell you click on, the opposite color is selected from what is already there and that color is applied to all cells you drag over until you let go and click somewhere else. You can also use the "Clear" button to white out all the inputs. The "A" through "F" outputs, as you might imagine, indicate the best matches for the input pattern. It's not a given that any will be checked for a given input, nor that only one will be checked. To see the stored template pattern for a character, right-click on the label for that letter and the pattern appears to the right (in Figure 3 the "D" pattern appears after right-clicking on the "D" check box).

One way to kick off the process is to show the pattern for one of the letters and to draw some semblance of that letter in the input grid. Keep in mind that it's not really the shape that counts, but the overlap of the black pixels you draw with the black pixels in the model image. To start with, none of the outputs will be checked, because we have to undergo training. Click on the "Train" button repeatedly (again, you can click once and hold down the "Enter" key to rapidly repeat this button) until you see that letter's check box checked.

Training can be facilitated by hand-crafting an XML file with all the training cases [10]. Instead, I created GIF images of the letters and wrote code to import them as training cases. In fact, each of the A - F neurons in the output layer have not just one training case to positively identify their own letters, but in fact have one case for each of the other letters to avoid accidentally recognizing them. For example, letter "B" has a training case to tend toward an output of 1 (true) when it recognizes a "B" in the input, but will tend toward an output of -1 (false) when it recognizes "A", "C", "D", "E", or "F" (or "blank", which is one extra test case they all share). The result is 6 * 6 = 36 total training cases. To test the system I initially trained only on

the 6 positive cases and got way too many false positives. Therefore, it is critical to also train on the negatives to avoid having every output neuron firing as soon as a few pixels get into the input grid.

Following are some example input patterns and the resulting neural network outputs. Most are pretty good.

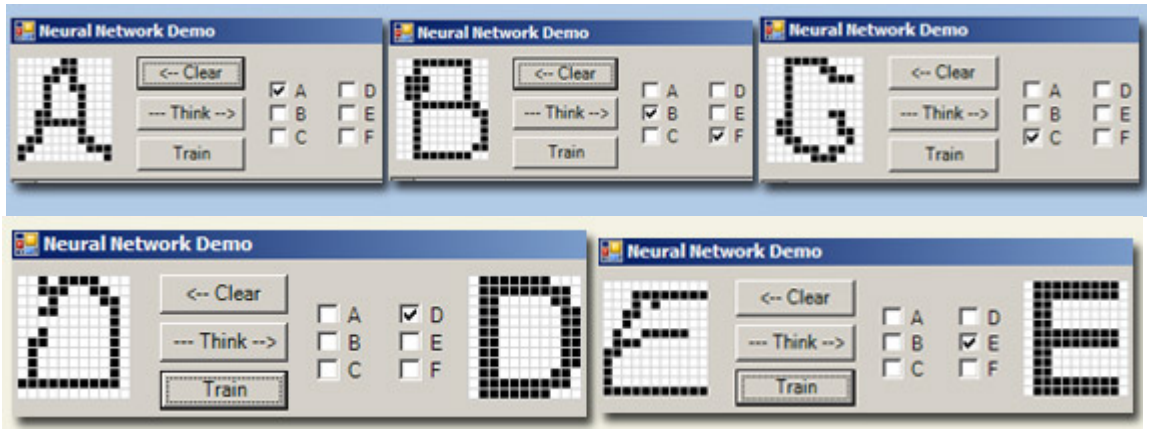


Figure 4**:** Sample cases of input and pretty good guesses by the neural network.

And now for some carefully chosen stinkers that should help illustrate the limitations of this solution.

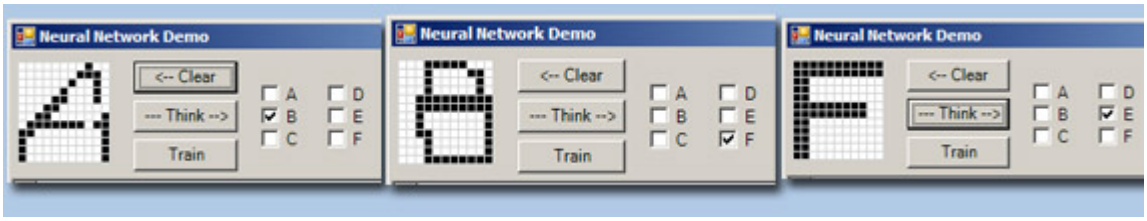


Figure 5: Sample cases of input and pretty bad guesses by the neural network.

Another Neural Network OCR with C#

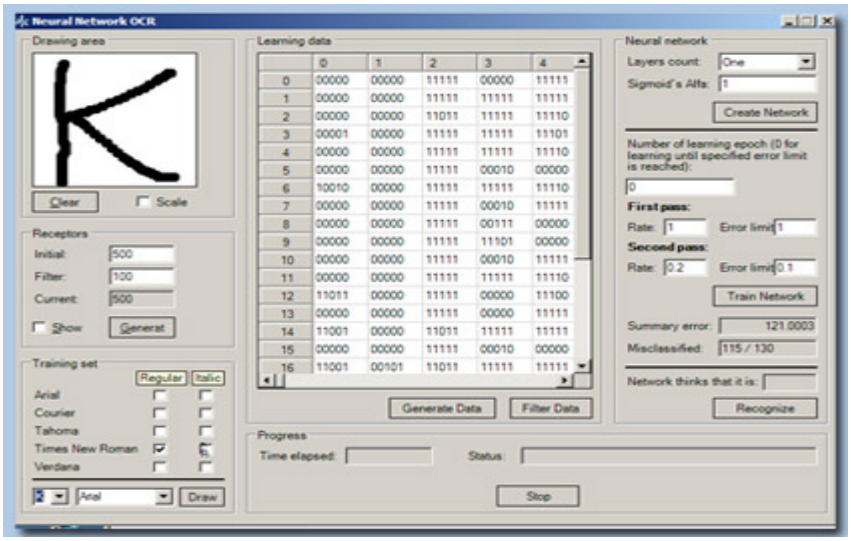
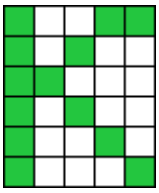


Figure 6: Another neural network OCR application [2,11].

A popular and simple approach to OCR problem is based on feed forward neural network with back propagation learning [3, 7]. The main idea is that a training set should be prepared and then train a neural network to recognize patterns from the training set. In the training step the taught network responds with

desired output for a specified input. For this purpose each training sample is represented by two components: possible input and the desired network's output for the input. After the training step is done, give an arbitrary input to the network and the network will form an output, from which can resolve a pattern type presented to the network.



Assume that t to train a network to recognize 26 capital letters represented as images of 5x6 pixels, something like this one: One of the most obvious ways to convert an image to an input part of a training sample is to create a vector (of size 30 for our case) containing "1" in all positions corresponding to the letter pixel and "0" in all positions corresponding to the background pixels. But, in many neural network training tasks, it's preferred to represent training patterns in so called "bipolar" way, placing into input vector "0.5" instead of "1" and "-0.5" instead of "0"[11]. Such sort of pattern coding will lead to a greater learning performance improvement.
For each possible input a desired network's output needs to be created to complete the training samples. For OCR task it's very common to code each pattern as a vector of size 26 (because we have 26 different letters), placing into the vector "0.5" for positions corresponding to the pattern's type number and "-0.5" for all other positions.

Test application

1. To generate initial receptors set. On application startup it's already generated, so it can skip this step, if are not planning to change the initial amount of receptors or the filtered amount.
2. Select fonts, which will be used for teaching network. Let it be the regular Arial font for the first time.
3. Generate data. In this step the initial training data will be generated.
4. Filter data. In this step the initial receptors set as well as the training data will be filtered.
5. Create network - a neural network will be created.
6. Train network - neural networks training.

AFCS-OCR -- A Real world OCR Application

One of the largest uses of OCR systems is the United States Postal Service. In the 1980's the US Postal Service had many Letter Sorting Machines (LSMs). These machines were manned by human clerks that would key the zip codes of sixty letters per minute. These human letter sorters have now been completely replaced by computerized letter sorting machine – AFCS-OCR (Advanced Facer Canceller System-Optical Character Reader) [9]. This new generation of letter sorting machines is enabled with OCR technology. These machines scan the incoming letters and read the zip code. Using the ZIP code these letters can be routed to their correct destination cities.

The **Enricher** (Unit 13) includes most of the electronics of the AFCS-OCR and serves as the communication interface with the new image recognition components, the ACP and RCR [9]. All of the new functionality of the AFCS-OCR is used in the Enricher unit. Images are processed by the ACP and RCR to resolve a 5-digit ZIP code result and determine if the destination is local or outgoing. Letter-mail destination type is used for sorting purposes. After a sorting decision has been made, the ACP and RCR will attempt to finalize the letter-mail address.

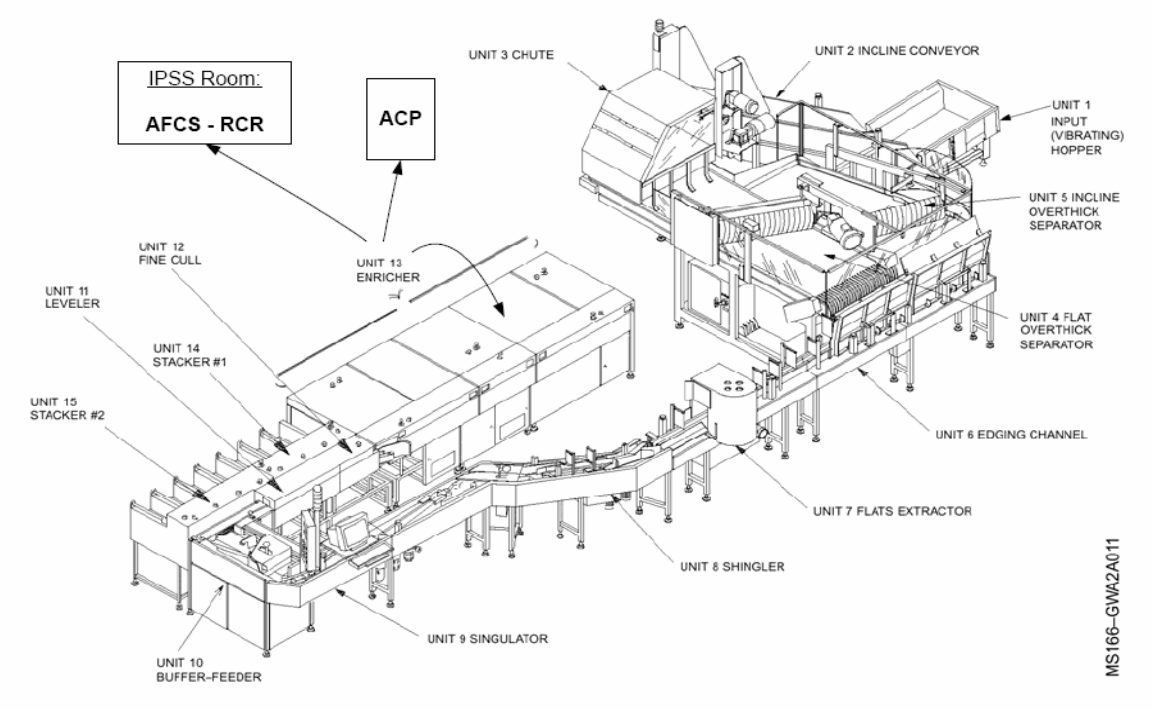


Figure 7: AFCS-OCR System Diagram [9].

Conclusion

Clearly the term artificial neural networks encompass a great variety of different software packages with many different types of artificial neurons, network architectures, and learning rules. These different networks can, in turn, be applied to a diverse range of functions in everything from beer manufacturing to better understanding the properties of the biological brains on which they are based.

References

[1] José C. Principe, Neil R. Euliano, Curt W. Lefebvre "Neural and Adaptive Systems: Fundamentals Through Simulations", ISBN 0-471-35167-9
[2] Microsoft web site: Microsoft web site: http://msdn2.microsoft.com/en-us/default.aspx
[3] Christopher M. Frenz,Touch All the Bases: Give Your .NET App Brains and Brawn with the Intelligence of Neural Networks, http://msdn.microsoft.com/msdnmag/issues/05/05/NeuralNetworks/#S2
[4] Articles –Design Patterns, Neural Networks, C#, Programming
http://amazedsaint-articles.blogspot.com/2006/05/brainnet-neural-network-library-part-i_22.html
[5] M.Villani, M.Curtin, Ngo J Simone, H.St. Fort, C.Tappert. S. –H Cha. Keystroke Biometric Recognition Studies on Long Text Input over the Internet, CSIS Pace University, Pleasantville, New York 10570
[6] Microsoft web site: SQL Server 2005 Books Online "Microsoft Neural Network Algorithm (SSAS)", http://msdn2.microsoft.com/en-us/library/ms174941.aspx
[7] Kingsley Tagbo, C# Neural Network Multilayer Feedforward Backpropagation Algorithm Version 1.0- Open source code release, http://www.kdkeys.net/blogs/kingsleytagbo/archive/2005/04/09/4514.aspx, 04/09/2005
[8] S. Marsland, J. Shapiro, U. Nehmzow. A self-organising network that grows when required. Neural Networks,
[9] US Postal Service@, Engineering Systems/Process Integration AFCS-OCR, http://blue.usps.gov/site/wcm/connect/resources/file/eb1591074c8f1ab/SPI%20AFCS-OCR%20Mail%20Flow%20Guide%20v1.1%20Sec%202%20System%20Overview.pdf?MOD=AJPERES
[10] Jim Carnicelli, Neural Network Demo, http://utopia.csis.pace.edu/cs615/2006-2007/team7/Software/OCR_vbnet.zip
[11] Andrew Kirillov, Neural Networks, http://utopia.csis.pace.edu/cs615/2006-2007/team7/Software/OCR_Csharp.zip
[12] NeuroIntelligence-Alyuda Research, http://www.alyuda.com/neural-network-software.htm
[13] NeuroDimension Inc web site. - Neural Network Software, http://www.nd.com/
[14] Learn How to Build A Provider Framework For .NET, Articles - Design Patterns, Neural Networks, C#, Programming, http://amazedsaint-articles.blogspot.com,March 15, 2007